Umeå University
Department of Computing Science
TDBC86 — Database Concepts
Examination: December 20, 2002

Name and ID number: ____

Signature: ____

1. All answers must be written in English.
2. A pocket engineering calculator (miniräknare) and English/X - X/English dictionary may be used. No other help materials are allowed.
3. The examination consists of two parts. In Part I, which includes Problems 1–4, on pages 1–8, the answers **must** be written on the sheets provided. In general, only the answers will be checked for these questions; derivations need not be provided and will not be evaluated. In Part II, which includes Problems 5–11, the answers must be written on separate sheets. If appropriate for Problems 5–11, work showing how the answers were obtained will be evaluated, and in some cases is required.
4. You must turn in this cover sheet, the answer sheets for Part I (pages 1–8), and the pages containing your answers for Problems 5–11. Collate the sheets in numerical order of the problems, and write on only one side of the paper. Write your name and personnummer on the top of each sheet. For questions 5–11, write the question number on the top of each sheet as well.
5. The examination has a total of 800 points, which constitutes 80% of the course grade. The other 20% is based upon the obligatory exercises, as explained in the course syllabus.
5. For Problems 5–11, you have the choice, for each part, to do the problem, or to skip it for partial credit. 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 15% 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 8(b) blank, your answer to that question will not be graded, even if it is completely correct. On the other hand, if you place an X in box 8(b), but provide no answer whatsoever to that question, you will not receive 15% of the points for that question. It is strongly recommended that you use a pencil, in case you change your mind! *Note that Problems 1–4 do not carry this option. No points are awarded for skipping them.*

Prob	5	6	7	8	9	10	11
(a)							
(b)							
(c)							
(d)							
(e)							

Name and ID number: ______________________________

(1: 80 points total) For each of the following statements, circle the T if the assertion is true, and the F if it is false. The point total will be computed as $4 \times C - 2 \times I$ with C = number of correct answers and I = number of incorrect answers; however, the score may never be less than zero. A blank answer is neither in C nor in I.

T F (a) The schema of a database in a modern relational system such as PostgreSQL is always in BCNF.

T F (b) If a single-relation schema which is constrained by functional dependencies is in 3NF, and it has only one candidate key, then it is in BCNF as well.

T F (c) If a single-relations schema which is constrained by functional dependencies is in 3NF, and each candidate key consists of a single attribute, then it is in BCNF as well.

T F (d) Every relational schema with exactly two attributes which is constrained by functional dependencies is in BCNF.

T F (e) Every relational schema which is constrained by functional dependencies has a dependency-preserving and lossless decomposition into 2NF.

T F (f) Every relational schema which is constrained by functional dependencies has a dependency-preserving and lossless decomposition into 3NF.

T F (g) With a pure deferred update strategy, it is never necessary to undo transactions.

T F (h) In the authorization model of PostgreSQL, if user A grants read privileges to user B on relation C, then user B automatically receives the authority to delete relation C as well.

T F (i) For every query Q in the relational calculus, there is an equivalent query Q' which does not use the for all ($\forall$) operator.

T F (j) SQL supports the division operation directly via the `DIVIDE` keyword.

T F (k) It is usually the case that an index on a secondary key is dense.

T F (l) To prevent cascading rollback, it suffices to ensure that no transaction reads a value which was written by a transaction which has not yet reached its commit point.

T F (m) In query optimization, it is generally a good idea to rearrange things so that joins are performed before select and project operations.

T F (n) In ODBC, the database server and application client must run the same operating system.

T F (o) The system catalog is a database which provides information on the relations of the underlying schema.

T F (p) An effective way to avoid livelock in the wait-die strategy is to restart aborted transactions with their old timestamps.

T F (q) A substantial limitation of the relational model is that there is no way to incorporate persistent data.

T F (r) A no-steal/force caching strategy is likely to be more efficient in terms of system resource use than a steal/no-force strategy.

T F (s) With the pure deferred update strategy, transactions never need be rolled back.

T F (t) In the authorization model of SQL, only the system administrator may grant privileges.

Name and ID number: ____

(2: 60 points total; 6 points for each question) For each of the questions below, place an X between the parentheses to the left of the correct responses to the question. A question will have at least one correct choice, but it may have several. Mark all correct choices.

(a) Unsafe queries:

( ) are impossible to express in the relational algebra.
( ) are impossible to express in SQL.
( ) are impossible to express in the relational domain calculus.
( ) do not have mathematically well-defined answers.

(b) With strict two-phase locking:

( ) All shared (read) locks are held until the transaction commits.
( ) All exclusive (write) locks are held until the transaction commits.
( ) Deadlock cannot occur.
( ) Livelock cannot occur.

(c) Relative to strict 2PL schedules, rigorous 2PL schedules:

( ) reduce the possibilities for concurrency.
( ) ensure recoverability under a wider variety of circumstances.
( ) offers no particular advantages other than possibly ease of implementation.
( ) rule out the possibility of blind writes.

(d) At a checkpoint:

( ) All transactions are temporarily suspended.
( ) All volatile data is written to nonvolatile storage.
( ) A checkpoint record is written to the log.
( ) All pin bits of cache pages are cleared.

(e) When a cache page is pinned:

( ) It contains correct data which has not yet been written to disk.
( ) It may contain incorrect data and so must not be written to the permanent database.
( ) It is often shadowed as well to facilitate the execution of checkpoints.
( ) The dirty bit is set.

(f) An object-oriented approach is likely to be superior to a relational approach in an application which involves:

( ) entities which have procedures associated with them.
( ) a great deal of concurrency.
( ) entities which have a natural hierarchy associated with them.
( ) many recursive queries.

(g) The seek time for a top-grade hard disk for use in a DBMS server:

( ) is approximately 3 to 5 milliseconds.
( ) is approximately 10 to 15 milliseconds.
( ) refers to the time required to position the heads over the correct cylinder.
( ) refers to the time required to spin the platters to the proper sector.

(h) Conservative 2PL schedules:

( ) can only be used in the context of deferred updates.
( ) allow nonexclusive (*i.e.*, read) locks to be acquired at any time.
( ) prevent deadlock from occurring.
( ) do not allow two distinct transactions to possess read locks on the same data item simultaneously.

(i) RAID arrays of disk drives:

( ) may be used to improve the speed of both reads and writes.
( ) may be used to decrease the damage resulting from a disk failure.
( ) require much more sophisticated checkpoint techniques than do single-disk architectures.
( ) are incompatible with relational database systems.

(j) A major advantage of a B^+-tree over a B-tree is that:

( ) The algorithm to insert a new data item is much simpler.
( ) A B^+-tree is more fully balanced.
( ) The data may be indexed on two keys at once.
( ) Sequential access to the data is supported.

Name and ID number: ______________________________

(3: 100 points total; 10 points for each part) Important: The answers to this question must be indicated below; do not write the answers to this question on a blank sheet.

The following relational schema with relation $R[ABCDEFG]$ and functional dependencies $\mathcal{F} = \{AB \to C,\ CD \to EF,\ CG \to F,\ F \to A,\ G \to BD\}$ applies to all ten parts of this problem.

Given below are six projections of $R[ABCDEFG]$. Assume that each such projection is constrained by exactly the functional dependencies in $\mathcal{F}^+$ which embed in the projection. In the space provided, give all of the candidate keys for the projection, as well as a basis of the functional dependencies from $\mathcal{F}^+$ which embed in it. Also, circle *all* of the normal-form properties which the projection possesses (even if some are implied by others). Circle the word none just in case none of the normal-form properties holds.

(a) Projection: $R[ABCDEFG]$

Candidate keys: ______________________________

FD basis: ______________________________

Normal forms: None 2NF 3NF BCNF

(b) Projection: $R[ABCDF]$

Candidate keys: ______________________________

FD basis: ______________________________

Normal forms: None 2NF 3NF BCNF

(c) Projection: $R[ACDEF]$

Candidate keys: ______________________________

FD basis: ______________________________

Normal forms: None 2NF 3NF BCNF

(d) Projection: $R[ABDF]$

Candidate keys: ______________________________

FD basis: ______________________________

Normal forms: None 2NF 3NF BCNF

(e) Projection: $R[ABCG]$

Candidate keys: ______________________________

FD basis: ______________________________

Normal forms: None 2NF 3NF BCNF

(f) Projection: $R[ABDEF]$

Candidate keys: ______________________________

FD basis: ______________________________

Normal forms: None 2NF 3NF BCNF

Shown below are four decompositions of the schema on the previous page. For each decomposition, indicate whether or not it has the properties of *lossless* and *dependency preserving* by circling the appropriate word(s). Circle the word *neither* just in case it is neither lossless nor dependency preserving.

(g) Decomposition: $\{R[ABCD], R[ABEFG]\}$.

Neither Lossless Dependency preserving

(h) Decomposition: $\{R[ABC], R[ABDEFG]\}$.

Neither Lossless Dependency preserving

(i) Decomposition: $\{R[ABCDFG], R[ACDEF]\}$.

Neither Lossless Dependency preserving

(j) Decomposition: $\{R[ABC], R[ACDEFG]\}$.

Neither Lossless Dependency preserving

Name and ID number: ____

(4: 80 points total) A simple database system has five data objects, named v, w, x, y, and z. The following four transactions are given.

$$\begin{aligned} T_1 &= \mathsf{r}_1(x)\mathsf{r}_1(z)\mathsf{w}_1(x)\mathsf{w}_1(z) \\ T_2 &= \mathsf{r}_2(w)\mathsf{r}_2(y)\mathsf{w}_2(z)\mathsf{w}_2(y) \\ T_3 &= \mathsf{w}_3(w)\mathsf{w}_3(x)\mathsf{r}_3(z)\mathsf{w}_3(z)\mathsf{r}_3(v) \\ T_4 &= \mathsf{r}_4(y)\mathsf{w}_4(y)\mathsf{w}_4(x) \end{aligned}$$

(a: 15 points) In the table below, for each transaction, circle the names of those data items on which it executes a blind write. Circle *none* just in case the transaction does not execute any blind writes.

T_1 blind writes: v w x y z none $\quad$ T_2 blind writes: v w x y z none

T_3 blind writes: v w x y z none $\quad$ T_4 blind writes: v w x y z none

(b: 5 points) Indicate the number of distinct serial schedules which are possible for this set of transactions. (No derivation is necessary; just give the number.)

Number of distinct serial schedules: __________

Shown on the next page are three schedules for this set of transactions. For each of these schedules, do the following **on that sheet**:

- Using the nodes provided, draw the edges of the conflict graph.
- Under the name of each property (*serial*, *view serializable*, *conflict serializable*), circle **T** (*true*) or **F** (*false*), depending upon whether or not the schedule possesses that property.
- If the schedule is view serializable, give the transaction sequence (*e.g.*, $T_4T_1T_3T_2$) of an equivalent serial schedule. If the schedule is not view serializable, no answer is required for this part.
- Assume that each transaction begins just before performing its first (read or write) operation, and that it commits just after performing its final (read or write) operation. Under these assumptions, indicates whether or not the schedule is recoverable by placing an $\times$ in the appropriate $\square$ as so: $\boxtimes$. In the case that it is not recoverable, also identify a pair of transactions which may cause a recovery problem by writing the appropriate subscripts in the spaces provided. (In the case that there are several such instances, only one need be identified.)

Name and ID number: ______________________________

(c: 20 points) $r_1(x)w_3(w)r_2(w)r_2(y)w_3(x)r_4(y)w_4(y)r_1(z)r_3(z)w_3(z)w_1(x)w_2(z)w_2(y)r_3(v)w_1(z)w_4(x)$

T_1 T_2 serial view serializable conflict serializable

T F T F T F

T_3 T_4 Equivalent serial schedule: ______________________

☐ This schedule is recoverable.

☐ This schedule is not recoverable because transaction T_____ may need to be rolled back after its commit point if transaction T_____ aborts.

(d: 20 points) $w_3(w)r_2(w)r_1(x)w_3(x)r_1(z)w_1(x)w_1(z)r_4(y)w_4(y)w_4(x)r_3(z)w_3(z)r_2(y)w_2(z)w_2(y)r_3(v)$

T_1 T_2 serial view serializable conflict serializable

T F T F T F

T_3 T_4 Equivalent serial schedule: ______________________

☐ This schedule is recoverable.

☐ This schedule is not recoverable because transaction T_____ may need to be rolled back after its commit point if transaction T_____ aborts.

(e: 20 points) $r_2(w)r_2(y)w_2(z)w_2(y)r_1(x)r_1(z)w_1(x)w_3(w)w_3(x)w_1(z)r_3(z)w_3(z)r_4(y)w_4(y)w_4(x)r_3(v)$

T_1 T_2 serial view serializable conflict serializable

T F T F T F

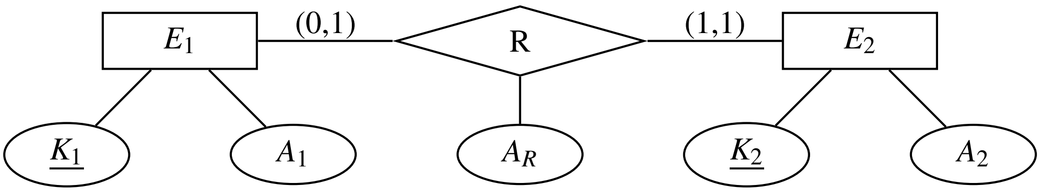
T_3 T_4 Equivalent serial schedule: ______________________

☐ This schedule is recoverable.

☐ This schedule is not recoverable because transaction T_____ may need to be rolled back after its commit point if transaction T_____ aborts.

(5: 60 points total; 30 points for each part)

(a) Consider an entity relationship subschema in which the two structural cardinality constraints are (0,1) and (1,1), respectively, as shown below. (Here (x,y) means that the relationship involves at least x and at most y entities.)



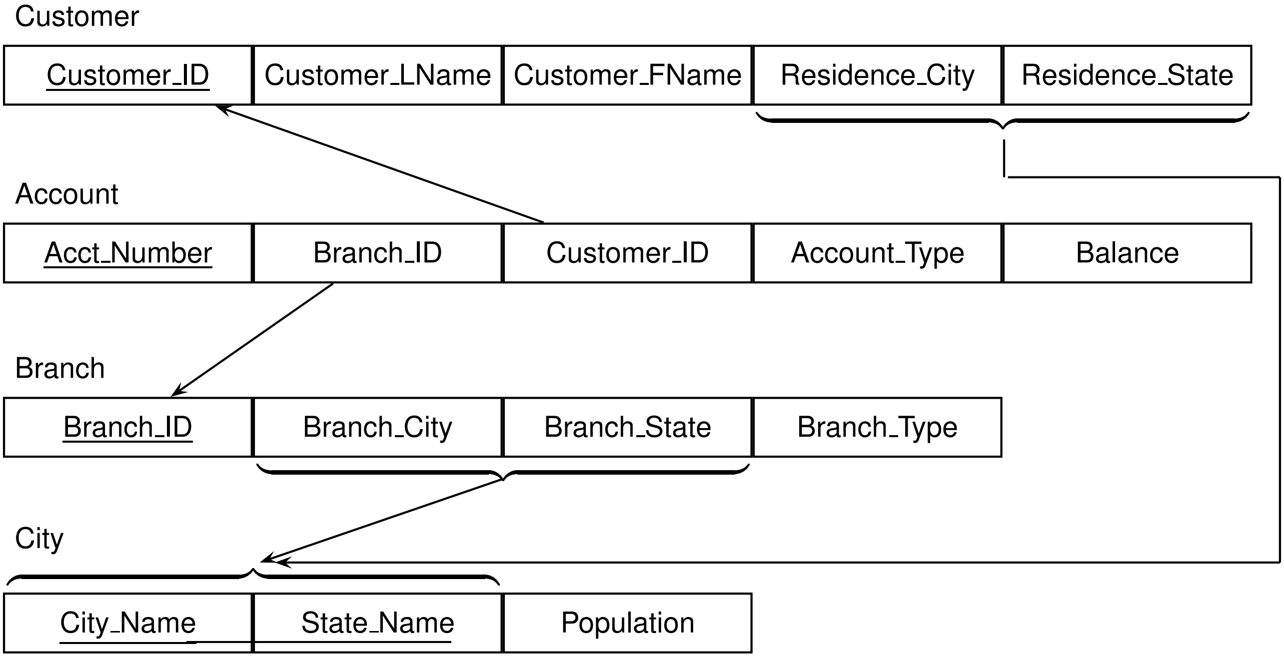
In this diagram, K_1 and K_2 are the keys of E_1 and E_2, respectively, while A_1, A_2, and A_R are other attributes.

(i) Give a simple, concrete example of such a situation, preferably within the context of the company database of the textbook. (It is most important to give concrete meaning, via the example, to the entities E_1 and E_2, the relationship R, and the keys K_1 and K_2. The other attributes are less important, and may be selected from any which are applicable.)

(ii) Illustrate two distinct ways in which this schema may be transformed to a relational model, and indicate clearly which is preferable, and why.

(b) In the company database of the textbook, a department may be housed at many locations. Illustrate two fundamentally different ways of representing this knowledge within entity-relationship modelling.

The following schema applies to Problems 7 through 10. In each case, the primary key is underlined, and the arrows run from foreign keys to their primary counterparts.



The following queries apply to Problems 7 through 9, and reference the schema defined above.

(a) Find the names (last and first) of those customers who have an account in a branch located in Burlington, Vermont and an account located in a branch in Burlington, Massachusetts. (Burlington is a city name, and Vermont and Massachusetts are names of states.)

(b) Find the names of those customers who have accounts in at least two different states. (The customer may have any number of accounts, but there must be at least two which are held in branches which are located in different states.)

(c) Find the names of those customers who have an account in a branch in every city named Burlington.
(Note: This means that for every city named Burlington, the customer must have an account in *some* branch in that city. It is not mean that that customer must have an account in *every* branch in every city named Burlington.)

Note: In the above schema, the attributes Customer_LName and Customer_FName identify the last and first names of the customer, respectively.

(6: 60 points total; 20 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. Use the join notation described in the course notes: (The symbol $\bowtie$ is used to denote join; in the absence of subscripts it denotes the natural join; any other join conditions must be specified explicitly using subscripts.)

(7: 60 points total; 20 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.

(8: 60 points total; 20 for each part) Solve each of the three queries (a) - (c) using SQL. To keep the solutions simple and consistent, the following restrictions apply:

(i) Directives involving the keyword JOIN may not be used. Express the join conditions directly using conditions in the WHERE clause.

(ii) The solution must consist of a single SQL directive. Creation and subsequent use of temporary tables is not permitted.

(iii) Aggregate operators, such as COUNT, SUM, AVG, MAX, and MIN may **not** be used.

Further note regarding Problems 6-8: Although it is stated in Problem 9 on the next page that $\{\mathsf{Customer\_LName}, \mathsf{Customer\_FName}\}$ is a candidate key for the relation Customer, your solutions in Problems 6-8 may not make this assumption. Your solutions must work correctly even in the case that Customer_ID is the only key for the relation Customer.

(9: 100 points total; 20 points for each part) Provide solutions, in SQL, to the following additional queries. Restrictions (i) and (ii) from Problem 8 on the previous page apply here as well. On the other hand, aggregate operations may be used, but only in the SELECT and HAVING clauses of queries and subqueries.

(a) Give an SQL CREATE TABLE command which creates the relation Customer of the figure on page 10. You may assume that the appropriate definitions for the other relations have already been made. Assume further the following:

- The field Customer_ID consists of a sequence of exactly ten characters.
- The fields Customer_LName and Customer_FName each consist of strings of at most fifteen characters.
- The field Residence_City consists of strings of at most twenty characters.
- The field Residence_State consists of a sequence of exactly two characters. (For the interminably curious, encodings of states are used here; *e.g.*, Vermont = VT and Massachusetts = MA.)
- In addition to the indicated primary key, the set {Customer_LName, Customer_FName} is a candidate key.
- All primary and candidate key fields must be non-null.
- All foreign key fields must be non-null.
- All other fields may be null.

(b) For each state which is listed in the City relation, give the average balance of all accounts whose owners (the customers) reside in that state. The result must list both the state and the average balance. If the state does not have a branch with accounts, nothing should be displayed for that state.

(c) For each customer who has a total balance in all accounts of at least 500, display the customer ID, the customer name (last and first), the total balance in all accounts for that customer, and the total number of accounts which that customer holds.

(d) Display the name of the state (as represented in the City relation) which houses that city named Burlington which has the greatest population over all such cities with that name. (In other words, find the city named Burlington with the greatest population, and display its state.) If there is population tie, display all states which house a Burlington with the maximal population.

(e) Increase by 100 the balance of each account whose owner resides in the state of New Hampshire. (The two-letter abbreviation for New Hampshire is NH.)

(10: 80 points total) The following parameters apply to a relation which is to be stored in a B^+-tree. Assume that each node, interior or leaf, is stored in one page, and that each tuple is stored as one record.

Number of tuples in the relation	4×10^{10}
Size of one tuple	200 bytes
Size of pointer to a B^+-tree node	6 bytes
Size of internal B^+-tree key	24 bytes
Page size	4096 bytes
Sequential-access pointers in the leaf nodes	12 bytes total

Answer the following. To receive credit, you must show your work, including the formulas used and the values which were substituted for the variables of the formula. (If you show just numerical computations, without an underlying formula, full credit will not be awarded, even if the final answer is correct.)

(a: 12 points) Compute the maximum number of indices per non-leaf node.

(b: 12 points) Compute the maximum number of records per leaf node.

(c: 12 points) Compute the maximum depth that such a B^+-tree may have. (The depth is defined as the length of a path from the root to a leaf.)

(d: 12 points) Compute the minimum depth that such a B^+-tree may have.

(e: 32 points) For a B^+-tree of the minimum depth computed in (d), compute the average number of records in each leaf node, assuming that each interior node, including the root, contains 55% of the maximum possible number of indices which was computed in part (a), rounded down to the nearest integer. State whether or not this number is consistent with the bounds on the number of records per leaf node, as computed in part (b).
Note: In the above, round down the number of indices per interior node, not the final number of records per leaf node.

(11: 60 points total) The following questions relate to the use of ODBC.

(a) In ODBC, there are (at least) two distinct ways to construct and submit an SQL query with parameters to a database. First, one can build a string representing the SQL query, with parameter bindings, directly in the programming language, and then submit it to the database system using the appropriate ODBC call. Second, one can use the ODBC call `SQLBindParameter`, together with other appropriate ODBC calls, to bind the parameters directly to a query with variables. Identify the circumstances under which the second approach is superior to the first, and explain why it is superior.

(b) In a C program with ODBC calls, the following directive occurs.

```
SQLINTEGER  salary;
```

You replace it with the following directive, and the program compiles and appears to run without problem, exactly as it did with the above directive.

```
int  salary;
```

Explain why this change is nonetheless not a good idea, and what might go wrong under suitable circumstances.

Appendix

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

$$R(m,r,d) = (m+1) \times (r+1)^d - 1$$

with

$$\begin{aligned}
m &= \text{the total number of indices at the root node.} \\
r &= \text{the total number of records in each non-root node.} \\
d &= \text{the depth of the tree; } \textit{i.e.}, \\
  & \text{ the length of a path from the root to a(ny) leaf node.} \\
R(m,r,d) &= \text{the total number of records in the tree.}
\end{aligned}$$

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

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

with

$$\begin{aligned}
m &= \text{the total number of indices at the root node.} \\
q &= \text{the total number of indices in each non-root, non-leaf node.} \\
r &= \text{the total number of records in each leaf node.} \\
d &= \text{the depth of the tree; } \textit{i.e.}, \\
  & \text{ the length of a path from the root to a(ny) leaf node.} \\
R(m,q,r,d) &= \text{the total number of records in the tree.}
\end{aligned}$$