CSC 321/621 Spring 2012 Name: KEY
Wake Forest University
Test #1 – March 6, 2012

Show all your work and reasoning for your answers in the space provided. Only one solution per question will be accepted, so be certain you clearly mark your final answer if you have a lot of scratch work for a problem. This exam is to be done individually according to the honor code at Wake Forest University.

Problem 1. *For the domain described below, answer the following questions concerning an Entity-Relationship (ER) model of the domain*.

We have been tasked with building a database representing information about academia and the structure of universities. There are requirements to model universities, colleges (administrative units within a university), departments (administrative units within a college) and buildings. As one example, the "Manchester Hall" building houses the "Computer Science" department of the "Undergraduate College" of the university named "Wake Forest University".

For a university, we wish to record the name and the year it was founded. For a college, we wish to record its name, its dean, and be able to associate it with the university it is a part of. For departments, we wish to record its name, its chairperson, and the college it is part of. For a building, we wish to record its name, its date of construction, and be able to associate it with the department(s) housed in it.

Names of universities are unique. Names of colleges are unique within a university. Names of departments are unique within a given college. Names of buildings are unique to a given university. A department can be housed in multiple buildings, with each department having a main office in one of the buildings it is housed in. A single building may house multiple departments. Not all buildings have to be associated with an academic department (for example, a gym on campus may just be for recreation). Assume the database maintains lists of all universities, colleges, departments, and buildings (presence in the database is not tied to participating in any given relationship).

a. At right is one part of the ER model. *Provide* and *justify* appropriate lower and upper bound multiplicity values for both sides of the "HousedWithin" relation between Department and Building. Don't forget to justify your choices!

The multiplicities should be 0..* next to the Department entity and 1..* next to the Building entity. Every department is in at least one building (where its main office is), but the description says it could be spread across multiple buildings. This leads to the 1 and * respectively. Some buildings may not house departments (such as the gym), while others may house 1 or more departments (Manchester Hall at WFU hosts both Math and CS). This leads to the 0..*.

b. List any weak entities suggested by the domain description above or write NONE if no weak entities exist.

College, Department, and Building are all weak entities, as they rely on other entities (University, College & University, and University respectively) to be uniquely identified.

Problem 2. You have been given a relationship R defined over five attributes (A,B,C,D,E). The following functional dependencies hold: {A→B; A→C; C,D→E; B→D; E→A}. Indicate whether or not you believe the attribute A is a candidate key for this relation and explain why or why not (using functional dependencies in your argument).

To show A is a candidate key, we need to show that all the other attributes are functionally dependent on A. That is, demonstrate A→B, A→C, A→D, A→E.

A→B and A→C are given in our list of known functional dependencies.

A→D can be inferred from transitivity. A→B and B→D are both given, meaning we can infer A→D.

Finally, via A→C (given) and A→D (just shown) [meaning A→CD], CD→E (given), and transitivity, we can show that A→E.

The fact that E→A is irrelevant. It really just means that E could be a candidate key as well (since all other attributes can be shown to be functionally dependent on E (using transitivity, the fact that E→ A, and the fact that we just showed A→everything else)).

Problem 3. A relation and its set of functional dependencies are provided below.

There is only one candidate key:
{employeeID, projectID}

These are the only functional dependencies:

employeeID → employeeLastName

employeeID,projectID → projectRole

employee ID	employee LastName	project ID	projectRole
1	Turkett	1	Designer
1	Turkett	2	Architect
43	Jones	1	Architect
79	Jackson	3	Programmer
82	Jones	2	Programmer

a. How could one state a reasonable contextual equivalent of the *employeeID,projectID → projectRole* functional dependency in plain English? (I am not looking for "The projectRole attribute is functionally dependent on attributes employeeId and projectID").

I was fairly generous in what I took as an answer here as long as it got across the notion that given a specific employeeID and projectID, we would find a unique projectRole. Ultimately what I was hoping for was:

A particular employee has exactly one role for a given project they are working on. (An employee doesn't have two roles on any one project).

b. If the relation above is in 2NF, justify why it is. If it is not, decompose it into multiple relations that are in 2NF.

The relation is not in 2NF as employeeLastNameis only partially dependent on the candidate key and this is a violation of the 2NF definition. The appropriate decomposition is the following two tables:

{employeeID, employeeLastName} {employeeID,projectID,projectRole}

Problem 4. On the last two pages of the exam is a set of relations with example data. Note that you don't have the full tables, but you have enough to see what type of data is in the tables. Also found on those pages is a key providing the meaning of the various relational algebra symbols. Answer the following questions using the provided relations and relational algebra symbols.

For each sub-problem below, provide an appropriate relational algebra expression:

a. Get the colors of all parts.

$\Pi_{P\_COLOR}(P)$

I also ended up taking $\Pi_{P\_ID,\ P\_COLOR}(P)$ since my statement was unclear on whether I wanted the list of parts with their colors or just the list of colors.

b. Get part ids for parts supplied to any job by a supplier in the same city as the job.

Either of these should work:

$\Pi_{P\_ID}((SPJ \bowtie J) \bowtie_{J.J\_City=S.S\_City} S)$

$\Pi_{P\_ID}(SPJ \bowtie (J \bowtie_{J.J\_City=S.S\_City} S))$

c. In plain English, what is returned from the following relational algebra expression? Again, provide a contextual answer, not just the words that map to each symbol.
$\Pi_{J\_NAME}(\sigma_{quantity > 30}(SPJ) \bowtie J)$

Show me the names of all jobs for which an order was made for that job to a given supplier for more than 30 units of a given part.

Saying things such as "jobs needing more than 30 parts" is not the same. The SPJ table is a list of orders made to a supplier for a particular part used for a particular job. Jobs needing more than 30 parts would have 30+ entries in the SPJ table (one row for each of the 30 parts).

Problem 5. Using the relations on the last page of the exam, answer the following questions.

a. Provide an SQL query that returns the requested information

The total number of jobs supplied by the supplier with S_ID 4.

SELECT COUNT (DISTINCT J_ID) FROM SPJ WHERE S_ID = 4; is the simplest version that should work. One needs DISTINCT J_ID, as it is possible that one could have made orders for multiple parts for a given job from the same supplier. COUNT is required instead of SUM – SUM would sum over the job ids (which are just arbitrary values assigned to jobs).

b. Explain in plain English what is returned from the following SQL statement:
SELECT DISTINCT P_NAME, P_WEIGHT FROM P WHERE P_ID IN (SELECT * FROM SPJ NATURAL JOIN S WHERE S_CITY = 'Raleigh') AS RSP;

This query returns the distinct part names and weights that were ordered for a job (having been ordered is the constraint that the part shows up in the SPJ table) and which were supplied by a company based in Raleigh (the S_CITY='Raleigh' constraint).

Problem 6: Answer the following questions concerning the relational algebra operators.

a. What is the fundamental difference between the Cartesian product relational algebra operation and the join operation (the general notion of a join, which covers all of the specializations of joins discussed in class)?

A join is a Cartesian product **followed by selection on some condition.**

b. Argue for or against the following statement: The following two applications of relational algebra operators are equivalent for arbitrary choices of *colA, c,* and *R*. Assume that the choices for *colA, c,* and *R*, whatever they are in a given instance, are applied to both expressions.

$$\Pi_{colA}(\sigma_c(R)) \qquad \text{vs.} \qquad \sigma_c(\Pi_{colA}(R))$$

These are NOT equivalent. The problem states that arbitrary choices can be made for setting colA, c, and R, but whatever the choices are for those parameters, they are applied to both expressions.

Assume that R has three attributes: D,E,F and *c* is a selection based on a property of D and *colA* is a projection to attribute E. The 2nd expression introduces a problem. Since the condition *c* involves an attributes other than the attribute specified in the projection, then the 2nd expression will fail (it becomes invalid). The first expression will run just fine. That is because the selection is done on the whole relation, so applying it is valid and will return a different relation. That returned relation will still be over D,E,F, to which the projection to E can be applied.

Problem 7: True/False and Explain. Fill in the blank with TRUE or FALSE, and then justify your choice.

a. FALSE Every entity in an ER model maps to a relation in a relational model, and every relation in a relational model maps to an entity in an ER model.

   Explanation: The 2nd half of the statement is the problematic part. Some relations in a relational model map to relationships in the ER model (the edges between entities), not to entities themselves.

b. TRUE The facts that row-ordering within a table is unimportant and that there are no repeated rows in a table directly stem from the mathematical definition of relations.

   Explanation: The mathematical definition of relations says they are **sets.** Two rules of set theory are that sets a) do not have repeated elements and b) sets do not have ordering (AKA ordering is irrelevant).

Problem 8: Answer the questions below with the terms *Inserts, Updates, Deletes,* or *None* as appropriate.

Assume that there is a referential integrity constraint (foreign key reference) from a column of table A to a column of table B.

a. What types of operations, if any, on A can violate the constraint?

Referential integrity says that a FK value must either be one of the values that actually currently exists for the attribute being referenced, or must be NULL.

Inserts and Updates are the problems here. These are the cases where one sets a FK attribute to a bogus value that is not actually in the list of valid values one can reference in B.

Most database systems handle these for you, blocking inserts or updates that violate this rule.

b. What types of operations, if any, on B can violate the constraint?

Deletes and Updates are the problem cases here. Delete is removing one of the values from B, meaning you may now have a value in A that references something in B that doesn't exist. Similarly, an update to B can cause the same problem.

These are the two cases you have to tell SQL how to manage (ON UPDATE RESTRICT/CASCADE ON DELETE RESTRICT/CASCADE).

Extra Credit: Provide a relational algebra expression that performs exactly the same work as the following SQL query (it can't just return the same values, it needs to do the same work in the same order): SELECT DISTINCT P_NAME FROM (P NATURAL JOIN (SELECT P_ID FROM (SPJ NATURAL JOIN J) WHERE J_CITY='Winston-Salem) AS WSP);

Π_{P_NAME} (P $\bowtie$ (Π_{P_ID} ($\sigma_{J\_CITY='Winston\text{-}Salem"}$ (SPJ $\bowtie$ J))))

Graduate Student Problems

Problem G1: Prove that every 2-column table that is in 1NF is in 3NF.

We need to first show that every 2-column table is first in 2NF (no non-candidate key attributes with partial dependencies on a candidate key).

A 2-column table must have either:

1. Both attributes composing the sole candidate key {a composite candidate key}
2. Both attributes being single (1-attribute) candidate keys.
3. One being the candidate key, the other not (which is which doesn't matter)

For case 1 and 2, there are no non-candidate key attributes, so it is automatically in 2NF. For case 3, the candidate key is a single attribute (not composite), so there can’t be a partial dependency. Thus, 2-column tables are guaranteed to be in 2NF.

Now we need to show that no non-candidate-key attributes are involved in transitive dependencies.

For transitive dependencies to exist, there must be at least 3 columns (A,B,C). Thus, no transitive dependencies exist in a 2-column table.

(Problem G2 is on reverse)

Problem G2: Pick two of the five primitive relational algebra operators and argue why they must be primitive (that is, they can't be constructed from the other primitive relational operators): *selection, projection, Cartesian product, union, difference*

To answer this question successfully, one had to first indicate a property of their chosen operators that was unique to the operators, and then demonstrate that the other operators alone or together could not do the same thing.

One example: projection

Projection is the only operator that has the property of being able to reduce the number of columns in a relation. Cartesian product can add columns (by joining tuples from two different relations) while selection, union, and difference will keep the same number of columns as the original relations provided as input, as these operators only deal with the rows of the relations.

The argument for Cartesian product is essentially the same as for projection, but arguing that only it can add additional columns.

For another example: selection

Selection is the only operator has the property of being able to reduce the number of tuples in a relation based on an arbitrary attribute-based condition. Cartesian product and union can only add rows. Projection may reduce rows (as duplicates tuples in a projected relation are dropped), but these removed tuples are removed not based on an arbitrary condition. Difference may reduce rows, but only based on tuple set relationships, not on an arbitrary attribute based condition.