CS145 Midterm Examination

Autumn 2010, Prof. Widom

- Please read all instructions (including these) carefully.
- There are 9 problems on the exam, with a varying number of points for each problem and subproblem for a total of 75 points to be completed in 75 minutes. *You should look through the entire exam before getting started, in order to plan your strategy.*
- The exam is closed book and closed notes, but you may refer to your three pages of prepared notes.
- Please write your solutions in the spaces provided on the exam. Make sure your solutions are neat and clearly marked. You may use the blank areas and backs of the exam pages for ungraded scratch work.
- *Simplicity and clarity of solutions will count.* You may get as few as 0 points for a problem if your solution is far more complicated than necessary, or if we cannot understand your solution.
- Throughout the exam you should assume and use "pure" SQL, XPath, XQuery, and XSLT as covered in class—not dialects of these languages supported by a particular implementation (such as SQLite or Saxon).

NAME ____

In accordance with both the letter and spirit of the Honor Code, I have neither given nor received assistance on this examination.

SIGNATURE: ____

Problem	1	2	3	4	5	6	7	8	9	TOTAL
Max. points	10	6	4	9	6	21	5	4	10	75
Points										

1. **Relational Algebra** (10 points, 1 per answer)

   Consider a relation $R(A, B)$ that contains r tuples and a relation $S(B, C)$ that contains s tuples; assume $r > 0$ and $s > 0$. Make no assumptions about keys. For each of the following relational algebra expressions, state in terms of r and s the minimum and maximum number of tuples that could be in the result of the expression.

Expression	minimum #tuples	maximum #tuples
$R \cup \rho_{S(A,B)} S$		
$\Pi_{A,C}(R \bowtie S)$		
$\Pi_B(R) - (\Pi_B(R) - \Pi_B(S))$		
$(R \bowtie R) \bowtie R$		
$\sigma_{A>B} R \cup \sigma_{A<B} R$		

2. **SQL** (6 points)

Consider two SQL tables `T1(A,B)` and `T2(C)`. You may assume there are no null values in either table. Make no assumptions about keys. Consider the following three SQL queries:

```
Q1: Select A
    From T1
    Where B In (Select C From T2)

Q2: Select A
    From T1, T2
    Where T1.B = T2.C

Q3: Select Distinct A
    From T1, T2
    Where T1.B = T2.C
```

Circle one of the following statements. (Note: two queries are equivalent if they are guaranteed to return the same answer on all possible databases.)

(a) `Q1`, `Q2`, and `Q3` are all equivalent.

(b) `Q1` and `Q2` are equivalent; `Q3` may produce a different answer on some databases.

(c) `Q1` and `Q3` are equivalent; `Q2` may produce a different answer on some databases.

(d) `Q2` and `Q3` are equivalent; `Q1` may produce a different answer on some databases.

(e) `Q1`, `Q2`, and `Q3` may all produce different answers on some databases.

If you chose any answer other than (a), show the *smallest* instances of `T1` and `T2` that demonstrate the nonequivalence(s). In addition to showing the tables, show the result of all three queries on your tables.

3. **More SQL** (4 points)

Consider a SQL table `T(A)`. You may assume there are no null values in `T`. Make no assumptions about keys. Consider the following two SQL queries:

```
Q1: Select A
    From T
    Where Not A >= Any (Select A From T)

Q2: Select A
    From T
    Where A < (Select Max(A) From T)
```

Circle one of the following statements. (Note: two queries are equivalent if they are guaranteed to return the same answer on all possible databases.)

(a) `Q1` and `Q2` are equivalent.

(b) `Q1` and `Q2` may produce a different answer on some databases.

If you chose (b), show the *smallest* instance of `T` that demonstrates the nonequivalence. In addition to showing the table, show the result of both queries on your table.

4. **More Relational Algebra and SQL** (9 points, 3 for each part)

For all three parts of this problem consider a relation $R(A, B, C)$. You may assume there are no null values or duplicates in R. In each part, state *in a few words or less* a property of R that is guaranteed to hold if the result of the given query is always empty. In each case there is a very concise correct answer corresponding to a property covered in class.

(a) $\sigma_{Y \neq V}(\rho_{R(X,Y,Z)} R \bowtie \rho_{R(X,V,W)} R)$

If the result is guaranteed to be empty, then:

(b)
```
Select A From R
Group By A
Having Count(*) > 1
```

If the result is guaranteed to be empty, then:

(c)
```
Select Distinct R1.A, R1.B, R2.C
From R R1, R R2
Where R1.A = R2.A
Except
Select Distinct * From R
```

If the result is guaranteed to be empty, then:

5. **DTDs and XML Schema** (6 points, 3 for each part)

Consider the following DTD and XML Schema Declarations:

```
<!DOCTYPE People [
   <!ELEMENT People (Prof?, Student?)>
   <!ELEMENT Prof (Name)>
   <!ATTLIST Prof pID ID #REQUIRED>
   <!ELEMENT Student (Name)>
   <!ATTLIST Student sID ID #REQUIRED>
   <!ELEMENT Name (#PCDATA)>
]>
```

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="People">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Prof" type="ProfType" />
        <xsd:element name="Student" type="StudType" />
      </xsd:sequence>
    </xsd:complexType>
    <xsd:key name="ProfKey">
      <xsd:selector xpath="Prof" />
      <xsd:field xpath="@pID" />
    </xsd:key>
    <xsd:key name="StudentKey">
      <xsd:selector xpath="Student" />
      <xsd:field xpath="@sID" />
    </xsd:key>
  </xsd:element>
  <xsd:complexType name="ProfType">
    <xsd:sequence>
      <xsd:element name="Name" type="xsd:string" />
    </xsd:sequence>
    <xsd:attribute name="pID" type="xsd:string" use="required" />
  </xsd:complexType>
  <xsd:complexType name="StudType">
    <xsd:sequence>
      <xsd:element name="Name" type="xsd:string" />
    </xsd:sequence>
    <xsd:attribute name="sID" type="xsd:string" use="required" />
  </xsd:complexType>
</xsd:schema>
```

(problem continues on next page...)

Here is a sample XML document that conforms to both specifications:

```
<People>
  <Prof pID="p1"> <Name>Jennifer</Name> </Prof>
  <Student sID="s1"> <Name>Izaak</Name> </Student>
</People>
```

You should not need to examine the XML Schema declaration in great detail to find concise and correct answers for the following two problems.

(a) Give the smallest XML document you can come up with that *does* conform to the DTD but *does not* conform to the XML Schema specification. Your solution will be graded on conciseness as well as correctness.

(b) Give the smallest XML document you can come up with that *does* conform to the XML Schema declaration but *does not* conform to the DTD. Your solution will be graded on conciseness as well as correctness.

6. **XPath, XQuery, and XSLT** (21 points, 2 each for parts (a)–(c), 3 each for the rest)

Consider the following XML document called "Students.xml" containing student information. If it makes a difference, you may assume attribute `sID` has type `ID` and attribute `Partner` has type `IDREFS`.

```
<Students>
  <Student sID="123" Partner="789">
    <Name>Ann</Name>
    <Major>Mechanical Engineering</Major>
    <Major>Electrical Engineering</Major>
  </Student>
  <Student sID="456" Partner="123 789">
    <Name>Bob</Name>
  </Student>
  <Student sID="789" Partner="123">
    <Name>Carl</Name>
    <Major>Mechanical Engineering</Major>
    <Major>English</Major>
  </Student>
</Students>
```

For each of the following queries/specifications in XPath, XQuery, or XSLT, show the result of execution on the XML document above. Don't worry about headers, whitespace, or fine details of result formatting; we are most interested in correct (and correctly structured) results.

(a) `doc("Students.xml")//Student[Major]/Name`

(problem continues on next page...)

(b)
```
doc("Students.xml")//Student[Major!="English"]/data(@Partner)
```

(c)
```
doc("Students.xml")//Student[@Partner=doc("Students.xml")//
                       Student[Major="English"]/@sID]/Name
```

(d)
```
for $s1 in doc("Students.xml")//Student
for $s2 in doc("Students.xml")//Student
where $s1/Major = $s2/Major
return $s1/Name
```

(problem continues on next page...)

(e)
```
for $x in doc("Students.xml")//Student/*
where some $y in $x/preceding-sibling::*
  satisfies contains($y,"Engineering")
order by $x/text() descending
return $x/parent::*/*[1]
```

(f)
```
for $s in doc("Students.xml")//Student
for $m in $s/Major
where contains($m,"Engineering") or contains($m,"English")
return <Eng> {$s/@sID} </Eng>
```

(problem continues on next page...)

(g)
```
<xsl:stylesheet ...>
<xsl:output ... />

<xsl:template match="Major">
</xsl:template>

</xsl:stylesheet>
```

(h)
```
<xsl:stylesheet ...>
<xsl:output ... />

<xsl:template match="Student">
  <S> <xsl:apply-templates select="@*|node()" /> </S>
</xsl:template>

<xsl:template match="@sID">
  <ID> <xsl:value-of select="." /> </ID>
</xsl:template>

<xsl:template match="@Partner" />
<xsl:template match="Name" />
<xsl:template match="Major" />

</xsl:stylesheet>
```

7. **Functional Dependencies and Keys** (5 points)

The following functional dependencies hold in a relation $R(A, B, C, D, E, F)$:

$AB \rightarrow C$
$DE \rightarrow F$
$F \rightarrow B$
$C \rightarrow E$

List all the minimal keys for R. Make sure to clearly denote when a set of attributes together constitute a key (as opposed to each attribute being a key on its own).

8. **Normal Forms** (4 points, 2 per part)

   Consider any relation R that never contains more than one tuple.

   (a) Is R in Boyce-Codd Normal Form (BCNF)? Circle one:

   - Yes
   - No
   - Can't tell without seeing the actual schema and FDs
   - Can't tell without seeing the actual data
   - Can't tell without seeing the actual schema, FDs, and data

   (b) Is R in Fourth Normal Form (4NF)? Circle one:

   - Yes
   - No
   - Can't tell without seeing the actual schema, FDs, and MVDs
   - Can't tell without seeing the actual data
   - Can't tell without seeing the actual schema, FDs, MVDs, and data

9. **BCNF Decomposition** (10 points, 2 for each part)

Here is the BCNF decomposition algorithm as presented in class:

```
Input: relation schema R and set of FD’s for R
(1) compute keys for R based on FD’s
(2) repeat until no more BCNF violations:
     (2a) pick any R’ with AA -> BB that violates BCNF
     (2b) decompose R’ into R1(AA,BB) and R2(AA,CC)
          where CC is all attributes in R’ except (AA U BB)
     (2c) compute FD’s for R1 and R2
     (2d) compute keys for R1 and R2 based on FD’s
```

For all problems parts, consider a relation $R(A, B, C, D)$ with the functional dependences $F = \{A \rightarrow B, A \rightarrow C\}$.

(a) Using the algorithm above, is it possible to get two different BCNF decompositions for R and F, depending which relation and violating FD is chosen in step (2a) at some point in the process? If so, write “two decompositions” and show the two different BCNF decompositions. If it is not possible to get two different decompositions, write “unique decomposition” and show it. Do not show the step-by-step process, just the final decomposition(s).

(b) If we omit step (2c), is it possible to get a non-BCNF decomposition for R and F? If so, show the non-BCNF decomposition. (Do not show the step-by-step process, just the final decomposition.) If it is not possible to get a non-BCNF decomposition, write “not possible.”

(c) Suppose we use the original algorithm but add the following step before step (1):

```
(0) add to F all FD's implied by F
```

Do we get different keys in step (1) from those we got without step (0)? If so, write "different keys" and show the two different sets of keys, clearly marking those gotten with step (0) and those gotten without step (0). If the keys are the same, write "same keys" and show the set of keys.

(d) Continue to suppose we add step (0) as in part (c). With step (0) is it possible to get a non-BCNF decomposition for R and F? If so, show the non-BCNF decomposition. (Do not show the step-by-step process, just the final decomposition.) If it is not possible to get a non-BCNF decomposition, write "not possible."

(e) Continue to suppose we add step (0) as in part (c). With step (0) is it possible to get a BCNF decomposition for R and F that we cannot get without step (0)? If so, show the new BCNF decomposition. (Do not show the step-by-step process, just the final decomposition.) If it is not possible to get a new BCNF decomposition, write "not possible."