# CS145 Midterm Examination

## Spring 2002, 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. *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 scratch work. Please do not use any additional scratch paper.
- *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.

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 | 9 | 6 | 8 | 10 | 12 | 6 | 8 | 75 |
| Points | | | | | | | | | | |

1. **Relational Algebra** (10 points)

   Consider the following relational schema:

```
Student(name, dorm)     // name is a key
Partners(name1, name2)  // name1 is a key, name2 is a key
```

   Write a relational algebra expression to find all pairs of students (by name) who are partners but do not live in the same dorm. You may assume that each partner pair is listed only once, i.e., if $(A, B)$ is in `Partners` then $(B, A)$ is not in `Partners`.

2. **Relational Algebra and SQL** (6 points)

   Consider the following relational schema:

```
Customer(ID, name, phone)           // ID is a key, <name,phone> is a key
Account(Num, custID, type, balance) // Num is the only key
```

   Suppose we wish to find the names and phone numbers of customers who have a checking or savings account with a negative balance. Each of the following five queries attempts to achieve this goal.

   (a) $\Pi_{\text{name, phone}}($
   Customer $\bowtie_{\text{ID = custID}} (\sigma_{\text{balance} < 0 \wedge (\text{type = 'checking'} \vee \text{type = 'savings'})}(\text{Account}))$

   (b)
```
select name, phone
from   Customer, Account
where  ID=custID and balance < 0
       and (type='checking' or type='savings')
```

   (c)
```
select name, phone
from   Customer
where  ID in
       (select custID from Account
        where balance < 0 and (type='checking' or type='savings'))
```

(d)
```
(select name, phone
 from Customer, Account
 where ID=custID and balance < 0 and type='checking')
union
(select name, phone
 from Customer, Account
 where ID=custID and balance < 0 and type='savings')
```

(e)
```
(select name, phone
 from Customer, Account
 where ID=custID and balance < 0 and type='checking')
union all
(select name, phone
 from Customer, Account
 where ID=custID and balance < 0 and type='savings')
```

Two queries are *equivalent* if they are guaranteed to produce the same result over every database conforming to the schema. Circle exactly one of the following five options, and if appropriate fill in the blank:

(i) All five queries are equivalent.

(ii) Four of the queries are equivalent to each other and one is not equivalent to those four. The query that is not equivalent is:

(iii) Three of the queries are equivalent to each other, and two of the queries are not equivalent to those three but are equivalent to each other. The three equivalent queries are:

(iv) Three of the queries are equivalent to each other, and two of the queries are not equivalent to those three and are not equivalent to each other either. The three equivalent queries are:

(v) None of the five queries are equivalent to each other.

3. **SQL** (9 points)

Consider the following relational schema:

```
Student(ID, GPA)   // ID is a key
Plays(ID, sport)   // <ID,sport> is the only key
```

and the following query over this schema:

```
select sport
from Student, Plays
where Student.ID=Plays.ID
group by sport
having min(GPA) < 2.0
```

Write a SQL query that is equivalent to the one above but does not use a HAVING clause. Write the simplest query you can think of.

(Scratch space)

4. **Programming with SQL** (6 points)

Consider a relation $R(A, B)$ that may contain duplicate tuples, and suppose we are interested in writing an operation that deletes exactly one copy of each tuple appearing in $R$. For example, if initially $R = \{(1,2), (1,2), (3,4), (5,6), (5,6), (5,6)\}$, then after our deletion operation we want $R = \{(1,2), (5,6), (5,6)\}$.

(a) Is it possible to perform this deletion operation using a single SQL DELETE statement?

Circle one: YES NO

(b) Is it possible to perform this deletion operation using SQL in conjunction with a programming language (i.e., using some form of embedded SQL, dynamic SQL, or DBMS programming language such as PSM)?

Circle one: YES NO

5. **XPath and XQuery** (8 points)

Consider querying XML documents that conform to the following DTD:

```
<!ELEMENT Bookstore (Book*)>
<!ELEMENT Book (Title, Authors)>
<!ATTLIST Book ISBN CDATA #REQUIRED Price CDATA #REQUIRED>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Authors (Author+)>
<!ELEMENT Author (First_Name, Last_Name)>
<!ELEMENT First_Name (#PCDATA)>
<!ELEMENT Last_Name (#PCDATA)>
```

For each of the following query pairs, circle YES if the XPath and XQuery expressions are equivalent (i.e., they are guaranteed return the same result over any XML document conforming to the above DTD), and circle NO if they are not equivalent (i.e., there is some document conforming to the DTD for which they will return different results). *For equivalence don't worry about details of answer presentation (such as* <result> *elements), just consider the content of the query results.*

(a)
```
XPath:
/Bookstore/Book[Authors/Author[Last_Name='Widom']]/Title

XQuery:
for $b in /Bookstore/Book
where every $n in $b/Authors/Author/Last_Name satisfies $n='Widom'
return $b/Title
```

Queries are equivalent? YES NO

(b)
```
XPath:
//*[@ISBN='1234']//Author

XQuery:
for $b in /Bookstore/Book
where $b/@ISBN='1234'
return $b/Authors/Author
```

Queries are equivalent? YES NO

(c)
```
XPath:
/Bookstore/Book
   [Authors/Author/First_Name=Authors/Author/First_Name
    and Authors/Author/Last_Name <> Authors/Author/Last_Name]/Title

XQuery:
for $b in /Bookstore/Book
for $a1 in $b/Authors/Author
for $a2 in $b/Authors/Author
where $a1/First_Name=$a2/First_Name
      and $a1/Last_Name <> $a2/Last_Name
return $b/Title
```

Queries are equivalent? YES NO

(d)
```
XPath:
/Bookstore/Book[@Price<'100'][3]/Title

XQuery:
for $b in /Bookstore/Book[3][@Price<'100']
return $b/Title
```

Queries are equivalent? YES NO

6. **More XPath and XQuery** (10 points)

Consider the following XML document containing student information. Assume that attribute "sID" has type ID and attribute "Partner" has type IDREF.

```
<Students>
  <Student sID="1234" Partner="5678">
    <Name>Ann</Name>
  </Student>
  <Student sID="4321" Partner="1234">
    <Name>Michael</Name>
    <Major>Chemical Engineering</Major>
    <Major>English</Major>
  </Student>
  <Student sID="5678" Partner="1234">
    <Name>Lilly</Name>
    <Major>Mechanical Engineering</Major>
    <Major>Electrical Engineering</Major>
  </Student>
</Students>
```

For each of the following XPath or XQuery expressions, write the result of the query on the XML document above. (We are more interested in correct query results than details of result formatting.)

(a) `/Students/Student[Major]/Name`

(b)
```
for $s in /Students/Student
where contains($s/Major, 'Engineering')
return <sID> $s/@sID </sID>
```

(c)
```
for $s1 in /Students/Student
for $s2 in /Students/Student
where $s1/@Partner=$s2/@sID and $s2/@Partner <> $s1/@sID
return $s1/Name
```

(d)
```
for $s in /Students/Student
return <NumMajors> count($s/Major) </NumMajors>
```

(e)
```
let $m := /Students/Student/Major
return <NumMajors> count($m) </NumMajors>
```

---

(Scratch space)

7. **English to Dependencies** (12 points)

Consider the following relational schema:

```
Car(make, model, year, color, dealer)
```

Each tuple in relation `Car` specifies that one or more cars of a particular make, model, and year in a particular color are available at a particular dealer. For example, the tuple

```
(Honda, Odyssey, 1998, Blue, Fred's Friendly Folks)
```

indicates that 1998 Honda Odysseys in blue are available at the Fred's Friendly Folks car dealer.

For each of the following English statements, write one nontrivial functional or multivalued dependency that best captures the statement. Consider each of parts (a)–(c) independently, i.e., except for part (d) do not assume the FD or MVD for one part of the problem holds when solving a different part.

(a) The model name for a car is trademarked by its make, i.e., no two makes can use the same model name.

(b) Each dealer sells only one model of each make of car.

(c) If a particular make, model, and year of a car is available in a particular color at a particular dealer, then that color is available at all dealers carrying the same make, model, and year.

Now suppose all three functional and/or multivalued dependencies you specified in parts (a), (b), and (c) hold, as do all the FD's and MVD's implied by them, but no other FD's or MVD's hold.

(d) Specify all keys for relation `Car`. (You may assume that `Car` cannot contain duplicate tuples.)

8. **Dependency Set Equivalence** (6 points)

Two sets of functional dependencies (FD's) $F$ and $F'$ are equivalent if all FD's in $F'$ follow from the ones in $F$, and all FD's in $F$ follow from the ones in $F'$. Consider the following three sets of functional dependencies over a relation $R(A, B, C, D)$:

$F_1 = \{A \to B, B \to C\}$
$F_2 = \{A \to B, A \to C\}$
$F_3 = \{A \to B, AB \to C\}$

(a) Are sets $F_1$ and $F_2$ equivalent? Circle one: YES NO

(b) Are sets $F_2$ and $F_3$ equivalent? Circle one: YES NO

(c) Are sets $F_1$ and $F_3$ equivalent? Circle one: YES NO

9. **More on Dependencies** (8 points)

Suppose we have three relations: `R(rID,...), S(sID,...), and T(rID,sID)`. The role of relation $T$ is to record relationships between elements of $R$ and elements of $S$ by listing a key of the $R$ element (`rID`) together with a key of the related $S$ element (`sID`).

(a) If $T$ encodes a relationship that is one-one between the elements of $R$ and the elements of $S$, state all nontrivial functional dependencies we know must hold on relation $T$.

(b) If $T$ encodes a relationship that is many-many between the elements of $R$ and the elements of $S$, state all nontrivial functional dependencies we know must hold on relation $T$.

(c) If $T$ encodes a relationship that is many-one from the elements of $R$ to the elements of $S$, state all nontrivial functional dependencies we know must hold on relation $T$.

(d) Among parts (a), (b), and (c), is there any case where relation $T$ is not in Boyce-Codd Normal Form (BCNF) or not in Fourth Normal Form (4NF)? If so, state which of (a), (b), and (c) violates a normal form, and which normal form is violated.