CMPT-354 D1 Fall 2008
Instructor: Martin Ester
TA: Gustavo Frigo

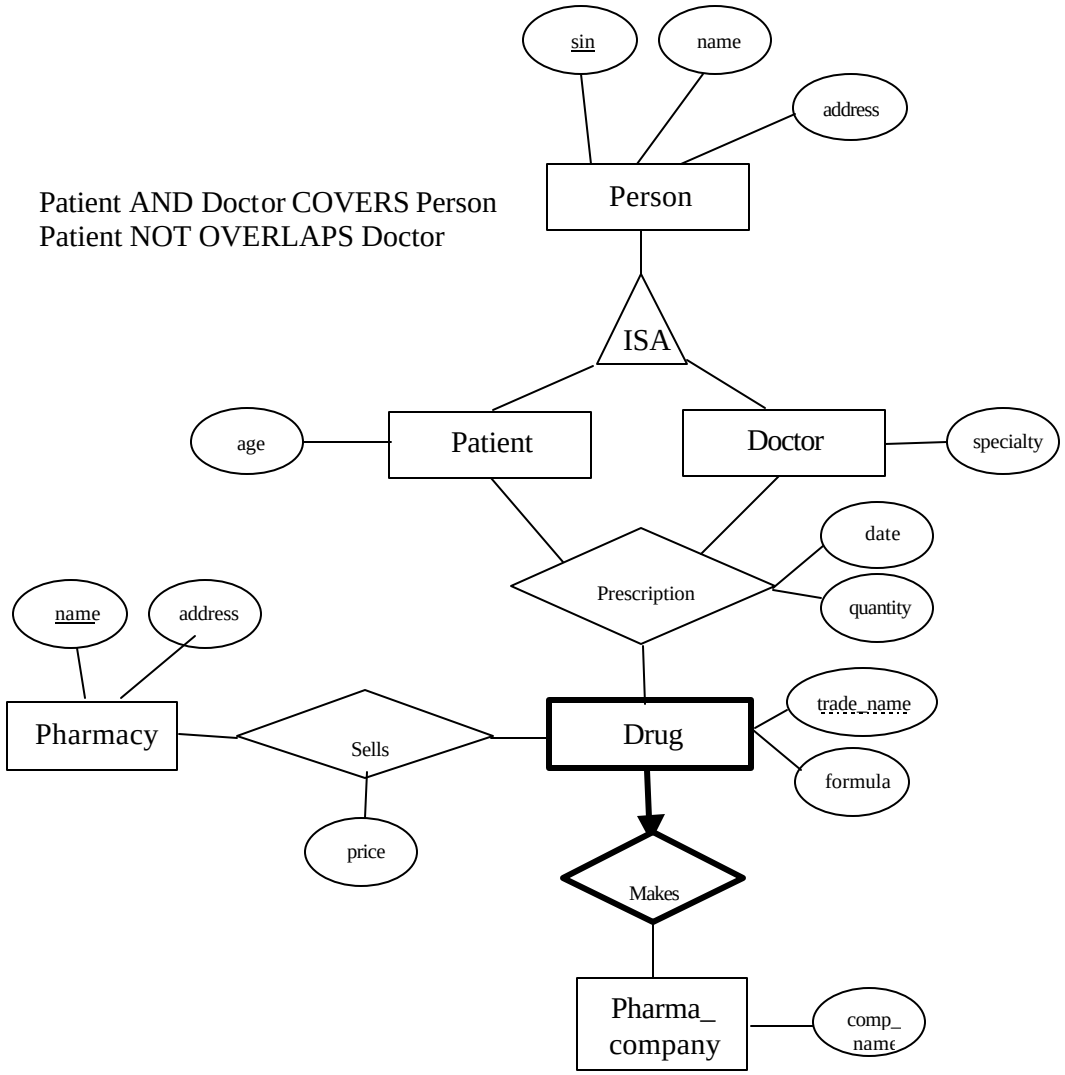
Sample Final Exam

Introduction

The first part of the final exam will cover the same topics as the midterm exam. This sample final exam therefore covers only the second part of the final exam that goes beyond the topics of the midterm exam.

Problem 1

Translate the following ER diagram of a pharma database into an equivalent relational schema.



Your relational schema should satisfy the following two design criteria:

- The number of tables should be minimal.
- As many integrity constraints from the ER diagram as possible should be expressed.

a) Write down the complete SQL statements to create the relational schema, including PRIMARY KEY, FOREIGN KEY and NOT NULL constraints. For the FOREIGN KEY constraints, you do not need to specify the reactions to violating updates.

```
CREATE  TABLE Patient (  sin INTEGER,
                         name CHAR (20),
                         address CHAR(20),
                         age INTEGER,
                         PRIMARY KEY (sin));

CREATE  TABLE Doctor (   sin INTEGER,
                         name CHAR (20),
                         address CHAR(20),
                         specialty CHAR(20),
                         PRIMARY KEY (sin));

CREATE  TABLE Prescription (     patient_sin INTEGER,
                                 doctor_sin INTEGER,
                                 drug_company CHAR(20),
                                 drug_name CHAR(20),
                                 date DATE,
                                 quantity INTEGER,
                         PRIMARY KEY (patient_sin, doctor_sin, drug_company, drug_name),
                         FOREIGN KEY (patient_sin) REFERENCES Patient,
                         FOREIGN KEY (doctor_sin) REFERENCES Doctor,
                         FOREIGN KEY (drug_company, drug_name) REFERENCES Drug);

CREATE  TABLE Drug (     comp_name CHAR(20),
                         trade_name CHAR(20),
                         formula CHAR(20),
                         PRIMARY KEY (comp_name, trade_name),
                         FOREIGN KEY (comp_name) REFERENCES Pharm_company);

CREATE  TABLE Pharm_company (comp_name CHAR(20),
                         PRIMARY KEY (comp_name);

CREATE  TABLE Pharmacy (name CHAR(20),
                         Address CHAR(20),
                         PRIMARY KEY (name));

CREATE  TABLE Sells (    pharmacy_name CHAR(20),
                         drug_company CHAR(20),
                         drug_name CHAR(20)
                         price REAL,
                         PRIMARY KEY (pharmacy_name, drug_company, drug_name),
                         FOREIGN KEY (pharmacy_name) REFERENCES Pharmacy,
                         FOREIGN KEY (drug_company, drug_name) REFERENCES Drug);
```

b) List the integrity constraints of the ER diagram that are not expressed in your relational schema.

The NOT OVERLAPS constraint on Patient and Doctor cannot be expressed in the relational schema.

Problem 2

Consider a relation R with five attributes ABCDE. For each of the following instances of R, state whether it violates the functional dependency (FD) BC→D. If an instance violates the FD, state two tuples which violate it. In the given instances, digits denote actual attribute values and "a" denotes a variable representing any attribute value. If an instance violates the FD only for certain values of a, state these values.

a) {(a,2,3,4,5), (2,a,3,5,5)}

If a = 2, then BC→ D is violated by the first and the second tuple, otherwise it is not violated.

b) {(a,2,3,4,5), (2,a,3,4,5), (a,2,3,6,5)}

For any value of a, the FD is violated by the first and the third tuple.

c) {(a,2,3,4,5), (a,2,3,6,5), (a,2,3,6,6), (2,a,3,4,6)}

For any value of a, the FD is violated by the first and the third tuple.

d) If each of the instances of R listed above is legal, what can we say about the FD A→ B: it holds, it does not hold or we don't know whether it holds?

No pair of two tuples violates A→B, i.e. we cannot conclude that the FD does not hold. However, we can never infer that a FD holds for all instances after having seen only some instances. Therefore, we do not know whether A→ holds.

Problem 3

Consider the following XML schema.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:simpleType name="stringtype">
                <xs:restriction base="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="inttype">
                <xs:restriction base="xs:positiveInteger"/>
        </xs:simpleType>
        <xs:simpleType name="dectype">
                <xs:restriction base="xs:decimal"/>
        </xs:simpleType>
        <xs:simpleType name="orderidtype">
                <xs:restriction base="xs:string">
                        <xs:pattern value="[0-9]{6}"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:complexType name="shiptotype">
```

```
                <xs:sequence>
                        <xs:element name="name" type="stringtype"/>
                        <xs:element name="address" type="stringtype"/>
                        <xs:element name="city" type="stringtype"/>
                        <xs:element name="country" type="stringtype"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="itemtype">
                <xs:sequence>
                        <xs:element name="title" type="stringtype"/>
                        <xs:element name="note" type="stringtype" minOccurs="0"/>
                        <xs:element name="quantity" type="inttype"/>
                        <xs:element name="price" type="dectype"/>
                </xs:sequence>
        </xs:complexType>

        <xs:complexType name="shipordertype">
                <xs:sequence>
                        <xs:element name="orderperson" type="stringtype"/>
                        <xs:element name="shipto" type="shiptotype"/>
                        <xs:element name="item" maxOccurs="unbounded"
                             type="itemtype"/>
                </xs:sequence>
                <xs:attribute name="orderid" type="orderidtype" use="required"/>
        </xs:complexType>
        <xs:element name="shiporderDB">
            <xs:complexType>
                <xs:sequence>
                        <xs:element name="shiporder" maxOccurs="unbounded"
                                    type="shipordertype"/>
                </xs:sequence>
             </xs:complexType>
        </xs:element>
</xs:schema>
```

Create a valid XML document with a "shiporderDB" root element and two "shiporder" elements.

We assume that the XML schema is stored in the file “shiporder.xsd” in the same directory as the XML document.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<shiporderDB
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="shiporder.xsd">
    <shiporder orderid="889923">
        <orderperson>John Smith</orderperson>
        <shipto>
                <name>Paul Larson</name>
                <address>1234 100th St.</address>
                <city>Surrey, BC</city>
                <country>Canada</country>
        </shipto>
        <item>
                <title>Database Systems: The Complete Book<title>
                <note>Second Edition</note>
                <quantity>1</quantity>
                <price>49.90</price>
        </item>
        <item>
                <title>Data Mining: Concepts and Techniques</title>
                <quantity>1</quantity>
```

```
            <price>59.90</price>
      </item>
   </shiporder>
   <shiporder orderid="712456">
      <orderperson>John Smith</orderperson>
      <shipto>
            <name>Christine Swartz</name>
            <address>777 Hastings St</address>
            <city>Vancouver, BC</city>
            <country>Canada</country>
      </shipto>
      <item>
            <title>The Kyte Runner</title>
            <note>Christmas sale</note>
            <quantity>1</quantity>
            <price>10.50</price>
      </item>
      <item>
            <title>Hide your heart</title>
            <quantity>1</quantity>
            <price>9.90</price>
      </item>
   </shiporder>
</shiporderDB>
```

Problem 4

Consider an XML document "ShiporderDB.xml" that conforms to the above XML schema. Formulate the following queries on this XML document in XQuery. Where applicable, eliminate duplicate results.

a) Find the items ordered in "Vancouver" with maximum price among these items, as a sequence of item elements.

```
let $items := doc("ShiporderDB.xml")/shiporderDB/shiporder
      [contains(shipto/city,'Vancouver')]/item
    $max := max($items/price)
for $i in doc("ShiporderDB.xml")/shiporderDB/shiporder
      [contains(shipto/city,'Vancouver')]/item
where $i/price = $max
return
      $i
```

b) For all shiporders that contain an item that has "computer" in its title, produce a result element consisting of the orderid and the number of items.

```
for $s in doc("ShiporderDB.xml")/shiporderDB/shiporder
where some $i in $s/item satisfies contains($i/title,"computer")
return <result>
            {$s/@orderid}
            {count($s/item)}
        </result>
```

c) For all orderpersons who have shipped an order to the US, find the orderperson and the sequence of all items shipped. Include all information belonging to the same orderperson into one result element.

```
for $o in (distinct-values(doc("ShiporderDB.xml")
      /shiporderDB/shiporder[shipto/country = "US"]/orderperson)
return <result>
            {$o}
            {for $s in (doc("ShiporderDB.xml")/shiporderDB/shiporder/
             where $o/parent::shiporder = $s
             return $s/item}
      </result>
```

d) Find all shiporders that contain an item that both has "Data" in the title and has a price of > $50.

```
for $i1 in doc("ShiporderDB.xml")/shiporderDB/shiporder/item
            [contains(title,'Data')]
    $i2 in doc("ShiporderDB.xml")/shiporderDB/shiporder/item[price > 50.0]
where $i1 = $i2
return $i1/parent::shiporder
```