Name: KEY

CS 589 Principles of Database Systems Winter 2011 Test 1

You can use the course text, course notes, assignments and solutions, but not other books or sources. Write your answers in the spaces provided. (I have extra blank pages if needed.) 75 points possible.

1. Query Languages

Out of the following 4 queries, 2 are equivalent. Here Needs(PName, Course) means the named person needs to take the course, and Offered(Course, Quarter) means the course is offered in the given quarter.

Q1 $\pi_{PName}(Needs) - \pi_{PName}(Needs \bowtie Offered)$

Q2 $\{p{:}PName \mid \exists y{:}Course\ (Needs(p, y) \wedge \forall q{:}Quarter\ (\neg Offered(y, q)))\}$

Q3 $\{N \mid N \in Needs \wedge \forall R \in Offered\ (\neg(N.Course = R.Course))\}$

Q4 Answer(p) :- Needs(p, y), ¬Avail(y).
Avail(c) :- Offered(c, q).

a. (5 points) Which two queries are equivalent?

Q2 & Q4 ; Q1 omits anyone with an available class, Q3 has a different schema

b. (9 points) Give example database states to show that the other two queries are not equivalent to each other, nor to the pair of equivalent queries. (Note that you might need up to 3 database states.)

Since Q3 has a different schema, pretty much any state works

Needs(PName, Course)	
Joe	Math 111

Offered(Course, Quarter)	
CS 201	Spring 11

to show Q3 $\not\equiv$ Q1, Q2, Q4

To show Q1 $\not\equiv$ Q2, consider

Needs(PName, Course)	
Joe	Math 111
Joe	CS201

Offered(Course, Quarter)	
CS201	Spring 11

Q1 gives an empty answer
Q2 gives <Joe>

c. (6 points) Modify each of the non-equivalent queries to make it equivalent to the pair of equivalent queries.

Q1 $\pi_{PName}(Needs - \pi_{PName\ Course}(Needs \bowtie Offered))$

Q3 $\{T \mid \exists N \in Needs \wedge T.PName = N.PName \wedge$
$\forall R \in Offered\ (\neg(N.Course = R.Course))\}$

2. Satisfaction of Dependencies (10 points)

Consider a relation r on schema ABCD that satisfies dependencies M = {B → A, AD → C, ⋈[ABC, CD]}. Explain why the set of tuples below *cannot* be part of r.

r(A B C D)

	A	B	C	D
	…			
t1	1	2	3	4
t2	1	8	9	5
t3	6	7	3	5
t4	10	11	12	4
	…			

By the JD, r will also have

t5 1 2 3 5

from t1[ABC] and t3[CD]

But then t2 + t5 violate AD → C

3. Dependency Checking

It is possible to use a query to check if a dependency is satisfied. For the answers below, you can use any query language from class, or SQL. (Note that an empty answer to a query will mean a dependency is satisfied.)

a. (6 points) Consider A → B on relation r(A B C). Write a query that returns a tuple <*a b1 b2*> for every A-value *a* that appears with different B-values *b1* and *b2*.

In domain calculus:

$$\{x\ y\ z \mid \exists c1\ \exists c2\ r(x\ y\ c1) \wedge r(x\ z\ c2) \wedge \neg(y = z)\}$$

b. (6 points) Consider ⋈[AB, BC, ACD] on relation s(A B C D). Write a query that produces the minimum set of tuples <*a b c d*> that must be added to s to make it satisfy the JD.

In relational algebra

$$\left(\pi_{AB}(s) \bowtie \pi_{BC}(s) \bowtie \pi_{ACD}(s)\right) - s$$

4. Implication of Dependencies

Let M be a set of FDs and JDs over schema R. Define the *completion* of a set of attributes X to be the maximum set of attributes X* such that M implies $X \to X^*$. (So X* is like X^+, except it also takes JDs into account.)

a. (10 points) Explain how to calculate X* from X and M.

Start with a tableau U with 2 rows w1 and w2 where w1[X] = w2[X] but distinct variables everywhere else.

Compute Chase(U, M)

Let X* be all the attributes where w1 and w2 agree in the chase.

b. (6 points) Demonstrate your procedure by calculating AB* for R = ABCDE and M = $\{B \to D, C \to \overset{E}{\cancel{D}}, \bowtie[ABC, BCD, DE]\}$.

U(A B C D E)

	A	B	C	D	E
w1	a	b	c	d	e
w2	a	b	c1	d1	e1

start.

U(A B C D E)

	A	B	C	D	E
w1	a	b	c	d	e
w2	a	b	c1	~~d1~~ d	~~e1~~ e
	a	b	c	d	~~e1~~ e
	a	b	c1	d	e

① d = d1 by B → D

② by JD

③ e = e1 by C → E

So AB* = ABDE

c. (2 points) What can you say about X* if M contains only FDs?

$X^* = X^+$

d. (2 points) What can you say about X* if M contains only JDs?

$X^* = X$ — no way to equate additional variables.

5. Null Values

An *existence constraint* (EC) limits where nulls can appear in tuples in a relation.
An EC has the form X ↓ Y (read as "X requires Y") where X and Y are sets of attributes.
A tuple t satisfies X ↓ Y if when t[X] has no nulls, then t[Y] has no nulls.
A relation r satisfies X ↓ Y if every tuple t in r satisfies it.

a. (4 points) Consider the relation contact-info below on the schema
{First, Middle, Last, AreaCode, Phone, HomeOrOffice}, where ∞ represents a null value.

contact-info(	F	M	L	A	P	H)
t1	Ray	∞	Ng	∞	∞	∞
t2	∞	∞	∞	∞	∞	∞
t3	Kay	Marie	Weil	∞	∞	Home
t4	∞	∞	Smith	∞	∞	∞
t5	∞	Ron	Doaks	213	887-8706	Home
t6	Mary	∞	Marty	∞	736-1117	Office
t7	Don	Jay	Allen	206	∞	Office

For each EC below, say whether contact-info satisfies it. If it doesn't, list one violating tuple.

i. M ↓ F   t5 violates: t5[M] = Ron, t5[F] = ∞

ii. P ↓ A   t6 violates: t6[P] = 736-1117 t6[A] = ∞

iii. AP ↓ H   OK. t5 is the only tuple defined on both A + P, and it is also defined on H

iv. ∅ ↓ L   t2 violates: no nulls in t2[∅], but t2[L] = ∞

b. (6 points) Say whether each of the inference axioms for ECs below is sound or not. If not, give a counterexample.

reflexivity: X ↓ X

OK

symmetry: X ↓ Y implies Y ↓ X

Consider ~~above~~ t1 above. Satisfies M ↓ F, but not F ↓ M

transitivity: X ↓ Y and Y ↓ Z imply X ↓ Z

OK

c. (3 points) Give one additional sound inference axiom for ECs.

Surprisingly, any axiom for FDs works for ECs.
For example, Additivity:
X ↓ Y and X ↓ Z imply X ↓ YZ.