# CSE-3421 Test #2
“Queries”

| | |
|---|---|
| **Family Name:** | ______________________ |
| **Given Name:** | ______________________ |
| **Student#:** | ______________________ |
| **CS&E Account:** | ______________________ |

- **Instructor:** Parke Godfrey
- **Exam Duration:** 75 minutes
- **Term:** winter 2009

Answer the following questions to the best of your knowledge. The exam is closed-book and closed-notes. Questions will not be answered during the test. Should you feel a question needs an assumption to be able to answer it, write the assumptions you need along with your answer. If you need more room to write an answer, clearly indicate where you are continuing the answer.

There are four major questions worth 10 points each for 40 points in total.

**Regrade Policy**

- Regrading should only be requested in writing. Write what you would like to be reconsidered. Note, however, that an exam accepted for regrading will be reviewed and regraded in entirety (all questions).

| Grading Box | |
|---|---|
| **1.** | /10 |
| **2.** | /10 |
| **3.** | /10 |
| **4.** | /10 |
| **Total** | /40 |

1. **Query Semantics.** *Say what?* (10 points) [Short Answer]

| **Customer** | |
|---|---|
| cust# | PK |
| cname | |
| fav_colour | |
| phone# | |

| **Item** | |
|---|---|
| item# | PK |
| prod# | FK to **Product** |
| cust# | FK to **Customer** |
| colour | |
| date_sold | |

| **Product** | |
|---|---|
| prod# | PK |
| pname | |
| cost | |
| maker | FK to **Company** |

| **Avail_Colours** | |
|---|---|
| prod# | PK, FK to **Product** |
| colour | PK |

Figure 1: **Colour** Schema.

Consider the schema in Figure 1 as we have used in class.

For each of the following, say briefly in *plain* English—as the descriptions for Question 2a–2d—what the query means, as though you are explaining it to a non-technical person.

Note that simply paraphrasing the query—e.g., "This query finds, for all cust#'s such that there exists a prod#, for all..."—is *not* adequate!

a. (2 points)

```
select C.cust#, C.cname, P.prod#, P.pname
    from Customer C, Item I, Product P
    where C.cust# = I.cust# and I.prod# = P.prod#
        and P.cost > 1000
        and I.date_sold >= '1 May 2008';
```

*Report customers by name and items they have bought on and after 1 May 2008 by product name that cost more than one thousand.*

b. (3 points)

```
select P.prod#, P.pname, count(*) as number, sum(cost) as amount
    from Item I, Product P
    where I.prod# = P.prod#
    group by P.prod#, P.pname;
```

*List each distinct product by name that has ever been sold and report how many items of it have been sold and the total (cost) spent for it.*

c. (2 points)

$\pi_{\mathsf{cname}}(\sigma_{\mathsf{colour}=\mathsf{fav\_colour}}(\textbf{Customer} \bowtie \textbf{Item}))$

*List customers by name who have ever bought an item in his or her favourite colour. (Note that* cust# *is not returned; this means if there are two customers of the same name each of whom qualifies, the name will be reported only once. This could lead to confusion.)*

d. (3 points)

```
select C.cust#, C.cname, P.prod#, P.pname
    from Customer C, Product P
    where not exists (
            select A.colour
                from Avail_Colour A
                where P.prod# = A.prod#
            except
            select I.colour
                from Item I
                where C.cust# = I.cust# and P.prod# = I.prod#
        );
```

*List customers by name with each product for which they own in all its available colours.*

2. **SQL** *Amazing queries.* (10 points) [Exercise]

You have just gone work for **Amazing.com**. Congratulations!

**Customer**(<u>cno</u>, *cname*, *address*)
**Author**(<u>ano</u>, *aname*, birth, country)
**Book**(<u>isbn</u>, *title*, *year*, *publisher*, *language*)
**Wrote**(<u>ano</u>, <u>isbn</u>)
  FK (ano) refs **Author**
  FK (isbn) refs **Book**
**Stock**(<u>isbn</u>, <u>from</u>, *qnty*, *price*)
  FK (isbn) refs **Book**
**Purchase**(<u>cno</u>, <u>isbn</u>, <u>when</u>, *from*, *qnty*)
  FK (cno) refs **Customer**
  FK (isbn, from) refs **Stock**
**Payment**(<u>cno</u>, <u>when</u>, *amount*)
  FK (cno) refs **Customer**

Figure 2: **Amazing.com** Schema.

The basic schema of the main database at **Amazing.com** is shown in Figure 2. The underlined attributes indicate a table's primary key (and are hence not nullable). Additionally, attributes that are in *italics* are not nullable. Foreign keys are indicated by FK.

The attribute price ($> 0$) in **Stock** tells how much a customer pays for that book. The attribute from in **Purchase** indicates where the stock was pulled from, and its attribute qnty ($> 0$) indicates the number of copies of the book the customer bought in that purchase. In **Payment**, attribute amount ($> 0$) indicates a payment the customer has made.

Write queries in SQL with respect to the schema in Figure 2 for the following.

a. (3 points) Report cno and cname for each customer who has bought a book written in *Japanese* whose author's country is *Canada*.

```
select distinct C.cno, C.cname
    from Customer C, Purchase P, Book B, Author A, Wrote W
    where C.cno = P.cno and P.isbn = B.isbn
        and P.isbn = W.isbn and W.ano = A.ano
        and B.language = 'Japanese'
        and A.country = 'Canada';
```

b. (2 points) Report cno, cname, paid for each customer who has ever made a payment, for which paid is the total of that customer's payments.

```
select C.cno, C.cname, sum(P.amount) as paid
    from Customer C, Payment P
    where C.cno = P.cno
    group by C.cno, C.cname;
```

c. (3 points) Report cno, cname, spent for each customer who has ever made a purchase, for which spent is the total of that customer's purchases.

```
select C.cno, C.cname, sum(P.qnty * S.price) as paid
    from Customer C, Purchase P, Stock S
    where C.cno = P.cno
        and P.isbn = S.isbn and P.from = S.from
    group by C.cno, C.cname;
```

d. (2 points)

Write an SQL query to report for each customer how much he or she owes the bookstore. For each row, report cname, cno, address, and amount. If the bookstore *owes* the customer money, report the amount as a negative number.

Be careful. A customer may never have bought anything, but has made a payment. (Weird.) Or a customer may have bought something, but not yet have made any payments. (More common.) Make certain that each customer gets reported.

Assume you have correct SQL for Questions 2b and 2c to use in composing your query.

```
with
    Paid (cno, cname, paid) as (
                ⋮
    ),
    Spent (cno, cname, spent) as (
                ⋮
    )
```

```
    PaidAll (cno, cname, paid) as (
        select cno, cname, max(paid)
            from (
                select cno, cname, 0
                    from Customer
                union
                select cno, cname, paid
                    from Paid
                ) as X
    ),
    SpentAll (cno, cname, spent) as (
        select cno, cname, max(spent)
            from (
                select cno, cname, 0
                    from Customer
                union
                select cno, cname, spent
                    from Spent
                ) as X
    )
select C.cno, C.cname, C.address, S.spent - P.paid as amount
    from Customer C, SpentAll S, PaidAll P
    where C.cno = S.cno and C.cno = P.cno;
```

3. **Query Logic.** *Careful what you ask for!* (10 points) [Multiple Choice]

The primary key of a relation is indicated by its underlined attributes. No attributes are nullable.

---

a. In relational algebra, the intersection operator ($\cap$) is *not* logically redundant if we only have additionally

**A.** join ($\bowtie$)
**B.** crossproduct ($\times$), select ($\sigma$), and project ($\pi$)
**C.** difference ($-$) and union ($\cup$)
**D.** crossproduct ($\times$) and difference ($-$)
$\boxed{\textbf{E.}}$ crossproduct ($\times$) and union ($\cup$)

---

b. Consider the relations $\mathbf{R}(\underline{\mathsf{A}}, \mathsf{B})$ and $\mathbf{S}(\mathsf{A}, \underline{\mathsf{B}})$, where **R** has a foreign key referencing **S** via B, and **S** has a foreign key referencing **R** via A.

Which of the following is guaranteed to produce fewer than, or at most the same, number of tuples as any of the others?

**A.** $\mathbf{R} \bowtie \pi_{\mathsf{B}}(\mathbf{S})$
**B.** $\pi_{\mathsf{A}}(\mathbf{R}) \bowtie \mathbf{S}$
$\boxed{\textbf{C.}}$ $\mathbf{R} \bowtie \mathbf{S}$
**D.** $\mathbf{R} \cup \mathbf{S}$
**E.** There is not enough information to answer this.

---

c. Consider the relations $\mathbf{R}(\mathsf{A}, \mathsf{B})$ and $\mathbf{S}(\mathsf{B}, \mathsf{C})$.

One of these is not like the others. That is, one can evaluate differently than the other four. Which one?

**A.** $\pi_{\mathsf{A}}(\mathbf{R} \bowtie \mathbf{S})$
$\boxed{\textbf{B.}}$ $\pi_{\mathsf{A}}(\mathbf{R}) - (\pi_{\mathsf{A}}(\mathbf{R} - \pi_{\mathsf{A,B}}(\mathbf{R} \bowtie \mathbf{S})))$
**C.** $\pi_{\mathsf{A}}(\mathbf{R}) - (\pi_{\mathsf{A}}(\mathbf{R}) - \pi_{\mathsf{A}}(\mathbf{R} \bowtie \mathbf{S}))$
**D.** $\pi_{\mathsf{A}}(\mathbf{R} \cap (\pi_{\mathsf{A}}(\mathbf{R}) \times \pi_{\mathsf{B}}(\mathbf{S})))$
**E.** $\pi_{\mathsf{A}}((\mathbf{R} \times \pi_{\mathsf{C}}(\mathbf{S})) \cap (\pi_{\mathsf{A}}(\mathbf{R}) \times \mathbf{S}))$

---

d. Consider the schema $\mathbf{R}(\underline{\mathsf{A}}, \underline{\mathsf{B}})$, and $\mathbf{S}(\underline{\mathsf{B}}, \underline{\mathsf{C}})$ (with no FKs).

One of these things is not like the other. That is, one of them may evaluate differently than the others. Which one?

**A.** $\{\langle A, C\rangle \mid \exists B(\langle A, B\rangle \in \mathbf{R} \rightarrow \langle B, C\rangle \in \mathbf{S})\}$
**B.** `select distinct R.A, S.C from R, S where R.B = S.B;`
**C.** $\pi_{\mathsf{A,C}}(\mathbf{R} \bowtie \mathbf{S})$
$\boxed{\textbf{D.}}$
```
select distinct R.A, S.C from R, S s1
    where R.B in (select s2.B from S s2 where s1.C = s2.C);
```
**E.** $\{\langle A, C\rangle \mid \neg \exists \mathrm{B}(\langle A, B\rangle \in \mathbf{R} \rightarrow \langle B, C\rangle \notin \mathbf{S})\}$

**R**

| A | B |
|---|---|
| 1 | a |
| 2 | b |
| 2 | c |
| 3 | d |

**S**

| B | C |
|---|---|
| b | 5 |
| b | 6 |
| c | 5 |
| d | 6 |
| e | 5 |

Two tables: **R** & **S**.

**I**

| A | B | C |
|---|---|---|

**II**

| A | B | C |
|---|---|---|
| 1 | a | 5 |
| 2 | b | 6 |
| 2 | c | 5 |
| 3 | d | 6 |

**III**

| A | B | C |
|---|---|---|
| 2 | b | 5 |
| 2 | b | 6 |
| 2 | c | 5 |
| 3 | d | 6 |

**IV**

| A | B | C |
|---|---|---|
| 1 | a | 5 |
| 1 | a | 6 |
| 2 | b | 5 |
| 2 | b | 6 |
| 2 | c | 5 |
| 2 | c | 6 |
| 3 | d | 5 |
| 3 | d | 6 |

**V**

| A | B | C |
|---|---|---|
| 1 | b | 5 |
| 1 | b | 6 |
| 1 | c | 5 |
| 1 | d | 6 |
| 1 | e | 5 |
| 2 | b | 5 |
| 2 | b | 6 |
| 2 | c | 5 |
| 2 | d | 6 |
| 2 | e | 5 |
| 3 | b | 5 |
| 3 | b | 6 |
| 3 | c | 5 |
| 3 | d | 6 |
| 3 | e | 5 |

Possible answer tables.

All the join operations below are natural joins.

e. What is the resulting table of $\{\langle A, B, C\rangle \mid \langle A, B\rangle \in \mathbf{R} \wedge \exists D(\langle D, C\rangle \in \mathbf{S})\}$?

**A.** I **B.** II **C.** III [✓ **D.** IV] **E.** V

f. What is the resulting table of $\{\langle A, B, C\rangle \mid \langle A, B\rangle \in \mathbf{R} \wedge \langle B, A\rangle \in \mathbf{S}\}$?

[✓ **A.** I] **B.** II **C.** III **D.** IV **E.** V

g. What is the resulting table of $\{\langle p.A, p.B, q.C\rangle \mid p \in \mathbf{R} \wedge q \in \mathbf{S} \wedge p.B = q.B\}$?

**A.** I **B.** II [✓ **C.** III] **D.** IV **E.** V

For 3i and 3h:

**I.** a lossless join decomposition
**II.** dependency preserving

h. For any schema,
[✓ **A.**] there is always a 3NF refinement that is both **I** and **II**.
**B.** there is always a 3NF refinement that is **I**, but not necessarily **II**.
**C.** there is always a 3NF refinement that is **II**, but not necessarily **I**.
**D.** there is never a 3NF refinement that is both **I** and **II**.
**E.** there is never a 3NF refinement that is **I** or **II**.

i. For any schema,
**A.** there is always a BCNF refinement that is both **I** and **II**.
[✓ **B.**] there is always a BCNF refinement that is **I**, but not necessarily **II**.
**C.** there is always a BCNF refinement that is **II**, but not necessarily **I**.
**D.** there is never a BCNF refinement that is both **I** and **II**.
**E.** there is never a BCNF refinement that is **I** or **II**.

j. Consider table **R** with attributes A, B, C, D, and E. What is the largest number of candidate keys that **R** could simultaneously have?
**A.** 1
**B.** 5
[✓ **C.**] 10
**D.** 31
**E.** 365

4. **Schema Refinement.** *I'm quite refined myself.* (10 points) [Exercise]

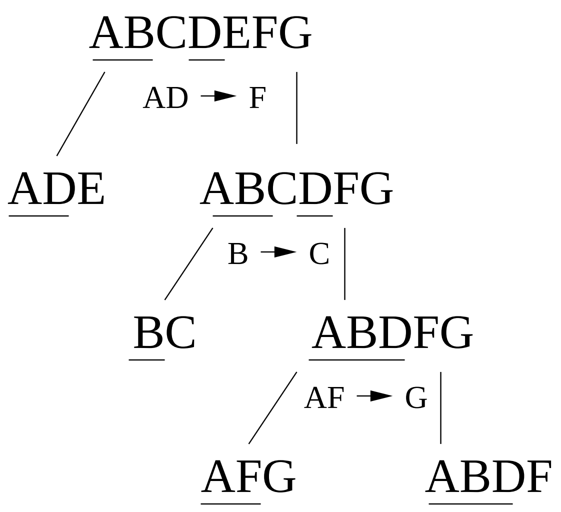
Consider the relation **R** with attributes A, B, C, D, E, F, and G and with the following functional dependencies (FDs):

$$\begin{array}{rr} AD \mapsto E & BE \mapsto F \\ B \mapsto C & AF \mapsto G \\ & F \mapsto E \end{array}$$

a. (2 points) What are the candidate keys of **R**?

$\{A, B, D\}$ *(or just ABD in shorthand).*
*(Continued on page 12.)*

b. (5 points) Provide a BCNF lossless-join decomposition for **R** (ABCDEFG). Show your steps and your decomposition tree.



c. (3 points) Provide a BCNF decomposition that is lossless *and* is dependency preserving, or explain why there is none in this case.

*There is none. We would need to add a table* BEF*. However, because of the functional dependency* $F \mapsto E$*, both* BE *and* BF *are candidate keys.*
$F \mapsto E$ *then is a violation of BCNF. We cannot decompose* BEF *and still preserve the dependency.*

Extra space.

*Continued answer for Question 4a:*
*We can find this in this case with the following reasoning.* A, B, *nor* D *show up on the right-hand side of any functional dependency (FD). So each must belong to* any *candidate key. We then check* ABD *to see which attributes it determines. In this case,* ABD *determines them all. So it is* a *candidate key.*
*Of course, generally, there can be more candidate keys. In this case, since* A, B, *and* D *have to be part of any candidate key, any other set of attributes that determines them all would contain* ABD *and so would be a super-key. Thus, there are no other candidate keys.*

Relax. Turn in your exam. Go home.