CS348 Sample Midterm (Fall 2019)

Question 1

(a) *Explain how physical data dependencies can increase the cost of maintaining an information system.*

If the system is dependent on physical data, then whenever the underlying physical information changes, the system have to be extensively updated to adapt to the new physical information format. This is a maintenance overhead.

(b) *What are the two general problems relating to data that are addressed by the user of database technology when implementing an information system?*

Physical independence and logical independence.

(c) *Explain each of the following terms:*

1. *Relational Completeness*: A query language is relationally complete if it is as least as expressive as relational calculus.
2. *Atomicity*: "All or nothing": To an external observer, it should appear as if either all operations in an atom have happened, or none have happened.
3. *Domain Independence*: A query φ is **domain independent** if for all database instance $\mathbf{DB}_{1,2} = (\mathbf{D}_i, =, \mathbf{R}_1, \ldots, \mathbf{R}_k)$ and valuation θ,
$$\mathbf{DB}_1, \theta \models \varphi \iff \mathbf{DB}_2, \theta \models \varphi$$

(d) *Describe the primary means of defining external views in the SQL language.*

Use the `create view` comamnd:

```
create view (<query>)
```

(e) *Assume relation R has numeric attributes $\{A, B, C\}$, with C as the primary key. Express the following query in the range restricted fragment of the relational calculus.*

```
select distinct r1.A
from R as r1, R as r2
and not exists (
  select * from R as r3
  where r3.B = r1.B or r3.B = r2.B )
```

Step 1: Convert the above query to relational calculus:

$$\{x_1 : \exists y_1, z_1, x_2, y_2, z_2 . R(x_1, y_1, z_1) \wedge R(x_2, y_2, z_2) \wedge \neg\exists x_3, y_3, z_3 . R(x_3, y_3, z_3) \wedge (y_3 = y_1 \vee y_3 = y_2)\}$$

This is not range restricted, since $y_3 = y_1$ is not range restricted. First we apply distributive law:

$$\neg\exists x_3, y_3, z_3 . R(x_3, y_3, z_3) \wedge (y_3 = y_1 \vee y_3 = y_2) \equiv \neg\exists x_3, y_3, z_3 . (R(x_3, y_3, z_3) \wedge y_3 = y_1) \vee (R(x_3, y_3, z_3) \wedge y_3 = y_2)$$

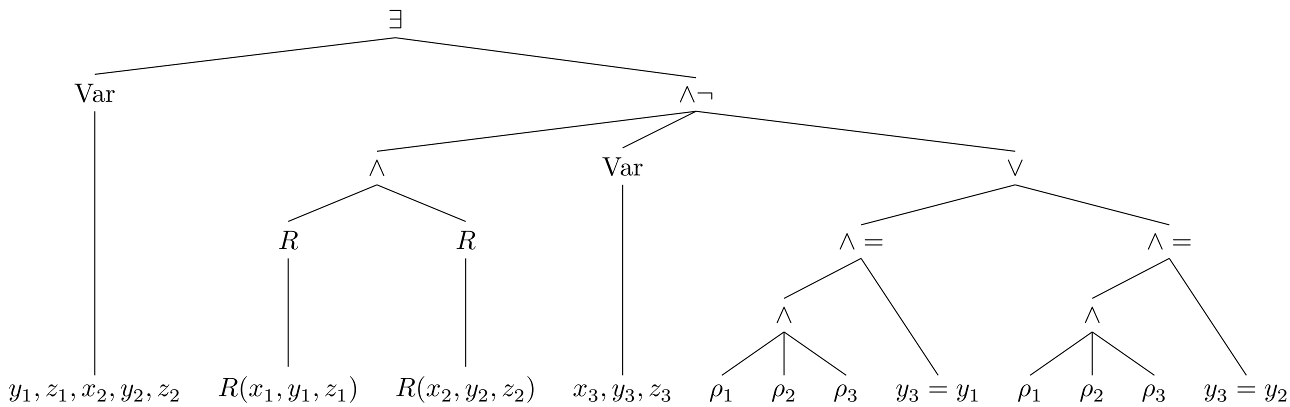
This is still not range restricted since the or causes problems. This can be mitigated by commuting the outer truth $R(x_1, y_1, z_1), R(x_2, y_2, z_2)$ into the inner clause: (we denote $R(x_i, y_i, z_i)$ by ρ_i)

$$\begin{aligned} \neg\exists x_3, y_3, z_3 . &(\rho_1 \wedge \rho_2 \wedge \rho_3 \wedge (y_3 = y_1)) \\ \vee &(\rho_1 \wedge \rho_2 \wedge \rho_3 \wedge (y_3 = y_2)) \end{aligned}$$

This produces the formula

$$\{x_1 : \exists y_1, z_1, x_2, y_2, z_2 . \rho_1 \wedge \rho_2 \wedge \neg\exists x_3, y_3, z_3 . (\rho_1 \wedge \rho_2 \wedge \rho_3 \wedge (y_3 = y_1)) \vee (\rho_1 \wedge \rho_2 \wedge \rho_3 \wedge (y_3 = y_2))\}$$

Anatomy of the above formula:



(f) *What are three features of SQL query language that makes it more expressive than the relational calculus?*
(Three of)

- Null
- Aggregation/Counting
- Duplicate semantics
- Recursive queries (not all dbs support them)

Question 2

Consider the following relational database schema for maintaining customer rental information

```
create table customer
  ( cnum   integer not null,
    cname  varchar(20) not null,
    city   varchar(20) not null,
  primary key (cnum) );

create table car
  ( license   integer not null,
    make      varchar(20) not null,
    model     varchar(20) not null,
    year      integer not null,
  primary key (license) );

create table pickup
  ( rnum      integer not null,
    cnum      integer not null,
    license   integer not null,
    fee       integer not null,
  primary key (rnum),
  foreign key (cnum)    references customer,
  foreign key (license) references car );

create table dropoff
  ( rnum      integer not null,
  primary key (rnum),
  foreign key (rnum) references pickup );
```

The database schema reflects two main events relating to car rentals:

(a) A customer takes posession of a car at the start of a rental agreement. In this case, a tuple is added to the `pickup` table.

(b) A customer returns the car at the end of a rental agreement and pays the agreed rental fee. In this case, a tuple is added to the `dropoff` table.

For each of the following queries, indicate if the query is a conjunctive query, and then translate the query to both ***relational calculus*** *and* ***SQL****.*

(a) *The number and name of each customer who has rented the same car at least twice in the past*

$$\{c, n : \exists\gamma.\mathsf{customer}(c, n, \gamma) \wedge (\exists l, \phi_1, \phi_2, r_1, r_2 : (r_1 \neq r_2) \wedge \mathsf{pickup}(r_1, c, l, \phi_1) \wedge \mathsf{pickup}(r_2, c, l, \phi_2)) \wedge \mathsf{dropoff}(r_1) \wedge \mathsf{dropoff}(r_2)\}$$

```
select c.cnum, c.cname
from customer as c
where exists (
    select * from car, pickup as p1, pickup as p1, dropoff as d1, dropoff as d2
    where (car.license = p1.license)
      and (car.license = p2.license)
      and (p1.rnum <> p2.rnum)
      and (p1.cnum = c.cnum)
      and (p2.cnum = c.cnum)
      and (p1.rnum = d1.rnum)
      and (p2.rnum = d2.rnum)
  )
```

(b) *The license and fee of cars that are currently rented by some customer in Waterloo.*

$$\{l, f : \exists \tau, \mu.\mathsf{car}(l, \tau, \mu) \wedge \exists c, n, \rho.\mathsf{pickup}(\rho, c, l, f) \wedge \neg\mathsf{dropoff}(\rho) \wedge \mathsf{customer}(c, n, \text{Waterloo})\}$$

```sql
select car.license, pickup.fee
from car, pickup, customer
where (car.license = pickup.license)
  and (customer.cnum = pickup.cnum)
  and (customer.city = 'Waterloo')
  and not exists (
    select * from dropoff
    where dropoff.rnum = pickup.rnum
  )
```

(c) Translate to ~~DDL~~DML: *The make and model of cars that have generated the lowest revenue. That is, for which the total rental fees for past rentals of cars of these makes and models is among the lowest*

```sql
select car.make, car.model
from car
where (
    select avg(p.fee)
    from pickup as p, car as c
    where (c.make = car.make) and (c.model = car.model) and
      (p.license = c.license)
  ) <= all (
    select avg(p.fee)
    from pickup as p, car as c
    where (p.license = c.license)
    group by c.license, c.model, c.make
  )
```

Question 3

Questions on the SQL standard, application development, and ER modeling. Answer each of the following using no more than a few sentences in each case.

(a) *What is the view update problem?*

Some views created by joining two tables cannot be updated unambiguously. i.e. Inserting a tuple into this view can correspond to multiple different changes of the tables in the database.

(b) *Is it possible for an information system using a call level interface (CLI) to a SQL database to factor the overhead of compiling SQL queries? Justify your answer*

The answer is yes and no. For some queries if the programmer knows that the query will be executed multiple times, the query's compiling stage can be extracted out of the loop to improve performance. This isn't possible for dynamic queries though.

(c) *Explain the main purpose of each of the following*

1. *A host variable*:
   Communicate information between the host language and the embedded SQL language.
2. *A preprocesssor for embedded SQL*
   Translates the embedded SQL into native C code which can be handled by an ordinary C compiler.
3. *the COMMIT WORK command*
   Commit the modifications of the database into the database so they become permanent changes.

(d) *Explain how many-to-one relationships can be captured using general cardinality constraints*

Suppose a relation R has two participating entities E_1, E_2, and E_1 is the "many" side and E_2 is the "one" side. No cardinality constraints is required on E_1, but E_2 needs a $(0, 1)$ cardinality constraint to limit its maximal participation in R to 1.