Winter 2022 - CS540 – Assignment 2(Group 15)

Full Name	OSU Email ID	ONID
Bharghav Srikhakollu	srikhakb@oregonstate.edu	934289891
Akhil Sai Chintala	chintala@oregonstate.edu	934409906

1.Query Containment

(a) Explain whether the homomorphism theorem holds for conjunctive queries with equality, e.g., (T(s): − R (x, y), x = y). If your answer is no, then modify the theorem to hold for these queries.

Given T(s): - R (x,y), x = y. let us consider ‘c’ and represent the query as below

Q_1(a): R(a,b,c) such that a=b and b=c

Q_2(a): R(a,b,c) such that a=b and a=c

The two queries form an equivalent pair but when we check the homomorphism it fails. Hence, we can say that in all situations homomorphism theorem does not hold for conjunctive queries with equality.

To hold these queries for homomorphism we must add literals that represent transitivity of equality and symmetry as a=b, b=a, a=c, c=a, b=c and c=b.

(b) Explain whether the homomorphism theorem holds for conjunctive queries with ≤, e.g., (T(s): − R (x, y), x ≤ y)

Homomorphism theorem does not hold good for conjunctive queries with ≤.

Like the above-mentioned queries, let's consider as below.

Q_1(k): R(i,j,k) such that $i \leq k$

Q_2(k): R(x,y,k), R(y,x,k) - In this relation we can say that either $x \leq y$ or $y \leq x$

So, from this we can conclude that Q_2 is a subset of Q_1 but no homomorphism exists between the two conjunctive queries.

(c) Prove that the algorithm given in the class to determine whether one nonrecursive Datalog without negation is contained in another one is correct. In other words, show that for two nonrecursive Datalog without negation queries q and p, q is contained in p if and only if for each rule r_q in q there is a rule r_p in p such that r_q is contained in rp

As per the given information, when we have two queries q and p where

--> rule r_q corresponds to q

--> rule r_p corresponds to p

This condition satisfies only when every variable appears in at least one positive predicate (true or false outcome). Rules are unions of conjunctive queries so there should be a dataset from r_q which is subset of r_p. With this statement as true we can conclude that given algorithm of one nonrecursive datalog without negation is contained in another one is correct.

(d) Characterize containment of queries of the form q1 - q2, where q1, q2 are relational algebra queries with selection, projection, Cartesian product and union. You may also assume that q1 and q2 are nonrecursive Datalog queries without negation.

Hint: First develop characterizations for the case in which q1 and q2 do not have union.

Let us assume q1 and q2 as nonrecursive datalog queries without negation.

For the queries of the form q1- q2

Let us consider q2' such that q1 – q2 is a subset of q1 – q2'

Which can be correlated as q2' is a subset of q2

From this we can say that the queries of the form q1-q2 are contained.

<u>2. Incomplete Information:</u>

(a) Show that naive tables are not a weak representation system for relational algebra queries.

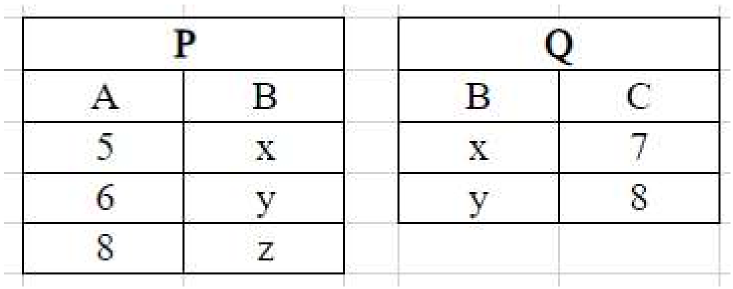
<u>Naïve Tables:</u> To handle null values in database, we work on certain(sure) answers where we will try to put distinct variables in place of null values – these tables are called 'Codd tables' although these tables form weak representation system it will no longer be valid system with union or join. So, in these Codd tables we will coincide with some of the variables which are referred to as marked nulls. Such representation of tables is called 'Naïve tables.'

In finding certain answers, we will try to work on the output as below in weak representation systems.

<u>sure (Q, T) = intersection{Q(Ri) | Ri ∈ POSS(T)}</u>

Q – Conjunctive query, R – Relations, POSS(T) - All possible solutions

This theory on Naïve tables forms a weak representation system for Select, Projection, Join and Union but will not work on intersection or difference (Intersect, Except, Not In, Not Exists)



P	
A	B
5	x
6	y
8	z

Q	
B	C
x	7
y	8

SELECT P.A from P

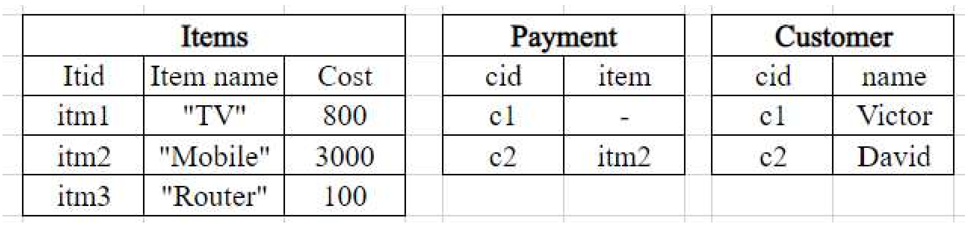
WHERE P.B NOT IN

(SELECT Q.B from Q)

Here we do not know the exact value of B in either of the table tuples since it is defined as variables. From the above highlighted statement to find weak representation, we try to find the certain answer by the intersection, here when we try to find the solution, the intersection (common solution) across the relations P, Q for the query returns null value. Similarly, we can find that we cannot get desired output when Intersect, Except, Not Exists is used with Naïve tables. So, we can say that naïve tables are not weak representation system for **all** relational algebraic queries.

(b) Find a SQL query that returns false negatives over databases with null values. Then, propose a transformation that translates this query to a SQL query that does not return any false negatives and retains all true positives delivered by the original query over databases with null values. You may also find a SQL query that returns false positives over databases with null values and transform it to a query that does not return any false positives and retains all true positives delivered by the original query over a database with null values.

- ✓ **False Negative:** If the result of a query "misses the correct results" (reject true null hypothesis) due to null values in the database then we call this a false negative condition.



Items		
Itid	Item name	Cost
itm1	"TV"	800
itm2	"Mobile"	3000
itm3	"Router"	100

Payment	
cid	item
c1	-
c2	itm2

Customer	
cid	name
c1	Victor
c2	David

To find unpaid orders,

SELECT Itid FROM Items

WHERE Itid NOT IN (SELECT item FROM Payment)

Here in Payment table, we expect that itm1 is missing for cid c1, so ideally the above query is expected to give output as itm3. But the result of the query will be EMPTY. So, we can modify the query as below so that we can handle the NULL values in the table and get positive output instead of EMPTY.

SELECT Itid FROM Items

WHERE Itid NOT IN (SELECT item FROM Payment WHERE **item IS NOT NULL**)

This will give output as itm1, itm3 which may be correct to some extent but not accurate since customer c1 could have bought either itm1 or itm3 which we are not sure. If by default c1 customer buy/do not buy only itm1 then we can modify the query further to make a union of the tables and get the output.

✓ **False Positive:** If the result of a query "returns incorrect results" (accept false null hypothesis) due to null values in the database then we call this a false positive condition, which is more dangerous than false negative.

For the above-mentioned database tables, we will try to find the customers without an order

SELECT C.cid from Customer C

WHERE NOT EXISTS (SELECT * FROM Items I, Payment P

WHERE C.cid = P.cid AND P.item = O.item)

The expected answer is EMTPY since both the customers made the payment, but because of the presence of NULL value in database, the output will be c2. So, we can modify the query as below to get the desired output.

SELECT C.cid from Customer C

WHERE NOT EXISTS (SELECT * FROM Items I, Payment P

WHERE C.cid = P.cid AND (P.item = O.item **OR P.item IS NULL**)

This will eliminate the NULL values fields and return the output as EMPTY.

(c) Prove that the family of queries UCQ (without inequality) does not return any false positives over a database with null values.

Let's assume the sequence of conjunctive queries (CQ) Q1, Q2, Q3, …... Qn.

The union of Conjunctive queries (UCQ) is defined as Q: - Q1 U Q2 U Q3……. U Qn.

Generally, UCQ with inequalities will always result in false-positive values in all databases due to the presence of null values in the database we accept false hypothesis by accepting the incorrect results out of the query.

So, by contradicting, the UCQ's without inequalities will never return false-positive values.

(d) Consider relational algebra queries with difference over relations with only one attribute. Show that these queries return false positive results over databases with null values. Then, suggest an algorithm that translates each query in this set of queries to a query that does not return any false positive and retains all true positives of the original query over a database with null values

Consider two relations P and Q with single attributes. P.A and Q.B

P.A = {5}, Q.B = {null}

Let's consider the relational difference between P and Q as below.

P - (P - Q): In Ideal scenario (without null), this will give the result of Q with values.

But in this case P – Q gives "{}" and P - (P - Q) results in different output which is not {} or null. We can say that this condition is a false positive result over databases with null values.

To avoid false positive outcomes, we can check for subqueries and its constraints, null values. If null values are to be included in the subquery, add conditions to check the null values so that it will eliminate the chance of getting false positive results.