Ferhat Elmas 214805

Advanced Databases Hw-13

1-)
Tableau form:

x	y1	z	w
x1	y1	z1	w1
x	y2	z1	z
x2	y	z	y
x	y	z	

For functional dependency with Chase, we get the w in the first row to z since A in first and third row is same x then D must be same and we can update z because it is in the query head.

Tableau form after chase update:

x	y1	z	**z**
x1	y1	z1	w1
x	y2	z1	z
x2	y	z	y
x	y	z	

if we have defined a map,

$\Psi(x1) = x$
$\Psi(z1) = z$
$\Psi(w1) = z$
we can get rid of the second row.

Moreover, due to $\Psi(z1) = z$, z1 in the third row becomes z.

x	y1	z	z
x	y1	**z**	**z**

x	y2	**z**	z
x2	y	z	y
x	y	z	

After one more iteration of Ψ(y2) = y1.

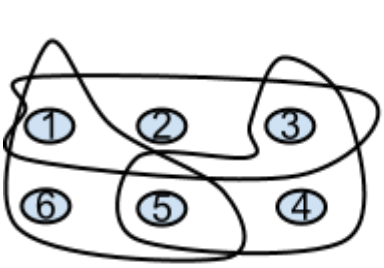
x	y1	z	z
x	y1	z	z
x	**y1**	z	z
x	y1	z	z
x	y	z	

As it is seen from above table, we can remove two of same three rows so minimized query becomes:

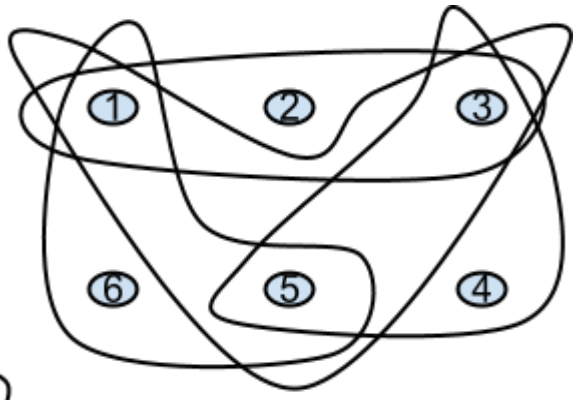
x	y1	z	z
x2	y	z	y
x	y	z	

2-) For an example of infinite sequence, we can utilize of the constants. For example, schema(S), each query can have a different natural number since natural numbers are infinite, we have an infinite sequences of queries. Moreover, since values are composed of only constants, there is no homomorphism between any of two queries.

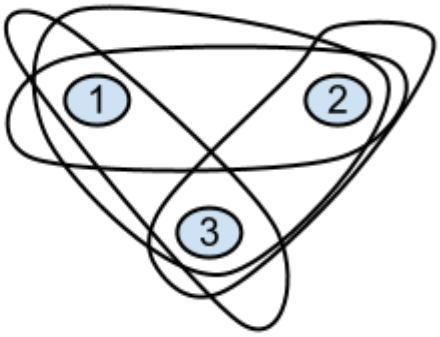
3-)
Here is the hypergraphs of the queries:



1

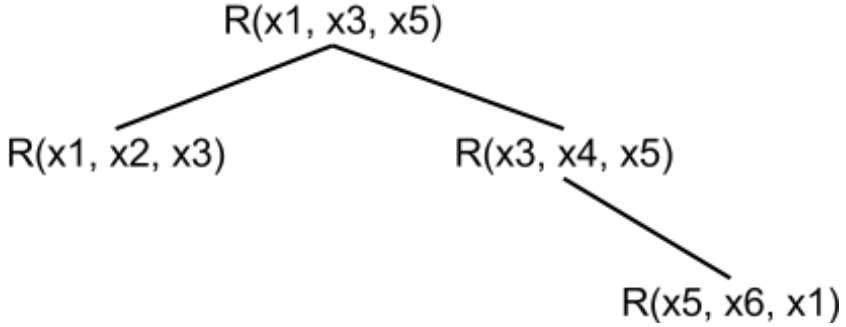


2



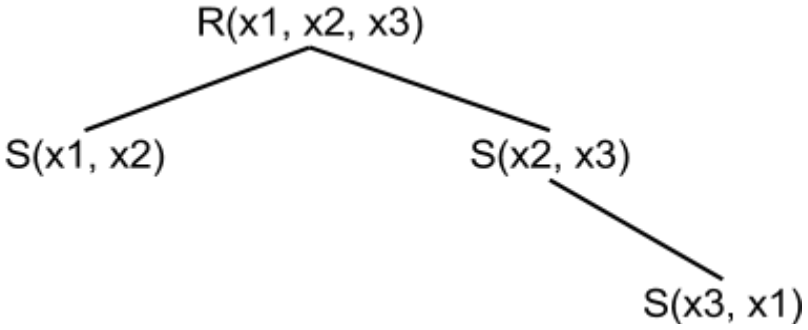
3

- ans(x1 , x3) ← R(x1 , x2 , x3), R(x3 , x4 , x5), R(x5 , x6 , x1)
    - there isn't any ear for this query so we cannot go further with gyo algorithm and since resultset isn't empty, it is cyclic.
- ans(x1 , x3) ← R(x1 , x2 , x3), R(x3 , x4 , x5), R(x5 , x6 , x1), R(x1 , x3 , x5)
    - f := R(x1, x2, x3) and f' := R(x1, x3, x5) then we have only x2 for f-f' and it isn't used in any other query.
    - f := R(x3, x4, x5) and f' := R(x1, x3, x5) then we have only x4 for f-f' and it isn't used in any other query.
    - f := R(x5, x6, x1) and f' := R(x1, x3, x5) then we have only x6 for f-f' and it isn't used in any other query.
    - R(x1, x3, x5) is isolated so remove it.
    - Since we can get an empty set, it is acyclic.
    - While applying gyo algorithm for reduction, we get a parse tree implicitly for execution plan. Therefore, now we have



- ans(x1 , x3) ← S(x1 , x2), S(x2 , x3), S(x3 , x1), R(x1 , x2 , x3)
    - f := S(x1, x2) and f' := R(x1, x2, x3) then we have only empty set which is fine where we can remove S(x1, x2).
    - f := S(x2, x3) and f' := R(x1, x2, x3) then we have only empty set which is fine where we can remove S(x2, x3).

- f := S(x3, x1) and f' := R(x1, x2, x3) then we have only empty set which is fine where we can remove S(x3, x1).
- Then, we have got the isolated R(x1, x2, x3) so we can also remove it.
- Since we can get an empty set, it is acyclic.



4-)

```
% alternating in black graph
alternateBlackArcs(A, B) :- arcBlack(A, C), alternateWhiteArcs(C, B), A != B.
alternateBlackArcs(A, B) :- arcBlack(A, B).

% alternating in white graph
alternateWhiteArcs(A, B) :- arcWhite(A, C), alternateBlackArcs(C, B), A != B.
alternateWhiteArcs(A, B) :- arcWhite(A, B).

path(A, B) :- alternateWhiteArcs(A, B).
```

5-)

```
% If Homer comes, Marge will come too.
visit("Marge") :- visit("Homer").

% At least one of their two daughters Lisa and Maggie will come.
visit("Lisa") v visit("Maggie").

% Either Marge or Bart will come, but not both.
visit("Marge") :- not visit("Bart").
visit("Bart") :- not visit("Marge").

% Either Bart and Lisa will come, or neither will come.
visit("Bart") :- visit("Lisa").
visit("Lisa") :- visit("Bart").

% If Maggie comes, then Lisa and Homer will also come.
visit("Lisa") :- visit("Maggie").
visit("Homer") :- visit("Maggie").
```