

# On Solving efficiently the View Selection Problem under Bag-semantics

---

Foto Afrati<sup>1</sup> Matthew Damigos<sup>1</sup> Manolis Gergatsoulis<sup>2</sup>

<sup>1</sup>*National Technical University of Athens*

<sup>2</sup>*Ionian University*

---

2nd International Workshop on  
*Business Intelligence for the Real Time Enterprise*  
(BIRTE 2008)

# Motivation

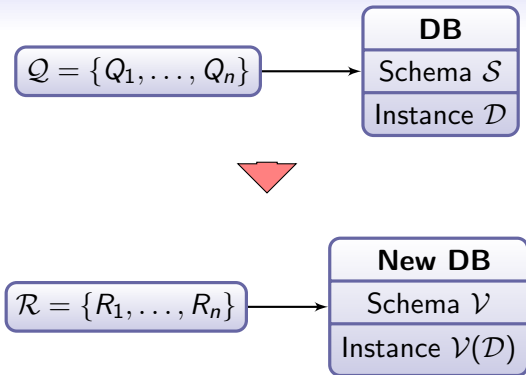
## Answering Set of Queries

- A Set of queries often involves repeated computation
  - parts of the same query
  - parts of different queries
- Improve query-evaluation performance
  - by precomputing an appropriate set of intermediate results of the queries' evaluation (materialized views)

## Web Sites

- connected to relational databases
- improve the performance
  - evaluate the queries on demand: **too much time**
  - compute every possible query in advance: **too much disk space**

## View Selection



- $\mathcal{R}(\mathcal{V}(\mathcal{D})) = \mathcal{Q}(\mathcal{D})$ ,
- the total time of getting the required results is reduced
- $\mathcal{V}(\mathcal{D})$  fits into available disk space.

# View Selection Example

- Database Schema  $S$ :  
PART(pName, wID)  
WAREHOUSES(wID, wLocation, wSupervisor)  
SUPPLIER(sName, sLocation)
- Database Instance  $D$ :

PART		WAREHOUSES			SUPPLIER	
pName	wID	wID	wLocation	wSupervisor	sName	sLocation
engine	1	1	Seattle	M.J.	Boeing	Seattle
engine	1	2	Boston	J.L.	Airbus	Paris
wing	2					
flap	1					

# View Selection Example

- Database Schema  $S$ :  
PART(pName, wID)  
WAREHOUSES(wID, wLocation, wSupervisor)  
SUPPLIER(sName, sLocation)
- Database Instance  $D$ :

PART		WAREHOUSES			SUPPLIER	
pName	wID	wID	wLocation	wSupervisor	sName	sLocation
engine	1	1	Seattle	M.J.	Boeing	Seattle
engine	1	2	Boston	J.L.	Airbus	Paris
wing	2					
flap	1					

- Bag Semantics: multiple occurrences of tuples**

## View Selection Example

- Database Schema  $S$ :  
 PART(pName, wID)  
 WAREHOUSES(wID, wLocation, wSupervisor)  
 SUPPLIER(sName, sLocation)

- Database Instance  $D$ :

PART		WAREHOUSES			SUPPLIER	
pName	wID	wID	wLocation	wSupervisor	sName	sLocation
engine	1	1	Seattle	M.J.	Boeing	Seattle
engine	1	2	Boston	J.L.	Airbus	Paris
wing	2					
flap	1					

- Q:  $q(X,Y) :- \text{SUPPLIER}(X,Z), \text{WAREHOUSES}(W,Z,U), \text{PART}(Y,W)$

Q(D)	
sName	pName
Boeing	engine
Boeing	engine
Boeing	flap

## View Selection Example

- Consider a set of views  $\mathcal{V} = \{V_1, V_2\}$ , where the definitions of  $V_1, V_2$  are:  
 $V_1: v_1(X,W) :- \text{SUPPLIER}(X,Z), \text{WAREHOUSES}(W,Z,U)$   
 $V_2: v_2(Y,W) :- \text{PART}(Y,W)$

$\mathcal{V}(D)$ :

$V_1(D)$		$V_2(D)$	
sName	wID	pName	wID
Boeing	1	engine	1
		engine	1
		wing	2
		flap	1

Database Instance  $D$ :

PART		WAREHOUSES			SUPPLIER	
pName	wID	wID	wLocation	wSupervisor	sName	sLocation
engine	1	1	Seattle	M.J.	Boeing	Seattle
engine	1	2	Boston	J.L.	Airbus	Paris
wing	2					
flap	1					

## View Selection Example

- Consider a set of views  $\mathcal{V} = \{V_1, V_2\}$ , where the definitions of  $V_1, V_2$  are:  
 $V_1: v_1(X,W) :- \text{SUPPLIER}(X,Z), \text{WAREHOUSES}(W,Z,U)$   
 $V_2: v_2(Y,W) :- \text{PART}(Y,W)$

$\mathcal{V}(D)$ :

$V_1(D)$		$V_2(D)$	
sName	wID	pName	wID
Boeing	1	engine	1
		engine	1
		wing	2
		flap	1

- Consider a query  $R$  defined on  $\mathcal{V}$  with definition:  
 $R: r(X,Y) :- v_1(X,W), v_2(Y,W)$

$R(\mathcal{V}(D))$	
sName	pName
Boeing	engine
Boeing	engine
Boeing	flap

## View Selection Example

- Consider a set of views  $\mathcal{V} = \{V_1, V_2\}$ , where the definitions of  $V_1, V_2$  are:  
 $V_1: v_1(X,W) :- \text{SUPPLIER}(X,Z), \text{WAREHOUSES}(W,Z,U)$   
 $V_2: v_2(Y,W) :- \text{PART}(Y,W)$

$\mathcal{V}(D)$ :

$V_1(D)$		$V_2(D)$	
sName	wID	pName	wID
Boeing	1	engine	1
		engine	1
		wing	2
		flap	1

- Consider **an equivalent rewriting**  $R$  of  $Q$  using  $\mathcal{V}$  with definition:  
 $R: r(X,Y) :- v_1(X,W), v_2(Y,W)$

$R(\mathcal{V}(D))$		$Q(D)$	
sName	pName	sName	pName
Boeing	engine	Boeing	engine
Boeing	engine	Boeing	engine
Boeing	flap	Boeing	flap

=

## View Selection Example

- Assuming available disk space equal to 5 tuples:
  - we can precompute and store  $\mathcal{V}(D)$
  - the query plan  $(R, \mathcal{V}(D))$  is more efficient than  $(Q, D)$
  - the tuple  $(\mathcal{V}, R)$  is a *solution* for the given problem input

## View Selection Example

- Assuming available disk space equal to 5 tuples:
  - we can precompute and store  $\mathcal{V}(D)$
  - the query plan  $(R, \mathcal{V}(D))$  is more efficient than  $(Q, D)$
  - the tuple  $(\mathcal{V}, R)$  is a *solution* for the given problem input

Is this an optimal query plan (i.e. an *optimal solution*)?

## View Selection Example

- Assuming available disk space equal to 5 tuples:
  - we can precompute and store  $\mathcal{V}(D)$
  - the query plan  $(R, \mathcal{V}(D))$  is more efficient than  $(Q, D)$
  - the tuple  $(\mathcal{V}, R)$  is a *solution* for the given problem input

Is this an optimal query plan (i.e. an *optimal solution*)? **NO**

Enough space to store the result of Q

# Problem Definition

- Query workload  $Q$  defined on a schema  $S$ , a database (instance)  $\mathcal{D}$
- find and precompute offline a *viewset*  $\mathcal{V}$ , defined on  $S$ , that when materialized
  - would satisfy a given storage limit  $L$
  - can be used to get equivalent rewritings of the queries in  $Q$
  - the evaluation cost of the queries is **minimized**
- $\mathcal{P} = (S, Q, \mathcal{D}, L)$ : *input of view selection problem*
- Bag-semantics
  - e.g. the instance of binary (bag-)relation  $r$  is  $\{((1,2);2),((2,3);1)\}$
- Language of  $Q$ ,  $\mathcal{V}$  and equivalent rewritings: CQs
- Size of a (bag-)relation: number of tuples
- The evaluation cost is increasing with the size of intermediate relations

## Related Work

- Problem Specification [CG2000]:
  - decidability for conjunctive queries under set semantics
- Search space under set semantics [CHS2001]:
  - each view has number of subgoals that is at most exponential in the size of the query workload
  - the number of the views required for an optimal solution is at most double exponential in the size of the query workload
- Search space under bag and bag-set semantics [ACGP2007]:
  - decision version of the problem under bag-, bag-set- (without self-joins) and set- (without self-joins) semantics is in NP
  - each viewset is obtained by generalizations of subexpressions of queries' bodies

# Contributions

- Restriction of the search space
  - reducing the number of candidate viewsets
- Further restriction (of the search space) for special cases:
  - unique occurrences of the base-relations in the query workload
  - the workload contains only path queries (path-query workload)
- Algorithm that outputs at least one optimal solution
  - LGG-VSB algorithm

## Preliminaries

- The definition of a **conjunctive query (or view)** is a rule of the form:

$$Q : q(\bar{X}) :- g_1(\bar{X}_1), \dots, g_n(\bar{X}_n)$$

- Head:  $q(\bar{X})$
  - Body:  $g_1(\bar{X}_1), \dots, g_n(\bar{X}_n)$
  - Subgoals:  $g_i(\bar{X}_i)$ , with  $1 \leq i \leq n$
- A **chain query** is a conjunctive query of the following form:

$$Q : q(X_0, X_n) :- r_1(X_0, X_1), r_2(X_1, X_2), \dots, r_n(X_{n-1}, X_n)$$

- **Path query of length n** (denoted as  $\mathbf{P}_n$ ) is a chain query of the following form:

$$P_n : q(X_0, X_n) :- \mathbf{r}(X_0, X_1), \mathbf{r}(X_1, X_2), \dots, \mathbf{r}(X_{n-1}, X_n)$$

## Queries' Subexpressions

**Query Workload:**  $\mathcal{Q} = \{Q_1, Q_2\}$

- $Q_1 : q_1(X, Y) :- e(X, X, X, Y)$
- $Q_2 : q_2(X, Y) :- e(X, Y, Y, Y)$

**Viewsets:**

- $\mathcal{V}_1 = \{V_{11}, V_{12}\}$ :  
 $V_{11} : v_{11}(X, Y) :- e(X, X, X, Y)$   
 $V_{12} : v_{12}(X, Y) :- e(X, Y, Y, Y)$
- $\mathcal{V}_2 = \{V_2\}$ :  
 $V_2 : v_2(Z_1, Z_2, Z_3) :- e(Z_1, Z_2, Z_2, Z_3)$
- $\mathcal{V}_3 = \{V_3\}$ :  
 $V_3 : v_3(W_1, W_2, W_3, W_4) :- e(W_1, W_2, W_3, W_4)$

- the above viewsets give equivalent rewritings of queries in  $\mathcal{Q}$
- $\mathcal{V}_1$  is set of queries' subexpressions

# Generalizations of Subexpressions

## Viewsets:

- $\mathcal{V}_1 = \{V_{11}, V_{12}\}$ :  
 $V_{11} : v_{11}(X, Y) :- e(X, X, X, Y)$   
 $V_{12} : v_{12}(X, Y) :- e(X, Y, Y, Y)$
- $\mathcal{V}_2 = \{V_2\}$ :  
 $V_2 : v_2(Z_1, Z_2, Z_3) :- e(Z_1, Z_2, Z_2, Z_3)$
- $\mathcal{V}_3 = \{V_3\}$ :  
 $V_3 : v_3(W_1, W_2, W_3, W_4) :- e(W_1, W_2, W_3, W_4)$

- $V_3$  is a generalization of  $V_2$

- i.e. substitution from  $V_3$  to  $V_2$ :

$$W_1 \rightarrow Z_1, W_2 \rightarrow Z_2, W_3 \rightarrow Z_2, W_4 \rightarrow Z_3$$

- $V_2$  and  $V_3$  are common generalizations of queries' subexpressions

e.g.

$$\frac{V_2 \rightarrow V_{11}}{Z_1 \rightarrow X, Z_2 \rightarrow X, Z_3 \rightarrow Y} \quad \Bigg| \quad \frac{V_2 \rightarrow V_{12}}{Z_1 \rightarrow X, Z_2 \rightarrow Y, Z_3 \rightarrow Y}$$

# Generalizations of Subexpressions

## Viewsets:

- $\mathcal{V}_1 = \{V_{11}, V_{12}\}$ :  
 $V_{11} : v_{11}(X, Y) :- e(X, X, X, Y)$   
 $V_{12} : v_{12}(X, Y) :- e(X, Y, Y, Y)$
  - $\mathcal{V}_2 = \{V_2\}$ :  
 $V_2 : v_2(Z_1, Z_2, Z_3) :- e(Z_1, Z_2, Z_2, Z_3)$
  - $\mathcal{V}_3 = \{V_3\}$ :  
 $V_3 : v_3(W_1, W_2, W_3, W_4) :- e(W_1, W_2, W_3, W_4)$
- $V_3$  is a generalization of  $V_2$
  - $V_2$  and  $V_3$  are common generalizations of queries' subexpressions  
e.g.

$$\frac{V_2 \rightarrow V_{11}}{Z_1 \rightarrow X, Z_2 \rightarrow X, Z_3 \rightarrow Y} \quad \Bigg| \quad \frac{V_2 \rightarrow V_{12}}{Z_1 \rightarrow X, Z_2 \rightarrow Y, Z_3 \rightarrow Y}$$

**$V_2$  is an least general generalization (or lgg) of subexpressions of the queries**

## Subexpressions do not suffice

- Consider  $\mathcal{D} = \{(e(a, a, a, a); 1)\}$

- $\mathcal{V}_1 = \{V_{11}, V_{12}\}$ :

$V_{11} : v_{11}(X, Y) :- e(X, X, X, Y)$

$V_{12} : v_{12}(X, Y) :- e(X, Y, Y, Y)$

$\mathcal{V}_1(\mathcal{D}) = \{(v_{11}(a, a); 1), (v_{12}(a, a); 1)\}$

- $\mathcal{V}_2 = \{V_2\}$ :

$V_2 : v_2(Z_1, Z_2, Z_3) :- e(Z_1, Z_2, Z_2, Z_3)$

$\mathcal{V}_2(\mathcal{D}) = \{(v_2(a, a, a); 1)\}$

- $\mathcal{V}_3 = \{V_3\}$ :

$V_3 : v_3(W_1, W_2, W_3, W_4) :- e(W_1, W_2, W_3, W_4)$

$\mathcal{V}_3(\mathcal{D}) = \{(v_3(a, a, a, a); 1)\}$

## Subexpressions do not suffice

- Consider  $\mathcal{D} = \{(e(a, a, a, a); 1)\}$   
and  $L = \text{size}(\mathcal{V}_2(\mathcal{D})) = 1$  tuple:

- $\mathcal{V}_1 = \{V_{11}, V_{12}\}$ :  
 $V_{11} : v_{11}(X, Y) :- e(X, X, X, Y)$   
 $V_{12} : v_{12}(X, Y) :- e(X, Y, Y, Y)$

$$\mathcal{V}_1(\mathcal{D}) = \{(v_{11}(a, a); 1), (v_{12}(a, a); 1)\} \Rightarrow \text{size}(\mathcal{V}_1(\mathcal{D})) > L$$

- $\mathcal{V}_2 = \{V_2\}$ :  
 $V_2 : v_2(Z_1, Z_2, Z_3) :- e(Z_1, Z_2, Z_2, Z_3)$

$$\mathcal{V}_2(\mathcal{D}) = \{(v_2(a, a, a); 1)\} \Rightarrow \text{size}(\mathcal{V}_2(\mathcal{D})) \leq L$$

- $\mathcal{V}_3 = \{V_3\}$ :  
 $V_3 : v_3(W_1, W_2, W_3, W_4) :- e(W_1, W_2, W_3, W_4)$

$$\mathcal{V}_3(\mathcal{D}) = \{(v_3(a, a, a, a); 1)\} \Rightarrow \text{size}(\mathcal{V}_3(\mathcal{D})) \leq L$$

## Subexpressions do not suffice

- Consider  $\mathcal{D} = \{(e(a, a, a, a); 1), (e(a, b, c, d); 5)\}$   
and  $\mathbf{L} = \text{size}(\mathcal{V}_2(\mathcal{D})) = 1$  tuple:

- $\mathcal{V}_1 = \{V_{11}, V_{12}\}$ :  
 $V_{11} : v_{11}(X, Y) :- e(X, X, X, Y)$   
 $V_{12} : v_{12}(X, Y) :- e(X, Y, Y, Y)$

$$\mathcal{V}_1(\mathcal{D}) = \{(v_{11}(a, a); 1), (v_{12}(a, a); 1)\} \Rightarrow \text{size}(\mathcal{V}_1(\mathcal{D})) > \mathbf{L}$$

- $\mathcal{V}_2 = \{V_2\}$ :  
 $V_2 : v_2(Z_1, Z_2, Z_3) :- e(Z_1, Z_2, Z_2, Z_3)$

$$\mathcal{V}_2(\mathcal{D}) = \{(v_2(a, a, a); 1)\} \Rightarrow \text{size}(\mathcal{V}_2(\mathcal{D})) \leq \mathbf{L}$$

- $\mathcal{V}_3 = \{V_3\}$ :  
 $V_3 : v_3(W_1, W_2, W_3, W_4) :- e(W_1, W_2, W_3, W_4)$

$$\mathcal{V}_3(\mathcal{D}) = \{(v_3(a, a, a, a); 1), (v_3(a, b, c, d); 5)\} \Rightarrow \text{size}(\mathcal{V}_3(\mathcal{D})) > \mathbf{L}$$

## Subexpressions do not suffice

- Consider  $\mathcal{D} = \{(e(a, a, a, a); 1), (e(a, b, c, d); 5)\}$   
and  $L = \text{size}(\mathcal{V}_2(\mathcal{D})) = 1$  tuple:

Is lgg always a better choice than subexpressions?

- $\mathcal{V}_1 = \{V_{11}, V_{12}\}$ :  
 $V_{11} : v_{11}(X, Y) :- e(X, X, X, Y)$   
 $V_{12} : v_{12}(X, Y) :- e(X, Y, Y, Y)$

$\mathcal{V}_1(\mathcal{D}) = \{(v_{11}(a, a); 1), (v_{12}(a, a); 1)\} \Rightarrow \text{size}(\mathcal{V}_1(\mathcal{D})) > L$

- $\mathcal{V}_2 = \{V_2\}$ :  
 $V_2 : v_2(Z_1, Z_2, Z_3) :- e(Z_1, Z_2, Z_2, Z_3)$

$\mathcal{V}_2(\mathcal{D}) = \{(v_2(a, a, a); 1)\} \Rightarrow \text{size}(\mathcal{V}_2(\mathcal{D})) \leq L$

- $\mathcal{V}_3 = \{V_3\}$ :  
 $V_3 : v_3(W_1, W_2, W_3, W_4) :- e(W_1, W_2, W_3, W_4)$

$\mathcal{V}_3(\mathcal{D}) = \{(v_3(a, a, a, a); 1), (v_3(a, b, c, d); 5)\} \Rightarrow \text{size}(\mathcal{V}_3(\mathcal{D})) > L$

## ... but lgg and subexpressions do

- Consider  $\mathcal{D} = \{(e(a, a, a, a); 1), (e(a, b, c, d); 5), (e(b, a, a, d); 8)\}$   
and  $L = \text{size}(\mathcal{V}_2(\mathcal{D})) = 2$  tuples:

Is lgg always a better choice than subexpressions? **NO**

- $\mathcal{V}_1 = \{V_{11}, V_{12}\}$ :  
 $V_{11} : v_{11}(X, Y) :- e(X, X, X, Y)$   
 $V_{12} : v_{12}(X, Y) :- e(X, Y, Y, Y)$

$\mathcal{V}_1(\mathcal{D}) = \{(v_{11}(a, a); 1), (v_{12}(a, a); 1)\} \Rightarrow \text{size}(\mathcal{V}_1(\mathcal{D})) \leq L$

- $\mathcal{V}_2 = \{V_2\}$ :  
 $V_2 : v_2(Z_1, Z_2, Z_3) :- e(Z_1, Z_2, Z_2, Z_3)$

$\mathcal{V}_2(\mathcal{D}) = \{(v_2(a, a, a); 1), (v_2(b, a, d); 8)\} \Rightarrow \text{size}(\mathcal{V}_2(\mathcal{D})) > L$

- $\mathcal{V}_3 = \{V_3\}$ :  
 $V_3 : v_3(W_1, W_2, W_3, W_4) :- e(W_1, W_2, W_3, W_4)$

$\mathcal{V}_3(\mathcal{D}) = \{(v_3(a, a, a, a); 1), (v_3(a, b, c, d); 5), (v_3(b, a, a, d); 8)\} \Rightarrow \text{size}(\mathcal{V}_3(\mathcal{D})) > L$

# Representative Set of Solutions

*Representative* is the *set of solutions* whose viewset can be constructed by considering only:

## Body

- **subexpressions** of queries in the workload
- **least-general-generalizations of subexpressions** of queries in the workload

## Head

- the minimum number of required head variables

## Theorem

The representative set of solutions contains **at least one optimal solution**

The minimum number of head variables is defined using the *shared variable property*

## Can we do better?

Let a view selection problem input  $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, L)$ :

### Proposition

- if every relation in  $\mathcal{S}$  appears at most once in a body of some query in  $\mathcal{Q}$ :
  - sets of queries' subexpressions suffice

### Theorem

- For path-query workloads:
  - path-viewsets suffice

### Proposition

- For chain-query workloads ( $\mathcal{Q}$  is a set of chain queries):
  - chain-viewsets **do not** suffice

# How much do we reduce the search space?

## Path query $P_3$ (of length 3):

$P_3 : q(X, Y) :- \text{edge}(X, Z), \text{edge}(Z, W), \text{edge}(W, Y)$

## Path-viewsets:

- $\mathcal{V}_1 = \{V_3\}$ , where:  
 $V_3 : v_3(X, Y) :- \text{edge}(X, Z), \text{edge}(Z, W), \text{edge}(W, Y)$
- $\mathcal{V}_2 = \{V_2, V_1\}$ , where:  
 $V_2 : v_2(X, Y) :- \text{edge}(X, Z), \text{edge}(Z, Y)$   
 $V_1 : v_1(X, Y) :- \text{edge}(X, Y)$
- $\mathcal{V}_3 = \{V_1\}$ , where:  
 $V_1 : v_{51}(X, Y) :- \text{edge}(X, Y)$

- The above viewsets give equivalent rewritings of  $P_3$
- Consider the (integer) partitions of the length of  $P_3$  (partitions of 3):  
 $\{3\} \quad \{2, 1\} \quad \{1, 1, 1\}$

# How much do we reduce the search space?

## Path query $P_3$ (of length 3):

$P_3 : q(X, Y) :- \text{edge}(X, Z), \text{edge}(Z, W), \text{edge}(W, Y)$

## Path-viewsets:

- $\mathcal{V}_1 = \{V_3\}$ , where:  $\longrightarrow \{3\}$   
 $V_3 : v_3(X, Y) :- \text{edge}(X, Z), \text{edge}(Z, W), \text{edge}(W, Y)$
- $\mathcal{V}_2 = \{V_2, V_1\}$ , where:  $\longrightarrow \{2, 1\}$   
 $V_2 : v_2(X, Y) :- \text{edge}(X, Z), \text{edge}(Z, Y)$   
 $V_1 : v_1(X, Y) :- \text{edge}(X, Y)$
- $\mathcal{V}_3 = \{V_1\}$ , where:  $\longrightarrow \{1, 1, 1\} = \{1\}$   
 $V_1 : v_{51}(X, Y) :- \text{edge}(X, Y)$

- The above viewsets give equivalent rewritings of  $P_3$
- For each viewset there is an integer partition that contains the lengths of its views

# How much do we reduce the search space?

## Path query $P_3$ (of length 3):

$P_3 : q(X, Y) :- \text{edge}(X, Z), \text{edge}(Z, W), \text{edge}(W, Y)$

## Path-viewsets:

- $\mathcal{V}_1 = \{V_3\}$ , where:  $\rightarrow \{3\}$   
 $V_3 : v_3(X, Y) :- \text{edge}(X, Z), \text{edge}(Z, W), \text{edge}(W, Y)$
- $\mathcal{V}_2 = \{V_2, V_1\}$ , where:  $\rightarrow \{2, 1\}$   
 $V_2 : v_2(X, Y) :- \text{edge}(X, Z), \text{edge}(Z, Y)$   
 $V_1 : v_1(X, Y) :- \text{edge}(X, Y)$
- $\mathcal{V}_3 = \{V_1\}$ , where:  $\rightarrow \{1, 1, 1\} = \{1\}$   
 $V_1 : v_{51}(X, Y) :- \text{edge}(X, Y)$

- The above viewsets give equivalent rewritings of  $P_3$
- For each viewset there is an integer partition that contains the lengths of its views
- The number of path-viewsets is exponential to the number of subgoals of queries in the workload

# LGG-VSB Algorithm

- Input:  $\mathcal{P} = \{\mathcal{S}, \mathcal{Q}, \mathcal{D}, L\}$ , where  $\mathcal{Q} = \{Q_1, Q_2, \dots, Q_n\}$
- Output: The representative set of optimal solutions

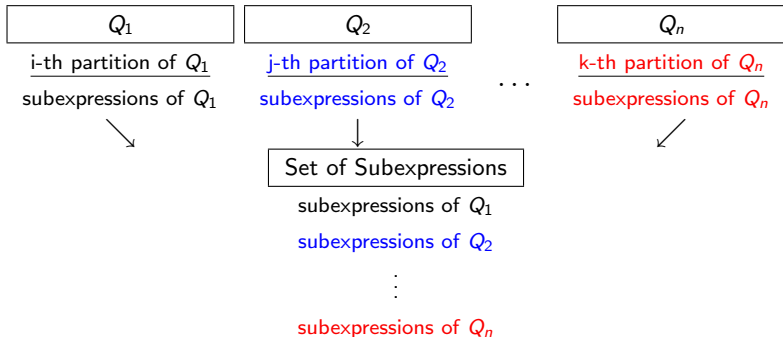
Step 1: Find all partitions of the subgoals of queries in  $\mathcal{Q}$

# LGG-VSB Algorithm

- Input:  $\mathcal{P} = \{\mathcal{S}, \mathcal{Q}, \mathcal{D}, L\}$ , where  $\mathcal{Q} = \{Q_1, Q_2, \dots, Q_n\}$
- Output: The representative set of optimal solutions

Step 1: Find all partitions of the subgoals of queries in  $\mathcal{Q}$

Step 2: Combine the partitions (sets of subexpressions)



No duplicate subexpressions in the set

# LGG-VSB Algorithm

- Input:  $\mathcal{P} = \{\mathcal{S}, \mathcal{Q}, \mathcal{D}, L\}$ , where  $\mathcal{Q} = \{Q_1, Q_2, \dots, Q_n\}$
- Output: The representative set of optimal solutions

Step 1: Find all partitions of the subgoals of queries in  $\mathcal{Q}$

Step 2: Combine the partitions (sets of subexpressions)

Step 3: For every distinct set of subexpressions construct all possible viewsets s.t. every view

- is either a subexpression of query in  $\mathcal{Q}$  or an lgg of subexpressions of queries in  $\mathcal{Q}$ , and
- has the minimum number of head variables

# LGG-VSB Algorithm

- Input:  $\mathcal{P} = \{\mathcal{S}, \mathcal{Q}, \mathcal{D}, L\}$ , where  $\mathcal{Q} = \{Q_1, Q_2, \dots, Q_n\}$
- Output: The representative set of optimal solutions

Step 1: Find all partitions of the subgoals of queries in  $\mathcal{Q}$

Step 2: Combine the partitions (sets of subexpressions)

Step 3: For every distinct set of subexpressions construct all possible viewsets s.t. every view

- is either a subexpression of query in  $\mathcal{Q}$  or an lgg of subexpressions of queries in  $\mathcal{Q}$ , and
- has the minimum number of head variables

Step 4: For every viewset  $\mathcal{V}$  that does not violate the storage constraint  $L$ :

- construct the equivalent rewritings using  $\mathcal{V}$  and compute the cost of them

# LGG-VSB Algorithm

- Input:  $\mathcal{P} = \{\mathcal{S}, \mathcal{Q}, \mathcal{D}, L\}$ , where  $\mathcal{Q} = \{Q_1, Q_2, \dots, Q_n\}$
- Output: The representative set of optimal solutions

Step 1: Find all partitions of the subgoals of queries in  $\mathcal{Q}$

Step 2: Combine the partitions (sets of subexpressions)

Step 3: For every distinct set of subexpressions construct all possible viewsets s.t. every view

- is either a subexpression of query in  $\mathcal{Q}$  or an lgg of subexpressions of queries in  $\mathcal{Q}$ , and
- has the minimum number of head variables

Step 4: For every viewset  $\mathcal{V}$  that does not violate the storage constraint  $L$ :

- construct the equivalent rewritings using  $\mathcal{V}$  and compute the cost of them

Step 5: Compare the costs of the rewritings and output the optimal solutions

# Future Work

- More special cases
  - e.g. Under which restrictions do the subexpressions suffice?
- Optimization of the LGG-VSB algorithm
- More complex classes queries:
  - queries with aggregation
  - CQs with arithmetic comparisons
- Adding more constraints:
  - view maintainance
- View and index selection

**Thank You**