National Engineering School of Mechanic & Aerotechnics
1, avenue Clément Ader - BP 40109 - 86961 Futuroscope cedex – France

La Fragmentation Horizontale Revisitée: Prise en Compte de l'Interaction de Requêtes

Amira KERKAD
amira.kerkad@ensma.fr

Ladjel BELLATRECHE
bellatreche@ensma.fr

Dominique GENIET
dominique.geniet@ensma.fr

EDA
Blois – June 13rd 2013

Data warehousing: Star Schema

- ❑ One fact table (volume)
- ❑ Many dimension tables

Star Join Queries

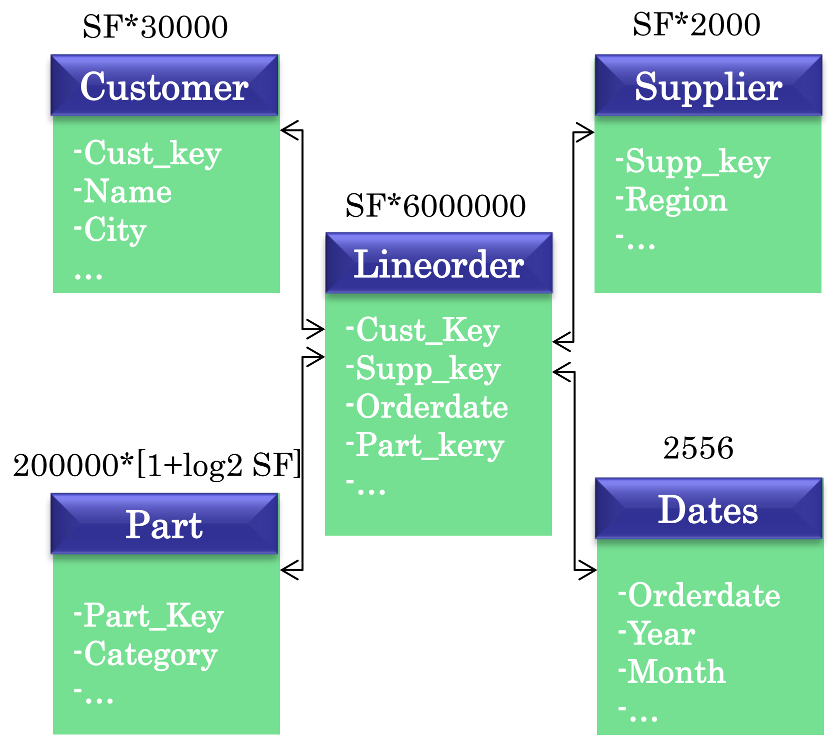
Selections on dimension tables

⇨ Multiple joins (between fact and dimension tables)

⇨ No direct joins between dimensions

Requirements

⇨ Lowering response time



Star Schema

Data Warehousing

Data warehousing: Star Schema

- ❑ One fact table (volume)
- ❑ Many dimension tables

Star Join Queries

Selections on dimension tables

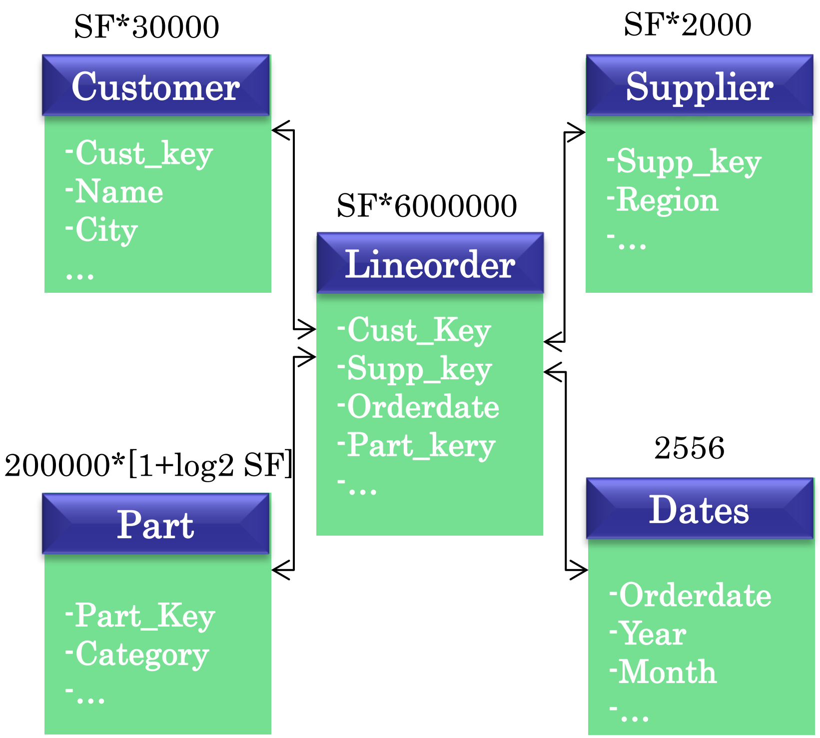
⇨ Multiple joins (between fact and dimension tables)

⇨ No direct joins between dimensions

Requirements

⇨ Lowering response time

➔ **Optimization is crucial**



Star Schema

Data Warehousing

Data warehousing: Star Schema

- ❑ One fact table (volume)
- ❑ Many dimension tables

Star Join Queries

Selections on dimension tables

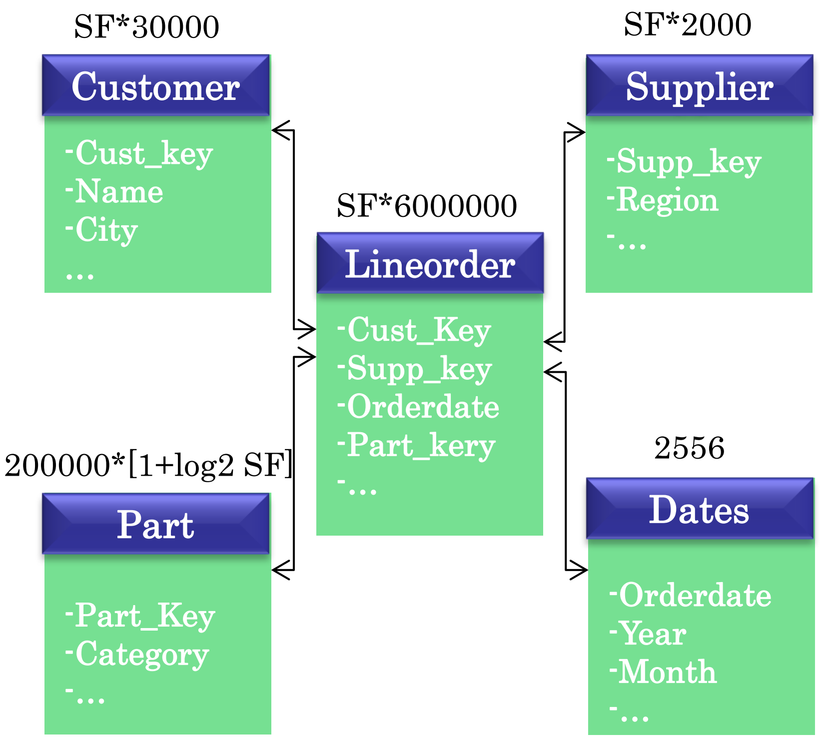
⇨ Multiple joins (between fact and dimension tables)

⇨ No direct joins between dimensions

Requirements

⇨ Lowering response time

➔ **Optimization is crucial**



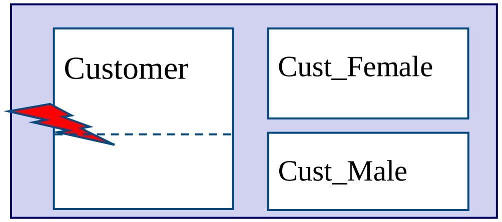
Star Schema

Horizontal Partitioning is well adapted for Star Join Queries

Horizontal Data Partitioning

❑Objective

Decompose table instances into disjoint groups of instances

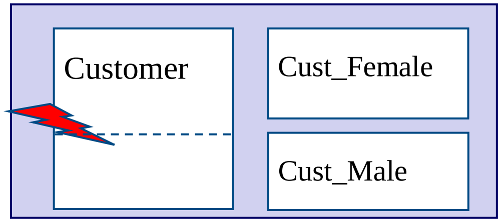


❑Two types :

Primary [Ceri'82]

Derived [Ceri'82]

Horizontal Data Partitioning



❑Objective

Decompose table instances into disjoint groups of instances

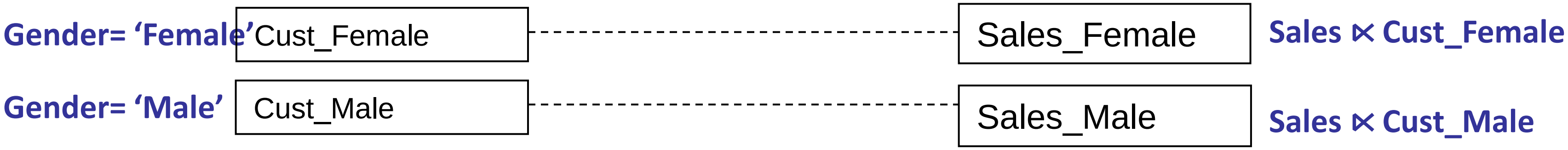
❑Two types :

Primary [Ceri'82]

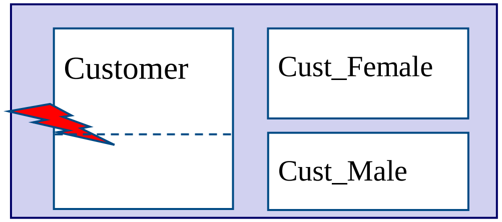
Customer(Id, Name, Gender)

Derived [Ceri'82]

Sales(Cust_Id, Prod_Id, Quantity)



Horizontal Data Partitioning



- ❑Objective

Decompose table instances into disjoint groups of instances

- ❑Two types :

Primary [Ceri'82]

Customer(Id, Name, Gender)

Derived [Ceri'82]

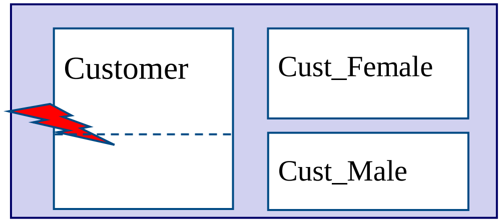
Sales(Cust_Id, Prod_Id, Quantity)

Gender= 'Female' Cust_Female — Sales_Female **Sales $\ltimes$ Cust_Female**

Gender= 'Male' Cust_Male — Sales_Male **Sales $\ltimes$ Cust_Male**

⇨Optimizing selections and joins

Horizontal Data Partitioning



❑Objective

Decompose table instances into disjoint groups of instances

❑Two types :

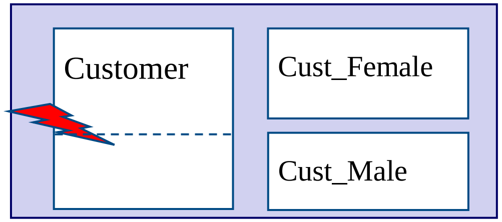
Primary [Ceri'82] — *Customer(Id, Name, Gender)*

Derived [Ceri'82] — *Sales(Cust_Id, Prod_Id, Quantity)*

Gender= 'Female'	Cust_Female	Sales_Female	**Sales ⋉ Cust_Female**
Gender= 'Male'	Cust_Male	Sales_Male	**Sales ⋉ Cust_Male**

⇨Optimizing selections and joins

Horizontal Data Partitioning



❑ Objective

Decompose table instances into disjoint groups of instances

❑ Two types :

Primary [Ceri'82]

Customer(Id, Name, Gender)

Derived [Ceri'82]

Sales(Cust_Id, Prod_Id, Quantity)

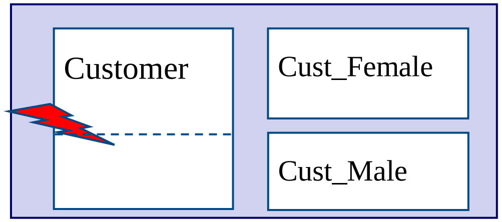
Gender= 'Female' Cust_Female - - - - Sales_Female **Sales ⋉ Cust_Female**

Gender= 'Male' Cust_Male - - - - Sales_Male **Sales ⋉ Cust_Male**

Input :

⇨ Optimizing selections and joins

Horizontal Data Partitioning



- Objective

Decompose table instances into disjoint groups of instances

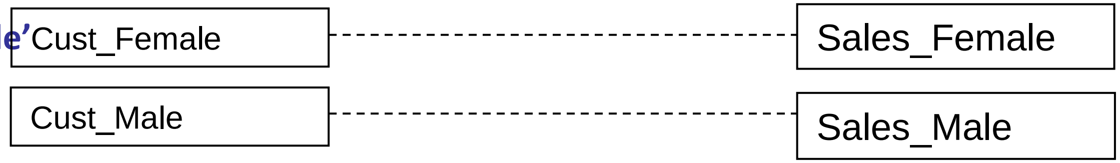
- Two types :

Primary [Ceri'82] — *Customer(Id, Name, Gender)*

Derived [Ceri'82] — *Sales(Cust_Id, Prod_Id, Quantity)*

Gender= 'Female'

Gender= 'Male'



Sales ⋉ Cust_Female

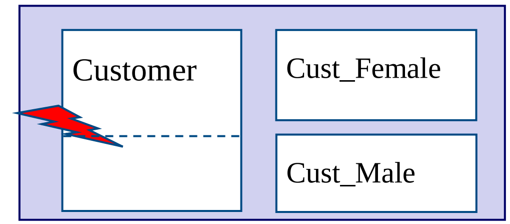
Sales ⋉ Cust_Male

Input :

Data warehouse {Fact, D_1, D_2, …, D_d}

⇨ Optimizing selections and joins

Horizontal Data Partitioning



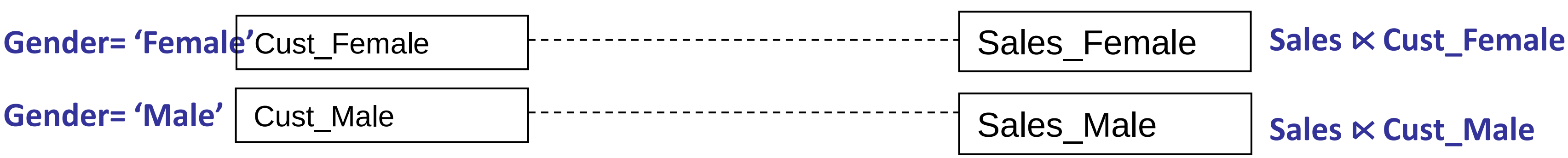
- Objective

Decompose table instances into disjoint groups of instances

- Two types :

Primary [Ceri'82] — *Customer(Id, Name, Gender)*

Derived [Ceri'82] — *Sales(Cust_Id, Prod_Id, Quantity)*



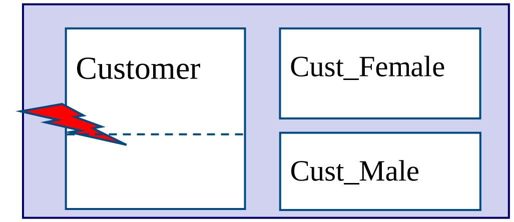
Input :

Data warehouse {Fact, D_1, D_2, …, D_d}
Workload Q

⇨ Optimizing selections and joins

Horizontal Data Partitioning

❑Objective



Decompose table instances into disjoint groups of instances

❑Two types :

Primary [Ceri'82]

Customer(Id, Name, Gender)

Derived [Ceri'82]

Sales(Cust_Id, Prod_Id, Quantity)

Gender= 'Female' Cust_Female — Sales_Female **Sales $\ltimes$ Cust_Female**

Gender= 'Male' Cust_Male — Sales_Male **Sales $\ltimes$ Cust_Male**

⇨Optimizing selections and joins

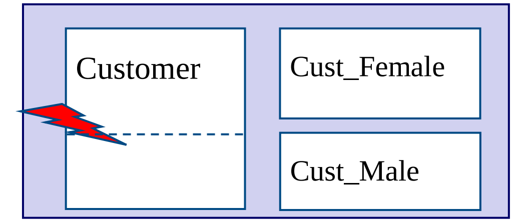
Input :

Data warehouse {Fact, D_1, D_2, …, D_d}
Workload Q
W : threshold (fixed byDBA)

Horizontal Data Partitioning

❑Objective

Decompose table instances into disjoint groups of instances



❑Two types :

Primary [Ceri'82]

Customer(Id, Name, Gender)

Derived [Ceri'82]

Sales(Cust_Id, Prod_Id, Quantity)

Gender= 'Female' Cust_Female ------ Sales_Female **Sales ⋉ Cust_Female**

Gender= 'Male' Cust_Male ------ Sales_Male **Sales ⋉ Cust_Male**

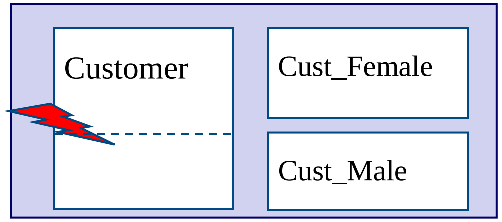
⇨Optimizing selections and joins

Input :

Data warehouse {Fact, D_1, D_2, …, D_d}
Workload Q
W : threshold (fixed byDBA)

Output:

Horizontal Data Partitioning



❑Objective

Decompose table instances into disjoint groups of instances

❑Two types :

Primary [Ceri'82]

Customer(Id, Name, Gender)

Gender= 'Female' Cust_Female

Gender= 'Male' Cust_Male

Derived [Ceri'82]

Sales(Cust_Id, Prod_Id, Quantity)

Sales_Female **Sales $\ltimes$ Cust_Female**

Sales_Male **Sales $\ltimes$ Cust_Male**

⇨Optimizing selections and joins

Input :

Data warehouse {Fact, D_1, D_2, …, D_d}

Workload Q

W : threshold (fixed byDBA)

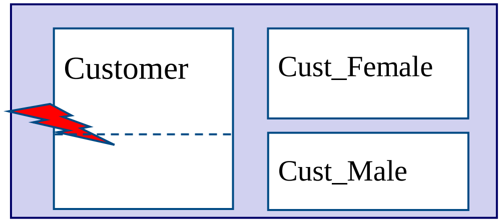
Output:

Set $D' \subseteq D$ of partitioned dimensions

Horizontal Data Partitioning

- Objective

Decompose table instances into disjoint groups of instances



- Two types :

Primary [Ceri'82]

Customer(Id, Name, Gender)

Derived [Ceri'82]

Sales(Cust_Id, Prod_Id, Quantity)

Gender= 'Female' Cust_Female ---- Sales_Female **Sales ⋉ Cust_Female**

Gender= 'Male' Cust_Male ---- Sales_Male **Sales ⋉ Cust_Male**

⇨Optimizing selections and joins

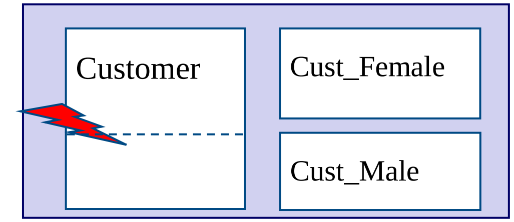
Input :

Data warehouse $\{Fact, D_1, D_2, \ldots, D_d\}$
Workload Q
W : threshold (fixed byDBA)

Output:

Set $D' \subseteq D$ of partitioned dimensions
Set of N fragments of facts $F_1, \ldots, F_N$

Horizontal Data Partitioning



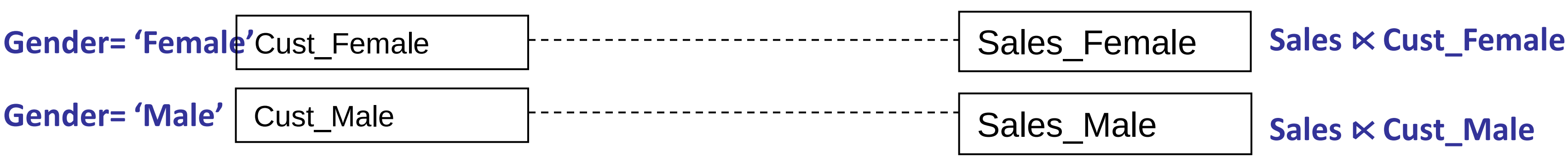
❑ Objective

Decompose table instances into disjoint groups of instances

❑ Two types :

Primary [Ceri'82] — *Customer(Id, Name, Gender)*

Derived [Ceri'82] — *Sales(Cust_Id, Prod_Id, Quantity)*



⇨ Optimizing selections and joins

Input :

Data warehouse $\{Fact, D_1, D_2, \dots, D_d\}$
Workload Q
W : threshold (fixed byDBA)

Output:

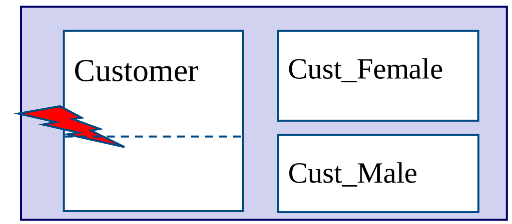
Set $D' \subseteq D$ of partitioned dimensions
Set of N fragments of facts $F_1, \dots, F_N$

Objectives :

Horizontal Data Partitioning

❑Objective

Decompose table instances into disjoint groups of instances



❑Two types :

Primary [Ceri'82]

Customer(Id, Name, Gender)

Gender= 'Female' Cust_Female

Gender= 'Male' Cust_Male

Derived [Ceri'82]

Sales(Cust_Id, Prod_Id, Quantity)

Sales_Female — Sales ⋉ Cust_Female

Sales_Male — Sales ⋉ Cust_Male

⇨Optimizing selections and joins

Input :

Data warehouse $\{Fact, D_1, D_2, \ldots, D_d\}$

Workload Q

W : threshold (fixed byDBA)

Output:

Set $D' \subseteq D$ of partitioned dimensions

Set of N fragments of facts $F_1, \ldots, F_N$

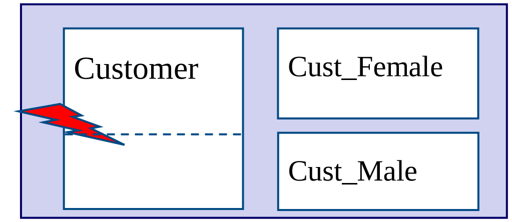
Objectives :

Lowering execution cost of Q

Horizontal Data Partitioning

- Objective

Decompose table instances into disjoint groups of instances



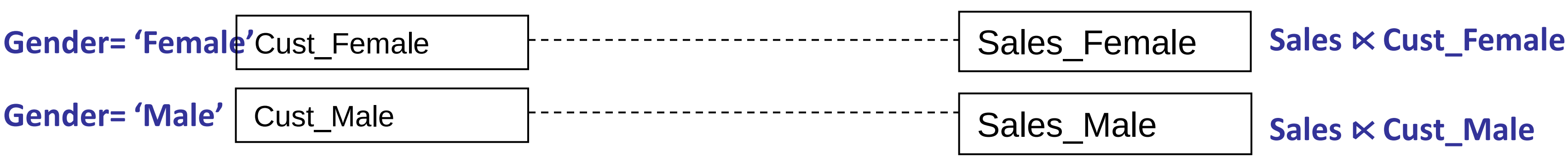
- Two types :

Primary [Ceri'82]

Customer(Id, Name, Gender)

Derived [Ceri'82]

Sales(Cust_Id, Prod_Id, Quantity)



⇨Optimizing selections and joins

Input :

Data warehouse {Fact, D_1, D_2, …, D_d}
Workload Q
W : threshold (fixed byDBA)

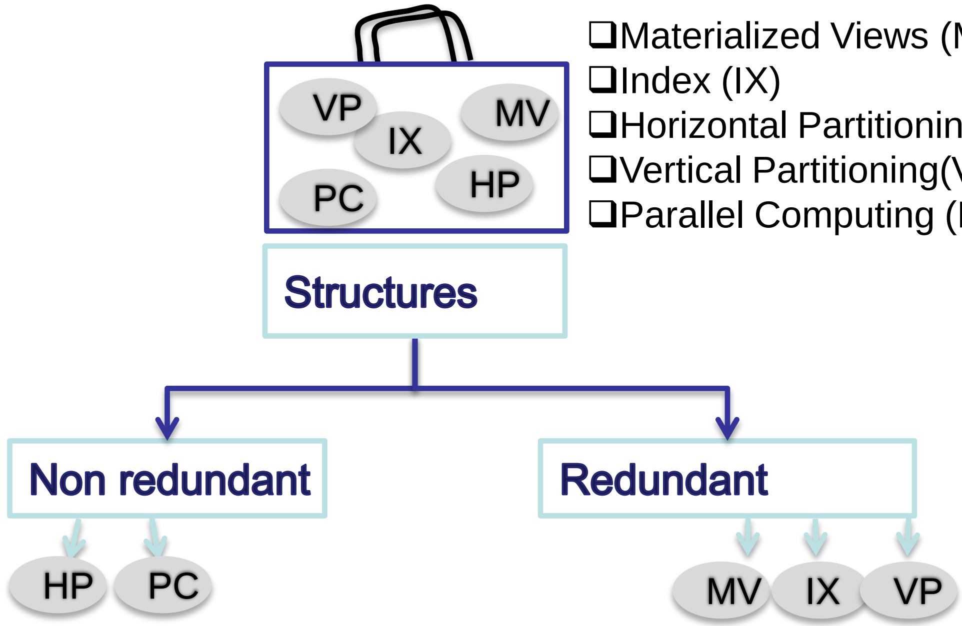
Output:

Set D'$\subseteq$ D of partitioned dimensions
Set of N fragments of facts F_1, …, F_N

Objectives :

Lowering execution cost of Q
$N \leq W$

Classification of Optimization Techniques



- Materialized Views (MV)
- Index (IX)
- Horizontal Partitioning (HP)
- Vertical Partitioning(VP)
- Parallel Computing (PC)

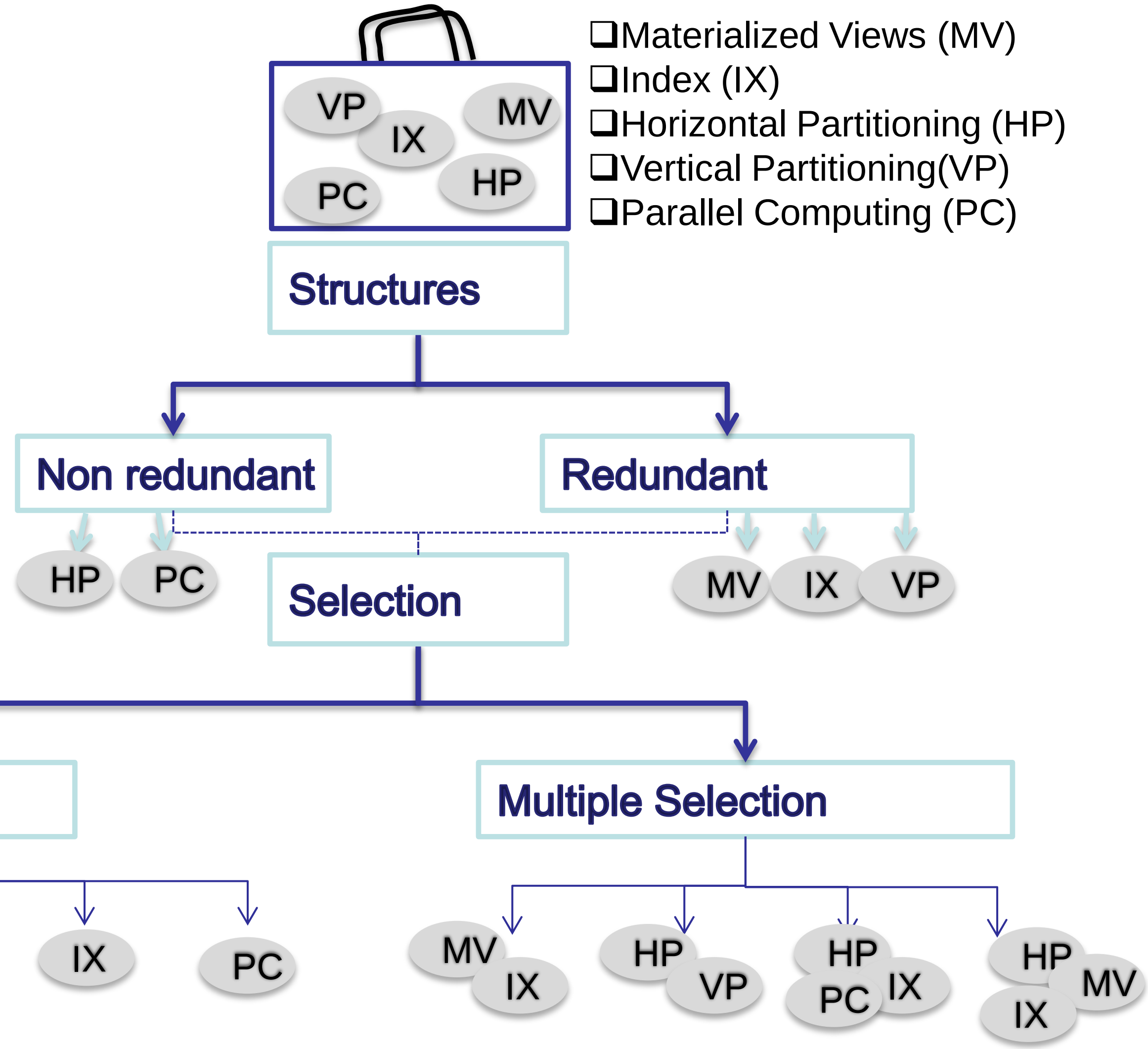
First Classification
[DEXA'07]

Classification of Optimization Techniques

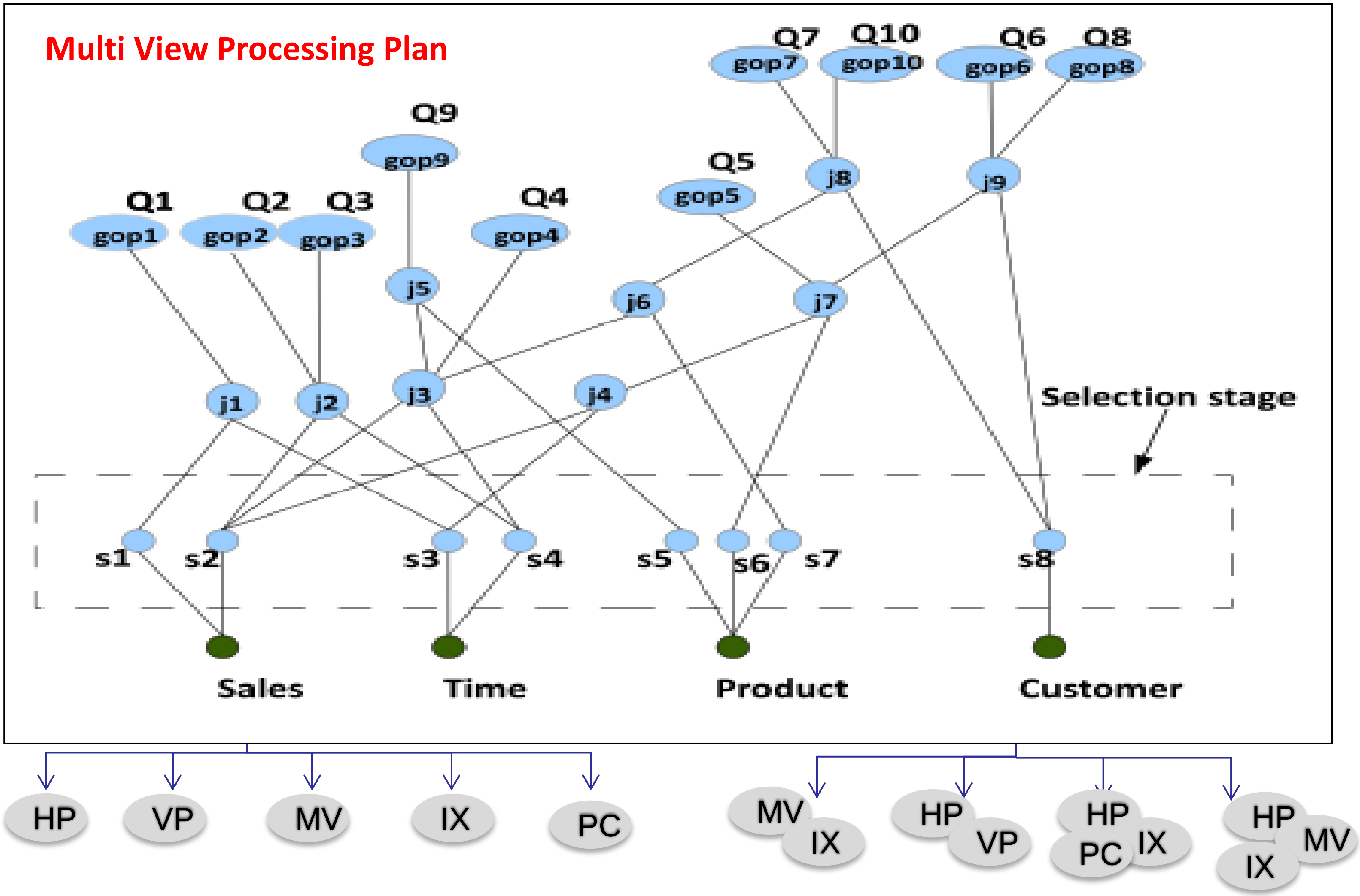
- ❑Materialized Views (MV)
- ❑Index (IX)
- ❑Horizontal Partitioning (HP)
- ❑Vertical Partitioning(VP)
- ❑Parallel Computing (PC)

First Classification
[DEXA'07]

Second Classification
[DAWAK'08]

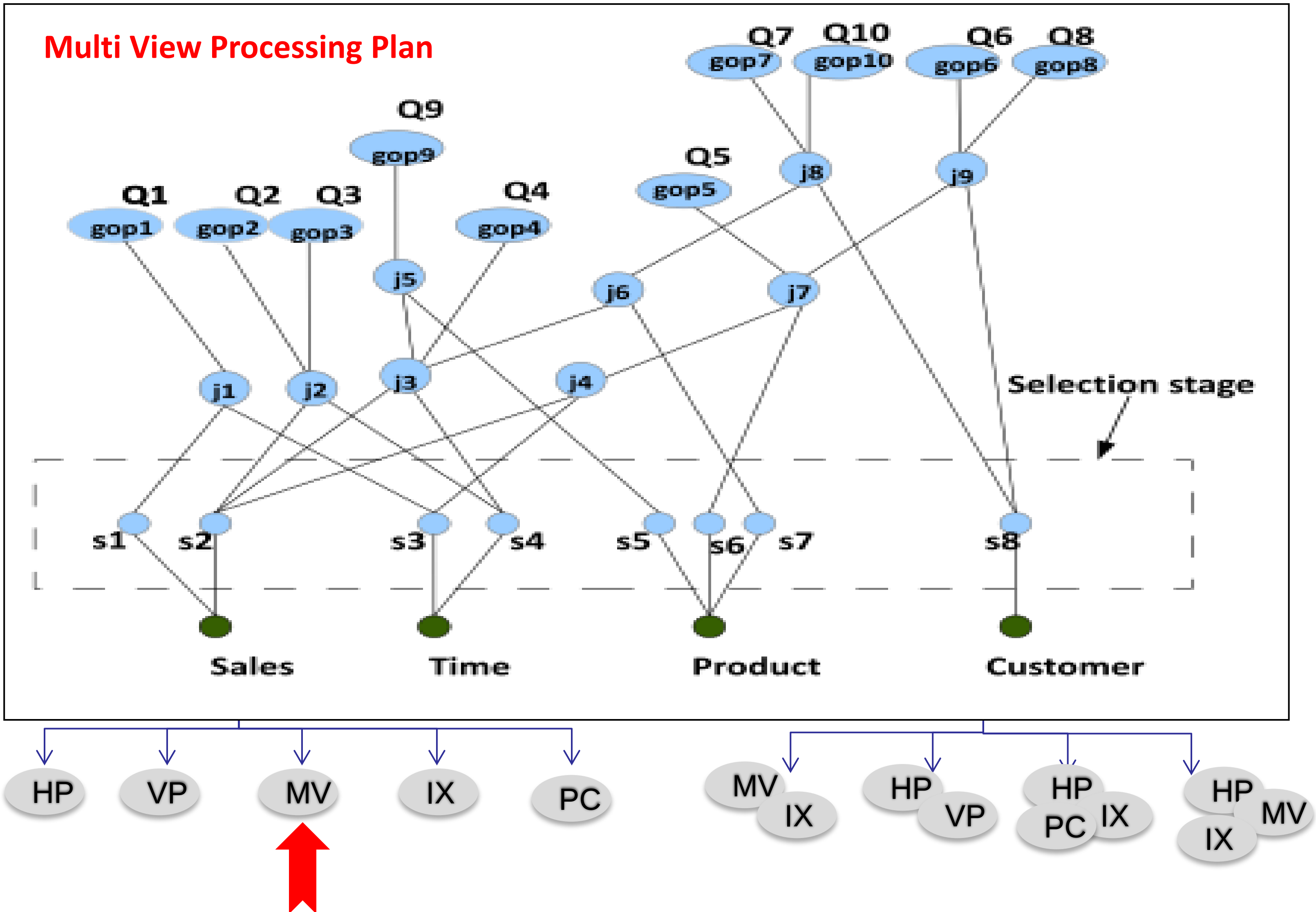


Classification of Optimization Techniques



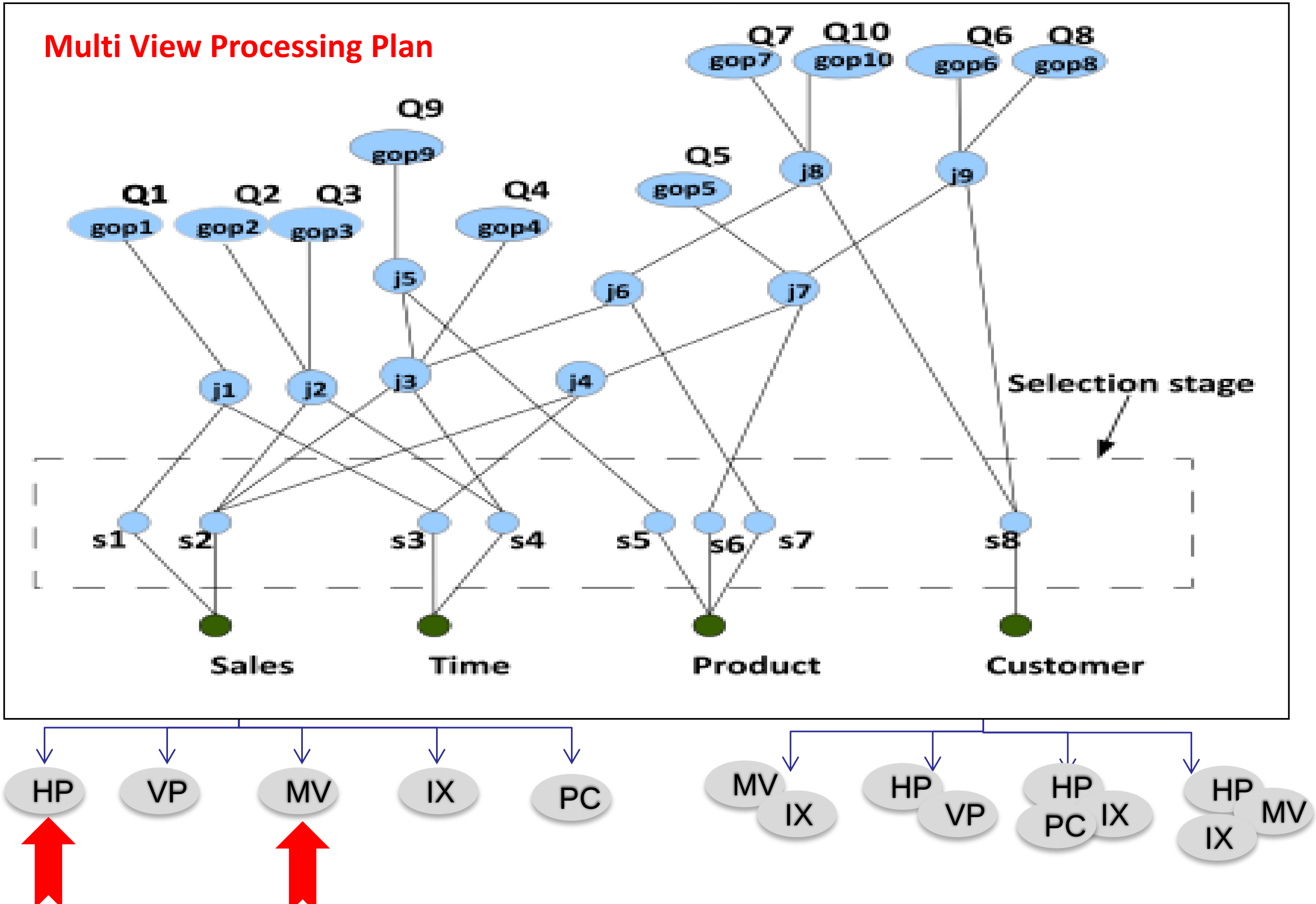
Query Interaction [Sellis'88]

Classification of Optimization Techniques



Query Interaction [Sellis'88]

Classification of Optimization Techniques



Query Interaction [Sellis'88]

Horizontal Data Partitioning Evolution

Classification of HDP approaches

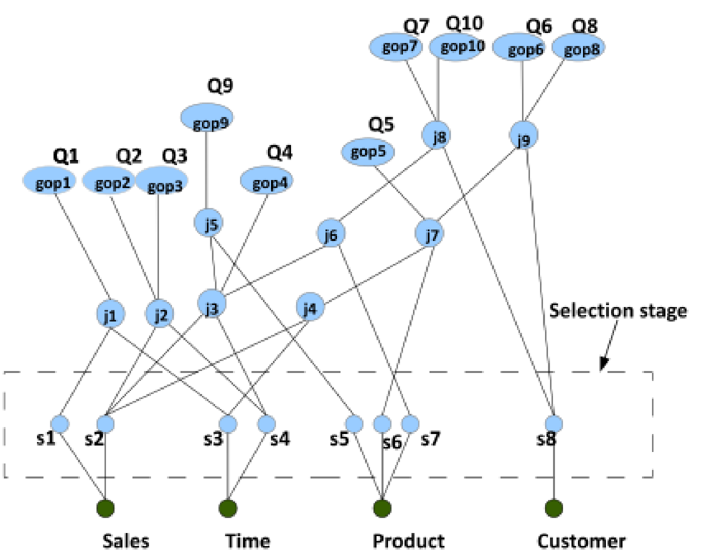
- Horizontal Data Partitioning
  - Unconstrained
    1. Affinity [Karlapalem, 1996]
    2. Minterms [Ozsu and Valduriez, 1999]
  - Constrained
    1. Cost Model [Bellatreche *et al.* 2000]
    2. Data Mining [Mahboubi and Darmont, 2009]

Horizontal Data Partitioning Evolution

Classification of HDP approaches

- Horizontal Data Partitioning
  - Unconstrained
    1. Affinity [Karlapalem, 1996]
    2. Minterms [Ozsu and Valduriez, 1999]
  - Constrained
    1. Cost Model [Bellatreche *et al.* 2000]
    2. Data Mining [Mahboubi and Darmont, 2009]

Query Interaction

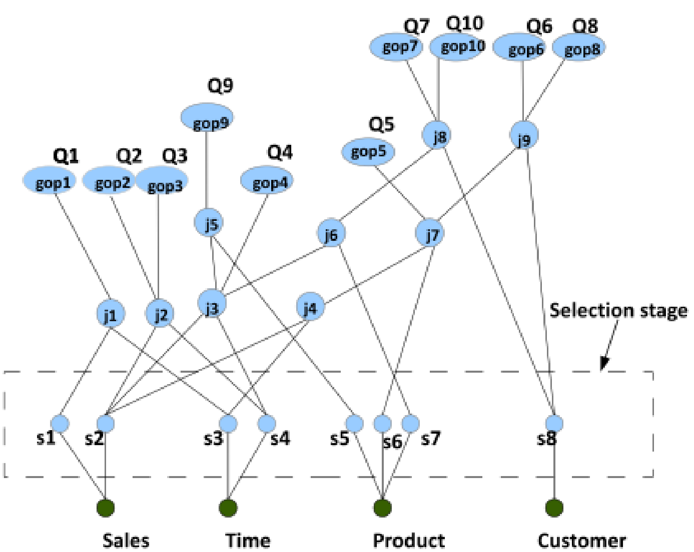


Horizontal Data Partitioning Evolution

Classification of HDP approaches

- Horizontal Data Partitioning
  - Ignoring Interaction
    - Algorithmic Level
      - Unconstrained
        1. Affinity [Karlapalem, 1996]
        2. Minterms [Ozsu and Valduriez, 1999]
      - Constrained
        1. Cost Model [Bellatreche *et al.* 2000]
        2. Data Mining [Mahboubi and Darmont, 2009]
  - Intercation-Aware
    1. Cost Model Level [Bellatreche *et al.* 2000]
    2. Algorithmic Level ?
    3. Data Structure Level ?

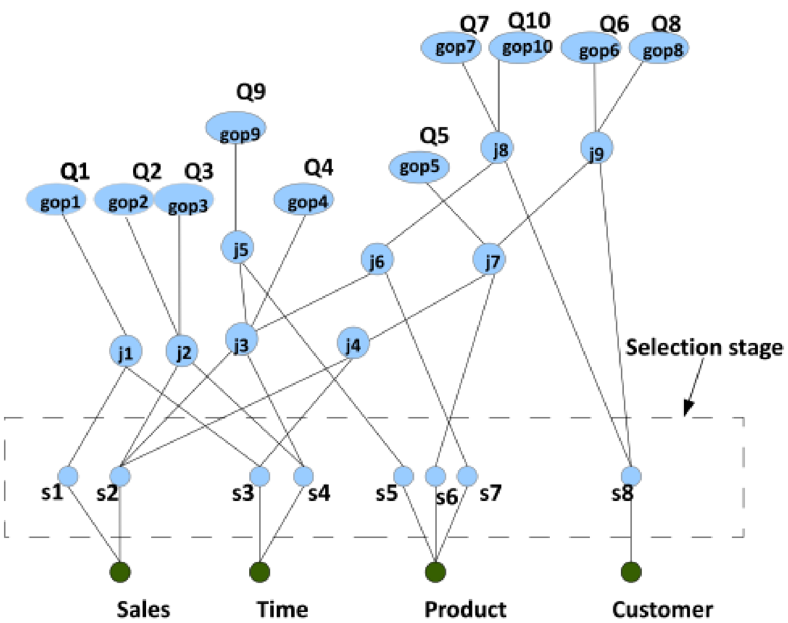
Query Interaction



Outlines

- **Motivating example**
- **Algebra**
- **EQHDP**
- **Experiments**
- **Conclusion & perspectives**

Motivating example

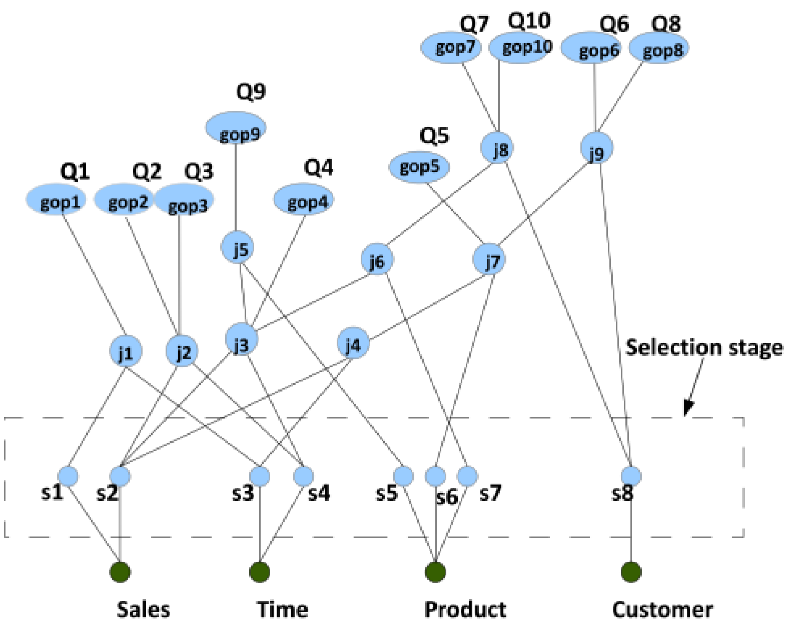


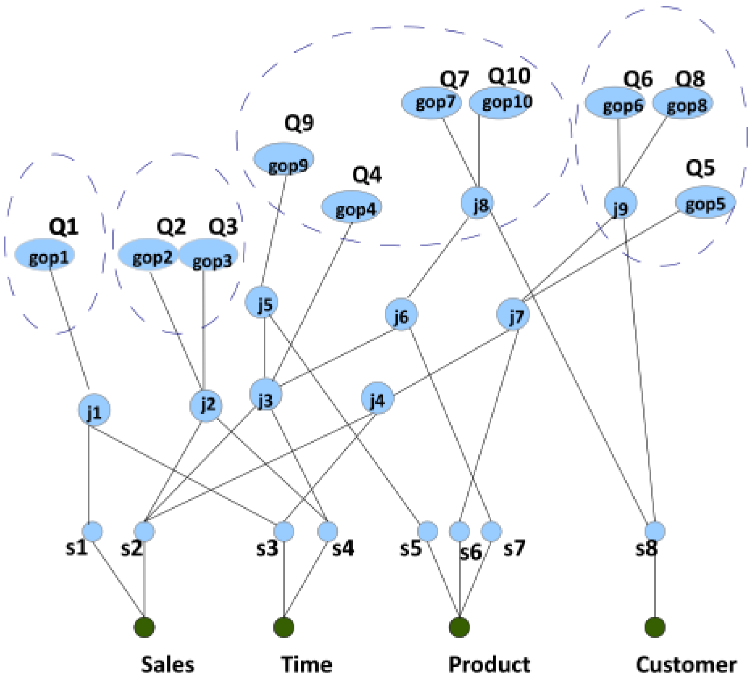
First join very expensive and needs optimization

➔ Selection stage (HDP)

➔ Spread benefit through the workload…

Motivating example



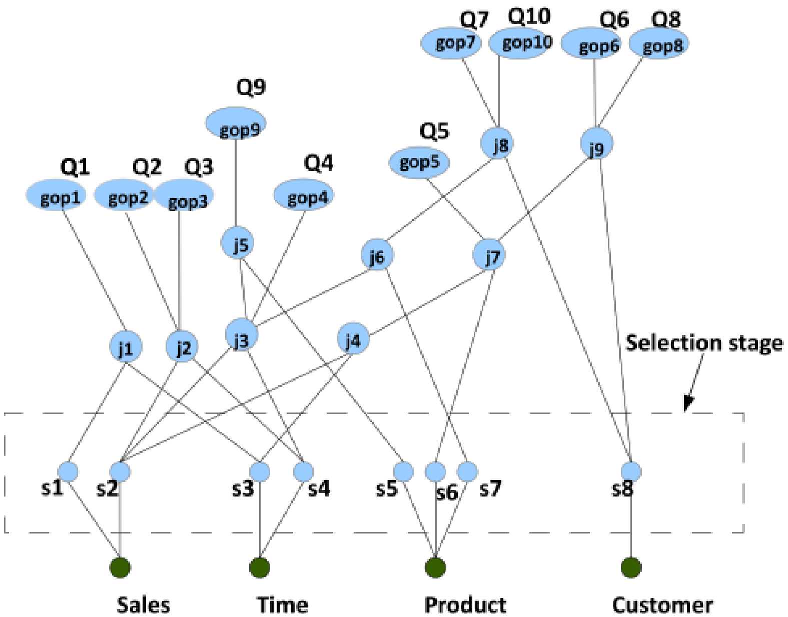


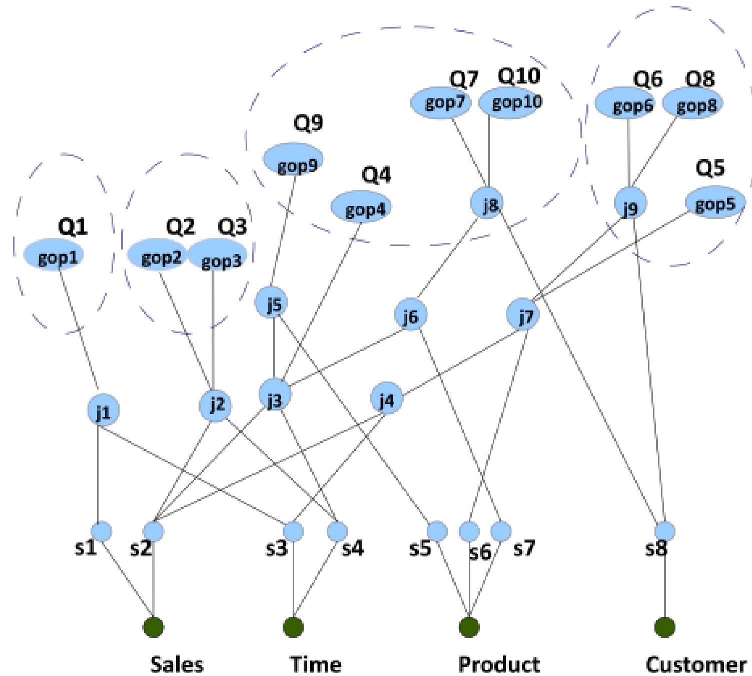
First join very expensive and needs optimization

➔ Selection stage (HDP)

➔ Spread benefit through the workload…

Motivating example





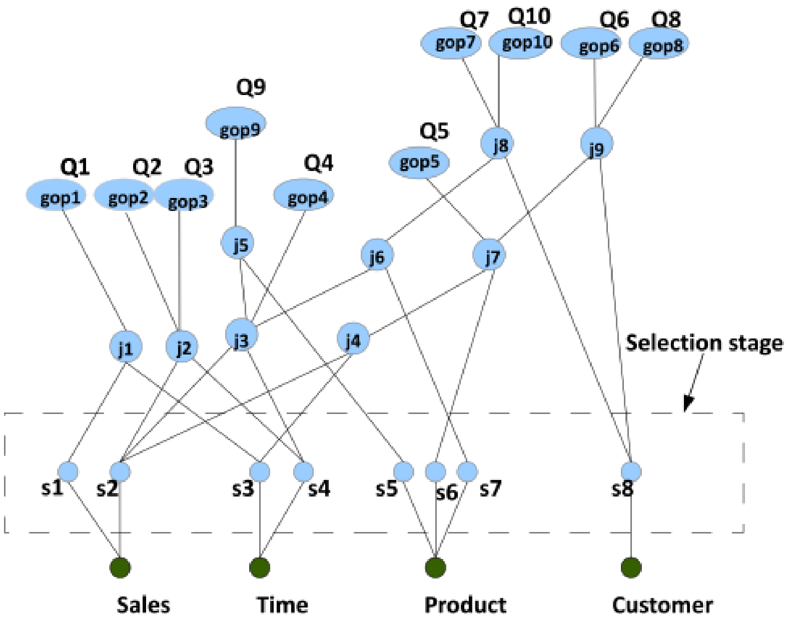
First join very expensive and needs optimization

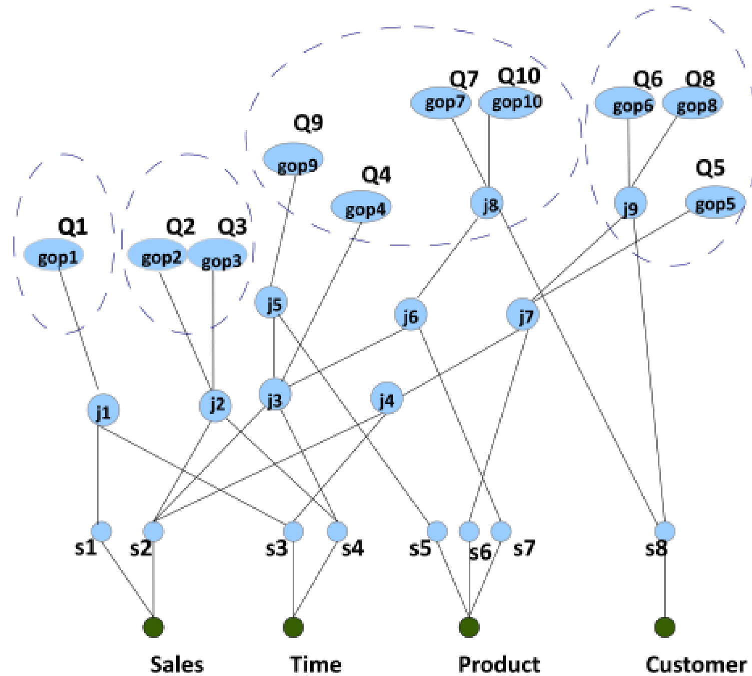
➔ Selection stage (HDP)

➔ Spread benefit through the workload…

1) Group queries
2) Elect one query in each group
3) Steer HDP process

Motivating example



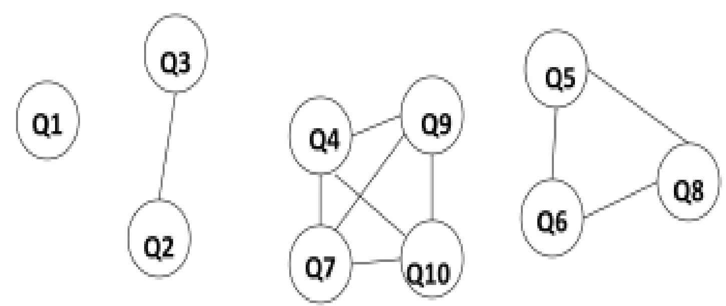


First join very expensive and needs optimization

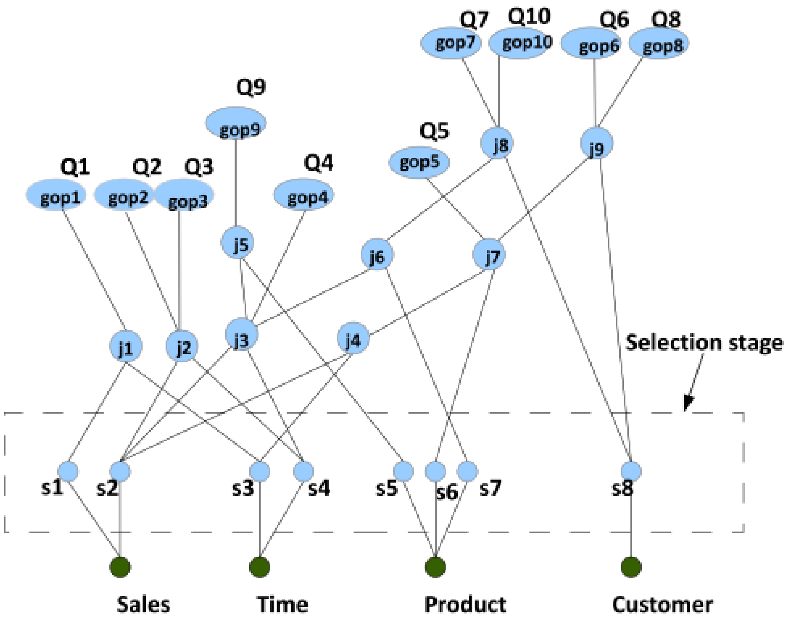
➔ Selection stage (HDP)

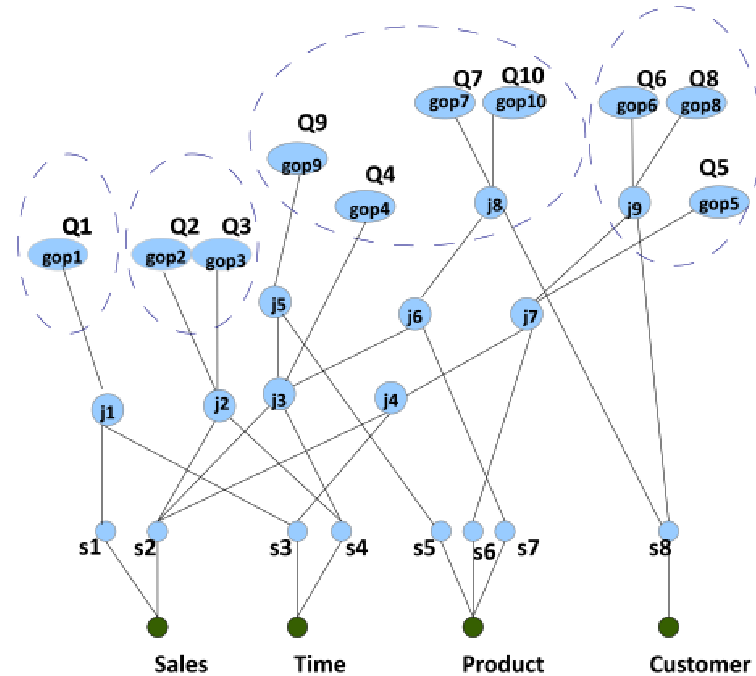
➔ Spread benefit through the workload…

1) Group queries
2) Elect one query in each group
3) Steer HDP process



Motivating example



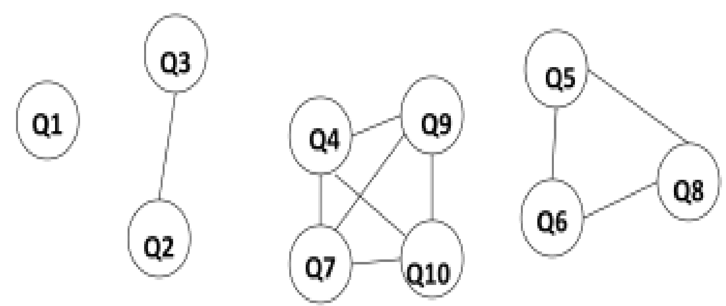


First join very expensive and needs optimization

➔ Selection stage (HDP)

➔ Spread benefit through the workload…

1) Group queries
2) Elect one query in each group
3) Steer HDP process



➔ How to elect query (criterion)?

➔ Algebra to handle generate HDP schema?

➔ Prune predicates and steer HDP by query interaction?

Algebra

Algebra allows to generate an encoding and a HDP schema

Generating incremental encoding

Att1	val1	else

→

Att1	val1	val2	Else

Horizontal Split

Att1	val1	val2	Else

→

Att1	val1	val2	Else
Att2	val1	else	

Vertical Split

Generating HDP schema

Quantity	1	1	1	
Season	1	2	1	3
Type	1	2	3	
Color	1	2		
Gender	1	1		

→

Quantity	1	1	1	
Season	1	2	4	3
Type	1	2	2	
Color	1	2		
Gender	1	2		

Merge

Split

Algebra

Algebra allows to generate an encoding and a HDP schema

Generating incremental encoding

Att1	val1	else	

→

Att1	val1	val2	Else

Horizontal Split

Att1	val1	val2	Else

→

Att1	val1	val2	Else
Att2	val1	else	

Vertical Split

Generating HDP schema

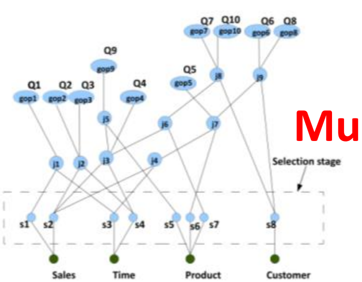
Quantity	1	1	1	
Season	1	2	1	3
Type	1	2	3	
Color	1	2		
Gender	1	1		

→

Quantity	1	1	1	
Season	1	2	4	3
Type	1	2	2	
Color	1	2		
Gender	1	2		

Merge

Split



Multiple View Processing Plan

Algebra

Algebra allows to generate an encoding and a HDP schema

Generating incremental encoding

Att1	val1	else	→	Att1	val1	val2	Else

Horizontal Split

Att1	val1	val2	Else	→ Att1	val1	val2	Else
				Att2	val1	else	

Vertical Split

Generating HDP schema

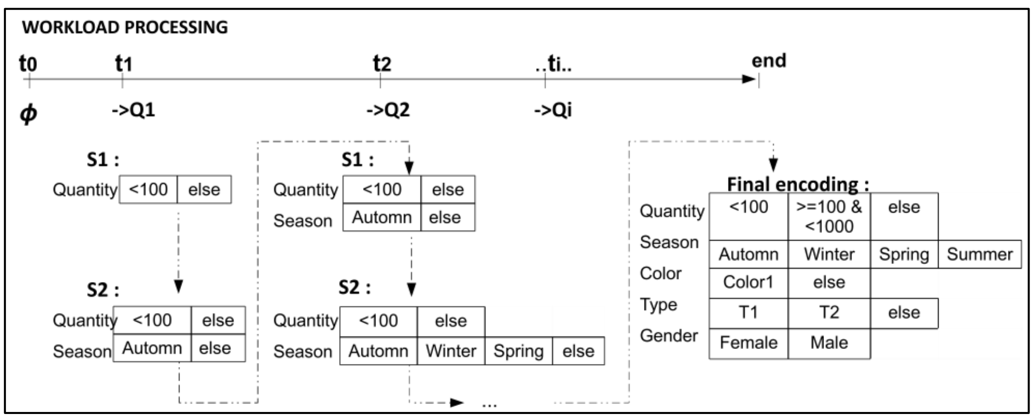
Quantity	1	1	1	
Season	1	2	1	3
Type	1	2	3	
Color	1	2		
Gender	1	1		

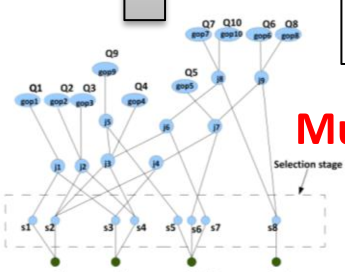
→

Quantity	1	1	1	
Season	1	2	4	3
Type	1	2	2	
Color	1	2		
Gender	1	2		

Merge

Split





Multiple View Processing Plan

Algebra

Algebra allows to generate an encoding and a HDP schema

Generating incremental encoding

Att1	val1	else

→

Att1	val1	val2	Else

Horizontal Split

Att1	val1	val2	Else

→

Att1	val1	val2	Else
Att2	val1	else	

Vertical Split

Generating HDP schema

Quantity	1	1	1	
Season	1	2	1	3
Type	1	2	3	
Color	1	2		
Gender	1	1		

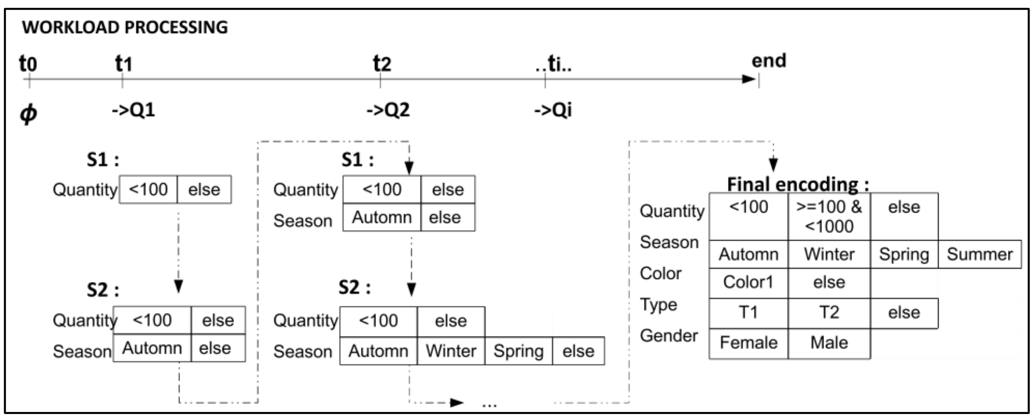
→

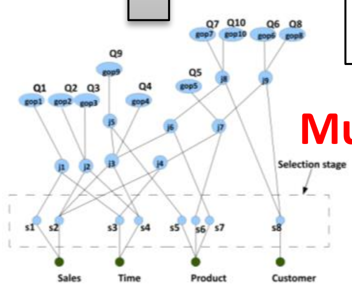
Quantity	1	1	1	
Season	1	2	4	3
Type	1	2	2	
Color	1	2		
Gender	1	2		

Merge

Split

Encoding Predicates





Multiple View Processing Plan

Algebra

Algebra allows to generate an encoding and a HDP schema

Generating incremental encoding

Att1	val1	else	

→

Att1	val1	val2	Else

Horizontal Split

Att1	val1	val2	Else

→

Att1	val1	val2	Else
Att2	val1	else	

Vertical Split

Generating HDP schema

Quantity	1	1	1	
Season	1	2	1	3
Type	1	2	3	
Color	1	2		
Gender	1	1		

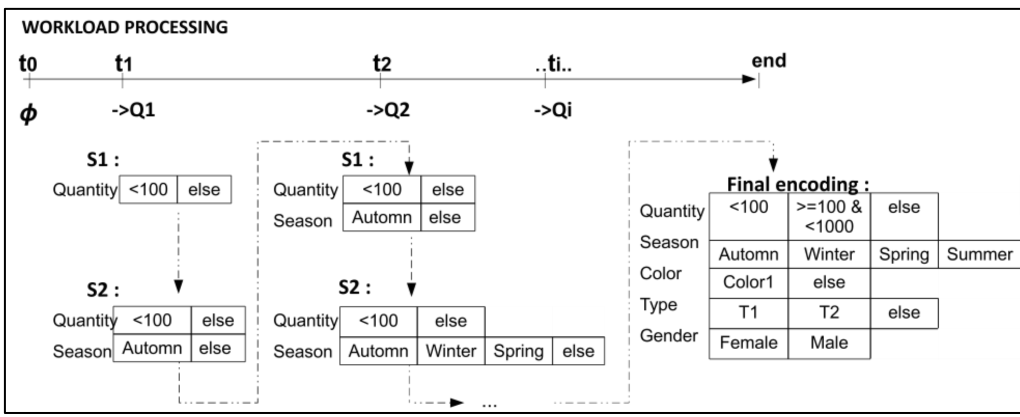
→

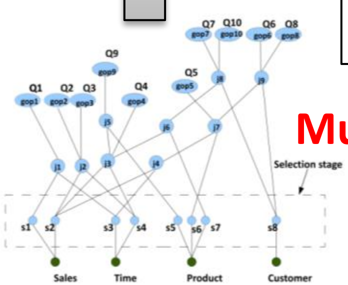
Quantity	1	1	1	
Season	1	2	4	3
Type	1	2	2	
Color	1	2		
Gender	1	2		

Merge

Split

Encoding Predicates





Multiple View Processing Plan

Quantity	1	1	1	
Season	1	2	1	3
Type	1	2	3	
Color	1	2		
Gender	1	1		

→

Quantity	1	1	1	
Season	1	2	4	3
Type	1	2	2	
Color	1	2		
Gender	1	2		

Algebra

Algebra allows to generate an encoding and a HDP schema

Generating incremental encoding

Att1	val1	else

→

Att1	val1	val2	Else

Horizontal Split

Att1	val1	val2	Else

→

Att1	val1	val2	Else
Att2	val1	else	

Vertical Split

Generating HDP schema

Quantity	1	1	1	
Season	1	2	1	3
Type	1	2	3	
Color	1	2		
Gender	1	1		

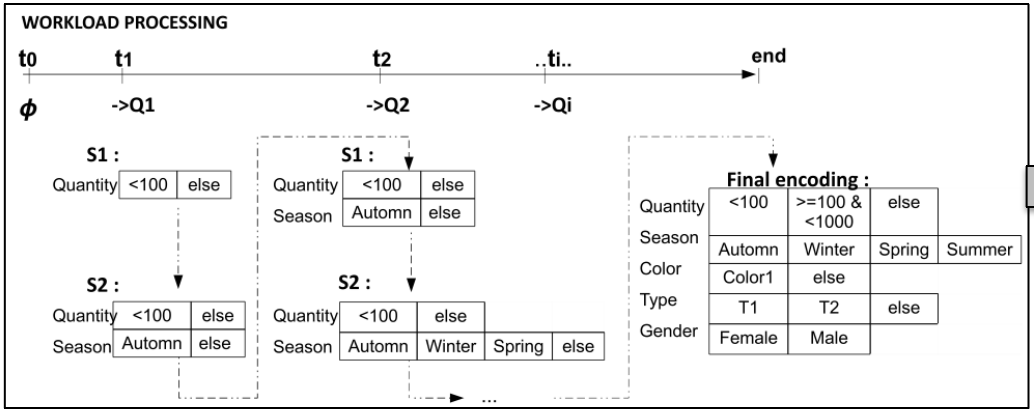
→

Quantity	1	1	1	
Season	1	2	4	3
Type	1	2	2	
Color	1	2		
Gender	1	2		

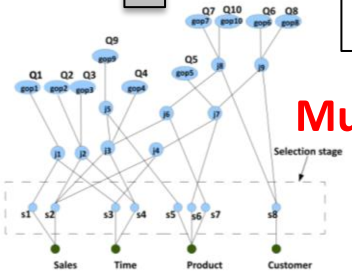
Merge

Split

Encoding Predicates



Generating HDP schema



Multiple View Processing Plan

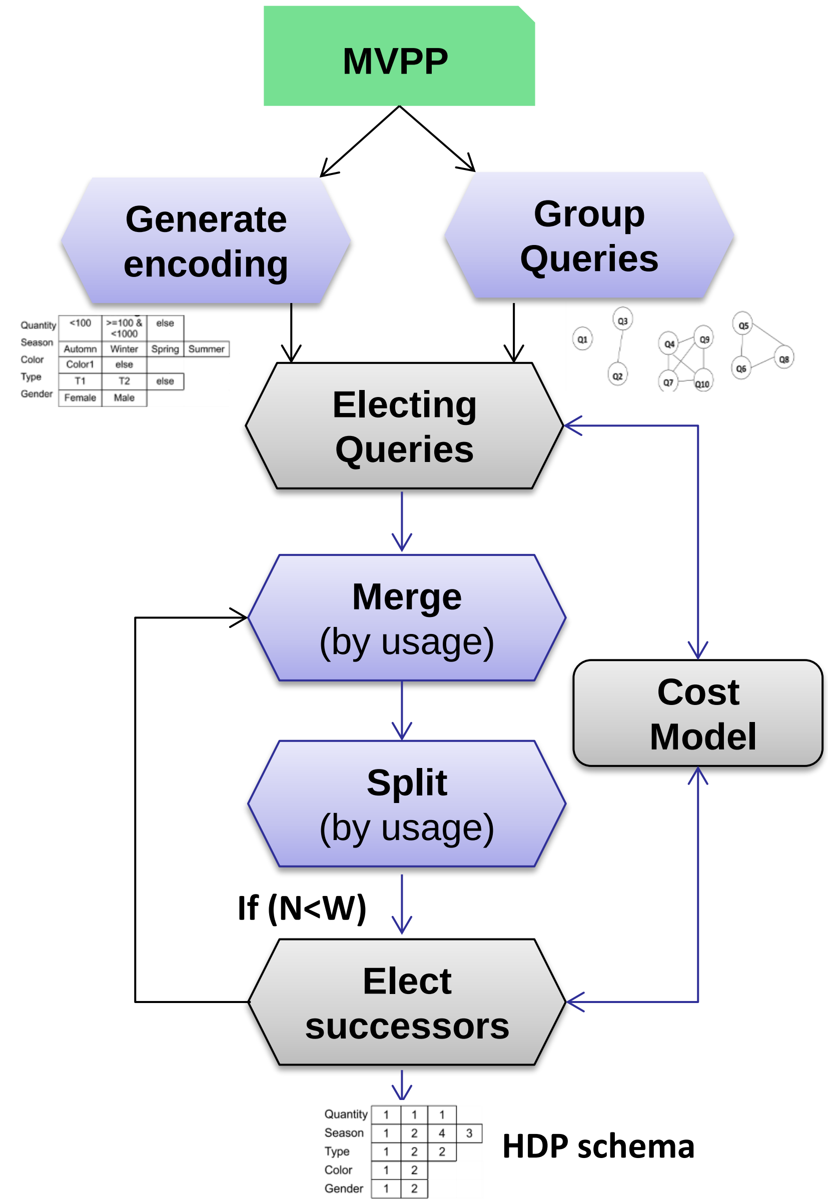
Quantity	1	1	1	
Season	1	2	1	3
Type	1	2	3	
Color	1	2		
Gender	1	1		

→

Quantity	1	1	1	
Season	1	2	4	3
Type	1	2	2	
Color	1	2		
Gender	1	2		

Electing Queries for HDP

```
Algorithm                EQHDP
generate_encoding();
EQA();
e := 1;
split_all();
E := elected(e)
while E not empty do
  prune_encoding(E);
  sort(E);
  S := required_attributes(E);
  usage(S);
  sort_attributes(S)
  for all a ∈ S do
    for all sd ∈ SubDomains(a) do
      if ((U(sd) = 0)and(N < W)) then
        merge(sd, P_0);
      end if
      if ((U(sd) = k)and(N < W)) then
        merge(sd, P_k);
      end if
    end for
  end for
  while k > 0 do
    for all a ∈ S do
      for all sd ∈ SubDomains(a) do
        if (N < W) then
          merge(sd, P_0);
        else
          merge(sd, P_k);
        end if
      end for
    end for
    k := k − 1;
  end while
  split_disjoint();
  e := e + 1;
  E := elected(e);
end while
```

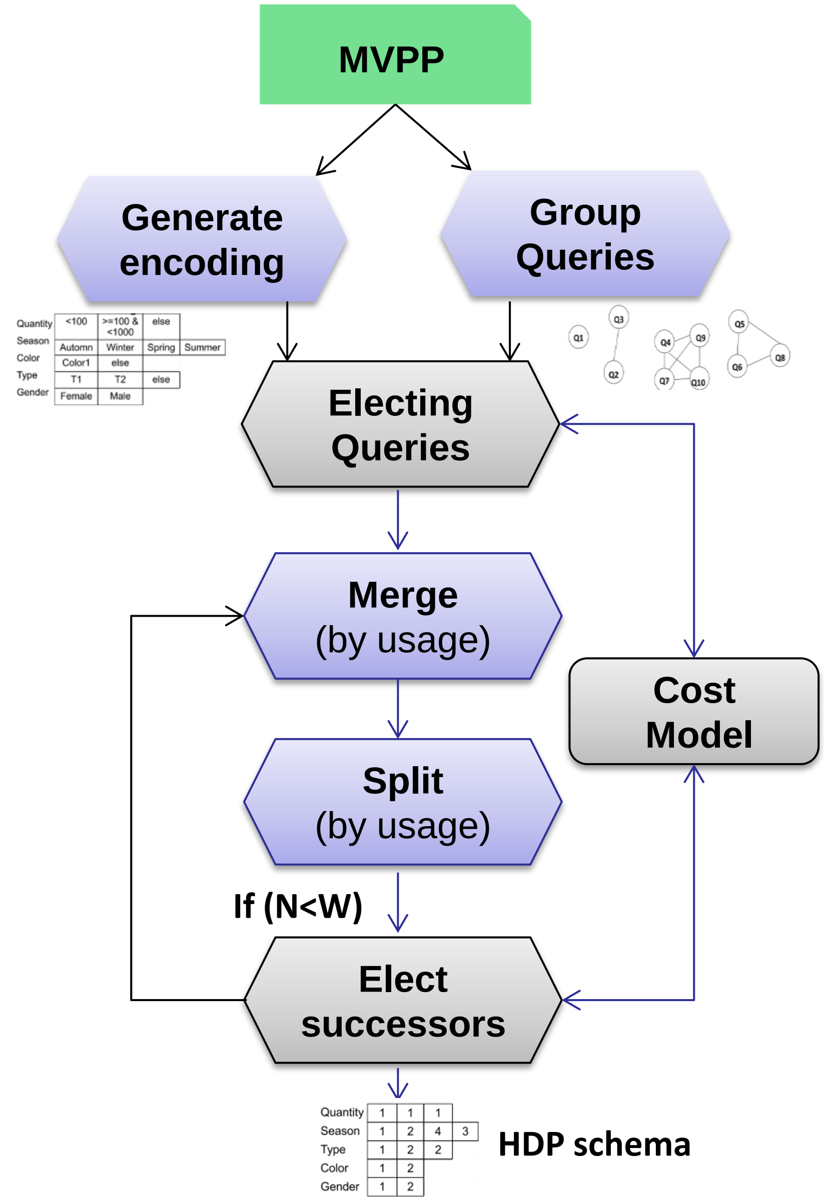


Electing Queries for HDP

Algorithm EQHDP

```
generate_encoding();
EQA();
e := 1;
split_all();
E := elected(e)
while E not empty do
  prune_encoding(E);
  sort(E);
  S := required_attributes(E);
  usage(S);
  sort_attributes(S)
  for all a ∈ S do
    for all sd ∈ SubDomains(a) do
      if ( (U(sd) = 0)and(N < W)) then
        merge(sd, P_0);
      end if
      if ( (U(sd) = k)and(N < W)) then
        merge(sd, P_k);
      end if
    end for
  end for
  while k > 0 do
    for all a ∈ S do
      for all sd ∈ SubDomains(a) do
        if ( N < W) then
          merge(sd, P_0);
        else
          merge(sd, P_k);
        end if
      end for
    end for
    k := k − 1;
  end while
  split_disjoint();
  e := e + 1;
  E := elected(e);
end while
```

Grouping Queries & Generating Encoding

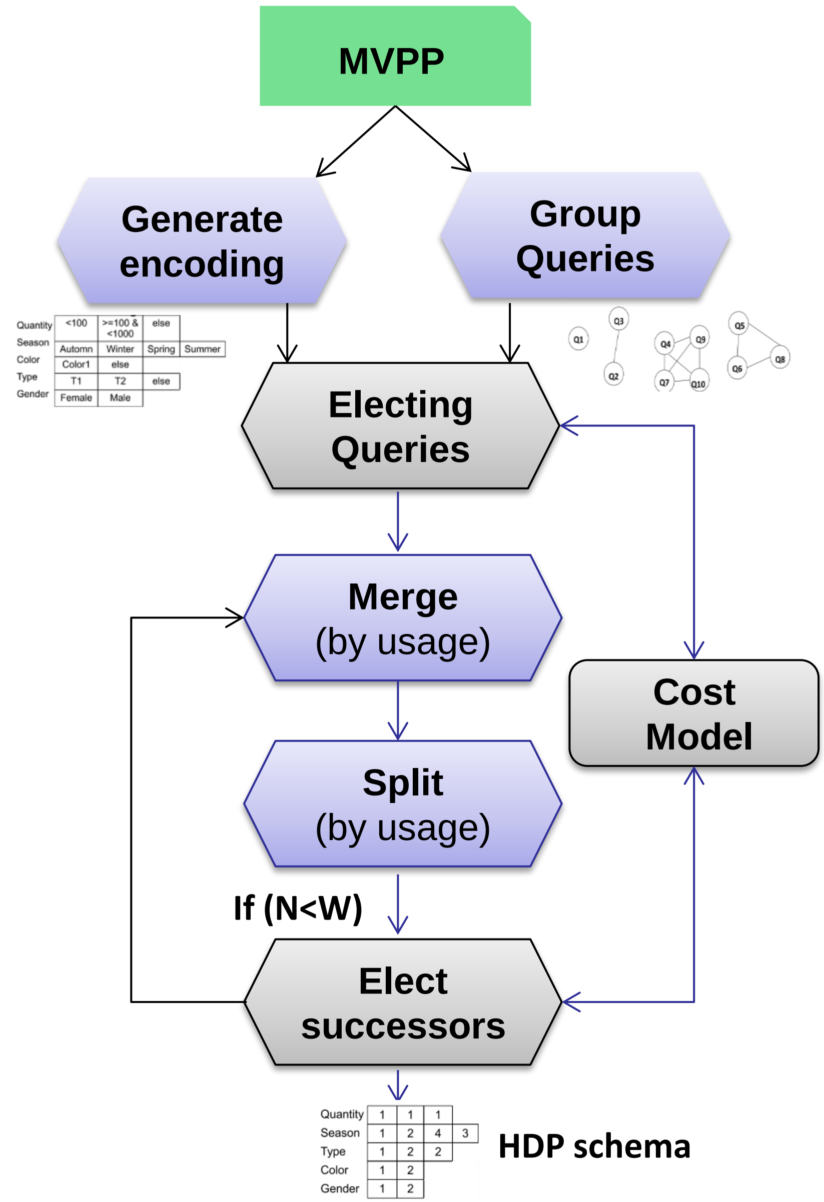


Electing Queries for HDP

```
Algorithm EQHDP
generate_encoding();
EQA();
e := 1;
split_all();
E := elected(e)
while E not empty do
   prune_encoding(E);
   sort(E);
   S := required_attributes(E);
   usage(S);
   sort_attributes(S)
   for all a ∈ S do
      for all sd ∈ SubDomains(a) do
         if ((U(sd) = 0)and(N < W)) then
            merge(sd, P_0);
         end if
         if ((U(sd) = k)and(N < W)) then
            merge(sd, P_k);
         end if
      end for
   end for
   while k > 0 do
      for all a ∈ S do
         for all sd ∈ SubDomains(a) do
            if (N < W) then
               merge(sd, P_0);
            else
               merge(sd, P_k);
            end if
         end for
      end for
      k := k − 1;
   end while
   split_disjoint();
   e := e + 1;
   E := elected(e);
end while
```

Grouping Queries & Generating Encoding

Elect queries & Prune encoding



Electing Queries for HDP

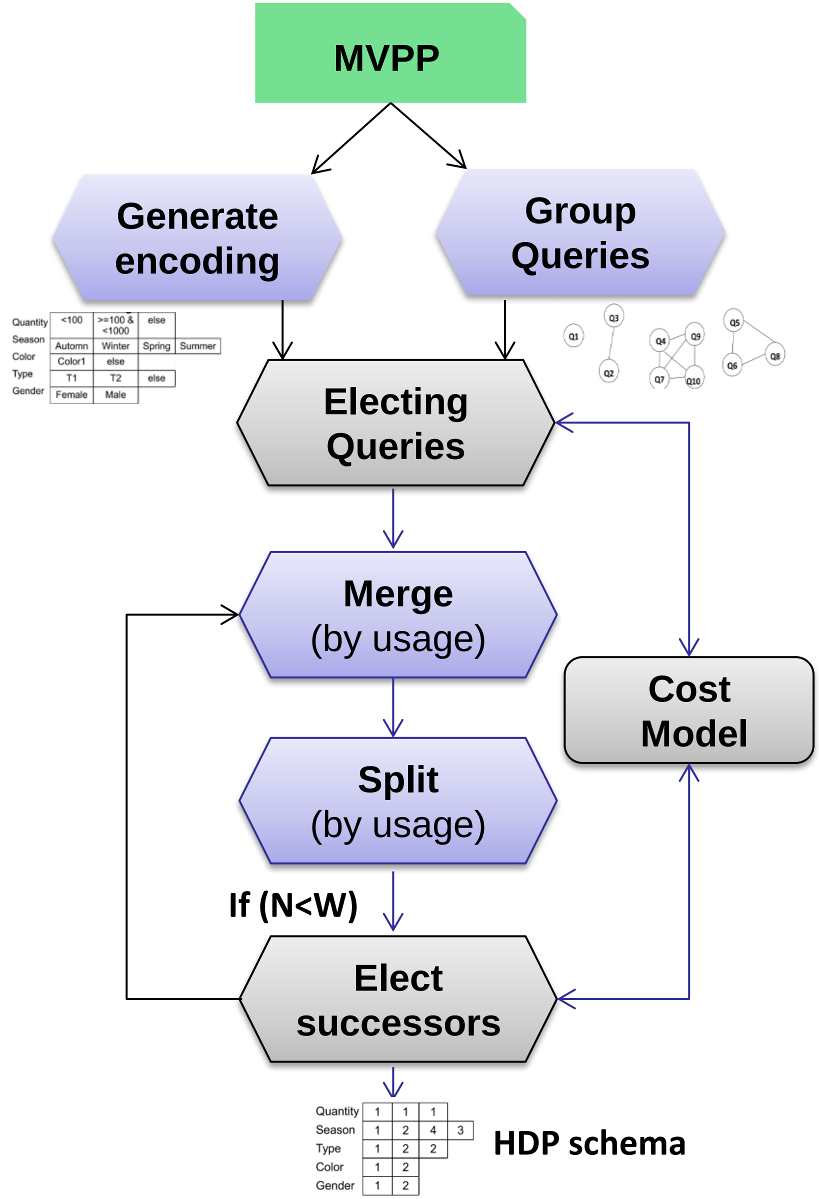
Algorithm EQHDP

```
generate_encoding();
EQA();
e := 1;
split_all();
E := elected(e)
while E not empty do
  prune_encoding(E);
  sort(E);
  S := required_attributes(E);
  usage(S);
  sort_attributes(S)
  for all a ∈ S do
    for all sd ∈ SubDomains(a) do
      if ((U(sd) = 0) and (N < W)) then
        merge(sd, P_0);
      end if
      if ((U(sd) = k) and (N < W)) then
        merge(sd, P_k);
      end if
    end for
  end for
  while k > 0 do
    for all a ∈ S do
      for all sd ∈ SubDomains(a) do
        if (N < W) then
          merge(sd, P_0);
        else
          merge(sd, P_k);
        end if
      end for
    end for
    k := k − 1;
  end while
  split_disjoint();
  e := e + 1;
  E := elected(e);
end while
```

Grouping Queries & Generating Encoding

Elect queries & Prune encoding

Sort elected queries by Cost & attributes by usage



Quantity	<100	>=100 & <1000	else	
Season	Automn	Winter	Spring	Summer
Color	Color1	else		
Type	T1	T2	else	
Gender	Female	Male		

Quantity	1	1	1	
Season	1	2	4	3
Type	1	2	2	
Color	1	2		
Gender	1	2		

Electing Queries for HDP

Algorithm EQHDP

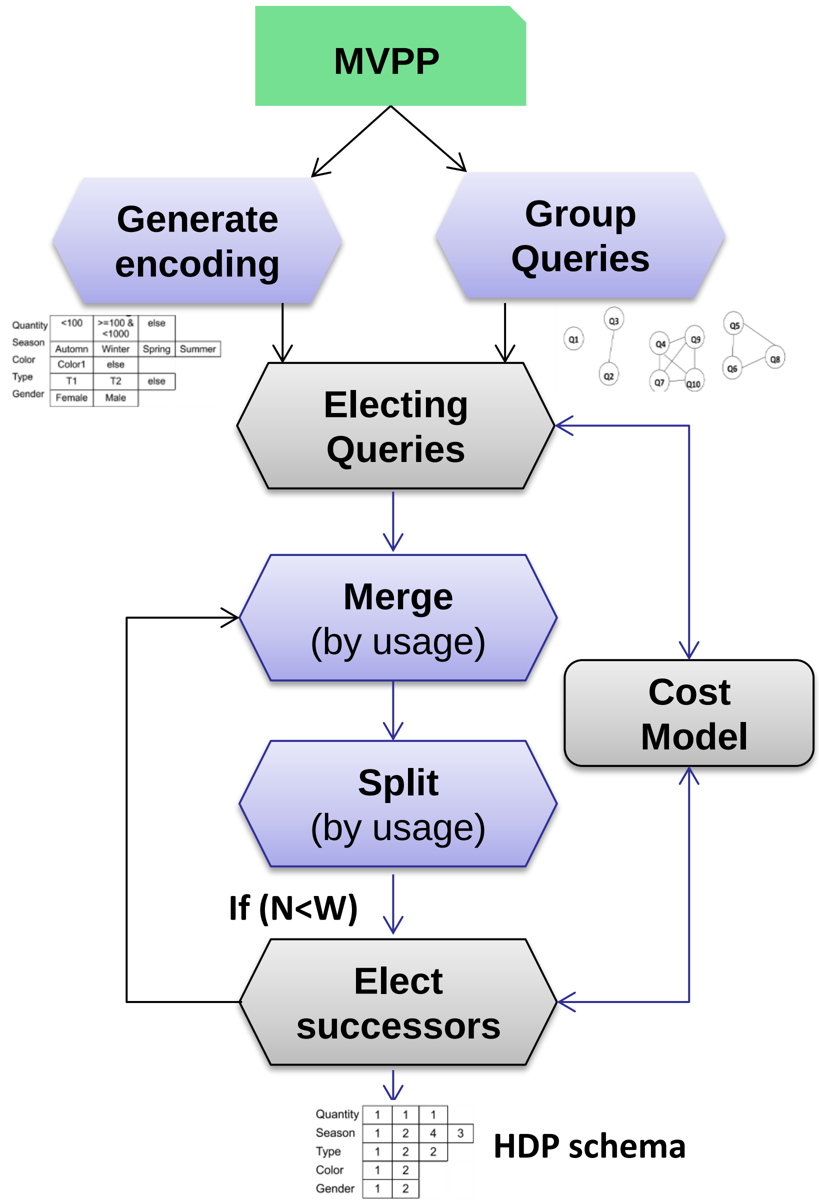
```
generate_encoding();
EQA();
e := 1;
split_all();
E := elected(e)
while E not empty do
  prune_encoding(E);
  sort(E);
  S := required_attributes(E);
  usage(S);
  sort_attributes(S)
  for all a ∈ S do
    for all sd ∈ SubDomains(a) do
      if ( (U(sd) = 0)and(N < W)) then
        merge(sd, P_0);
      end if
      if ( (U(sd) = k)and(N < W)) then
        merge(sd, P_k);
      end if
    end for
  end for
  while k > 0 do
    for all a ∈ S do
      for all sd ∈ SubDomains(a) do
        if ( N < W ) then
          merge(sd, P_0);
        else
          merge(sd, P_k);
        end if
      end for
    end for
    k := k − 1;
  end while
  split_disjoint();
  e := e + 1;
  E := elected(e);
end while
```

Grouping Queries
&
Generating Encoding

Elect queries
& Prune encoding

Sort elected queries by
Cost & attributes by usage

From highest usage rate
(by elected queries):
Merge subdomains
Having the same rate



Quantity	<100	>=100 & <1000	else	
Season	Automn	Winter	Spring	Summer
Color	Color1	else		
Type	T1	T2	else	
Gender	Female	Male		

Quantity	1	1	1	
Season	1	2	4	3
Type	1	2	2	
Color	1	2		
Gender	1	2		

Electing Queries for HDP

Algorithm EQHDP

```
generate_encoding();
EQA();
e := 1;
split_all();
E := elected(e)
while E not empty do
  prune_encoding(E);
  sort(E);
  S := required_attributes(E);
  usage(S);
  sort_attributes(S)
  for all a ∈ S do
    for all sd ∈ SubDomains(a) do
      if ( (U(sd) = 0)and(N < W)) then
        merge(sd, P_0);
      end if
      if ( (U(sd) = k)and(N < W)) then
        merge(sd, P_k);
      end if
    end for
  end for
  while k > 0 do
    for all a ∈ S do
      for all sd ∈ SubDomains(a) do
        if ( N < W ) then
          merge(sd, P_0);
        else
          merge(sd, P_k);
        end if
      end for
    end for
    k := k − 1;
  end while
  split_disjoint();
  e := e + 1;
  E := elected(e);
end while
```

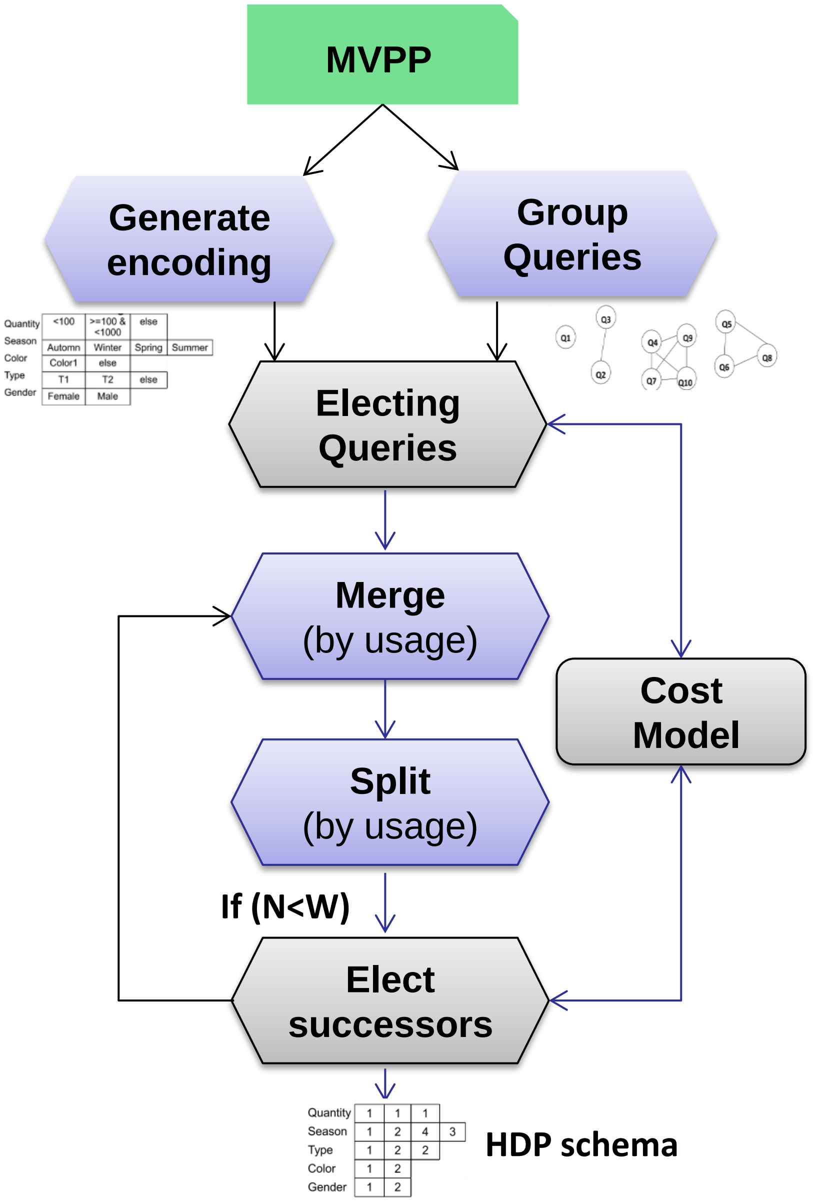
Grouping Queries
&
Generating Encoding

Elect queries
& Prune encoding

Sort elected queries by
Cost & attributes by usage

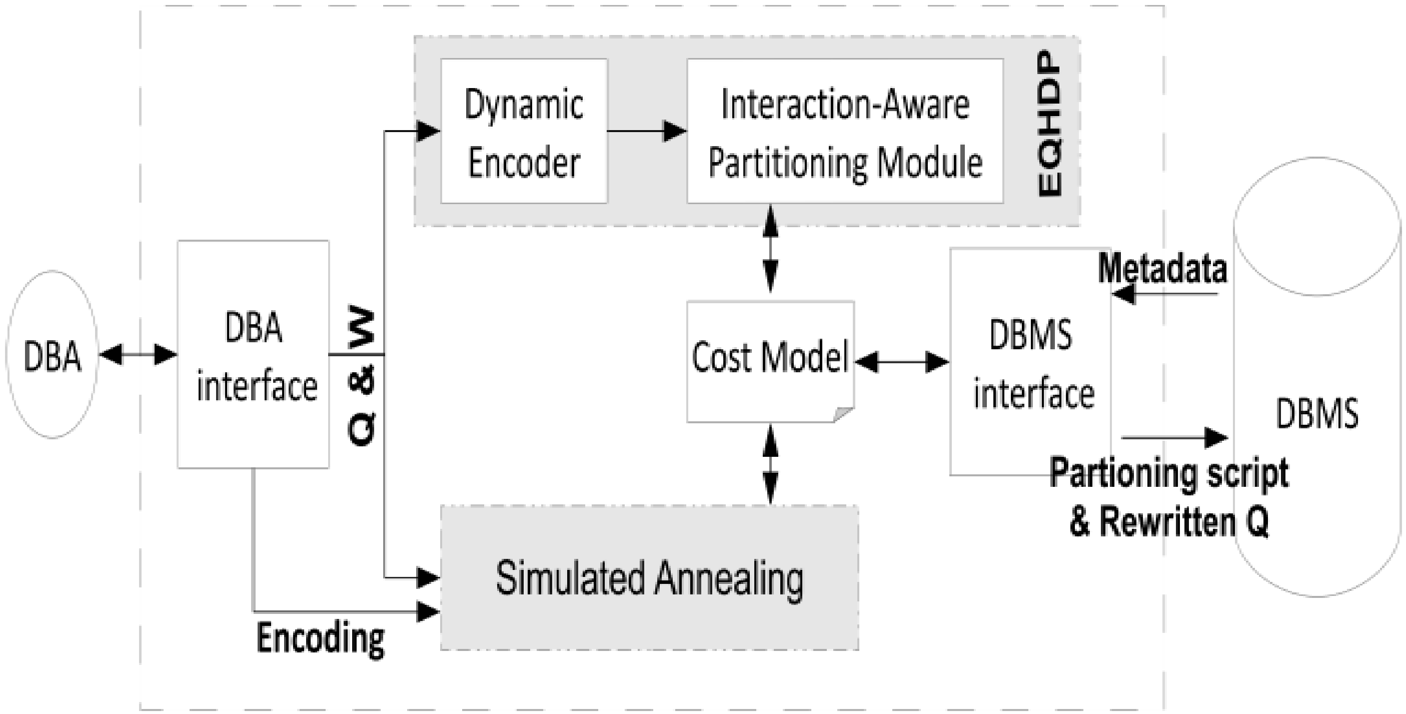
From highest usage rate
(by elected queries):
Merge subdomains
Having the same rate

Improvement:
move to successors of
elected queries (e ← e+1)



Experimental Study

Configuration of experiments



SSB of 100 GB

Workload1: 12 queries (no interaction)

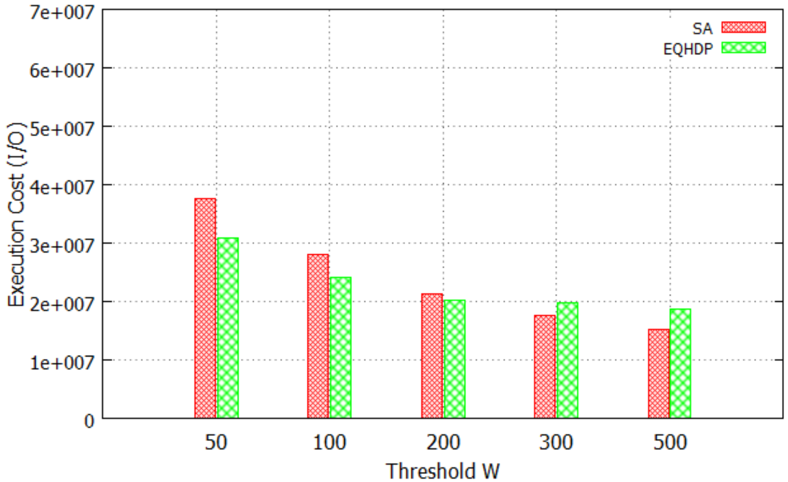
Workload2: 22 queries (with interaction)

Oracle11g DBMS

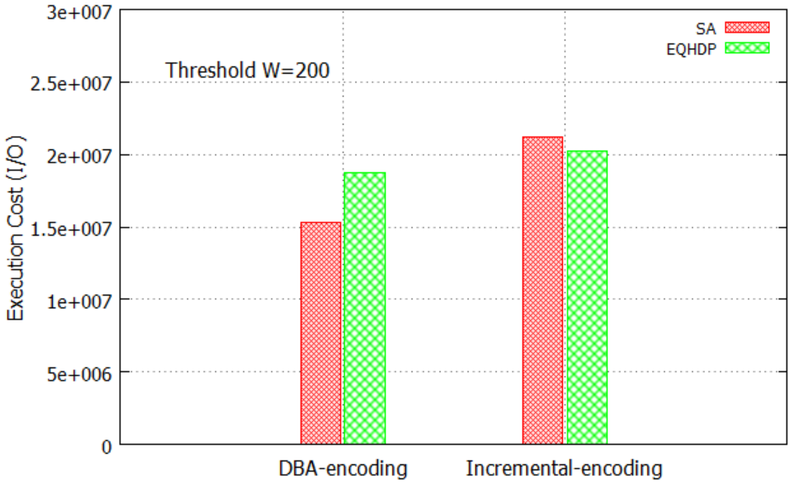
Server of 32 GB of RAM

Intel Xeon CPU : 2x2.45 GHz

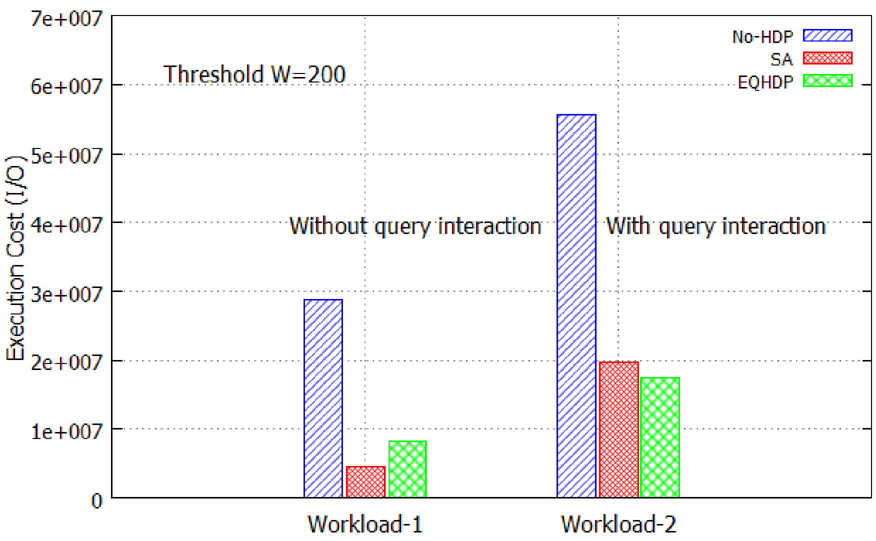
Experimental Study



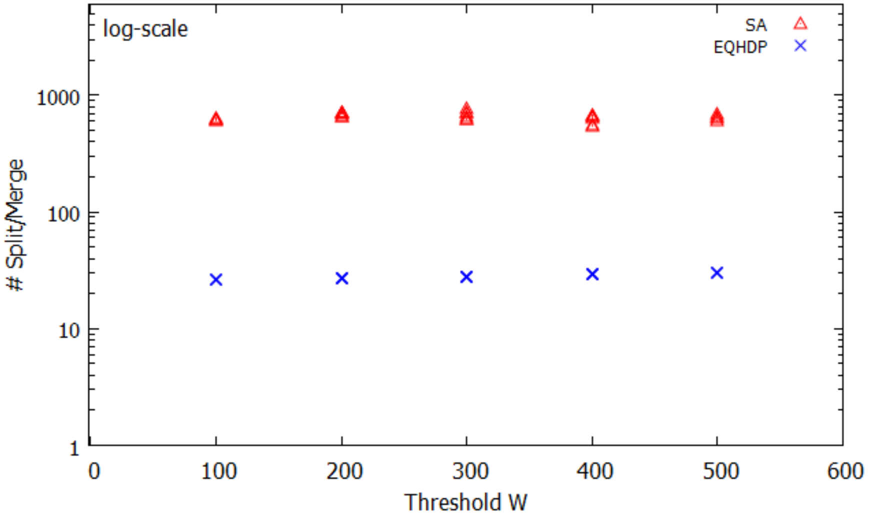
EQHDP Vs. SA



Impact of incremental encoding on performance

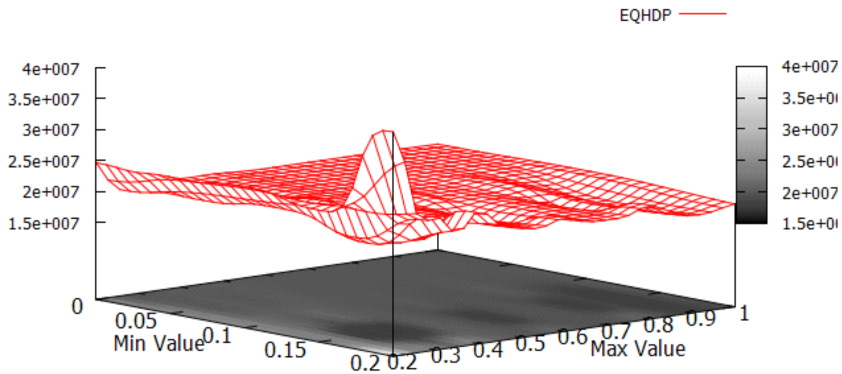


Impact of query interaction on performance

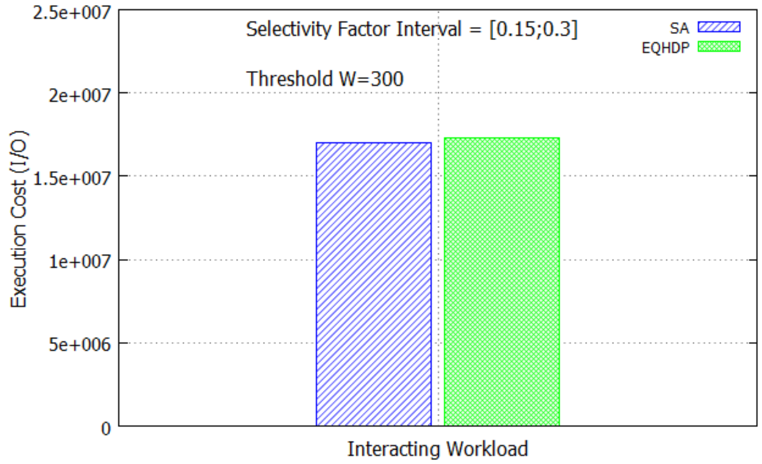


Number of Split/Merge to reach the solution

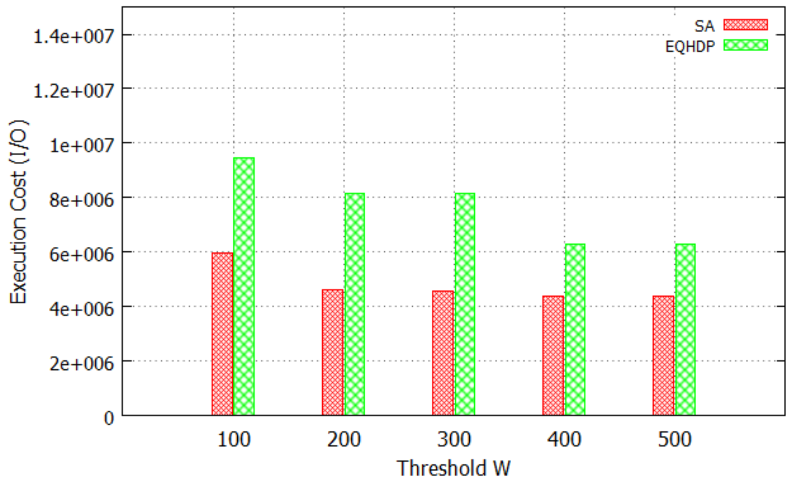
Experimental Study



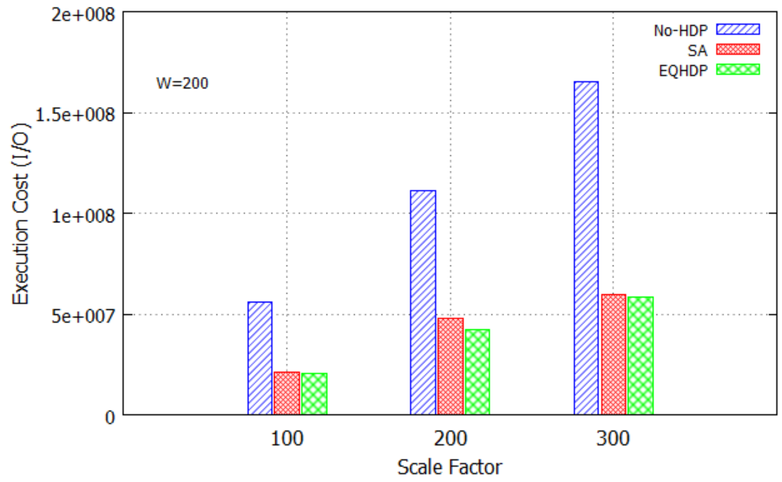
Best Selectivity Factor intervals



Improving EQHDP by SF interval



Validating algorithms' performance on Oracle11g



Scaling-Up and impact of data volume

Conclusion & Future Work

- ✓ Optimization in RDW by HDP
- ✓ Considering query interaction
- ✓ Incremental encoding for representing schemas
- ✓ Pruning predicates and steering HDP by elected queries
- ➢ Considering query interaction in other optimization techniques
- ➢ Include MVPP optimization in Physical Design
- ➢ **InterPhase** project

Thank you