An Approach to Optimize Data Processing in Business Processes

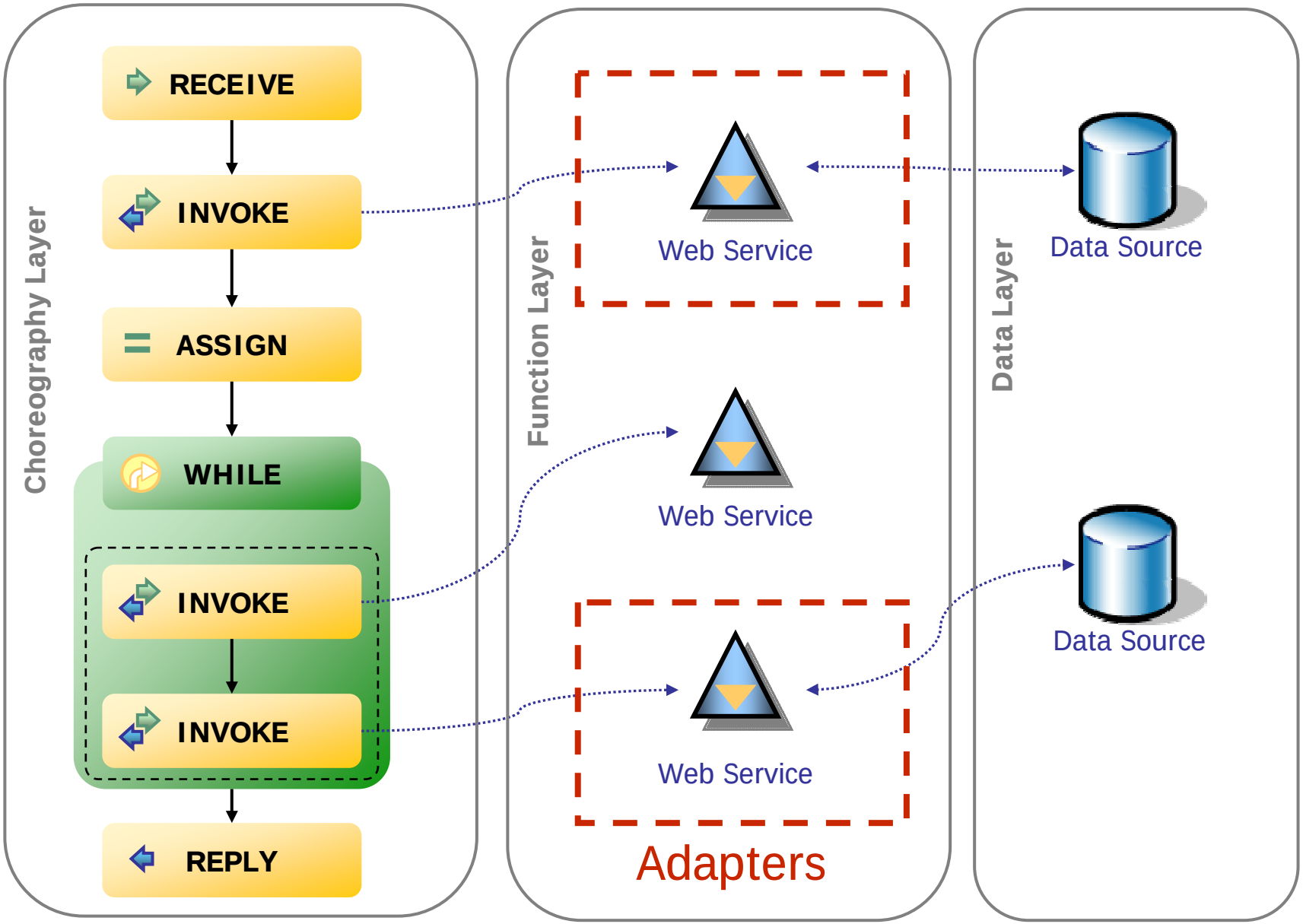
Marko Vrhovnik[1], **Holger Schwarz**[1], Oliver Suhre[2], Bernhard Mitschang[1], Volker Markl[3], Albert Maier[2], Tobias Kraft[1]

[1]Universität Stuttgart

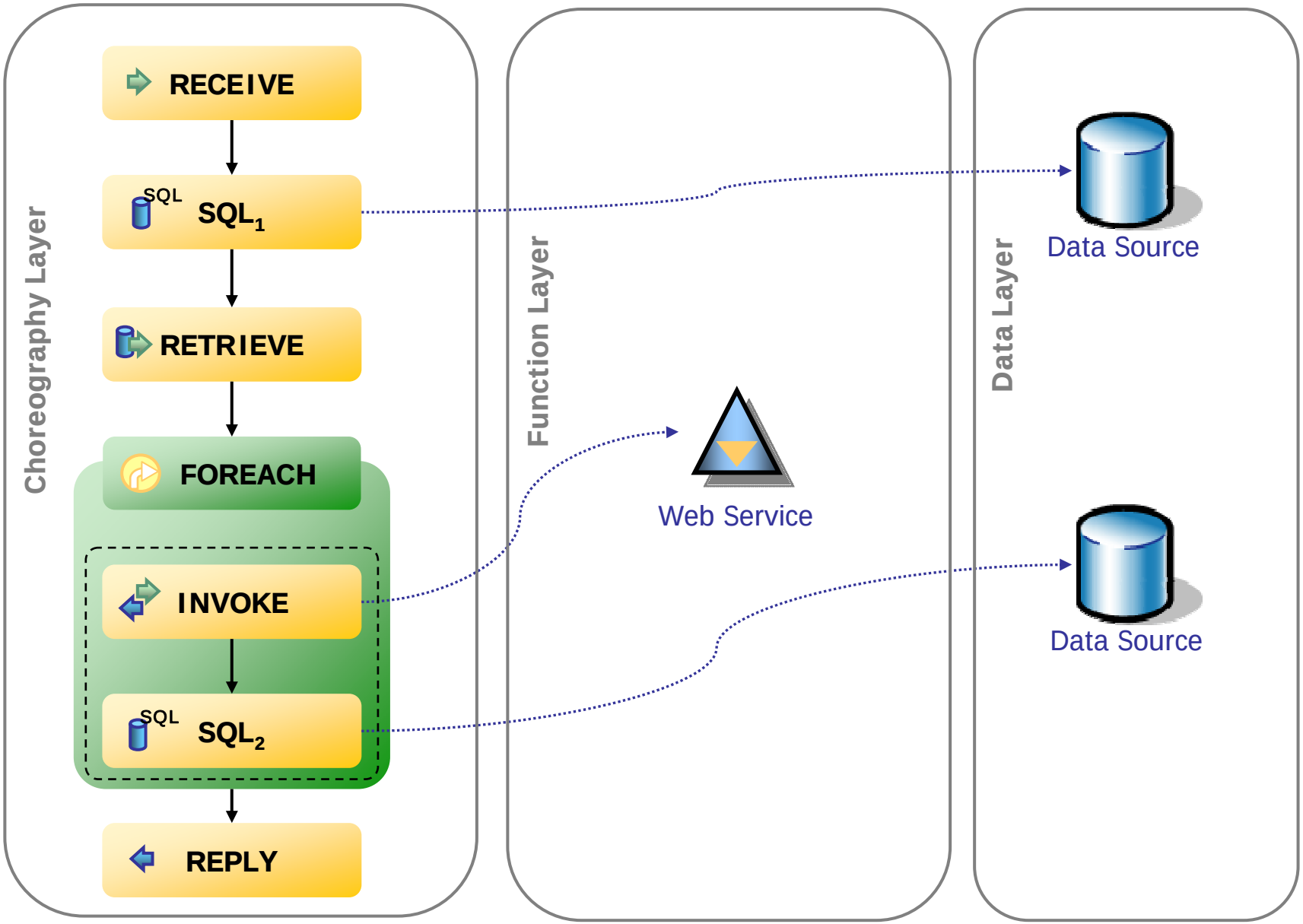
[2]IBM Böblingen

[3]IBM Almaden

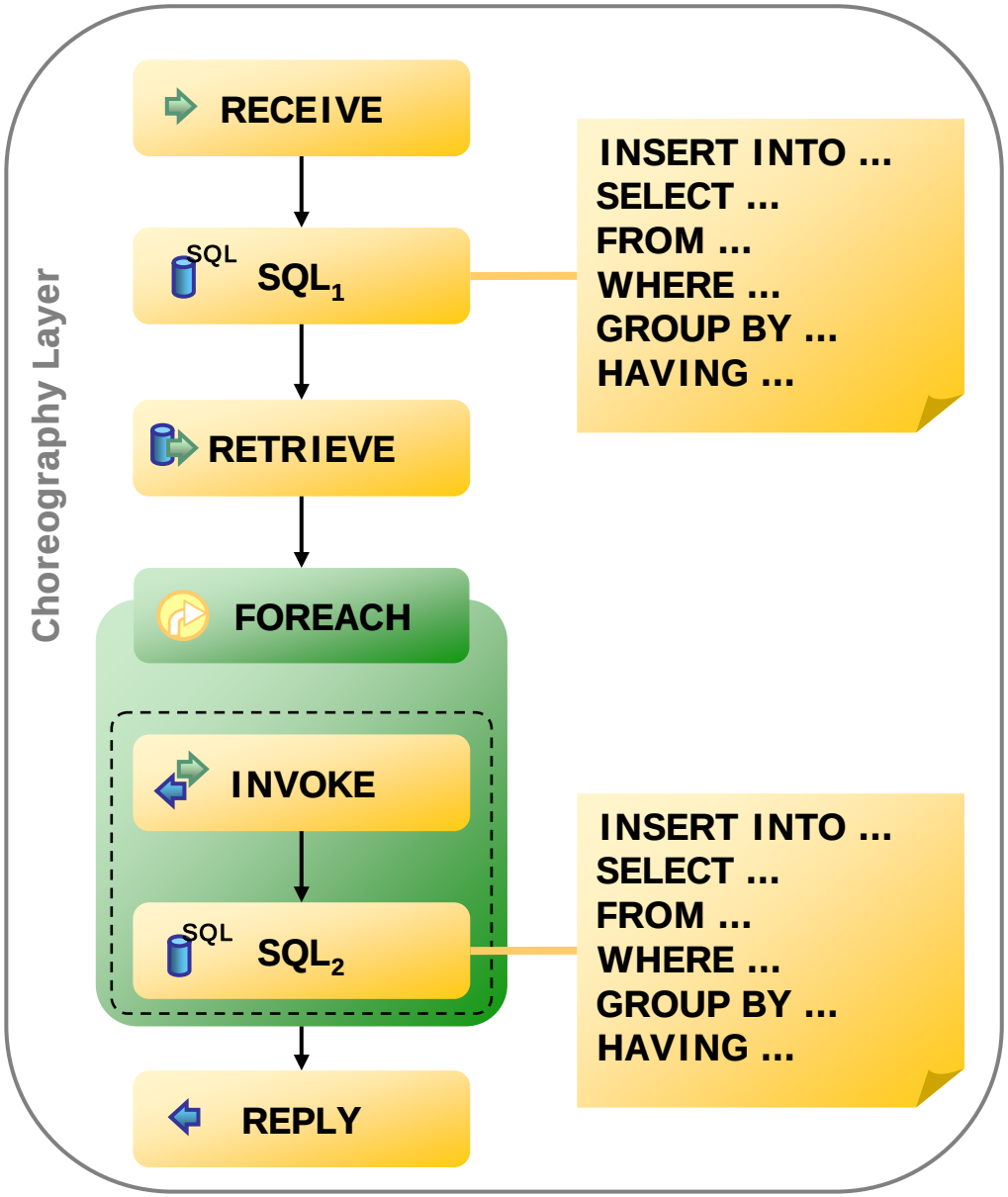
BPEL and Data Management: Adapters



BPEL and Data Mgmt.: Direct Integration



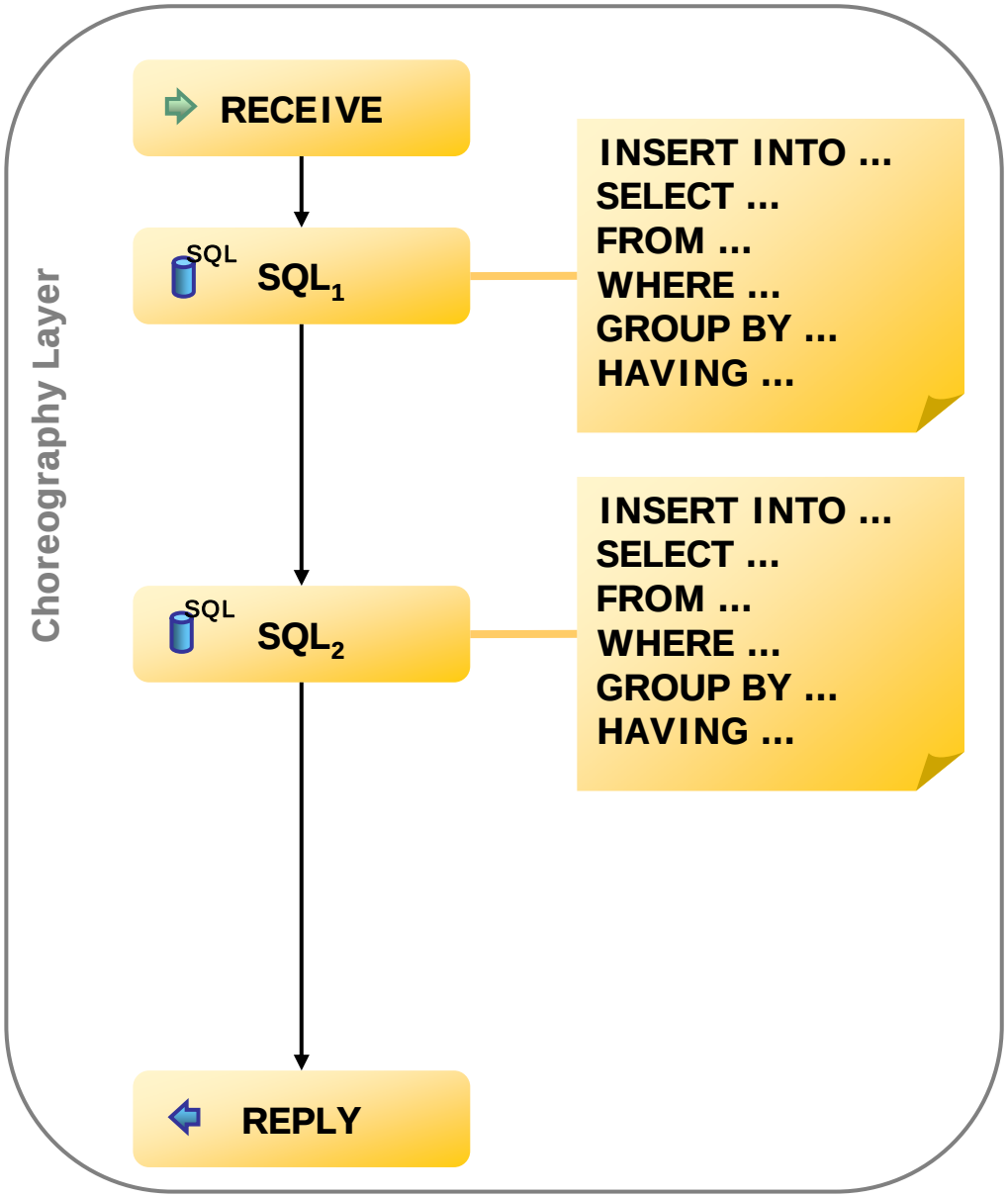
BPEL and Data Mgmt.: Direct Integration



- Data management is defined at the choreography layer
- Opportunities for optimizing data management arise

Rewrite the process definition

BPEL and Data Mgmt.: Direct Integration



Rewrite the process definition

- Appropriate rewrite rules?
- Components of an optimizer engine?
- Performance improvements?

Outline

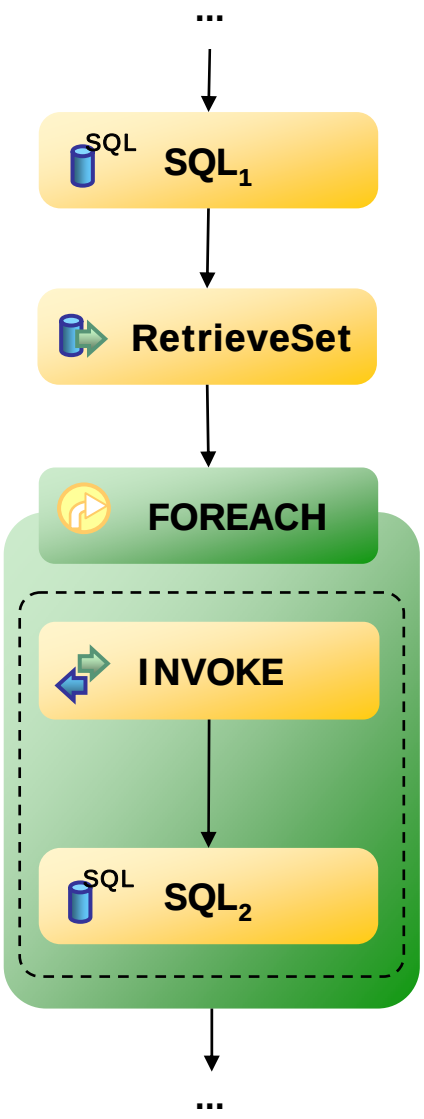
- Data management at the choreography level
- Sample scenario and optimization opportunities
- Classification of rewrite rules
- Control strategy
- Optimizer Engine
- Experimental results
- Conclusion

Alternatives for Data Management at the Choreography Level

- Database vendors pursue various approaches
- IBM WebSphere Process Server
  - Adds information service activities to BPEL
  - Adds set-oriented BPEL variables
  - BPEL/SQL
- Oracle BPEL Process Manager
  - XPath extension functions embedded in assign activities
  - Functions support any valid SQL statement
  - Query results are stored in set-oriented process variables
- Microsoft Windows Workflow Foundation
  - SQL activities as part of business processes
  - Entire workflow is for example described by XOML

Sample Scenario

Flow Logic



Partners

SELECT ItemID, SUM(Quantity) **AS** ItemQuantity
FROM #SR_Orders#
WHERE Approved=1
GROUP BY ItemID
→ #SR_ItemList#

Orders

ItemList
(Temporary Table)

RETRIEVE #SR_ItemList# **INTO** #SV_ItemList#

FOREACH #CurrentItem# **IN** #SV_ItemList#

OrderFromSupplier

INVOKE OrderFromSupplier
IN: #CurrentItem.ItemID#, #CurrentItem.ItemQuantity#,
OUT: #OrderConfirmation#

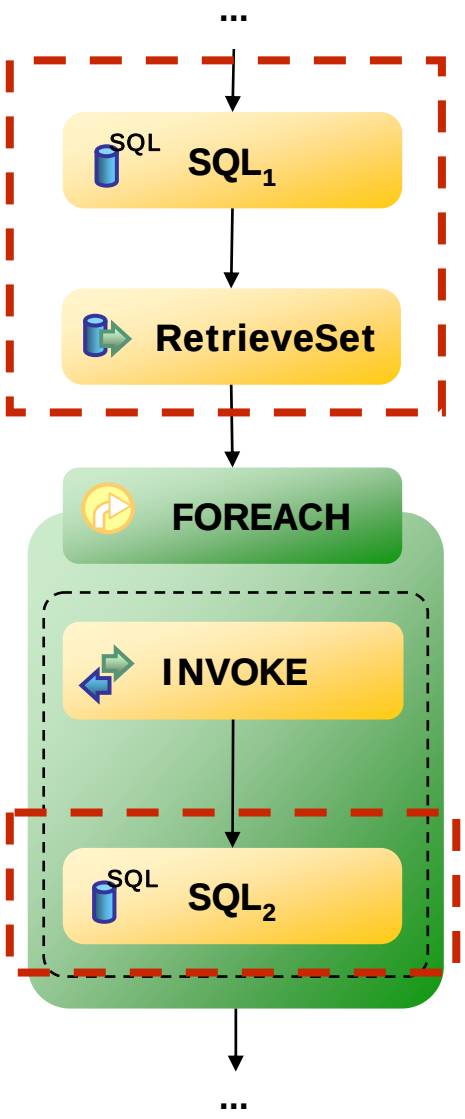
INSERT INTO #SR_ OrderConfirmations#

VALUES (#CurrentItem.ItemID#,
#CurrentItem.ItemQuantity#,
#OrderConfirmation#)

OrderConfirmations

Sample Scenario

Flow Logic



Partners

```
SELECT  ItemID, SUM(Quantity) AS ItemQuantity
FROM    #SR_Orders#
WHERE   Approved=1
GROUP BY ItemID
→ #SR_ItemList#
```

Orders

ItemList
(Temporary Table)

```
RETRIEVE #SR_ItemList# INTO #SV_ItemList#
```

Data Management Activities

```
FOREACH #CurrentItem# IN #SV_ItemList#
```

OrderFromSupplier

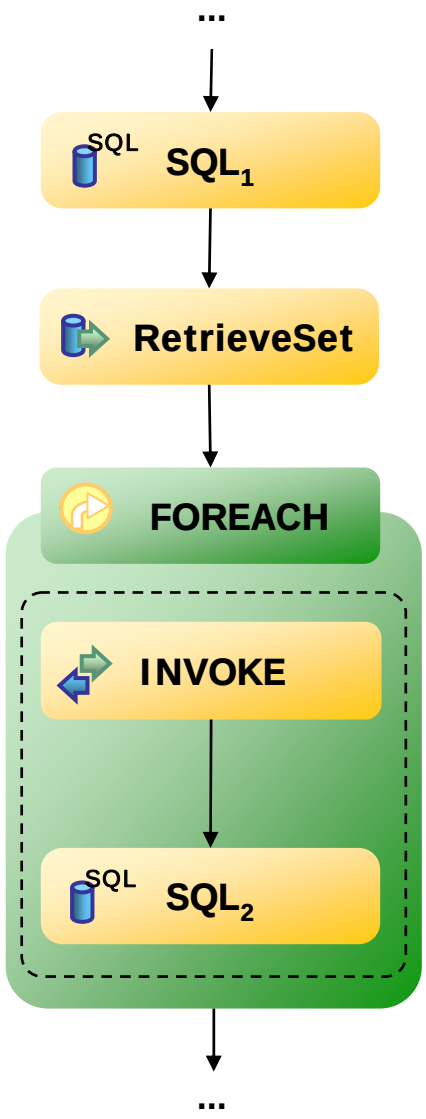
```
INVOKE OrderFromSupplier
IN:   #CurrentItem.ItemID#, #CurrentItem.ItemQuantity#,
OUT: #OrderConfirmation#
```

```
INSERT INTO #SR_ OrderConfirmations#
VALUES (#CurrentItem.ItemID#,
        #CurrentItem.ItemQuantity#,
        #OrderConfirmation# )
```

OrderConfirmations

Sample Scenario

Flow Logic



Partners

```
SELECT ItemID, SUM(Quantity) AS ItemQuantity
FROM   #SR_Orders#
WHERE  Approved=1
GROUP BY ItemID
→ #SR_ItemList#
```

Orders

ItemList
(Temporary Table)

```
RETRIEVE #SR_ItemList# INTO #SV_ItemList#
```

Set References

```
FOREACH #CurrentItem# IN #SV_ItemList#
```

OrderFromSupplier

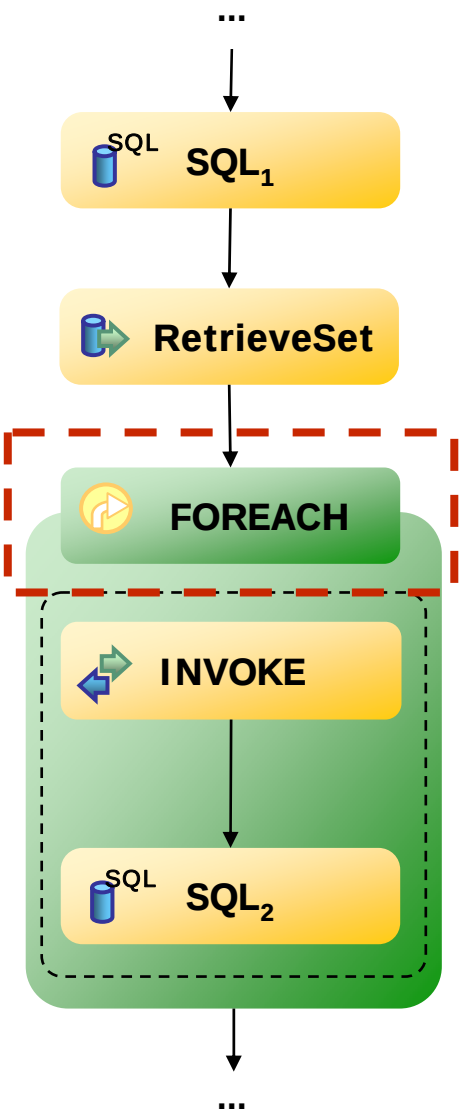
```
INVOKE OrderFromSupplier
IN:   #CurrentItem.ItemID#, #CurrentItem.ItemQuantity#,
OUT: #OrderConfirmation#
```

```
INSERT INTO #SR_ OrderConfirmations#
VALUES (#CurrentItem.ItemID#,
        #CurrentItem.ItemQuantity#,
        #OrderConfirmation# )
```

OrderConfirmations

Sample Scenario

Flow Logic



Partners

SELECT ItemID, SUM(Quantity) **AS** ItemQuantity
FROM #SR_Orders#
WHERE Approved=1
GROUP BY ItemID
→ #SR_ItemList#

Orders

ItemList
(Temporary Table)

RETRIEVE #SR_ItemList# **INTO** #SV_ItemList#

Iterator

FOREACH #CurrentItem# **IN** #SV_ItemList#

Set Variable

INVOKE OrderFromSupplier
IN: #CurrentItem.ItemID#, #CurrentItem.ItemQuantity#,
OUT: #OrderConfirmation#

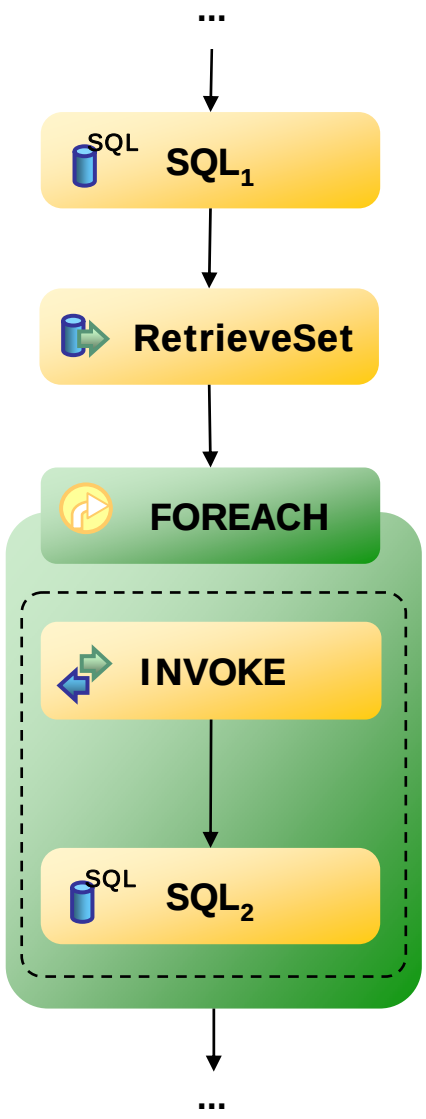
OrderFromSupplier

INSERT INTO #SR_ OrderConfirmations#

VALUES (#CurrentItem.ItemID#,
#CurrentItem.ItemQuantity#,
#OrderConfirmation#)

OrderConfirmations

Optimization of Sample Process



SELECT ItemID, SUM(Quantity) **AS** ItemQuantity
FROM #SR_Orders#
WHERE Approved=1
GROUP BY ItemID
→ #SR_ItemList#

Orders

ItemList
(Temporary Table)

RETRIEVE #SR_ItemList# **INTO** #SV_ItemList#

FOREACH #CurrentItem# **IN** #SV_ItemList#

OrderFromSupplier

INVOKE OrderFromSupplier
IN: #CurrentItem.ItemID#, #CurrentItem.ItemQuantity#,
OUT: #OrderConfirmation#

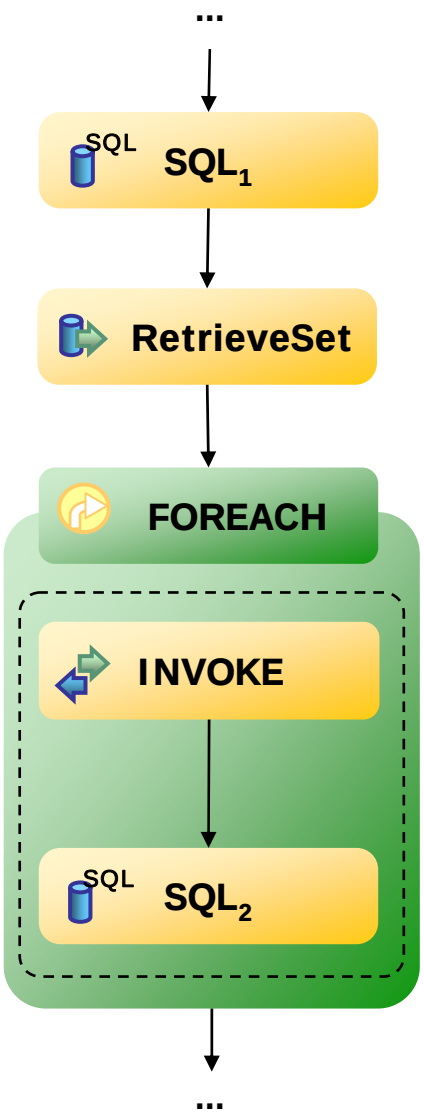
INSERT INTO #SR_OrderConfirmations#
VALUES (#CurrentItem.ItemID#,
#CurrentItem.ItemQuantity#,
#OrderConfirmation#)

OrderConfirmations

Variables ⇢ Data Dependencies

Optimization of Sample Process

Web Service Pushdown



SELECT ItemID, SUM(Quantity) **AS** ItemQuantity
FROM #SR_Orders#
WHERE Approved=1
GROUP BY ItemID
→ #SR_ItemList#

Orders

ItemList
(Temporary Table)

RETRIEVE #SR_ItemList# **INTO** #SV_ItemList#

FOREACH #CurrentItem# **IN** #SV_ItemList#

OrderFromSupplier

INVOKE OrderFromSupplier
IN: #CurrentItem.ItemID#, #CurrentItem.ItemQuantity#,
OUT: #OrderConfirmation#

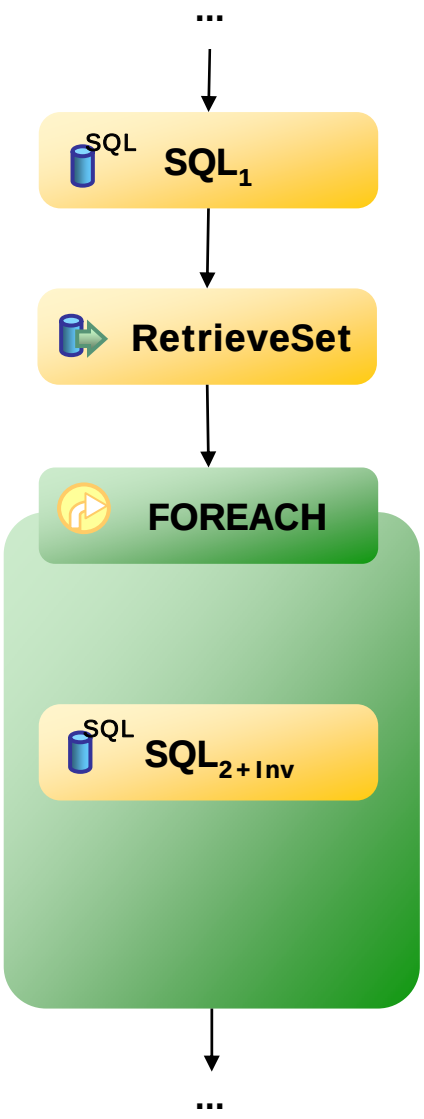
INSERT INTO #SR_ OrderConfirmations#
VALUES (#CurrentItem.ItemID#,
#CurrentItem.ItemQuantity#,
#OrderConfirmation#)

OrderConfirmations

Variables ⋯▸ Data Dependencies

Optimization of Sample Process

Web Service Pushdown




```
SELECT  ItemID, SUM(Quantity) AS ItemQuantity
FROM    #SR_Orders#
WHERE   Approved=1
GROUP BY ItemID
→ #SR_ItemList#

RETRIEVE #SR_ItemList# INTO #SV_ItemList#

FOREACH #CurrentItem# IN #SV_ItemList#

INSERT INTO #SR_ OrderConfirmations#
VALUES (#CurrentItem.ItemID#,
        #CurrentItem.ItemQuantity#,
        OrderFromSupplier(#CurrentItem.ItemID#,
                          #CurrentItem.ItemQuantity#))
```

Orders

ItemList
(Temporary Table)

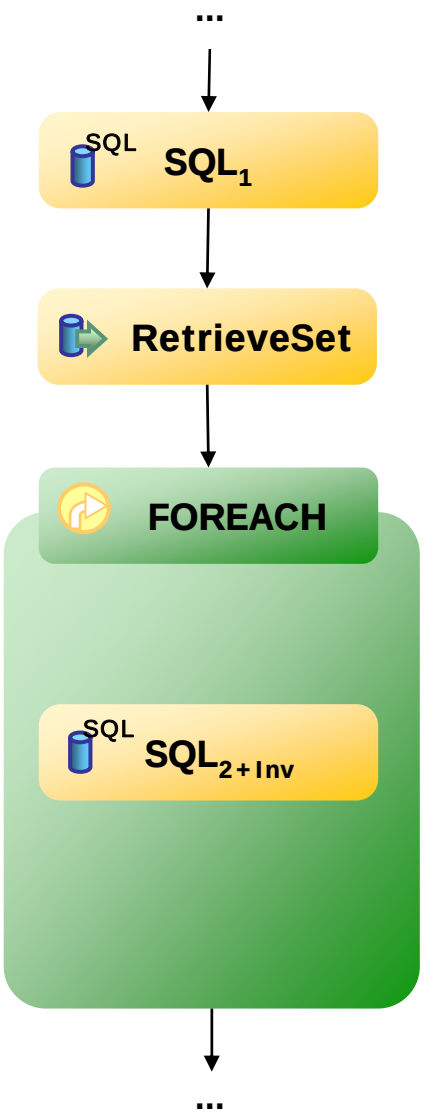
OrderFromSupplier

OrderConfirmations

Variables ▪▪▪▪▪▪▶ Data Dependencies

Optimization of Sample Process

Tuple-To-Set




```
SELECT  ItemID, SUM(Quantity) AS ItemQuantity
FROM    #SR_Orders#
WHERE   Approved=1
GROUP BY ItemID
→ #SR_ItemList#

RETRIEVE #SR_ItemList# INTO #SV_ItemList#

FOREACH #CurrentItem# IN #SV_ItemList#

INSERT INTO #SR_ OrderConfirmations#
VALUES (#CurrentItem.ItemID#,
        #CurrentItem.ItemQuantity#,
        OrderFromSupplier(#CurrentItem.ItemID#,
                          #CurrentItem.ItemQuantity#))
```

Orders

ItemList
(Temporary Table)

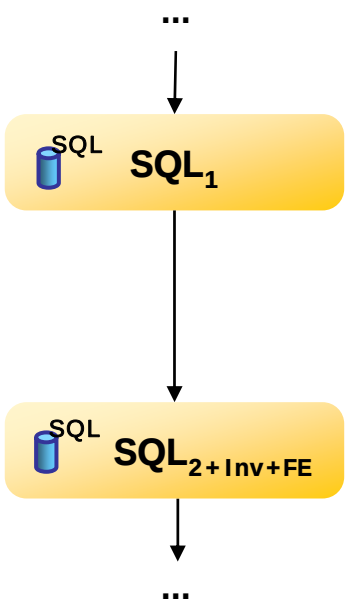
OrderFromSupplier

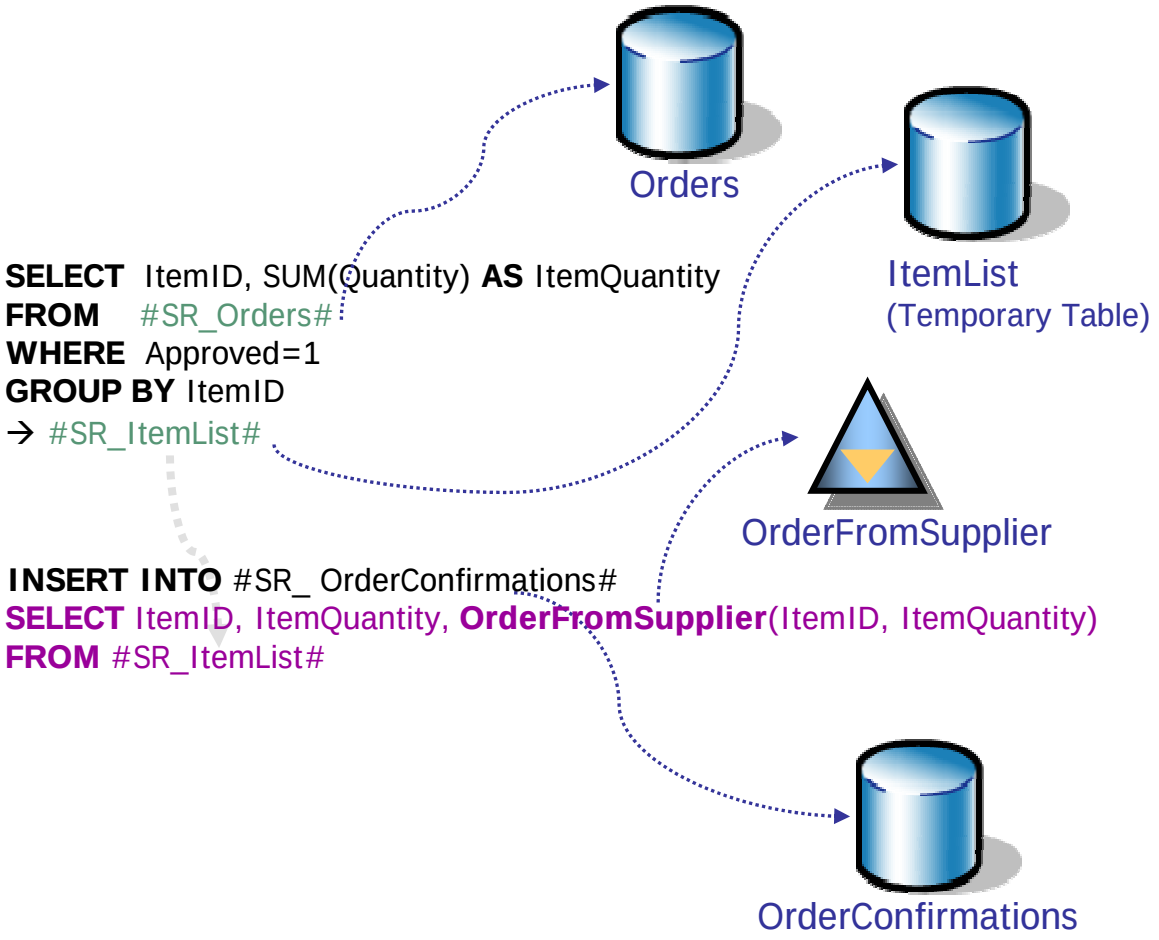
OrderConfirmations

Variables ⇢ Data Dependencies

Optimization of Sample Process

Tuple-To-Set

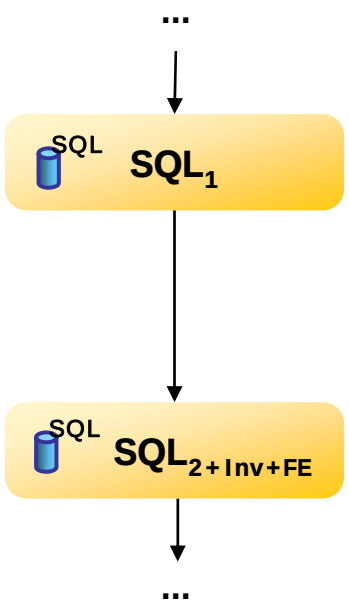


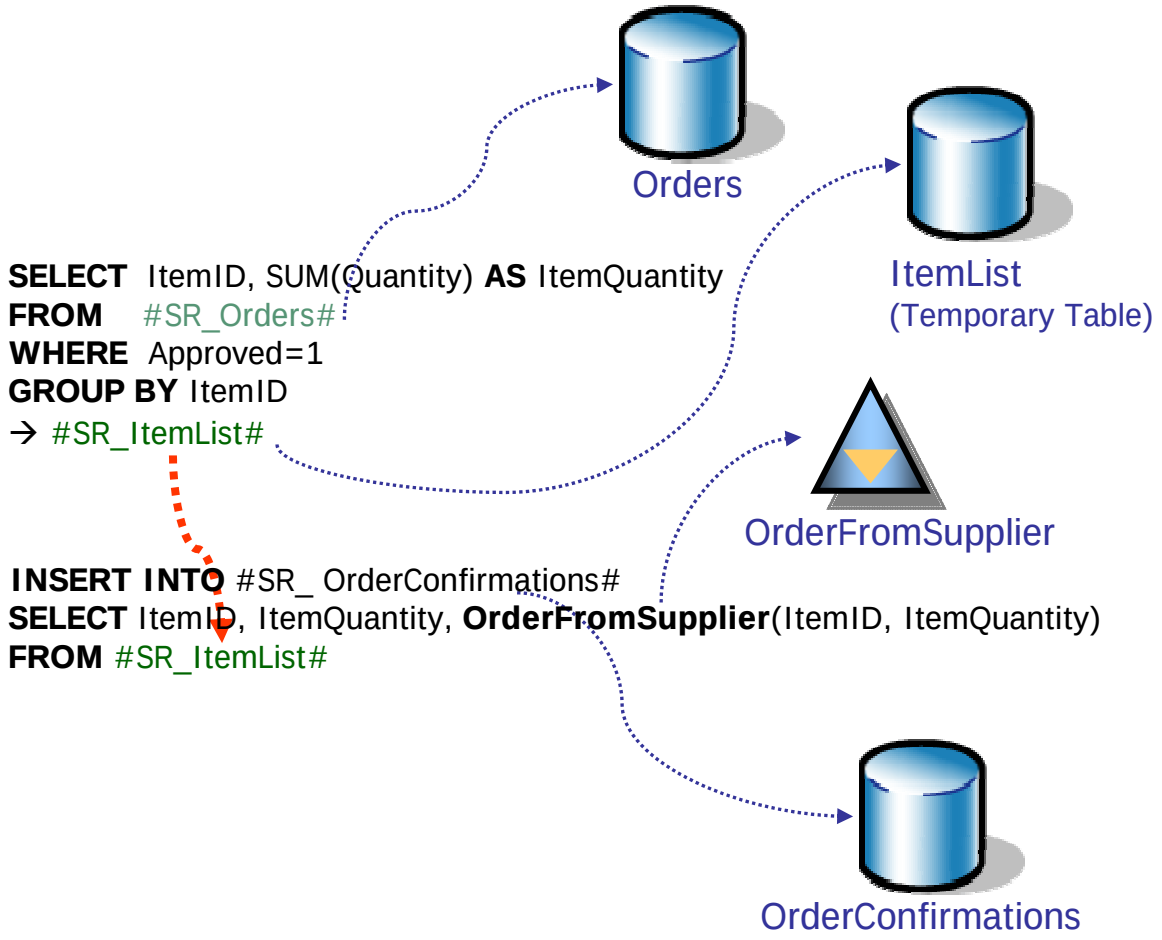


Variables ▪▪▪▪▪▪▶ Data Dependencies

Optimization of Sample Process

Eliminate Temporary Table

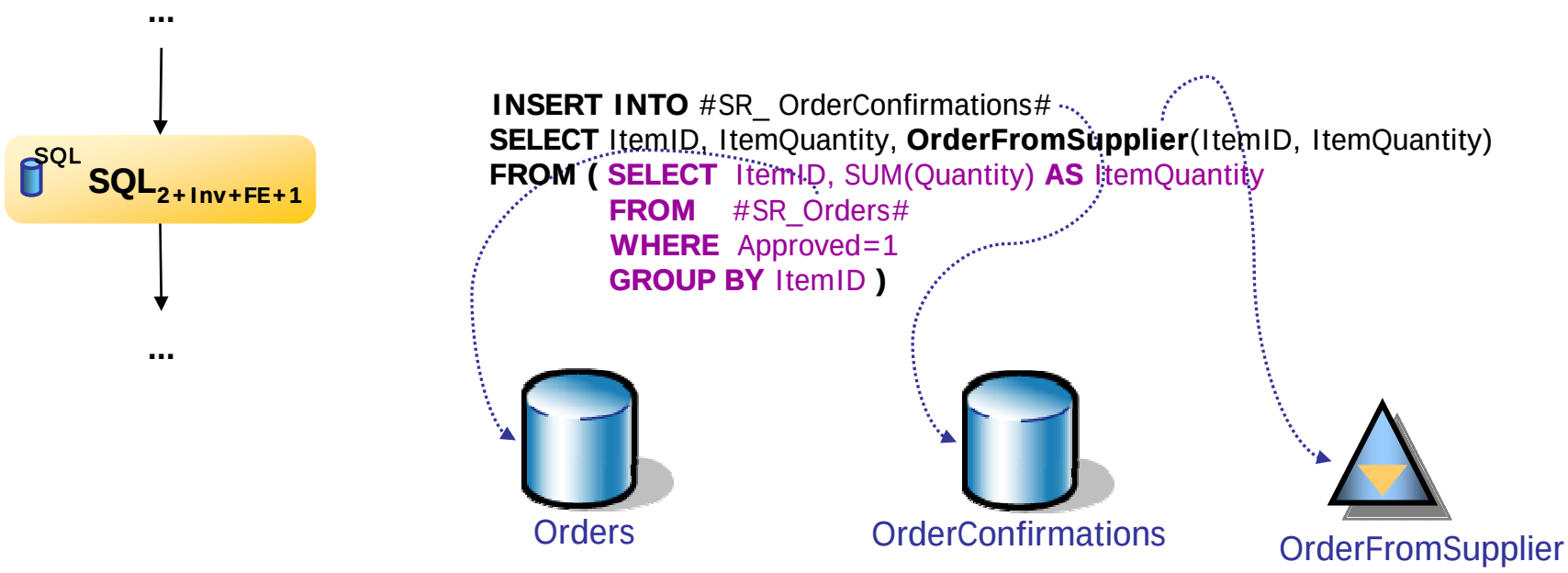




Variables ▪▪▪▪▪▪▶ Data Dependencies

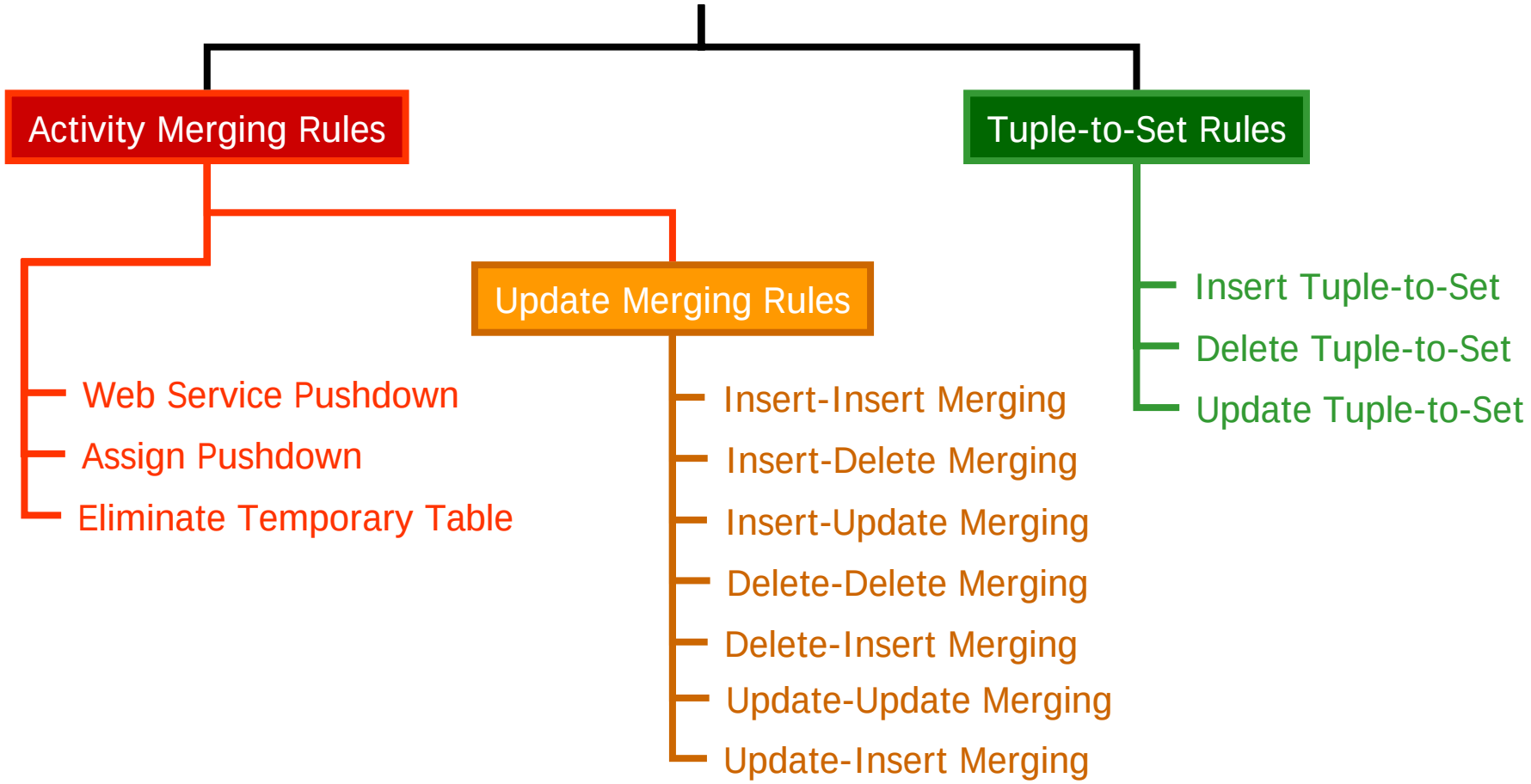
Optimization of Sample Process

Eliminate Temporary Table

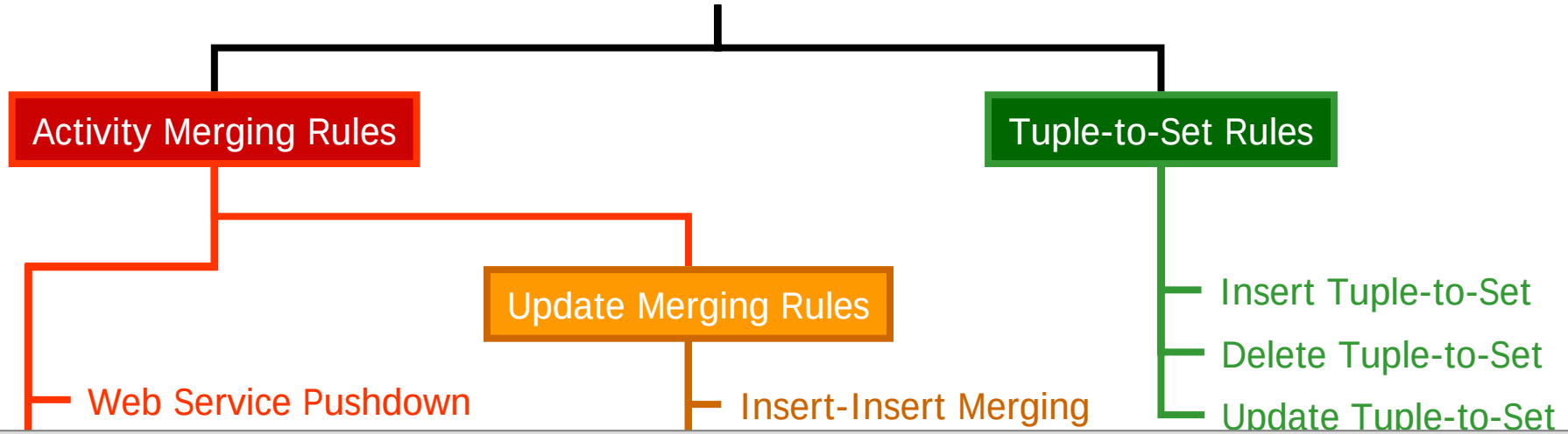


Variables ▪▪▪▪▪▪▶ Data Dependencies

Classification of Rewrite Rules



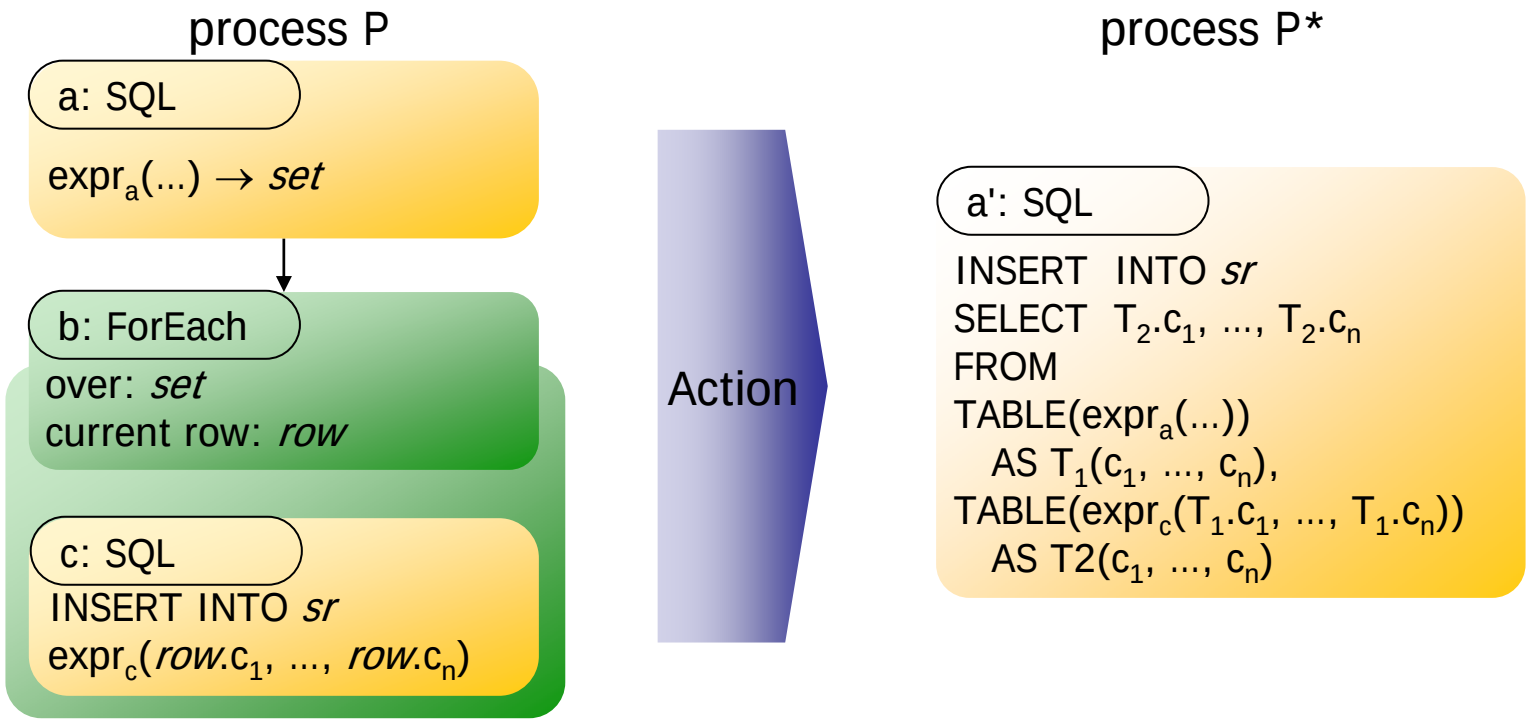
Classification of Rewrite Rules



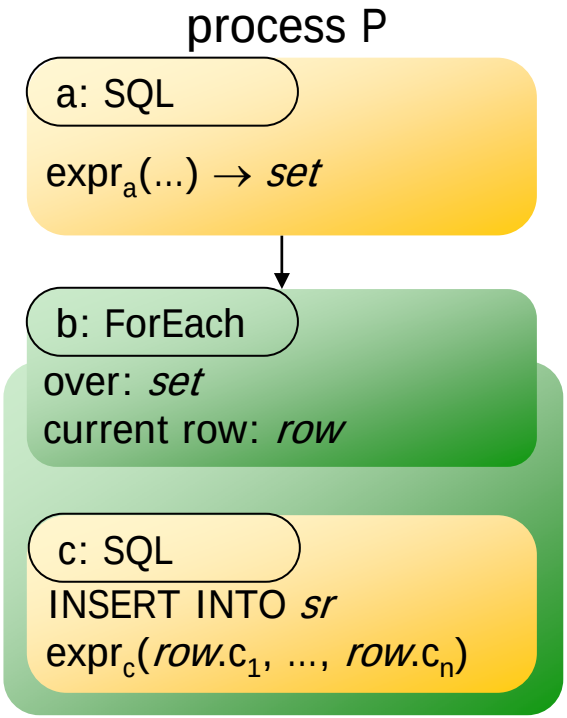
Properties of this rule set

- Performance improvement due to ...
  - reduced number of activities
  - reduced overhead for translating, optimizing and processing SQL
  - reduced data transfer between choreography layer and database level
  - higher potential for optimization at the database level
- Semantics preserving
- Enabling relationships

Insert Tuple-to-Set Rule

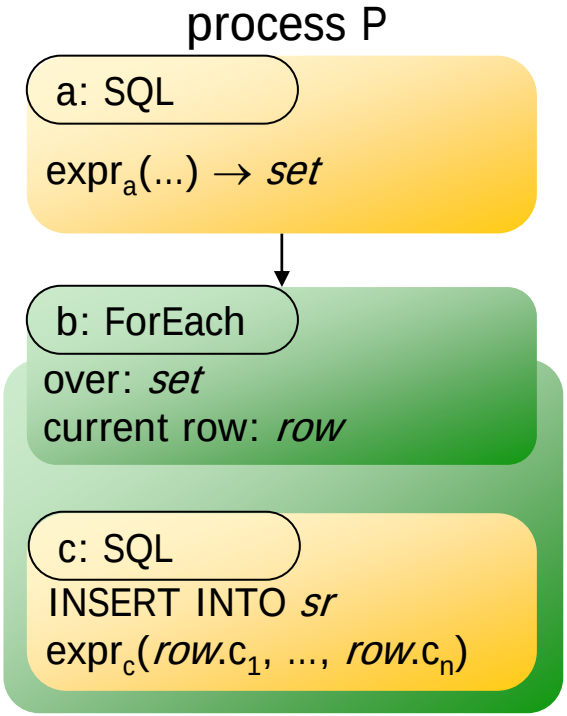


Insert Tuple-to-Set Rule

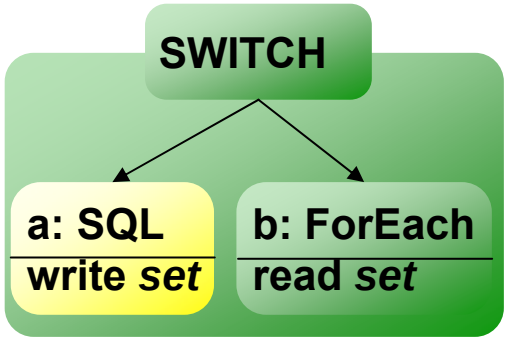


- Activity Condition A1:
  - Activity *a* is of type SQL
  - providing the results of query expression expr$_a$ in a set variable *set*.
- Activity Condition A2:
  - ForEach activity *b* iterates over *set* and
  - provides the current row in variable *row* (columns row.c$_l$).
- Activity Condition A3:
  - SQL activity *c* is the only activity in the loop body.
  - It executes an INSERT statement (query expression or VALUES expression)

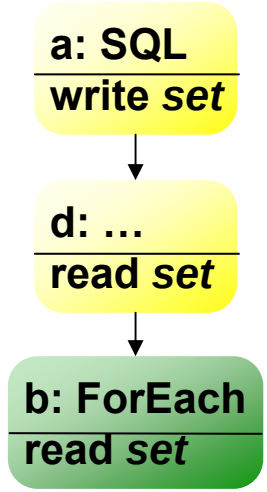
Insert Tuple-to-Set Rule



processes not matching this condition



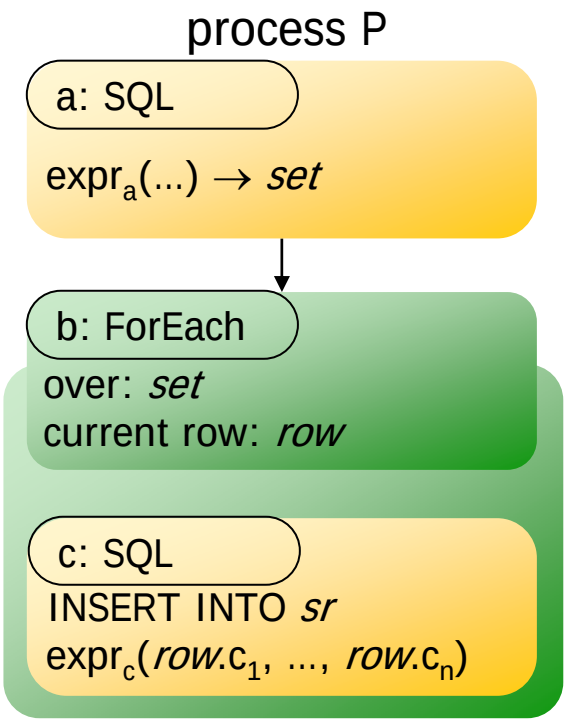
no write-read dependency



multiple dependencies

- Data Dependency Condition D1:
  - There is a single write-read data dependency between a and b based on *set*.
  - Activity a writes *set* before b reads *set*.

Insert Tuple-to-Set Rule



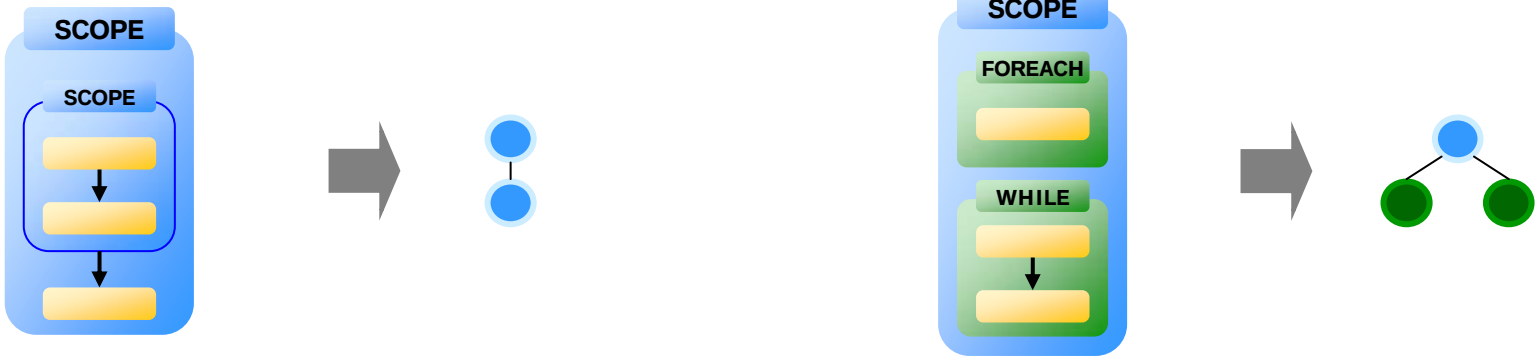
- Data Dependency Condition D1:
  - There is a single write-read data dependency between a and b based on variable *set*.
  - Activity a writes *set* before b reads *set*.
- Value Stability Condition S1:
  - *set* is stable, i.e., it does not change between its definition in a and its usage in b.
- Data Dependency Condition D2:
  - There is a single write-read data dependency between b and c based on variable *row*.
  - Activity b writes *row* before c reads *row*.
- Value Stability Condition S2:
  - In each iteration of b, c reads that value of *row* that is provided by b.

Optimization Spheres

- Define where rewrite rules are applied
- All effects of rewrite rules are local to an optimization sphere

Scope Optimization Spheres (SOS)
- defined by BPEL scope activities
- BPEL scope activities may cover fault- and compensation handlers
- Activity Merging rules only

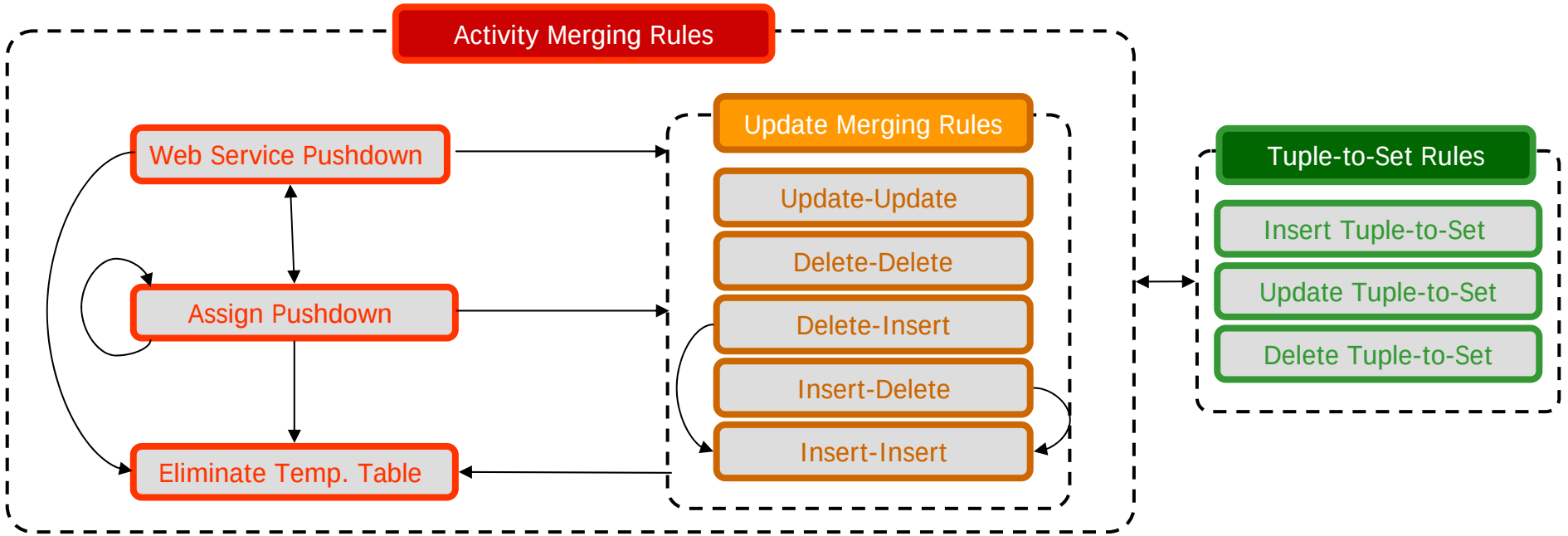
Loop Optimization Spheres (LOS)
- defined by loop activities, e.g. ForEach
- Activity Merging Rules and Tuple-to-Set Rules



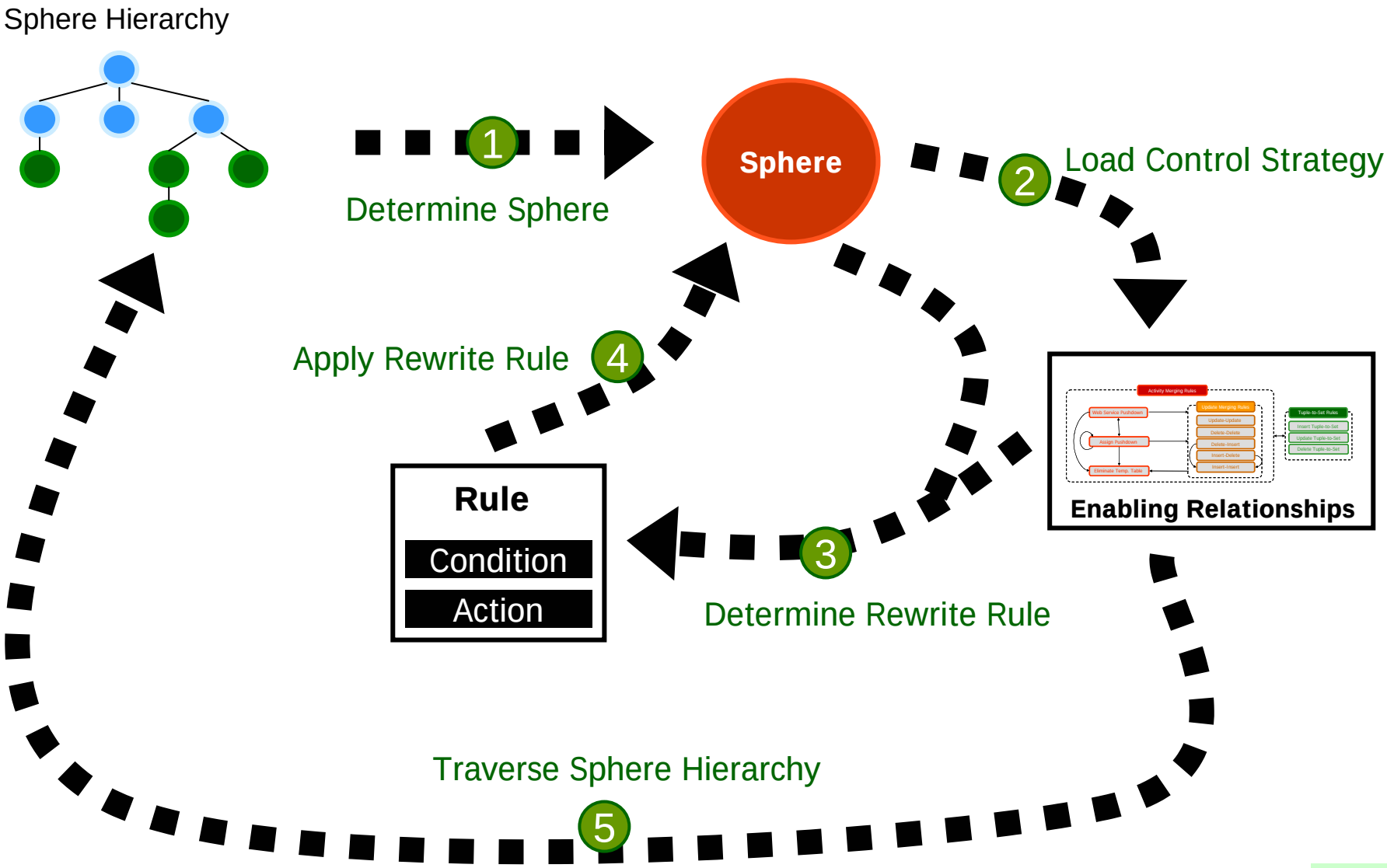
- Optimization process exploits the sphere hierarchy

Enabling Relationships

- Define the order in which rewrite rules are applied

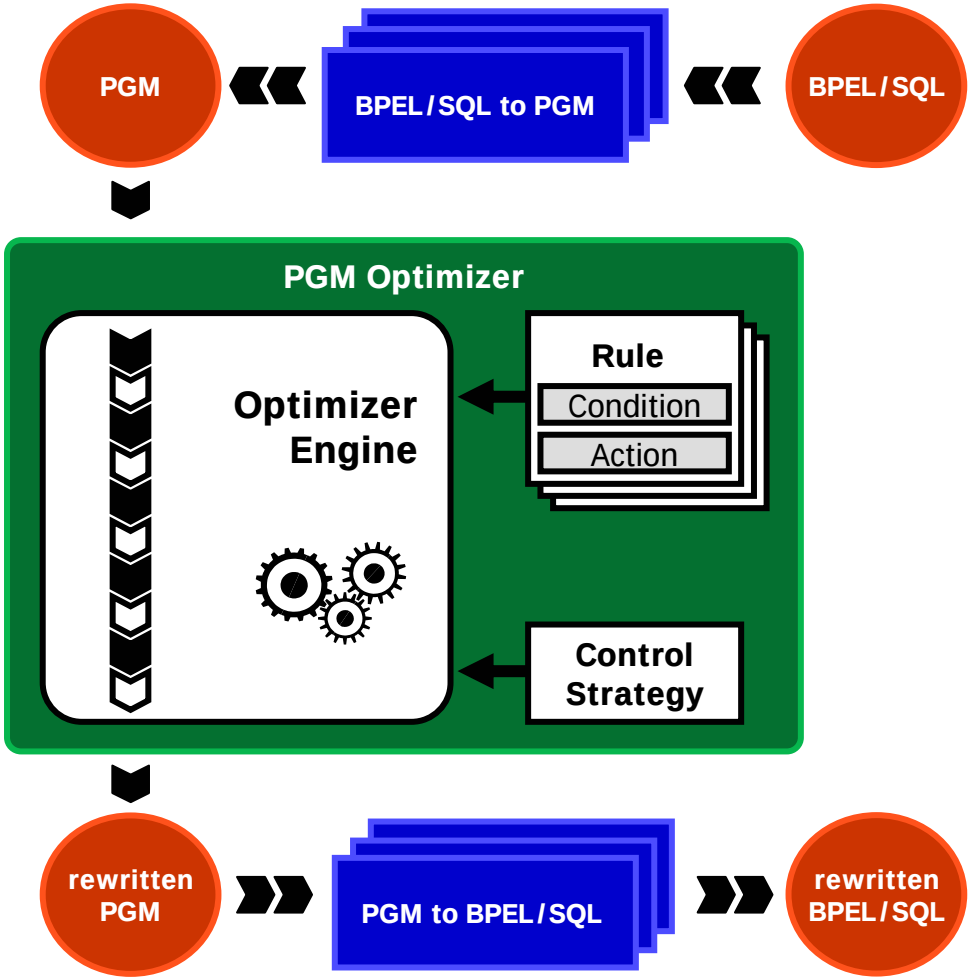


Overall Control Strategy



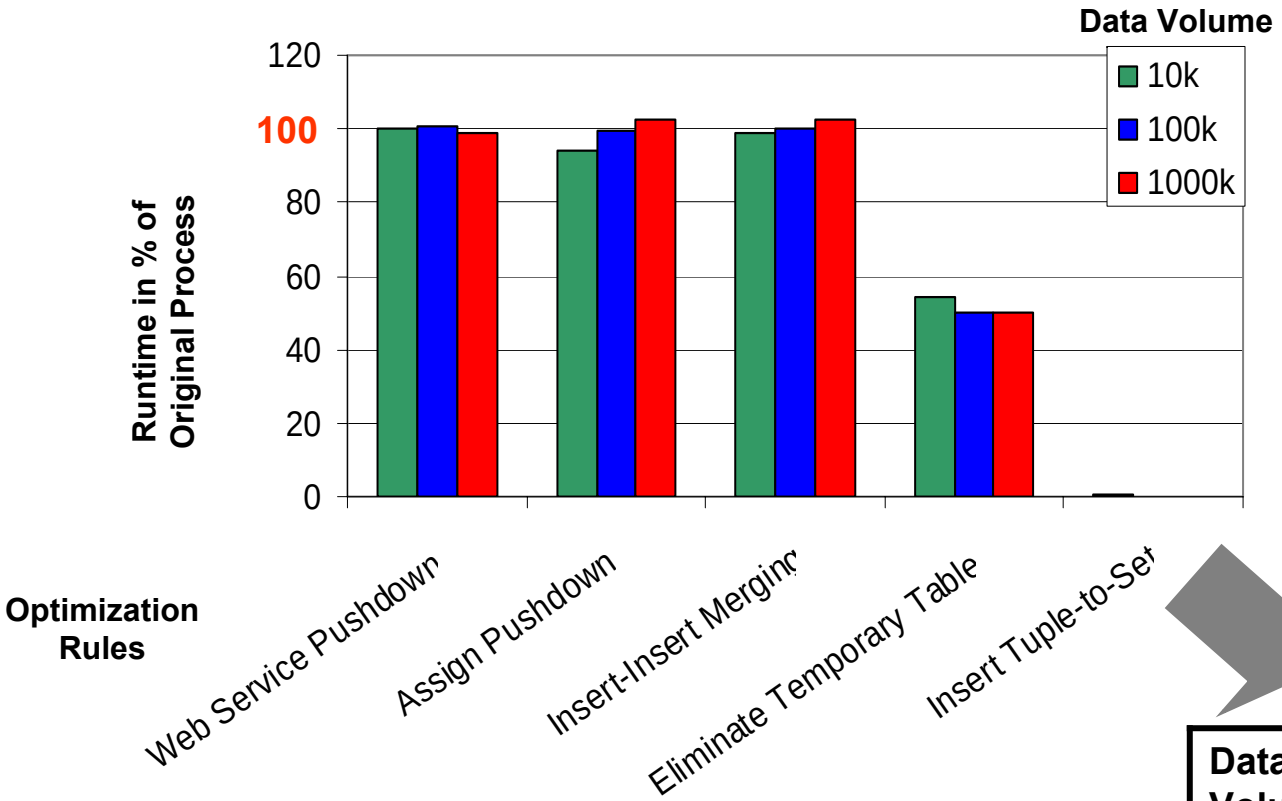
Optimizer Engine

- Generality issues:
  PGM optimizer is independent
  - from a specific workflow language
  - from the underlying database system
- Main steps to adjust the optimizer:
  - provide appropriate mapping components
  - add activity types to PGM
  - add rewrite rules
  - adjust control strategy



Effectiveness of Rewrite Rules

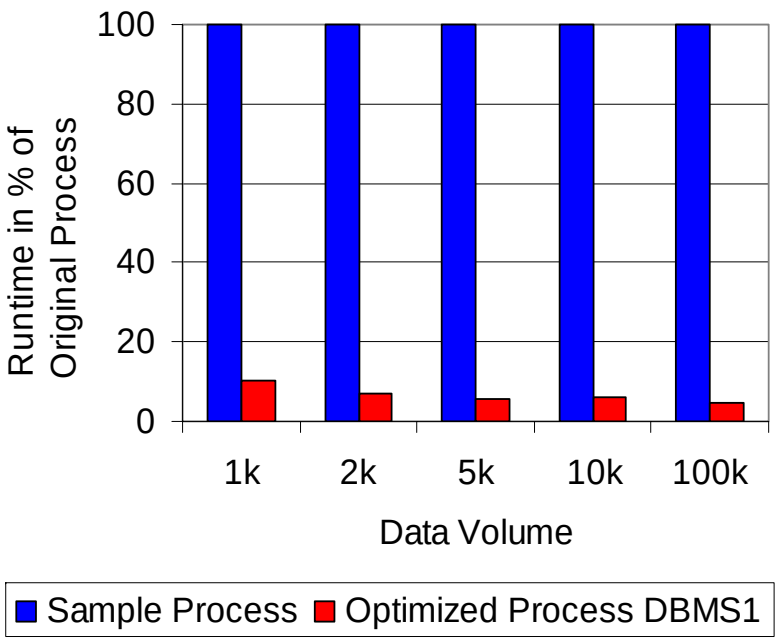
- Sample processes: 1 rewrite step / minimum set of activities

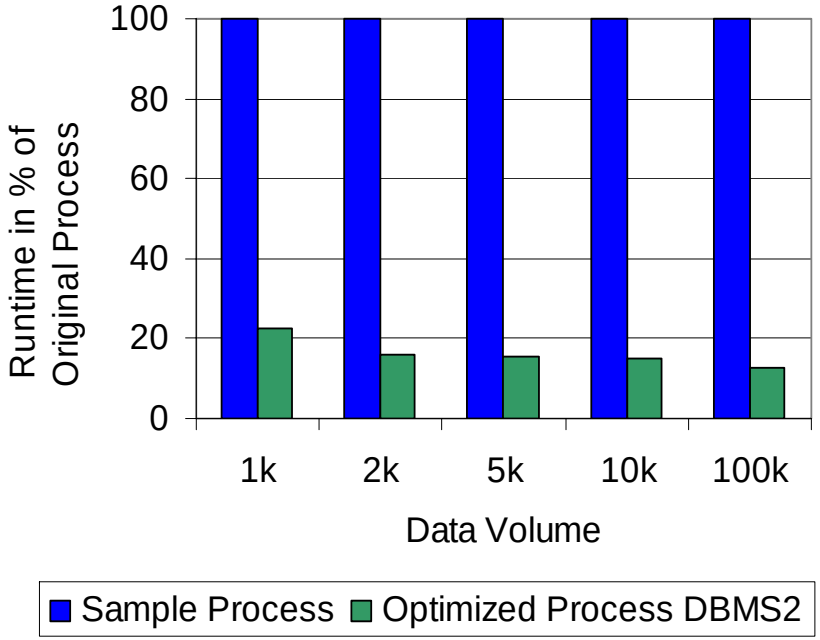


Data Volume	10k	100k	1000k
Factor	245	2090	≈15000

Performance Improvements

- Sample scenario: 5 activities / 3 rewrite steps





- Performance Improvements
  - are also achievable in more complex scenarios
  - are largely independent of table cardinality
  - are independent of underlying database system

Conclusion

- Data management tasks are increasingly treated as first class citizens in workflow languages.
- New optimization opportunities arise.
- Applying rewrite rules to the definition of business processes results in remarkable performance improvements
- Main components of the optimizer engine:
  - set of rewrite rules
  - process graph model as internal representation of workflows
  - control strategy