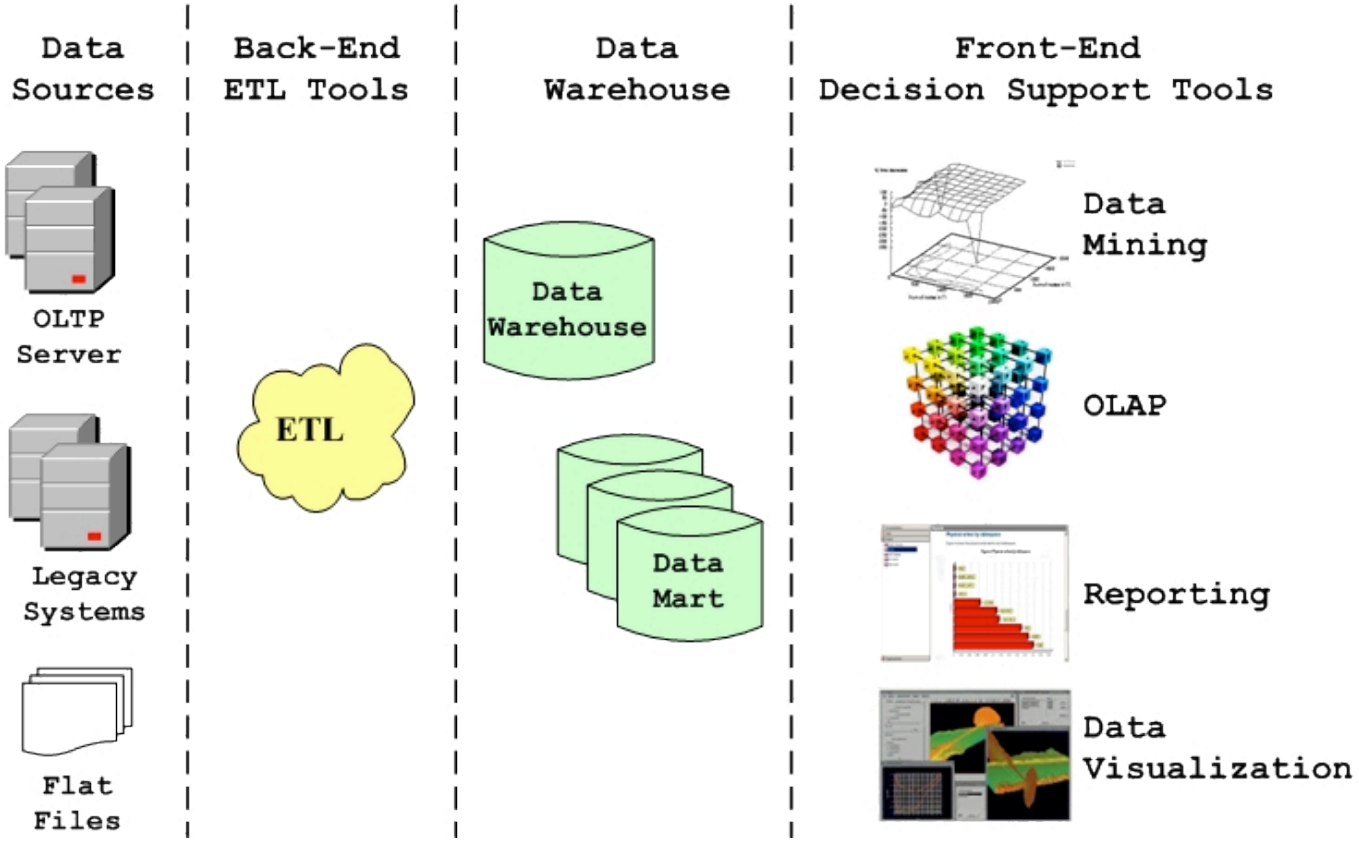
ETL Workflows:
From Formal Specification to Optimization

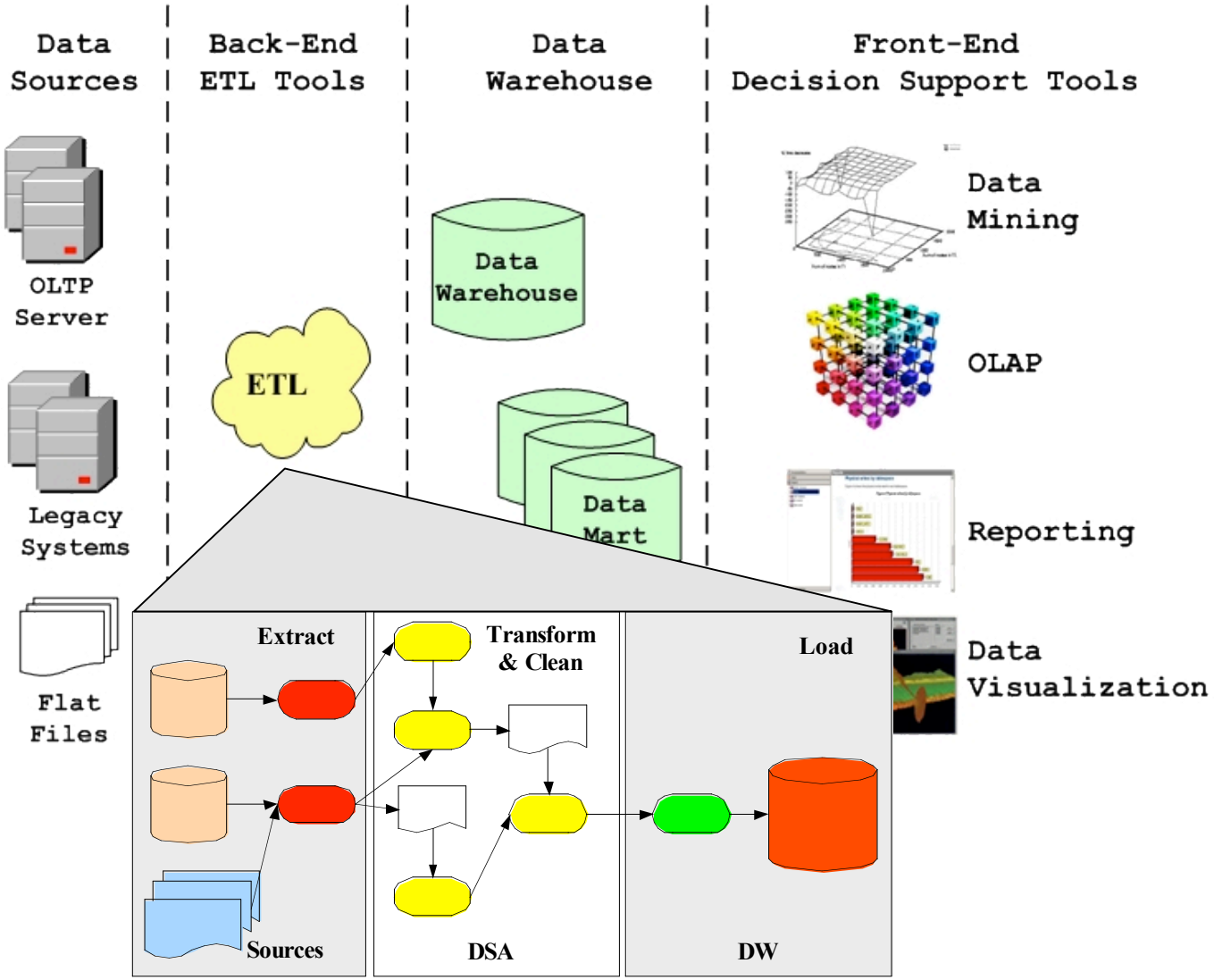
Timos Sellis
Institute of the Management of Information Systems (R.C. “Athena”)
and National Technical University of Athens

(joint work with Alkis Simitsis, Panos Vassiliadis-Univ. of Ioannina
and Dimitris Skoutas-NTUA)

Data Warehouse Environment



Extract-Transform-Load (ETL)



Motivation

- ETL and Data Cleaning tools cost
  - **30%** of effort and expenses in the budget of the DW
  - **55%** of the total costs of DW runtime
  - **80%** of the development time in a DW project
- ETL market: a multi-million market
- ETL tools in the market
  - software packages
  - in-house development
- No standard, no common model
  - most vendors implement a core set of operators and provide GUI to create a data flow

Problems

- The key factors underlying the main problems of ETL processes are:
  - **vastness** of the data volumes
  - **quality problems**, since data is not always clean and has to be cleansed
  - **performance**, since the whole process has to take place within a specific time window
  - **evolution** of the sources and the data warehouse can eventually lead, even to daily maintenance operations

Modeling Work – Why?

- Conceptual
  - we need a simple model, sufficient for the early stages of the data warehouse design; we need to be able to model what our sources “talk” about
- Logical
  - we need to model a workflow that offers formal and semantically founded concepts to capture the characteristics of an ETL process
- Execution
  - we need to find a good execution strategy for ETL processes, not in an ad-hoc way

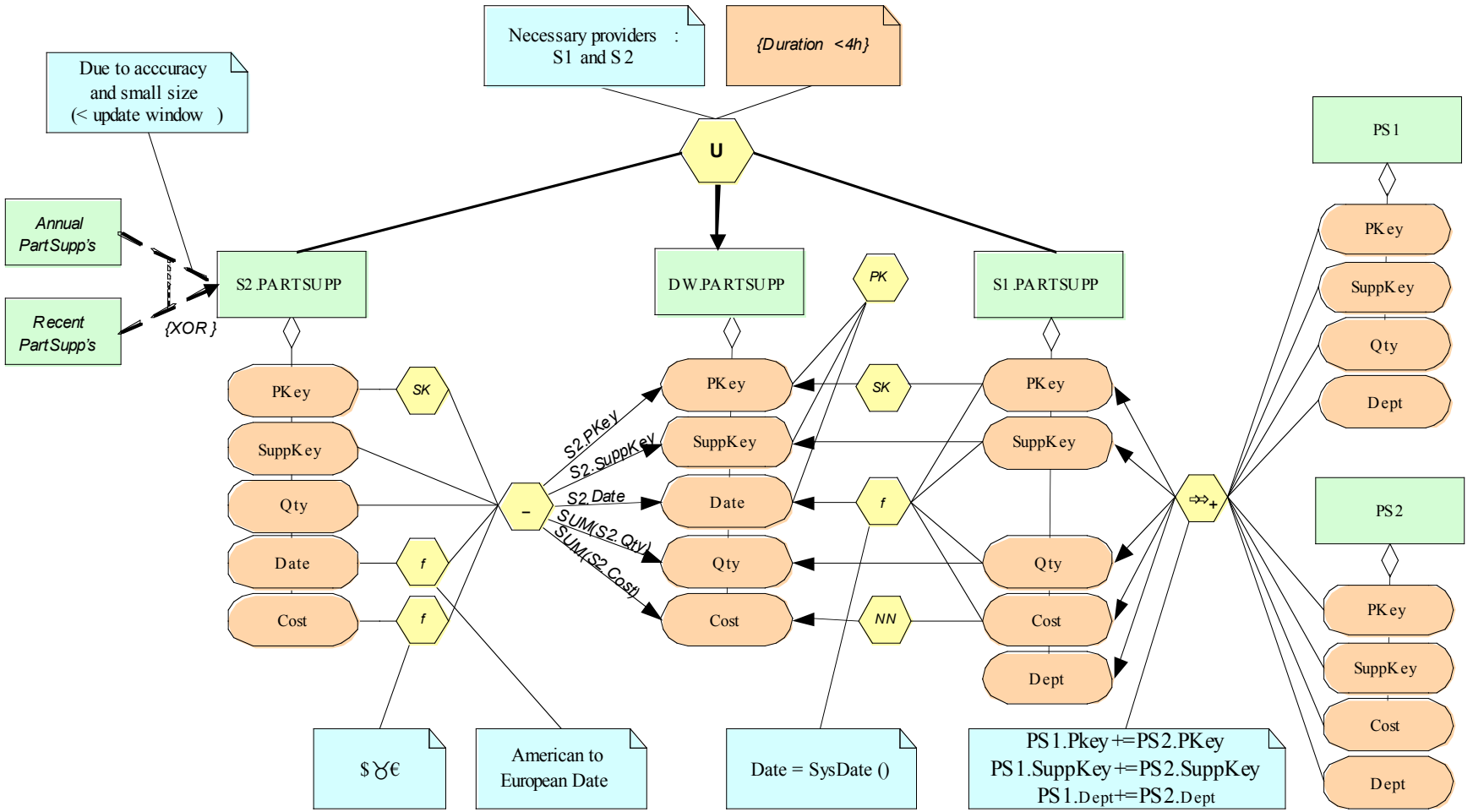
Outline

- **Conceptual Model**
- Logical Model
- Optimization of ETL Workflows
- Research Challenges

Conceptual Model

- Design goals
  - we need a **simple model**, sufficient for the early stages of the data warehouse design
  - we need convenient **means of communication** among different groups of people involved in the DW project (e.g., dba's and business managers)
  - we need to be able to model **what our sources "talk" about**
- Semantic goals
  - we need **richer semantics** to
    - describe sources
    - reason about them

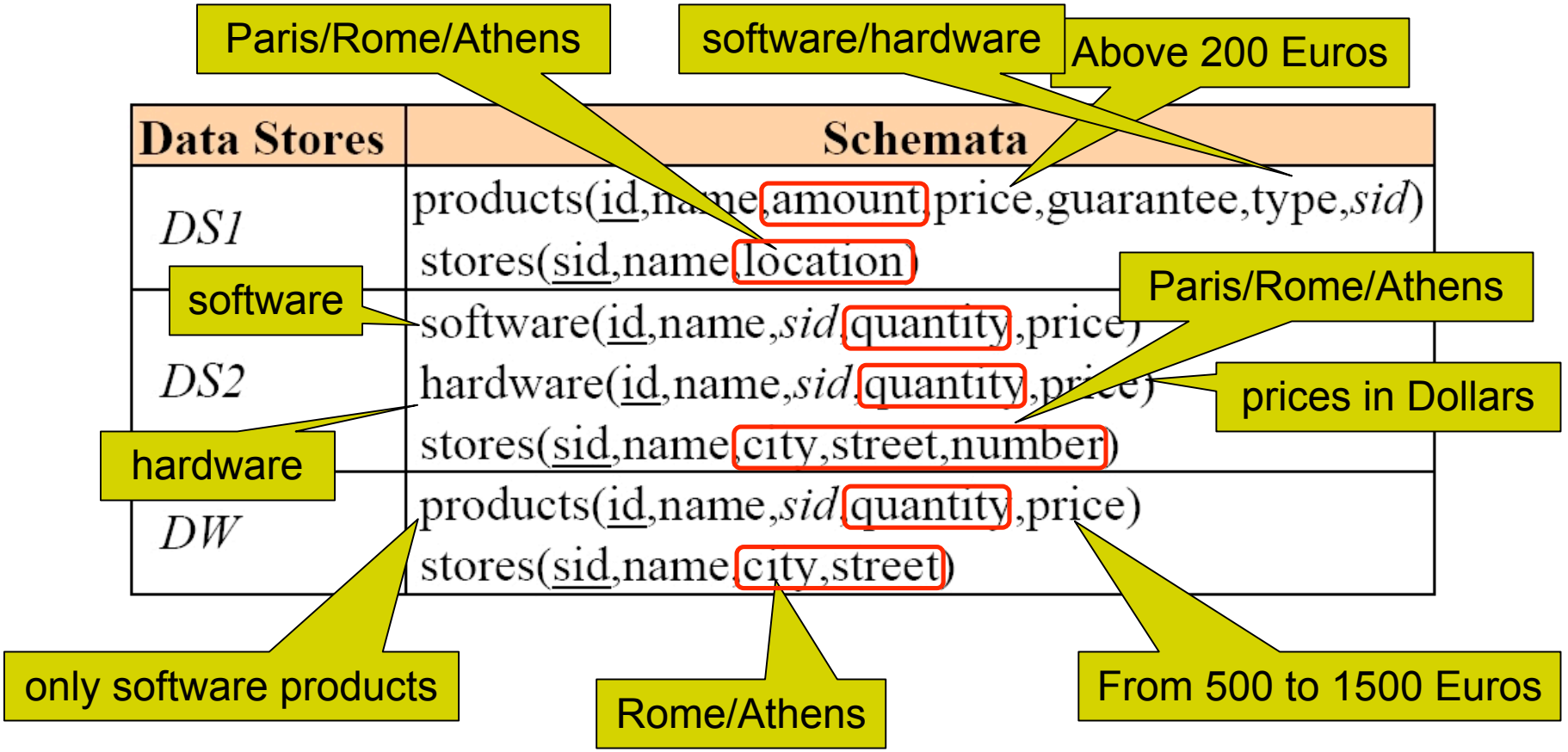
Conceptual Model



Conceptual Model

- Key idea
  - an ontology-based approach to facilitate the conceptual design
- An ontology
  - is a “formal, explicit specification of a shared conceptualization”
  - describes the knowledge in a domain in terms of classes, properties, and relationships between them
  - machine processable
  - formal semantics
  - reasoning mechanisms
- Method
  - construct an appropriate application vocabulary
  - annotate the data sources
  - generate the application ontology
  - apply reasoning techniques to select relevant sources and identify required transformations

Conceptual Model



Conceptual Model

Application vocabulary

V_C = {product, store}
$V_{Pproduct}$ = {pid, pName, quantity, price, type, storage}
V_{Pstore} = {sid, sName, city, street}
V_{Fpid} = {source_pid, dw_pid}
V_{Fsid} = {source_sid, dw_sid}
V_{Fprice} = {dollars, euros}
V_{Ttype} = {software, hardware}
V_{Tcity} = {paris, rome, athens}

Datastore mappings

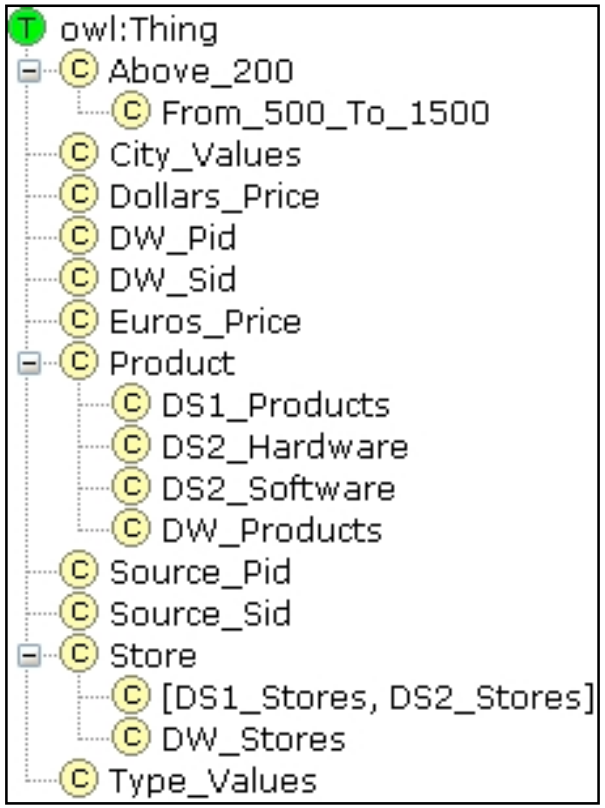
DS1		
products → product	id → pid name → pName amount → quantity	price → price type → type sid → storage
stores → store	sid → sid name → sName	location → city location → street

Datastore annotation

DS1		φ	min	max	T	n	R´	Γ_f	Γ_α
products	I_{pid}	source_pid	-	-	-	1	-	-	-
	I_{pName}	-	-	-	-	1	-	-	-
	$I_{quantity}$	-	-	-	-	-	-	-	-
	I_{price}	euros	200	-	-	1	-	-	-
	I_{type}	-	-	-	{software, hardware}	1	-	-	-
	$I_{storage}$	-	-	-	-	1	store	-	-
stores	I_{sid}	source_sid	-	-	-	1	-	-	-
	I_{sName}	-	-	-	-	1	-	-	-
	I_{city}	-	-	-	{paris, rome, athens}	1	-	-	-
	I_{street}	-	-	-	-	1	-	-	-

Conceptual Model

- The class hierarchy



- Definition for class DS1_Products

OWL-Class: DS1_Products

Intersection of:
(∀pid . Source_Pid)
Product
(= 1 storage)
(∀storage . DS1_Stores)
(= 1 pid)
(= 1 pName)
(= 1 price)
(∀type . ((∃hasValue . {hardware}) ∪ (∃hasValue . {software})))
(∀price . (Euros_Price ∩ Above_200))
(= 1 type)

Conceptual Model

- Reasoning on the mappings

c(street, number, street)

DS1		
products → product	id → pid name → pName amount → quantity	price → price type → type sid → storage
stores → store	sid → sid name → sName	location → city location → street

DS2		
software → product	id → pid name → pName sid → storage	quantity →quantity price → price
hardware → product	id → pid name → pName sid → storage	quantity →quantity price → price
stores → store	sid → sid name → sName	city → city street → street number → street

Conceptual Model

- Reasoning on the definitions

σ(type, {software})

OWL-Class: DS1_Products

Intersection of:
(∀pid . Source_Pid)
Product
(= 1 storage)
(∀storage . DS1_Stores)
(= 1 pid)
(= 1 pName)
(= 1 price)
(∀type . ((∃hasValue . {hardware})
∪ (∃hasValue . {software})))
(∀price . (Euros_Price ∩
Above_200))
(= 1 type)

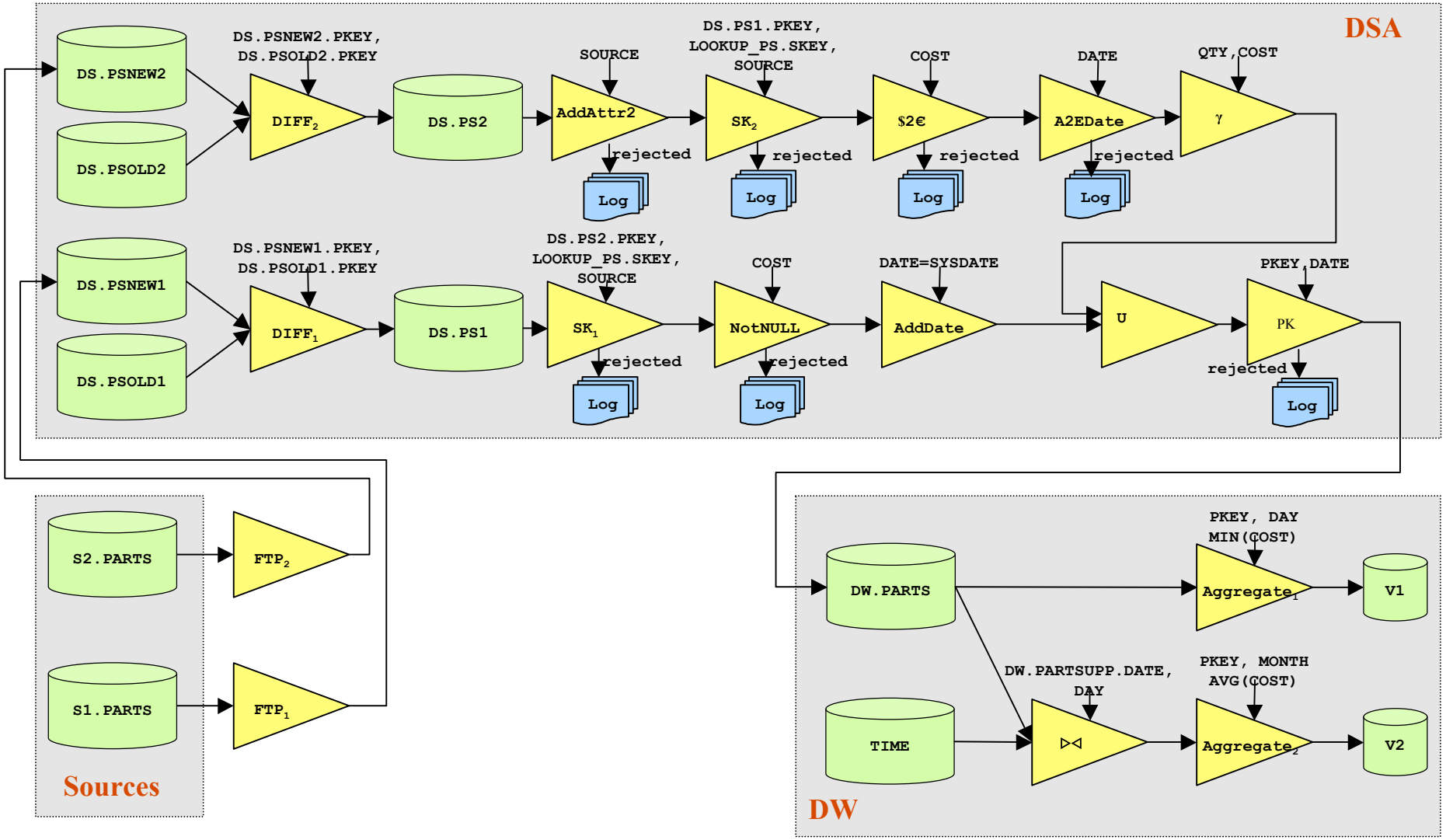
OWL-Class: DW_Products

Intersection of:
(= 1 quantity)
(∀pid . DW_Pid)
Product
(∀price . (Euros_Price ∩
From_500_To_1500))
(= 1 storage)
(= 1 pid)
(= 1 pName)
(= 1 price)
(= 1 type)
(∀type . (∃hasValue . {software}))
(∀storage . DW_Stores)

Outline

- Conceptual Model
- **Logical Model**
- Optimization of ETL Workflows
- Research Challenges

Logical Model



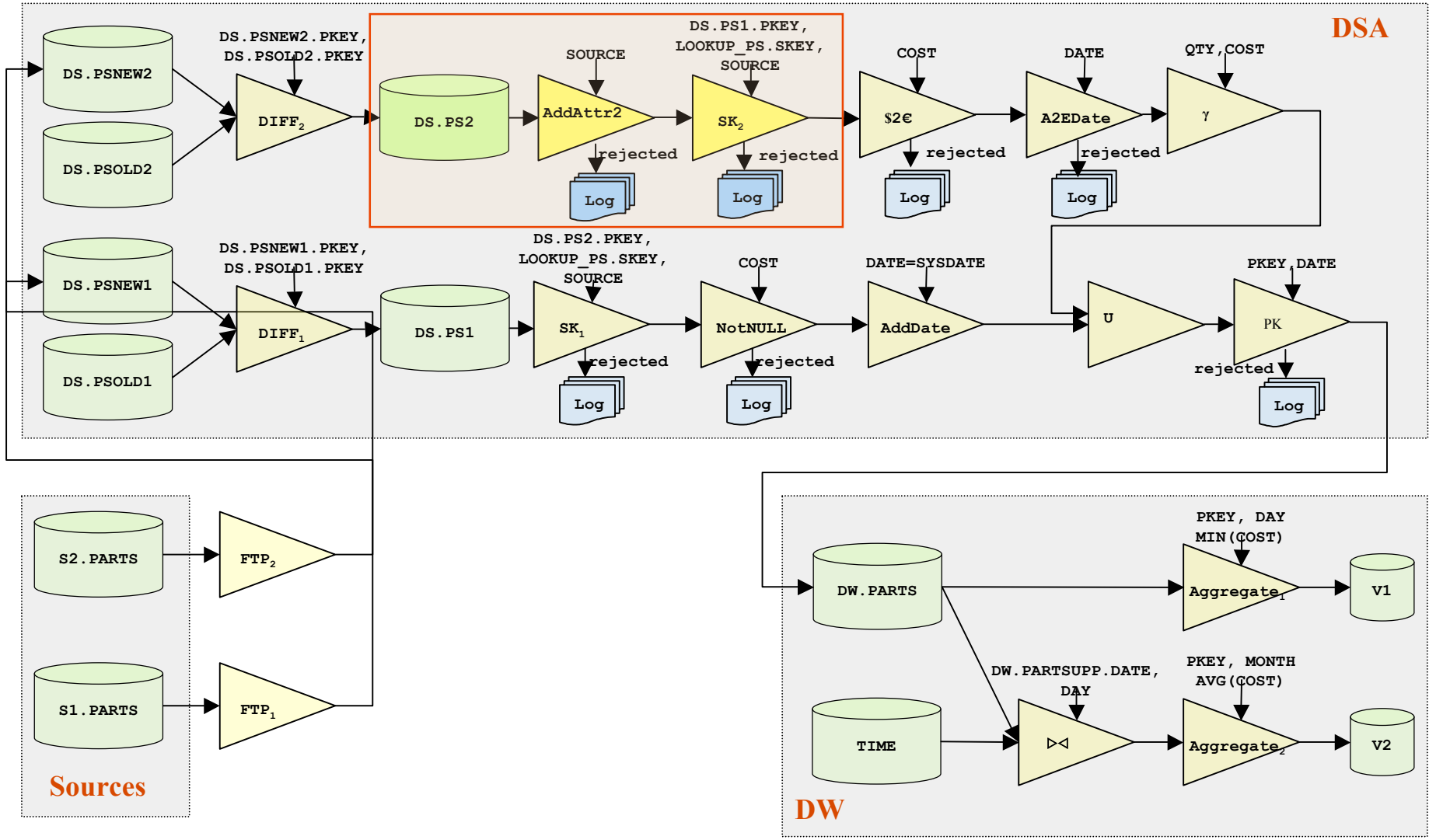
Logical Model

- *Main question*:

  What information should we put inside a metadata repository to be able to answer questions like:

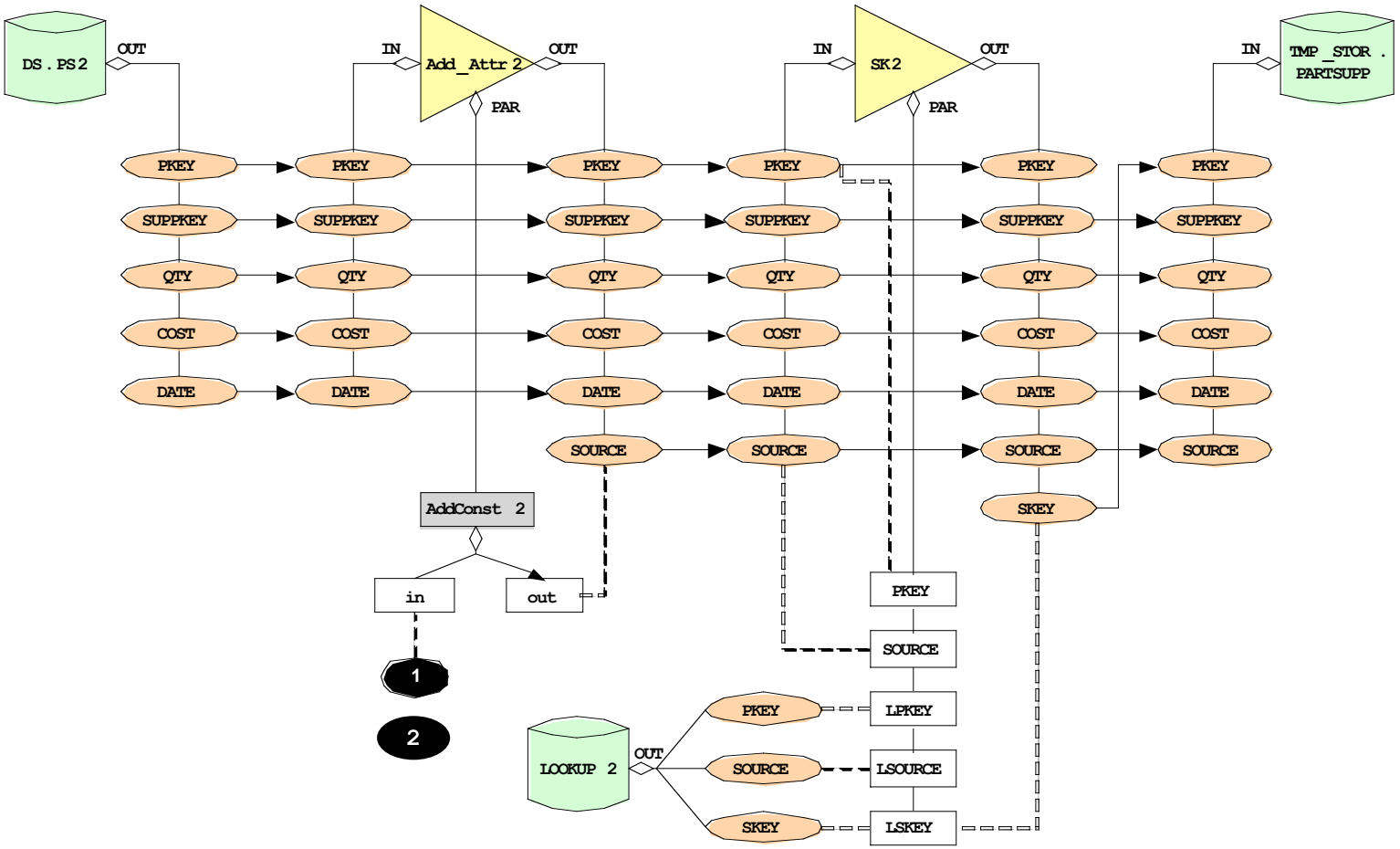
  - what is the architecture of my DW back stage?
  - which attributes/tables are involved in the population of an attribute?
  - what part of the scenario is affected if we delete an attribute?

Architecture Graph



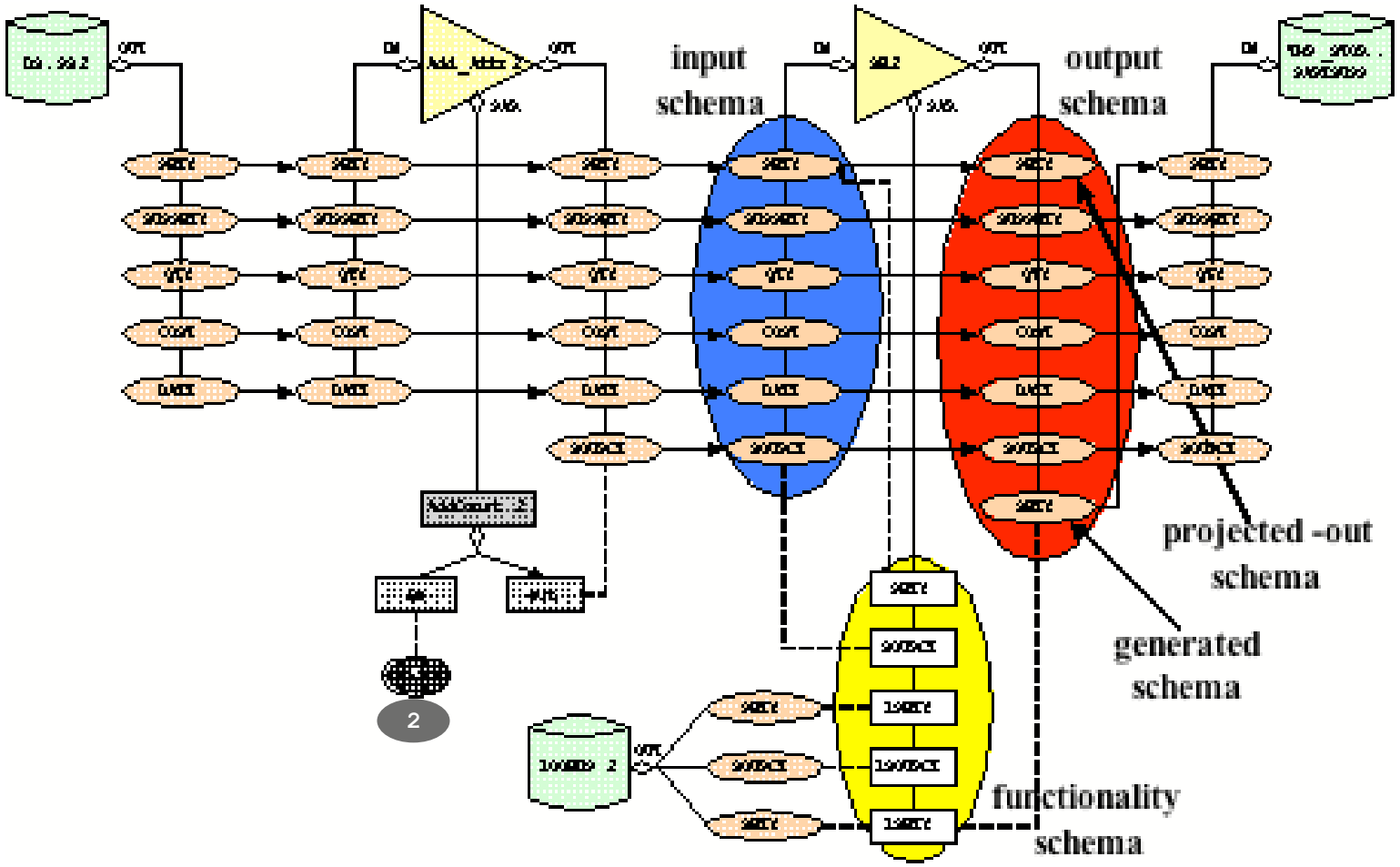
Architecture Graph

Example



Architecture Graph

Example



Semantics

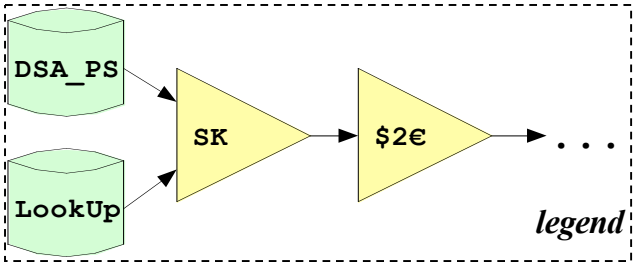
- The semantics of each activity is given by a declarative program expressed in LDL++
  - each program is a finite list of LDL++ rules
  - each rule is identified by an (internal) rule identifier

- We consider three types of programs
  - intra-activity programs
    - characterize the operational semantics, i.e., the internals of activities
  - inter-activity programs
    - link the input/output of an activity to a data provider/consumer
  - side-effects programs
    - characterize whether the provision of data is an insert, update, or delete action

Semantics

Example

- LDL++ for a small part of a scenario

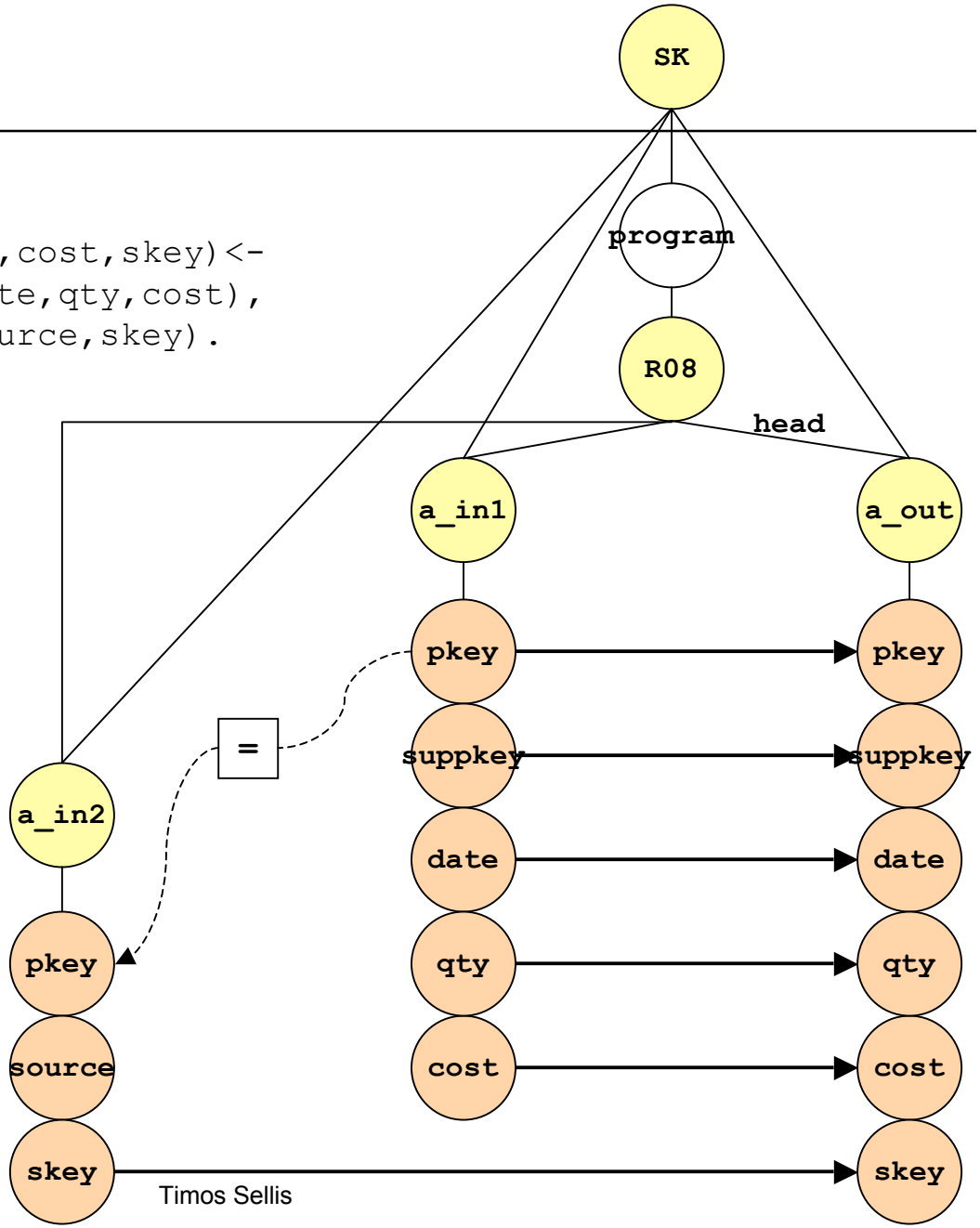



```
R06: sk.a_in1(pkey,suppkey,date,qty,cost)<-
         dsa_ps(pkey,suppkey,date,qty,cost).

R07: sk.a_in2(pkey,source,skey)<-
         lookUp(l_pkey,source,l_skey),
         pkey=l_pkey,
         skey=l_skey,
         source=1.

R08: sk.a_out(pkey,suppkey,date,qty,cost,skey)<-
         add_sk1.a_in1(pkey,date,qty,cost),
         add_sk1.a_in2(pkey,source,skey).

R09: dollar2euro.a_in(skey,suppkey,date,qty,cost)<-
         sk.a_out(pkey,suppkey,date,qty,cost,skey).
```

Semantics

R08:

```
sk.a_out(pkey,suppkey,date,qty,cost,skey)<-
          add_sk1.a_in1(pkey,date,qty,cost),
          add_sk1.a_in2(pkey,source,skey).
```



Semantics

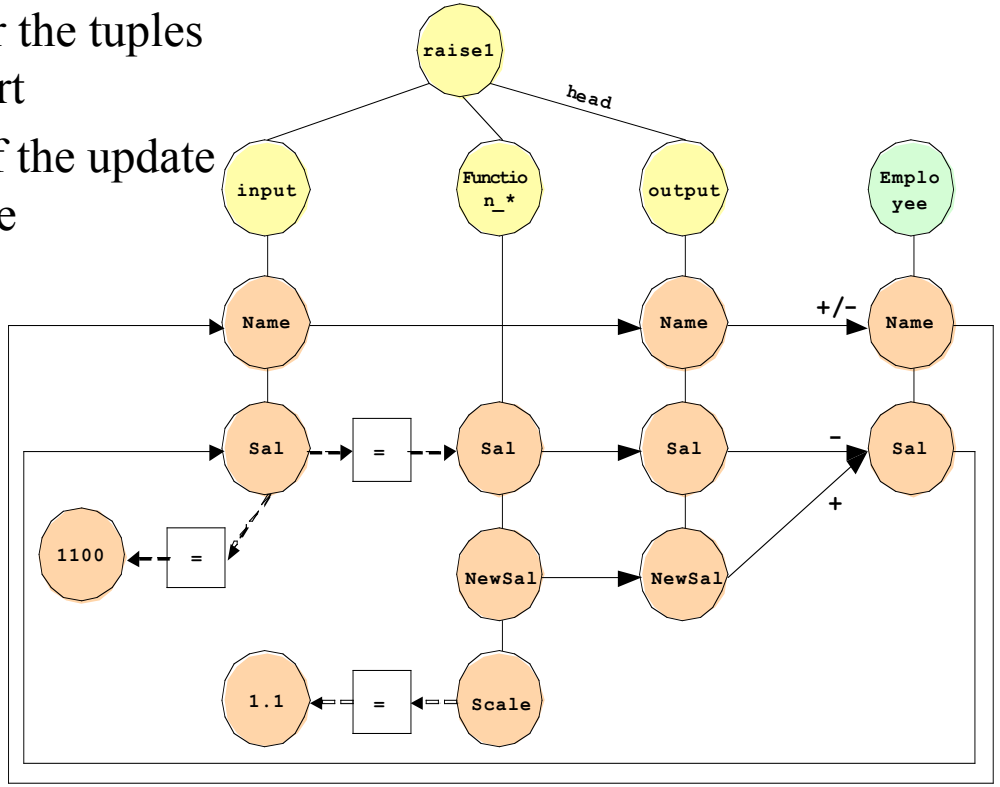
- We provide graph modeling techniques for several kinds of activities:
  - update (INS, UPD, DEL) activities
  - aggregates
  - rules employing negation and aliases
  - functions

Semantics

- Updates to the database
  - An update expression is of the form

    `head <- query part, update part`

    with the following semantics:
    - a query to the database for the tuples that abide by the query part
    - we update the predicate of the update part as specified in the rule




```
raise1(Name, Sal, NewSal) <-
  employee(Name, Sal),Sal=1100,   (a)
  NewSal = Sal * 1.1,             (b)
  - employee(Name, Sal),          (c)
  + employee(Name, NewSal).       (d)
```

Logical Model

- *Question revisited*

  What information should we put inside a metadata repository to be able to answer questions like:

  - what is the architecture of my DW back stage?
    - it is described as the Architecture Graph
  - which attributes/tables are involved in the population of an attribute?
  - what part of the scenario is affected if we delete an attribute?
    - follow the appropriate path in the Architecture Graph

Outline

- Conceptual Model
- Logical Model
- **Optimization of ETL Workflows**
- Research Challenges

Optimization of ETL Workflows

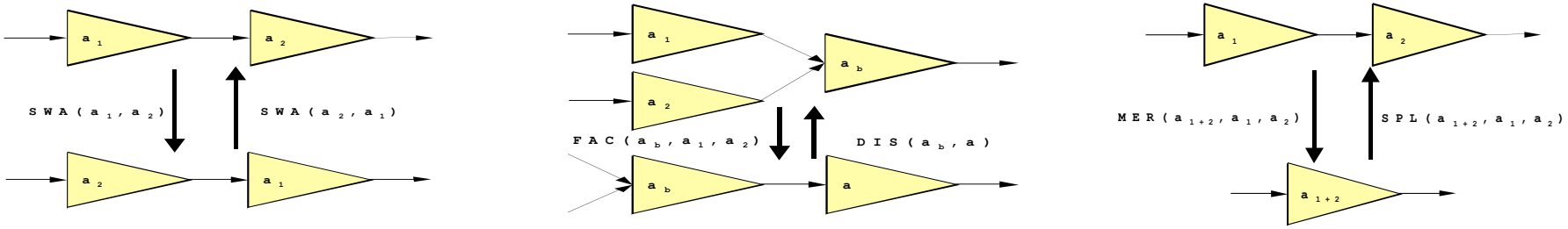
- Common settlement
  - ad-hoc optimization based on the experience of the designer
  - execute ETL workflow as it is; hopefully, the optimizer of the DBMS would improve the performance
- An ETL workflow is **NOT** a *big query*
  - ETL is very procedural in nature, each ETL operator is a low-level operator in procedural languages like PL/SQL
- Traditional query optimization techniques are not enough
  - existence of functions
    - where it is allowed to push an activity before/after a function?
  - existence of black-box activities
    - unknown semantics
    - can not interfere in their interior
  - naming conflicts

Optimization of ETL Workflows

- How can we improve an ETL workflow in terms of execution time?
- We model the ETL processes optimization problem as a state search problem
  - we consider each ETL workflow as a state
  - we construct the search space
  - the optimal state is chosen according to our cost model's criteria, in order to minimize the execution time of an ETL workflow

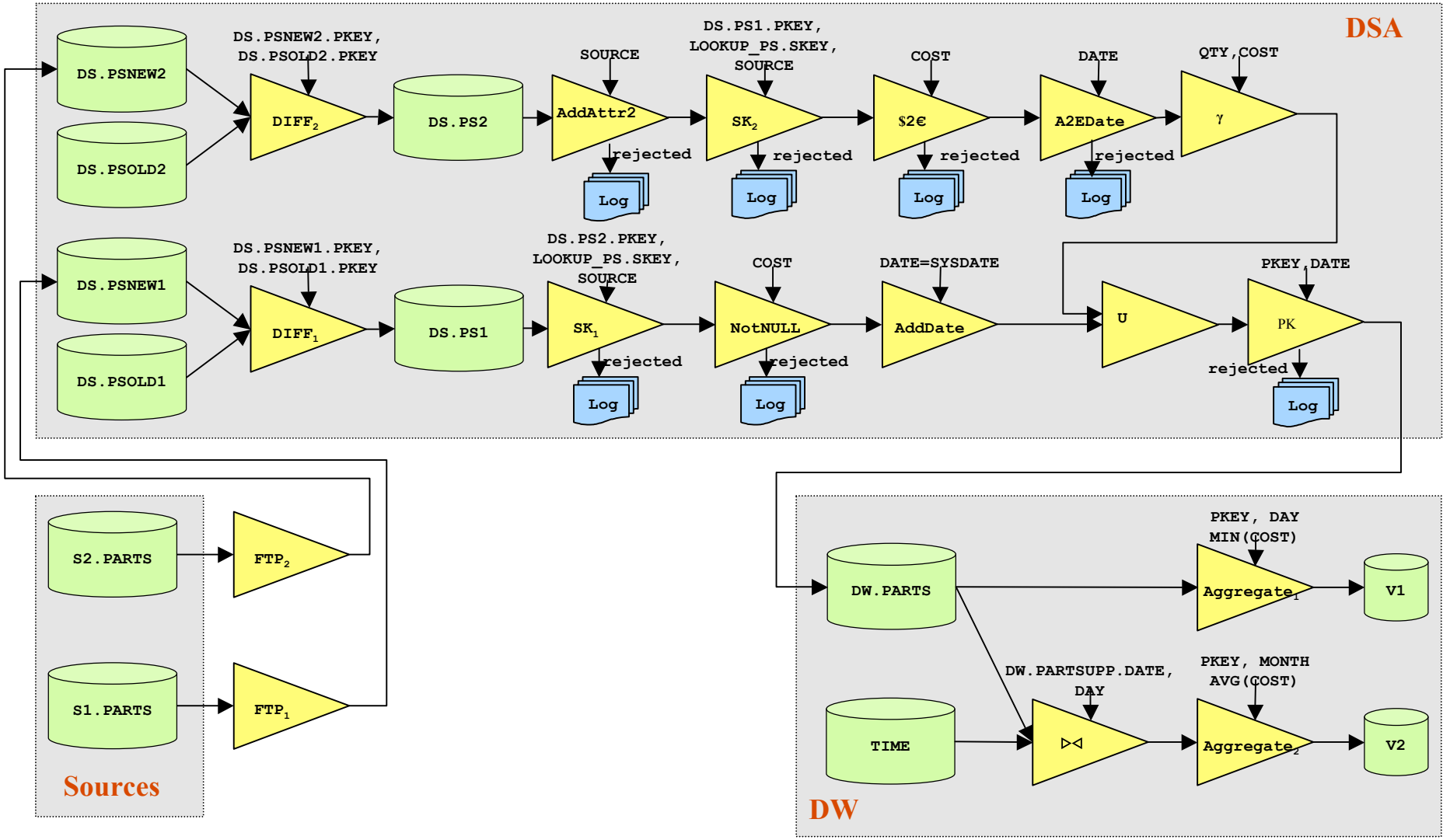
Optimization of ETL Workflows

- Transition from one state to the other
  - SWA: interchange two activities of the workflow
  - FAC: replace homologous tasks in parallel flows with an equivalent task over a flow to which these parallel flows converge
  - DIS: divide tasks of a joint flow to clones applied to parallel flows that converge towards the joint flow
  - MER / SPL: merge / split group of activities



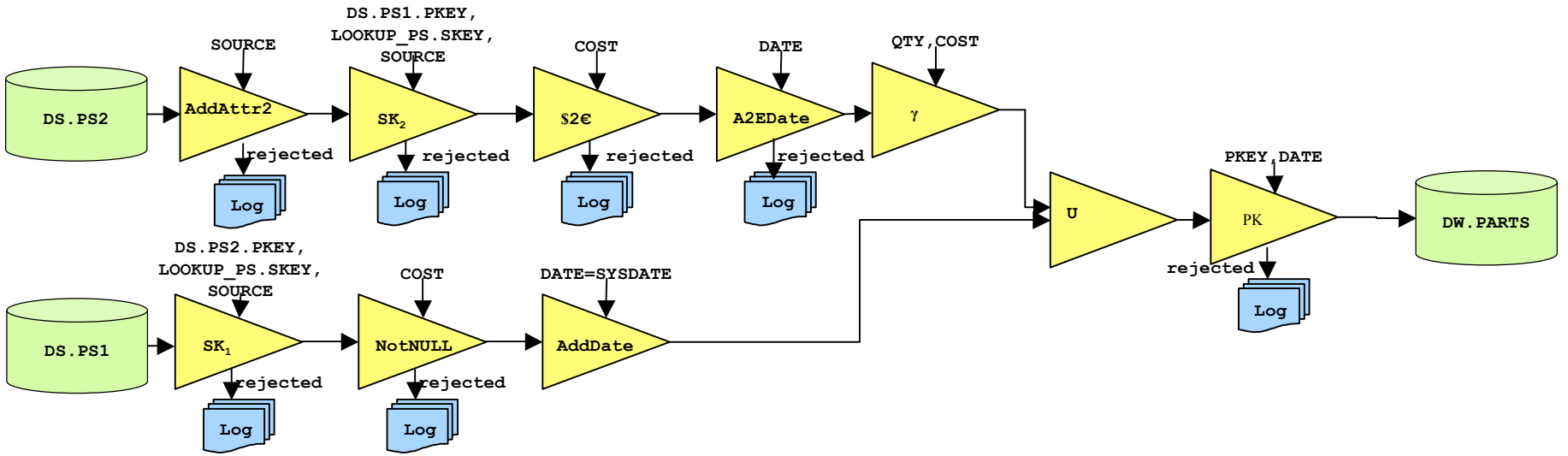
a, a_1, a_2: homologous activities

Optimization of ETL Workflows



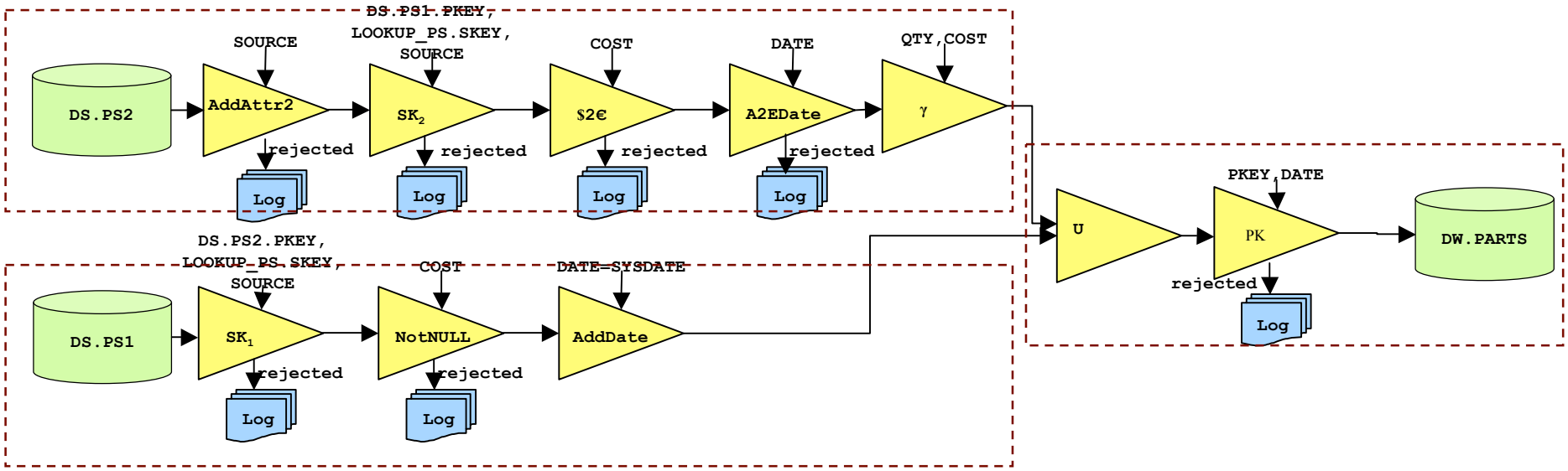
Optimization of ETL Workflows

Pre-Processing → Phase I: SWA's in local groups → Phase II: FAC's homologous → Phase III: DIS's → Phase IV: SWA's in local groups → Post-Processing



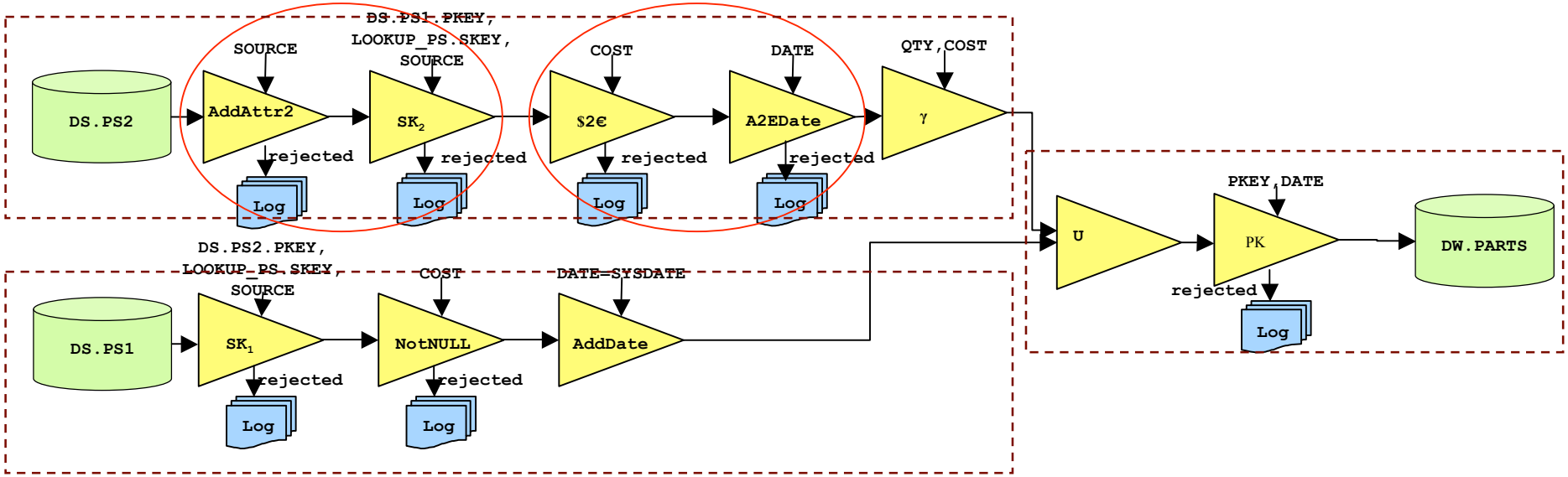
Optimization of ETL Workflows

Pre-Processing → Phase I: SWA's in local groups → Phase II: FAC's homologous → Phase III: DIS's → Phase IV: SWA's in local groups → Post-Processing



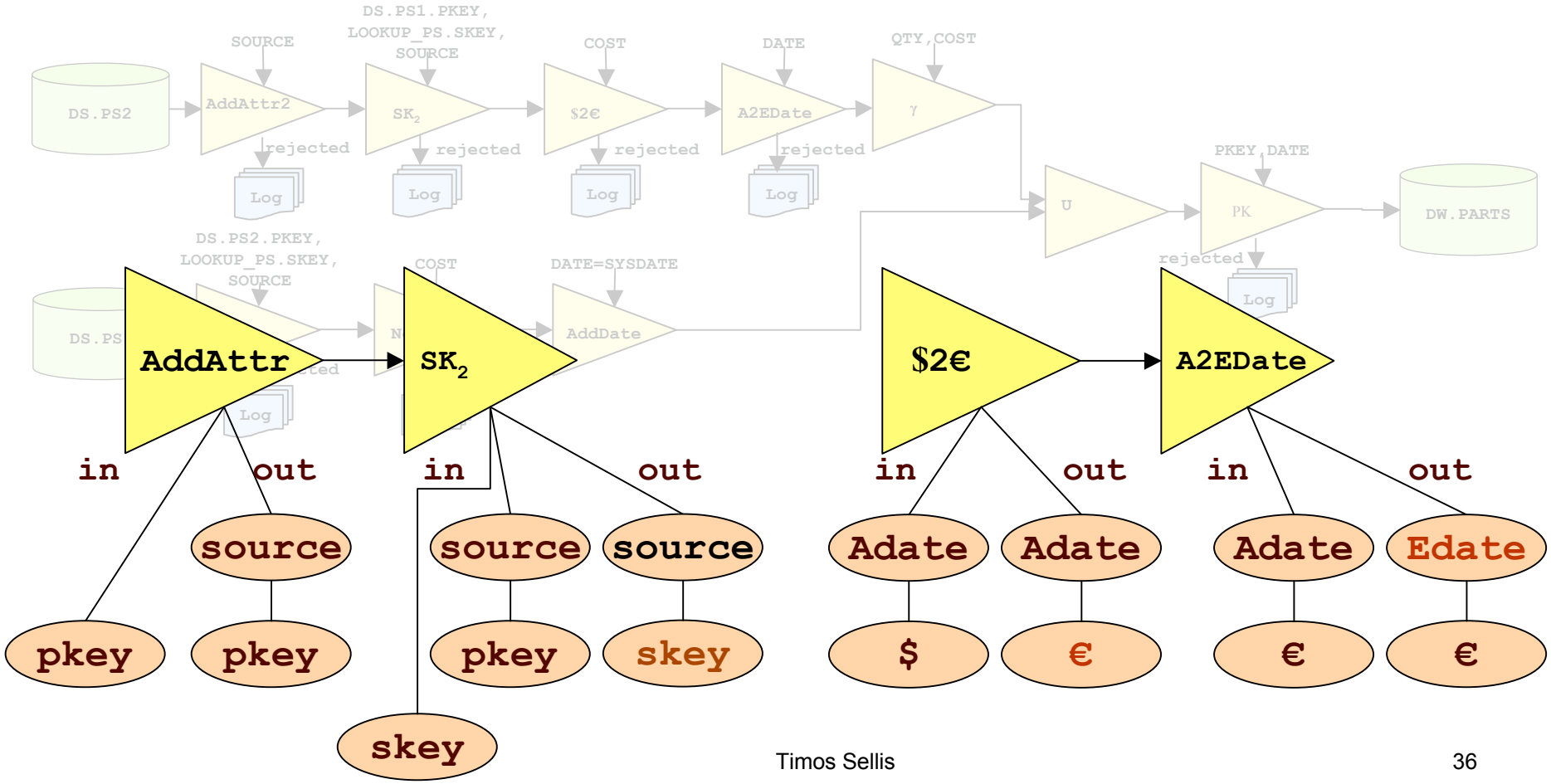
Optimization of ETL Workflows

Pre-Processing → Phase I: SWA's in local groups → Phase II: FAC's homologous → Phase III: DIS's → Phase IV: SWA's in local groups → Post-Processing



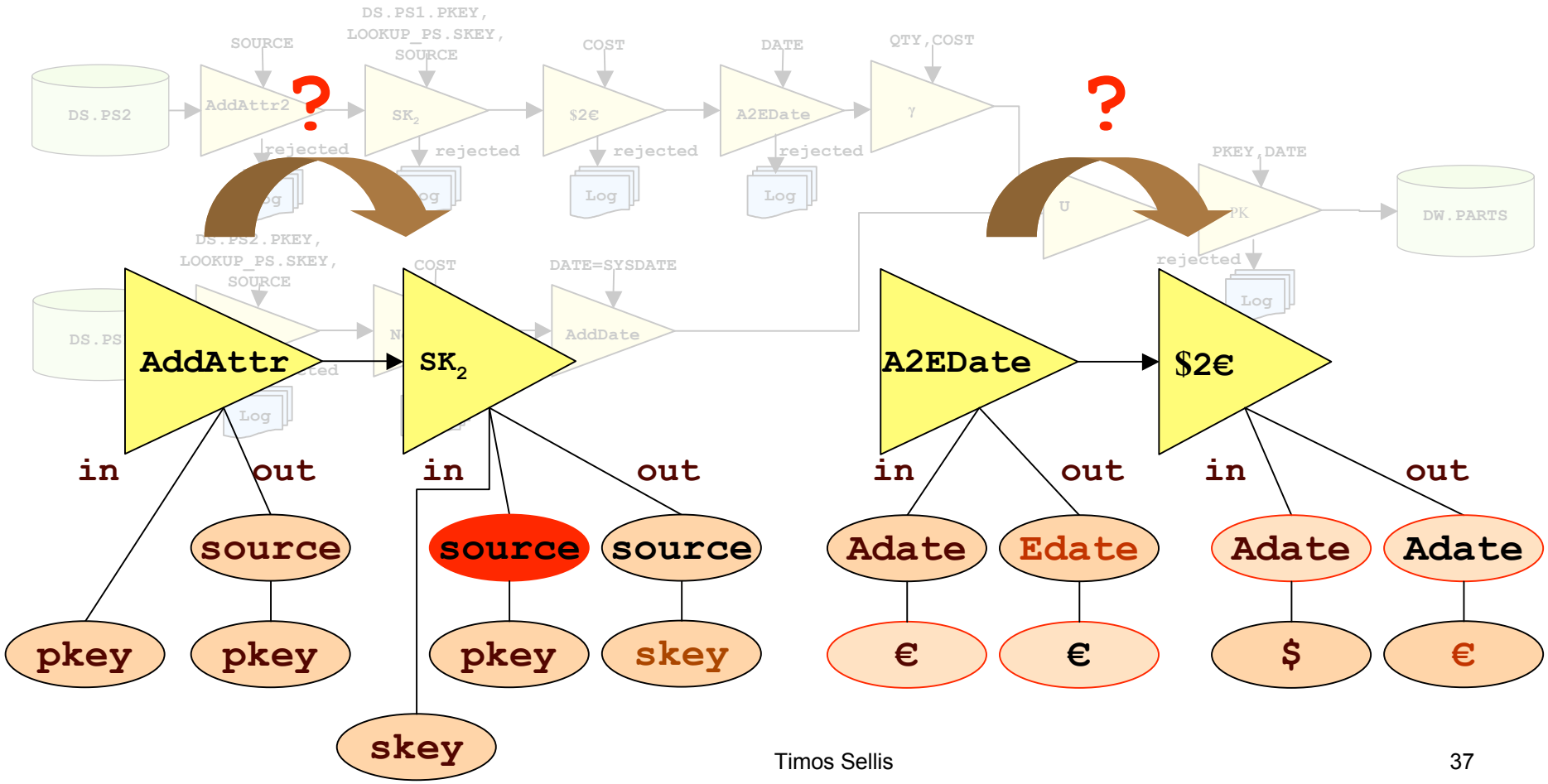
Optimization of ETL Workflows

Pre-Processing → Phase I: SWA's in local groups → Phase II: FAC's homologous → Phase III: DIS's → Phase IV: SWA's in local groups → Post-Processing



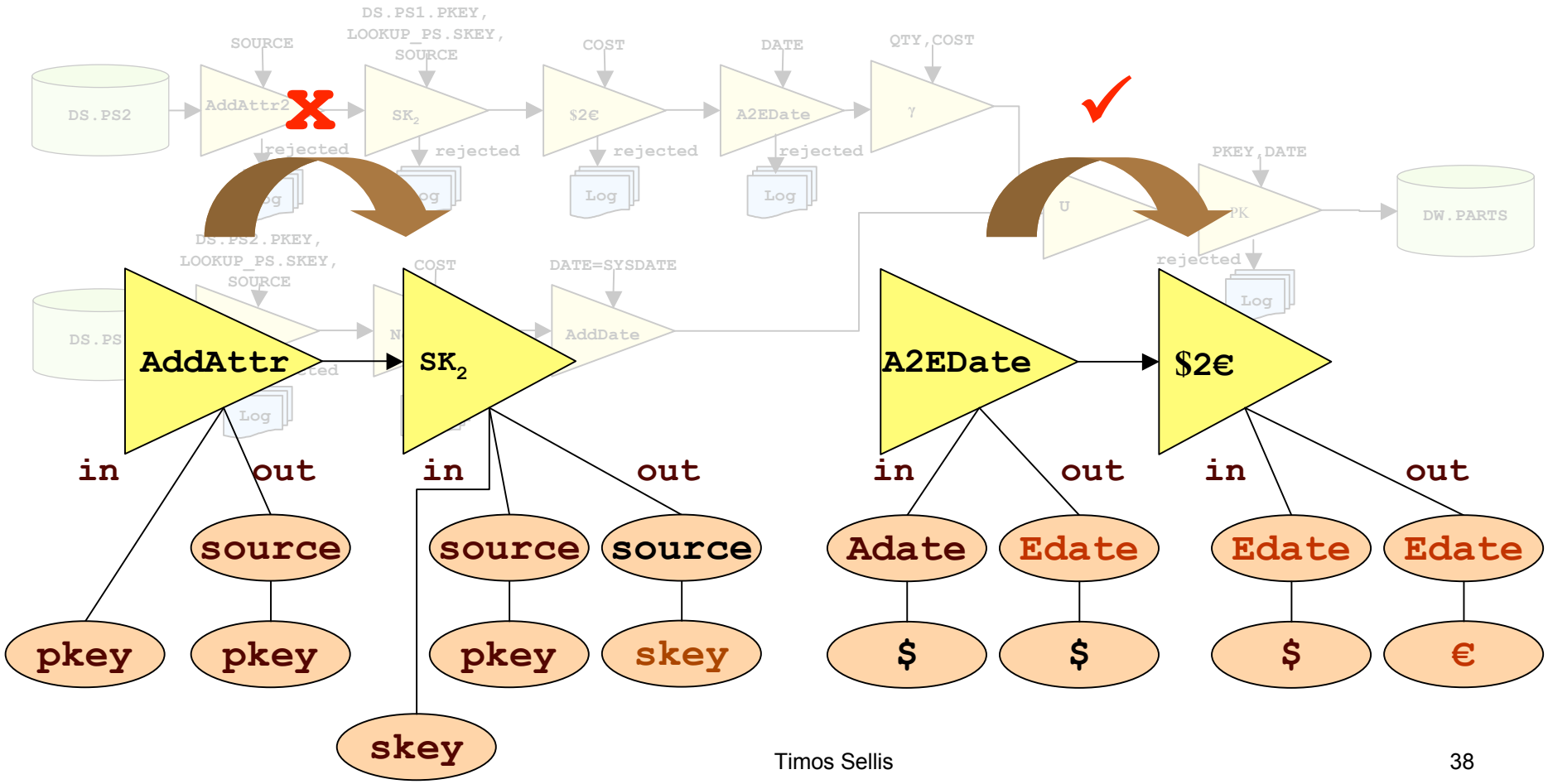
Optimization of ETL Workflows

Pre-Processing → Phase I: SWA's in local groups → Phase II: FAC's homologous → Phase III: DIS's → Phase IV: SWA's in local groups → Post-Processing



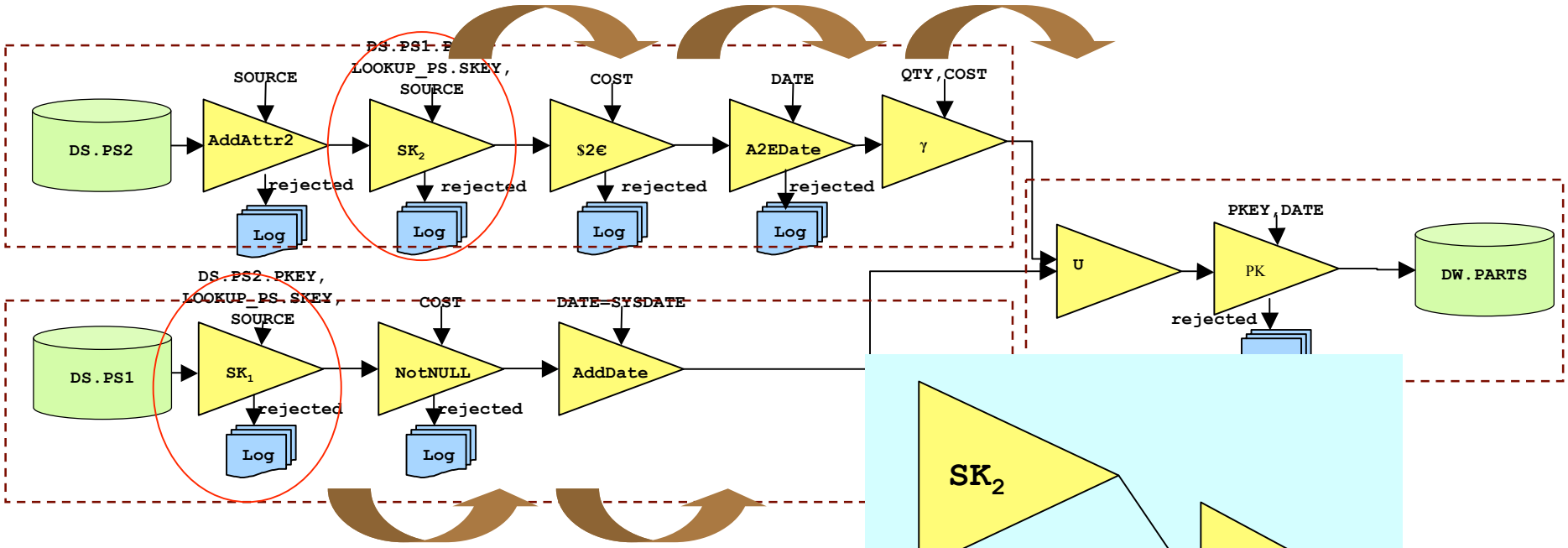
Optimization of ETL Workflows

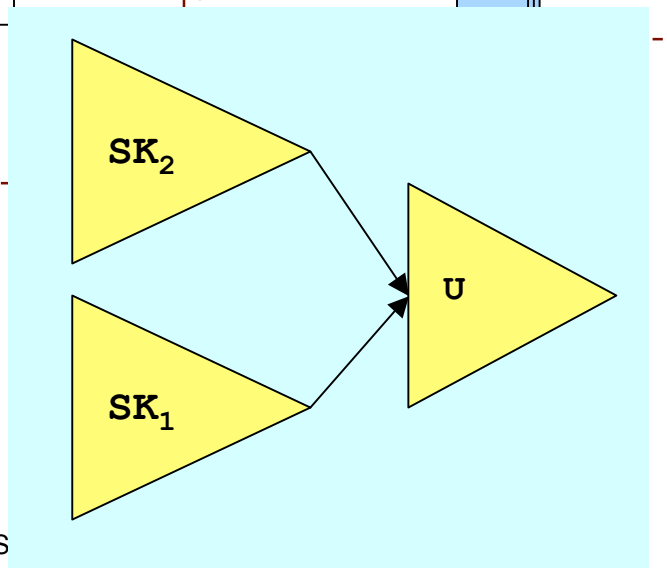
Pre-Processing → Phase I: SWA's in local groups → Phase II: FAC's homologous → Phase III: DIS's → Phase IV: SWA's in local groups → Post-Processing



Optimization of ETL Workflows

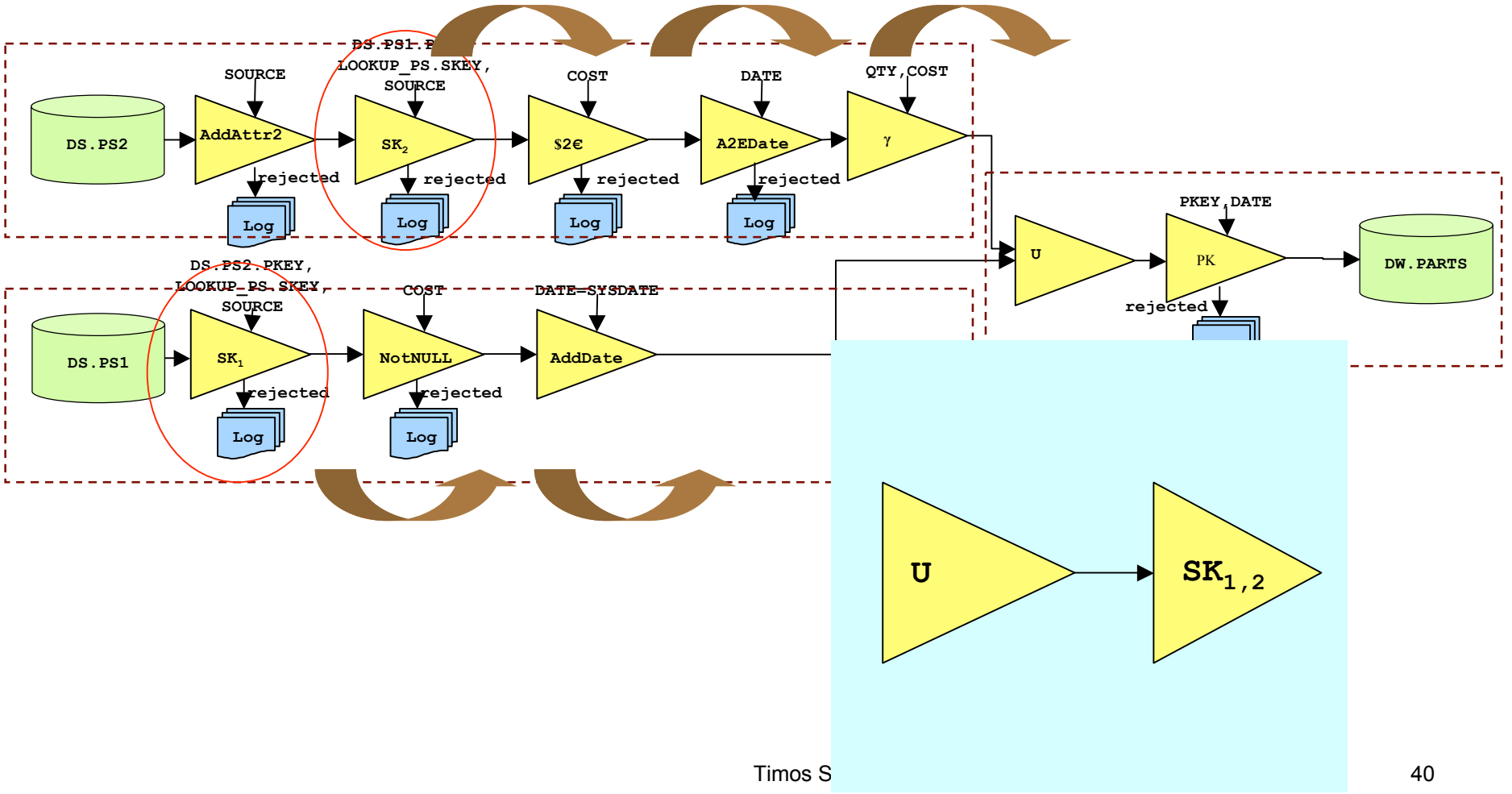
Pre-Processing → Phase I: SWA's in local groups → Phase II: FAC's homologous → Phase III: DIS's → Phase IV: SWA's in local groups → Post-Processing





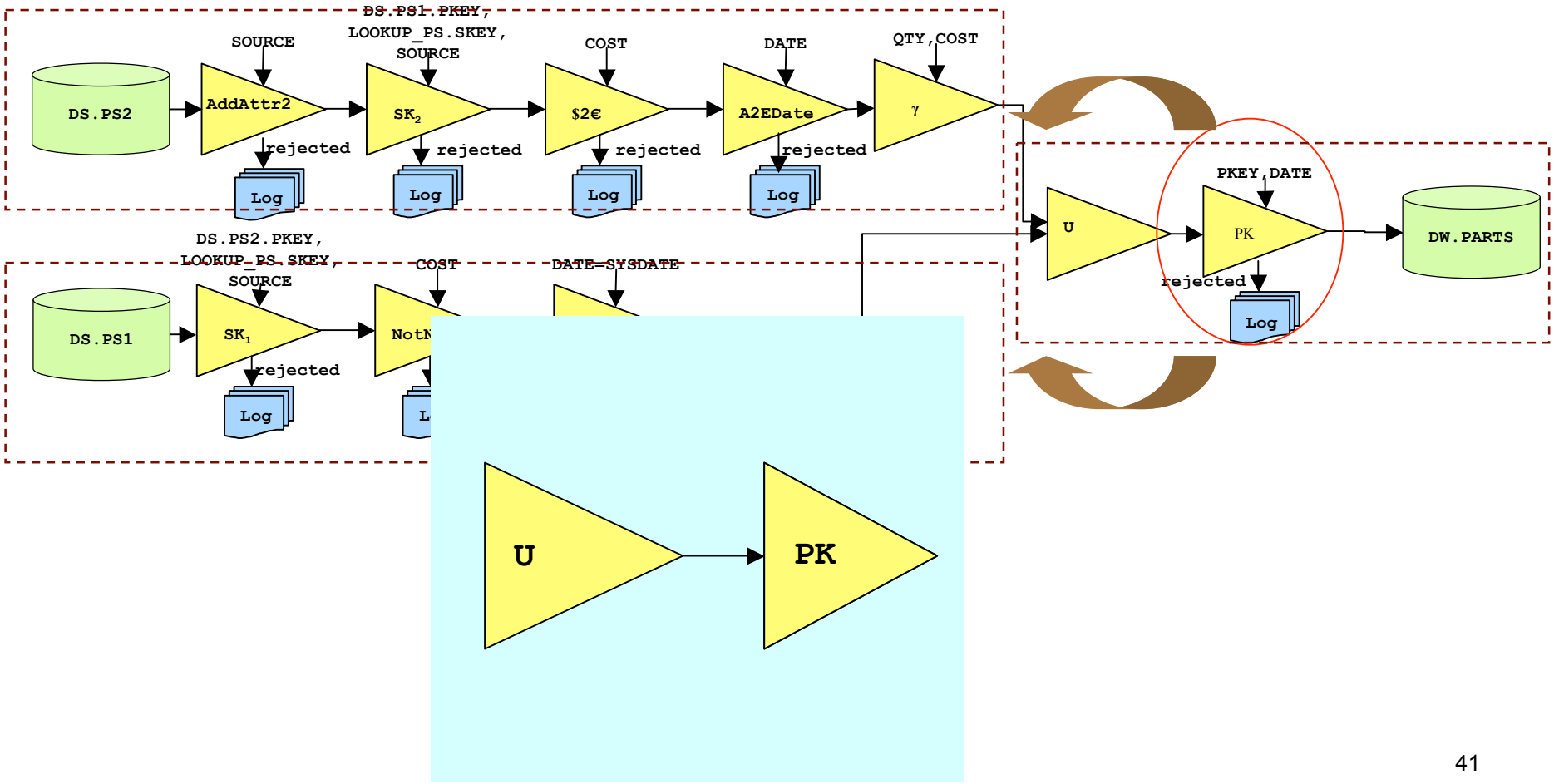
Optimization of ETL Workflows

Pre-Processing → Phase I: SWA's in local groups → Phase II: FAC's homologous → Phase III: DIS's → Phase IV: SWA's in local groups → Post-Processing



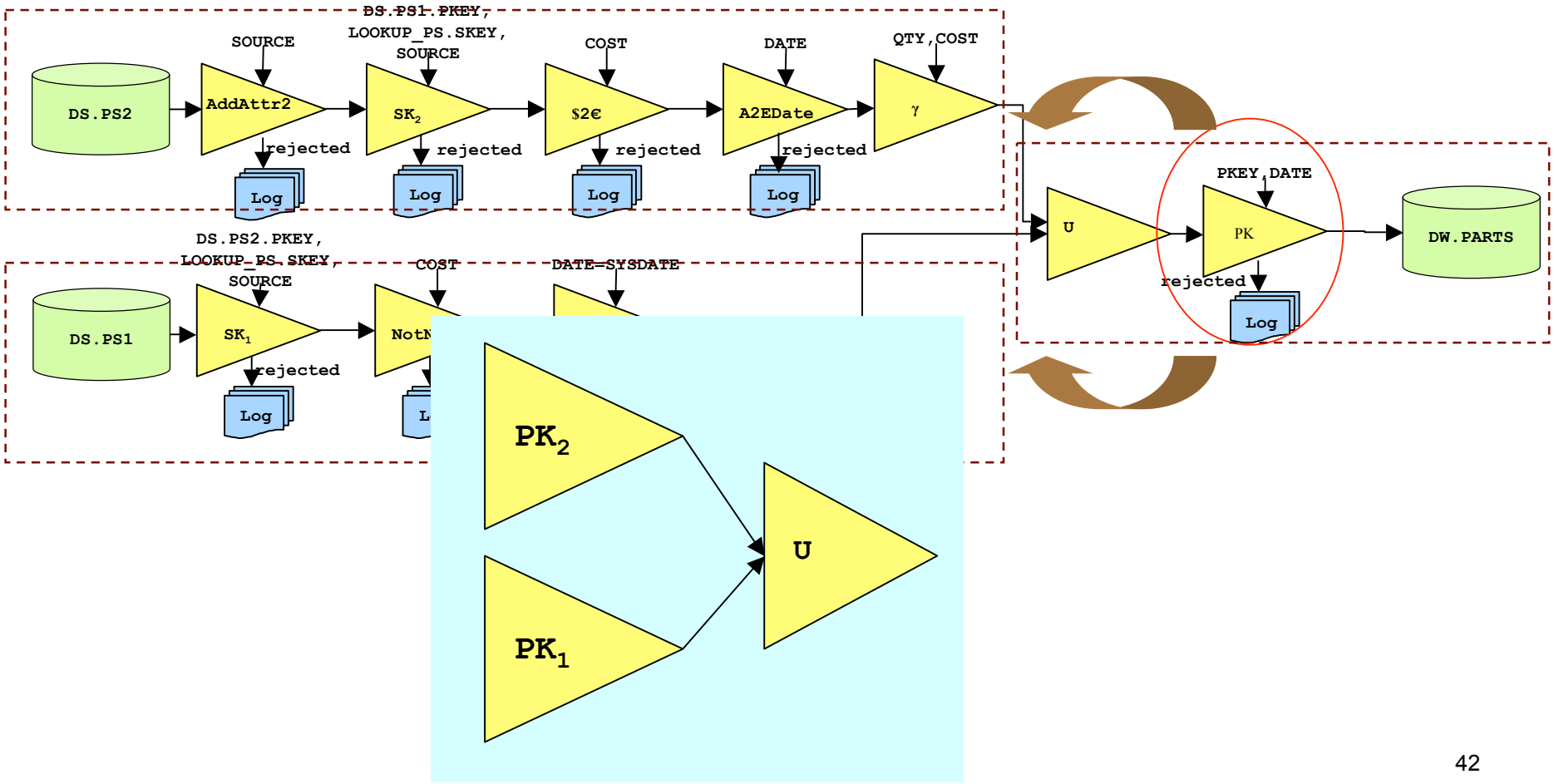
Optimization of ETL Workflows

Pre-Processing → Phase I: SWA's in local groups → Phase II: FAC's homologous → Phase III: DIS's → Phase IV: SWA's in local groups → Post-Processing

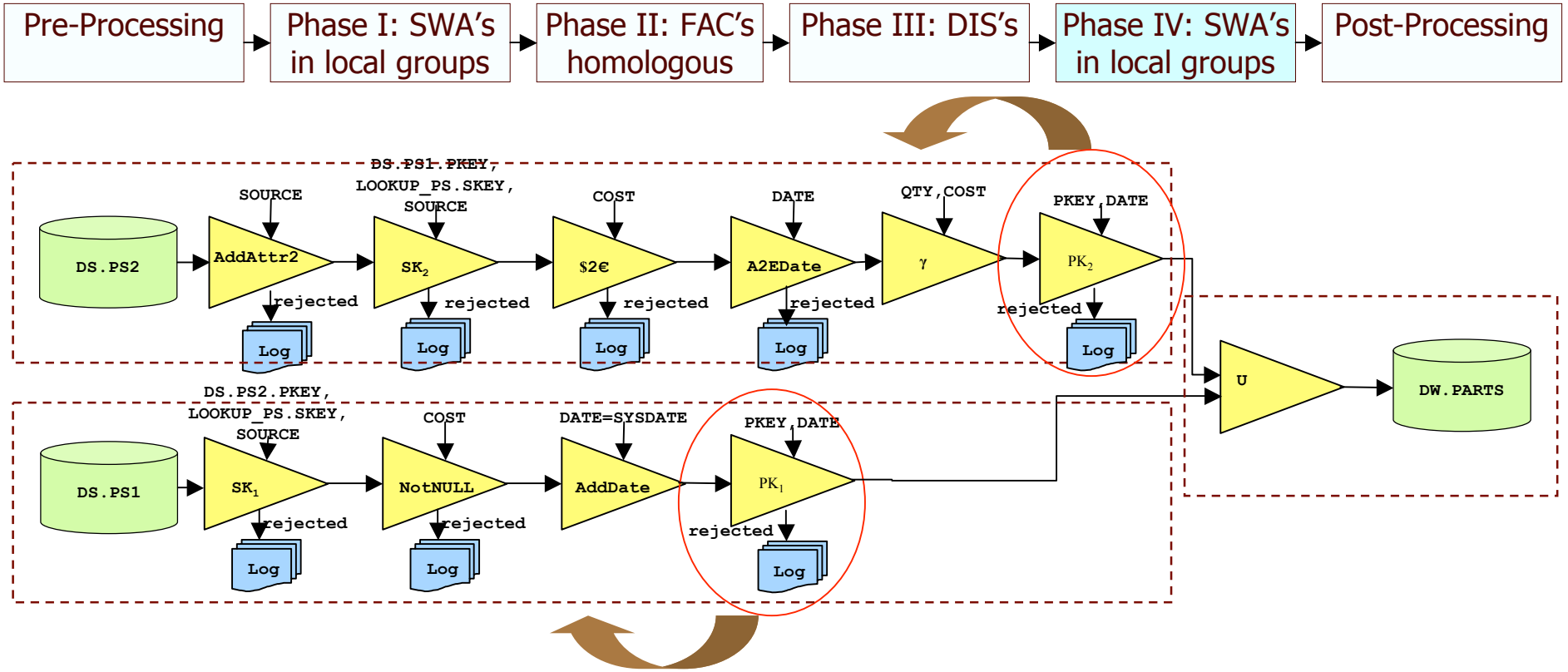


Optimization of ETL Workflows

Pre-Processing → Phase I: SWA's in local groups → Phase II: FAC's homologous → Phase III: DIS's → Phase IV: SWA's in local groups → Post-Processing

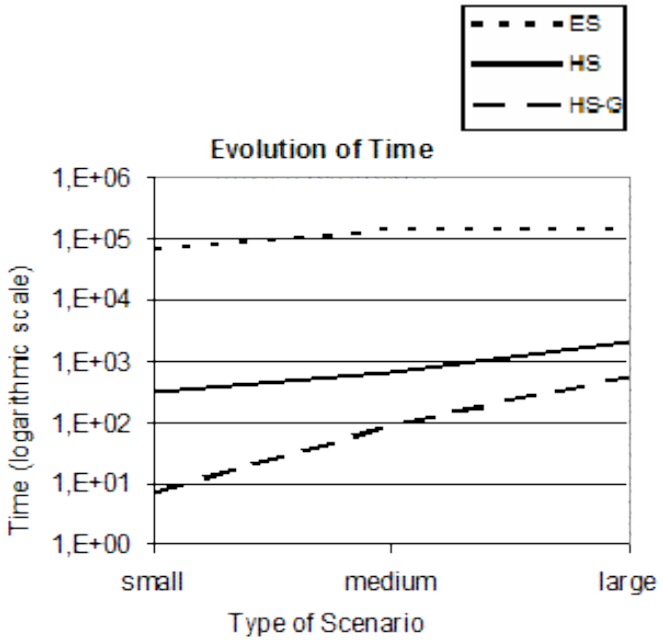


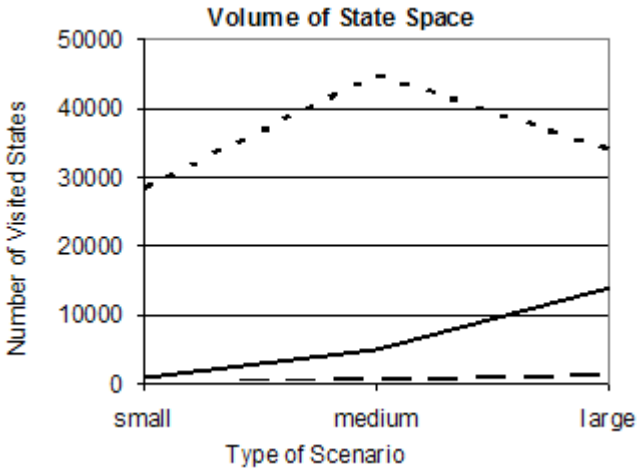
Optimization of ETL Workflows

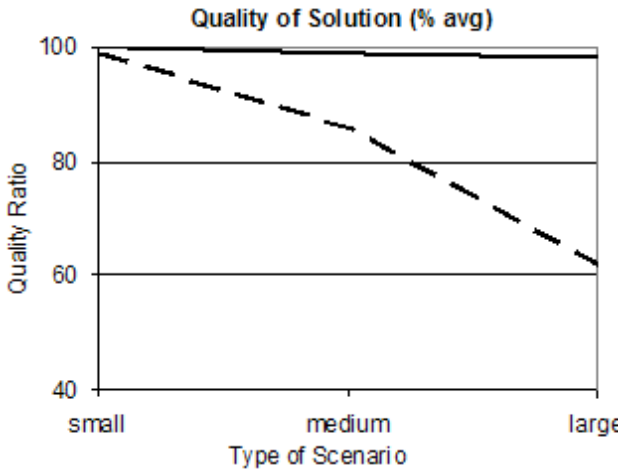


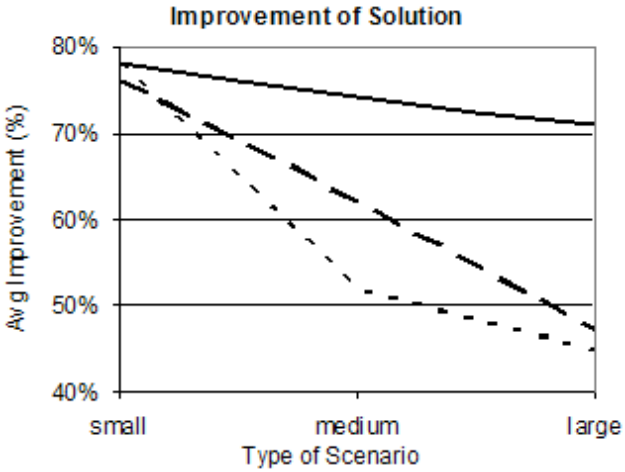
Optimization of ETL Workflows

- Algorithms
  - exhaustive, ES
  - heuristic, HS
  - greedy, HS-G
    - swap only if we gain in cost
- Results
  - algorithms HS, HS-G improve the performance of ETL workflows over 70% (avg) during a satisfactory for DW's period of time









Optimization of ETL Workflows

- Physical optimization
  - several physical operators implement the same semantic operation
    - e.g., a logical join can be performed in more than one way: nested loops, sort-merge join, hash-join
- Solution
  - employ different alternatives for the physical execution of each logical-level activity
  - take into consideration
    - the effects of possible system failures to the workflow operation
    - the introduction of sorter activities in the physical design
- Algorithms
  - exhaustive search of the search space
  - state-space pruning based on experimental observations and the benefits of sorter introduction given by the formula:

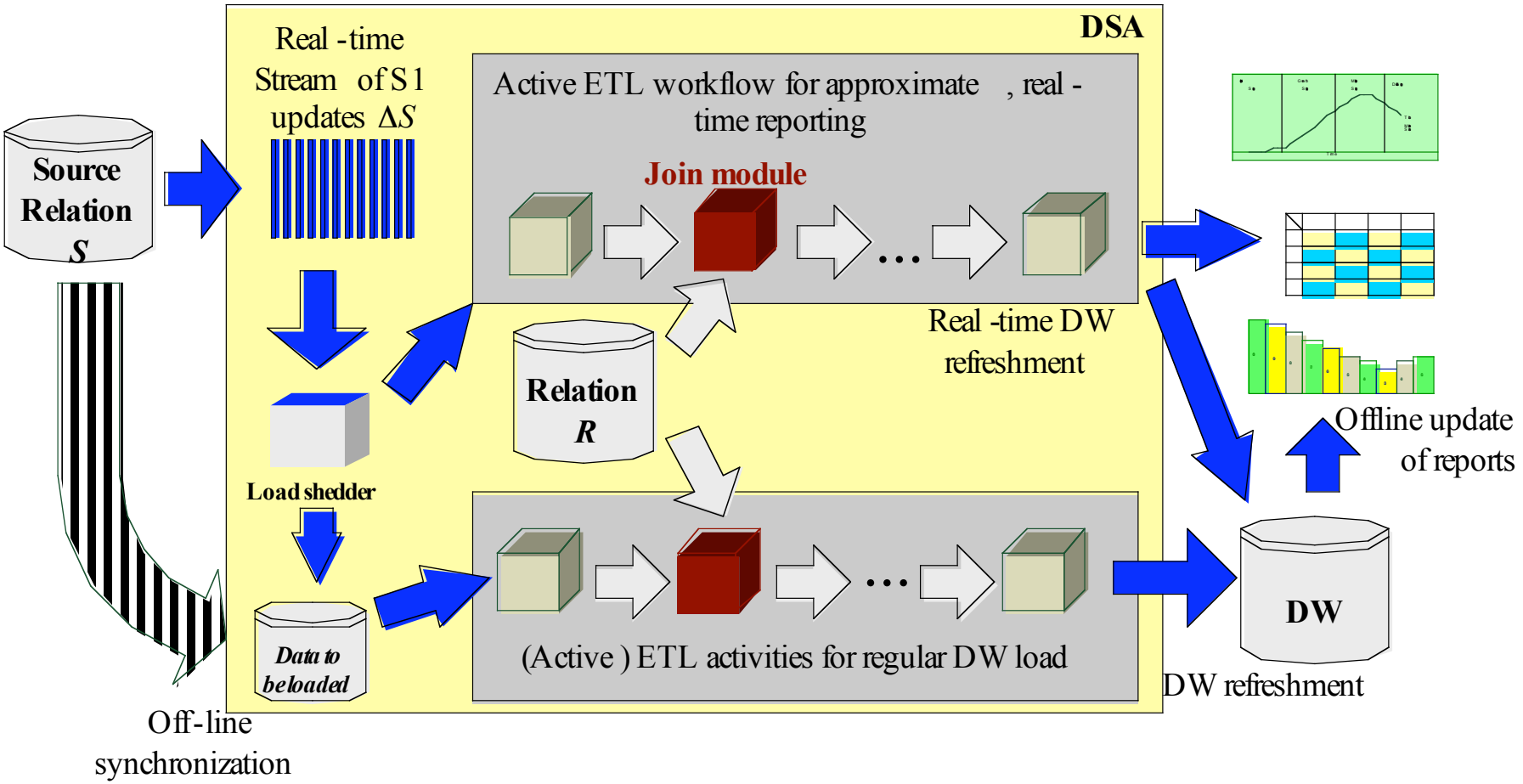
Outline

- Conceptual Model
- Logical Model
- Optimization of ETL Workflows
- **Research Challenges**

Research Challenges

- A unified way to represent ETL processes
  - an algebra or a declarative language
  - a **benchmark** for ETL processes
    - useful for evaluating optimization techniques, implementation algorithms for specific ETL activities, and so on
      (a first attempt was presented in QDB'07, in conj. w/ VLDB'07)
- Extension of the ETL mechanisms for **non-traditional data**
  - XML/HTML, spatial, biomedical data, …
- Apply the techniques described to **similar environments**
  - e.g., Active DWs
    - meet the high demand of applications for up-to-date information
    - are refreshed on-line and achieve a higher consistency between the stored information and the latest data updates

Real time ETL



Research Challenges

- Take into consideration
  - privacy issues, physical constraints
- Evolution of ETL processes
  - changes may occur in the sources, DW, business requirements, …
    (a first attempt was presented in DaWaK'07)
- Can it scale?
  - think of new models for the case of large distributed environments with many sources, e.g. P2P
    - can the techniques scale?
    - can they adapt to the different semantics, like approximate and incomplete answers?
    - can we make the techniques "goal"-driven rather than strict: e.g. I want to have 100% over this week's data, 80% over last week's, etc?
    - how to integrate static and dynamic cases (peers come and leave, others stay there for a long period)?

Conclusions

- Conceptual Model — DOLAP '02,'06 – NLDB '07 – J. IJSWIS '07
- Logical Model
  - Architecture Graph — CAiSE '02 – DMDW '02
  - Operational Semantics — CAiSE '03 - J. Inf.Sys. '05
  - Metrics — ER '05 – DaWaK '05
- Conceptual to Logical — DOLAP '05 – J. DSS '05
- Optimization of ETL Workflows — ICDE '05 – J. TKDE '05 – DOLAP '07
- Research Challenges
  - Real-time ETL — Real ETL – MeshJoin, ICDE '07, TKDE '07
  - Evolution and Data Provenance — DaWaK '07 – *ProP '07*