Logical Optimization of ETL Workflows

Outline

- Introduction
- Modeling ETL Optimization as a State-Space Problem
  - Problem formulation
  - State Generation
  - Transition Applicability
- State Space Search & Experimental Evaluation
- Conclusions & Future Work
Outline

- **Introduction**
- Modeling ETL Optimization as a State-Space Problem
  - Problem formulation
  - State Generation
  - Transition Applicability
- State Space Search & Experimental Evaluation
- Conclusions & Future Work

---

**Extraction – Transformation – Loading (ETL) Workflows**

- **Extract**
- **Transform & Clean**
- **Load**

---
ETL workflows are not “big” queries
The well known query optimization techniques are not sufficient, mostly due to the existence of:

- data manipulation functions
  - when is it valid to push an ETL activity in front of a function?
- “black-box” activities
  - unknown semantics
  - difficult/impossible/meaningless to express in relational algebra
  - we cannot interfere with their interior (e.g., their source code)
- naming conflicts
  - e.g., an attribute named COST in source “1” contains values in Dollars, while in source “2” contains values in Euros

Related Work

- ETL and optimization
  - Data Cleaning (Rahm & Do 2000)
  - Duplicate detection (Monge 2000)
  - Cleaning of data from the web (AJAX 2000)
  - CPU optimization for ETL (Potter’s Wheel 2001)
  - Extraction, Recovery and Lineage Tracing
    - Detection of differentials (Labio et al. 1997)
    - Recovery (Labio et al. 2000)
    - Lineage tracing (Cui 2001)
- Streams and optimization
  - AURORA 2003
Equivalent workflows

Can we push selection early enough?
Can we aggregate before $2\text{€}$ takes place?
What about naming conflicts?

Aims

- How can we minimize the execution time of an ETL workflow?
- How can we deal with naming problems?
- When is it valid to change the position of an activity in an ETL workflow?
- How can we produce equivalent ETL workflows?

In summary, the two problems that have risen are
- to determine which modifications of the workflow are legal
- which configuration is the best in terms of performance gains
Problem Formulation and Contribution

- We set up the theoretical framework for the problem of the optimization of an ETL workflow by modeling the problem as a state-space search problem.
  - Each state represents a particular design of the workflow as a graph.
  - We define transitions from one state to another.
  - We construct the state-space.
  - We search for a state that minimizes the execution time of the workflow.

Outline

- Introduction
- Modeling ETL Optimization as a State-Space Problem
  - Problem formulation
  - State Generation
  - Transition Applicability
- State Space Search & Experimental Evaluation
- Conclusions & Future Work
States

- Each **activity** is characterized by
  - A unique ID
  - Input schemata
  - Output schemata
  - Semantics, expressed in an extended relational algebra with black-box functions

- Each ETL workflow is a **state**, i.e., a DAG:
  - Nodes: activities and relations
  - Edges: provider relations

State Generation through Transitions

- **State Generation**: a **transition** is a modification to a state (i.e., a variant of the workflow) that generates a semantically equivalent workflow

- Three categories for five transitions
  - Swap
  - Distribute and Factorize
  - Merge and Split
Transitions – Swap two activities

- Locally swap two unary activities

\[ \text{SWA}(a_1, a_2) \]

- SWA \((a_2, a_1)\)

Transitions – Distribute and Factorize

- Factorize: replacement of two homologous activities placed in two converging data flows by a new one

\[ \text{FAC}(a_p, a_1, a_2) \]

- Distribute: distribution of an activity into two converging data flows

\[ \text{DIS}(a_p, a) \]

\( a, a_1, a_2 \): homologous activities
Transitions – Distribute and Factorize

- An example
  - A simple cost model: \#rows
  - Cost: \( c_{\text{sk}} = n \log_2 n \) and \( c_\sigma = n \)
  - Application:
    - Input: 8 records in each flow
    - Selectivity: 50% for \( \sigma \), 100% for the rest

\[
\begin{align*}
(1) & \quad \text{sk} \quad \sigma \quad \text{sk} \\
(2) & \quad \text{sk} \quad \sigma \quad \text{sk} \\
(3) & \quad \text{sk} \quad \sigma \quad \text{sk}
\end{align*}
\]

- \( C_1 = 2n \log_2 n + n = 56 \)
- \( C_2 = 2(n + (n/2) \log_2(n/2)) = 32 \)
- \( C_3 = 2n + (n/2) \log_2(n/2) = 24 \)

Transitions – Merge and Split

- Merge / Split unary activities

\[
\begin{align*}
\text{MER}(a_{1+2}, a_1, a_2) \\
\text{SPL}(a_{1+2}, a_1, a_2)
\end{align*}
\]
We resolve naming problems by reducing all attribute names (for all activity and relation schemata) to a common terminology.

We annotate each activity with three extra schemata:
- **Functionality schema**: the set of input attributes that participate in the computation
- **Generated / Projected-out schemata**

Easy to obtain these through a set of templates ([Vassiliadis et al. @ CAiSE’03, Inf. Systems](#)).

E.g., the Surrogate Key Transformation requires an input production key, a generated surrogate key and a parameter specifying the source of the data.

When is it valid to swap two activities?
- The **functionality schema** should be a subset of the input schema.

![Diagram](#)
Transition Applicability

- When is it valid to swap two activities?
  - the functionality schema should be a subset of the input schema
  - input schemata should be subset of provider schemata

Transition Correctness

- To prove correctness we employ
  - $\text{cond}$: post-condition for activities
    - true, when the activity is successfully executed
    - involves activity’s functionality schema
    - e.g., $\$2\v(\#vrb11)$
  - $\text{Cond}$: post-condition for ETL workflows
    - conjunction of the post-conditions of all the workflow activities, arranged in the order of their execution

- ETL workflow equivalence:
  - the schema of the data propagated to each target recordset, at the end of each flow, is identical
  - the flows have the same post-conditions

- Theorem: All transitions produce equivalent workflows
Outline

- Introduction
- Modeling ETL Optimization as a State-Space Problem
  - Problem formulation
  - State Generation
  - Transition Applicability
- State Space Search & Experimental Evaluation
- Conclusions & Future Work

Algorithms

- Simple to produce an Exhaustive search (ES algorithm)
  - We generate all the possible states that can be generated by applying all the applicable transitions to every state
- Heuristic search (HS algorithm)
  - Pre-Processing: Merge before any other transition
  - Phase 1: Swap only in linear paths
  - Phase 2: Factorize only homologous activities placed in two converging paths
  - Phase 3: Distribute only if transformation is applicable
  - Phase 4: Swap again only in the linear paths of the new states produced in phases ‘2’ and ‘3’
- Greedy variant of the Heuristic search (HS-G algorithm)
  - Swap only if we gain in cost
Experiments

- **Setup**
  - Three categories for 40 different ETL workflows
    - Small: 20 activities (avg)
    - Medium: 40 activities (avg)
    - Large: 70 activities (avg)
  - **Quality of solution**
    - HS: near optimal
    - HS-G: good for small/medium and over 60% for large workflows
  - **Improvement of solution**
    - HS: over 70%
    - HS-G: over 60% for small and medium, and over 47% for large workflows

- **Volume of state space**
  - ES: exponentially increases
  - HS: grows for large scenarios
  - HS-G: keeps state space small
- **Evolution of time**
  - ES: in medium/large scenarios did not terminate within 40h
  - HS: avg worst case is ~35min for large scenarios, while the gain in the execution time outreaches 70%
  - HS-G: for large scenarios is quicker than HS, but the gain is only 47% (avg)
Conclusions & Future Work

- We set up the problem of optimization of ETL workflows as a state-space search problem, by modeling an ETL workflow as a state.
- We define transitions in the search space, along with their applicability and prove the correctness of applicability rules.
- We provide three search algorithms and explore their performance.
- Future work can be pursued in different directions:
  - Optimization at the physical level
  - ETL Optimization for non-traditional data (XML, biomedical, …)
  - Parallel ETL processing
Thank you