POZNAN UNIVERSITY OF TECHNOLOGY

Still Open Issues in ETL Design and Optimization

Robert Wrembel
Poznan University of Technology
Institute of Computing Science
Poznań, Poland
Robert.Wrembel@cs.put.poznan.pl
www.cs.put.poznan.pl/rwrembel

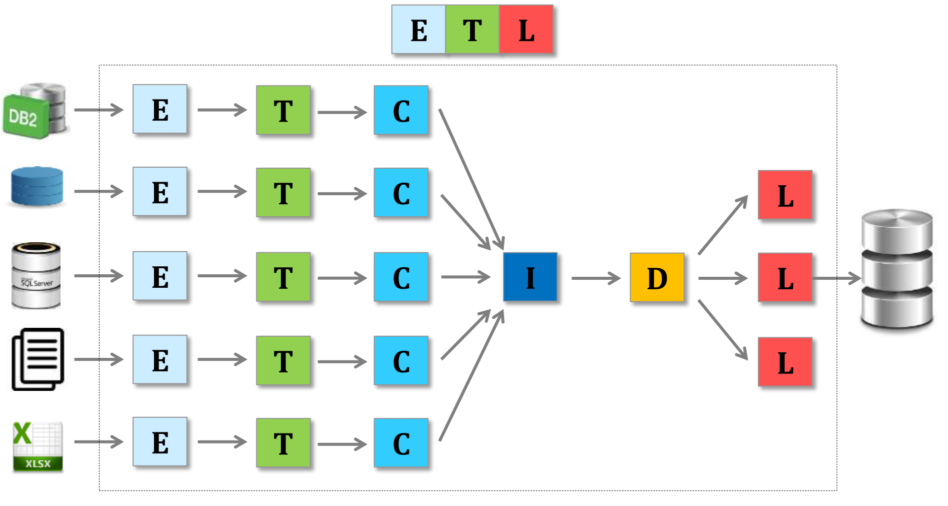
Outline

- **Evolving ETL workflows**
- **Optimizing ETL workflows**

Part 1

Evolving ETL workflows

Evolving DSs in DW architecture



DSs change in time

➲ **Structures of data sources change frequently**
- **Wikipedia: every 9-10 days during the last 4 years → 171 schema versions**
- **telecom: every 7-13 days**
- **banking: every 2-4 weeks**

- D. Sjøberg : Quantifying schema evolution. IST 35(1), 1993
- C.A. Curino, L. Tanca, H.J. Moon, C. Zaniolo: Schema evolution in wikipedia: toward a web information system benchmark. ICEIS, 2008
- H. J. Moon, C. A. Curino, A. Deutsch, C.-Y. Hou, C. Zaniolo: Managing and querying transaction-time databases under schema evolution. VLDB, 2008
- P. Vassiliadis, A. V. Zarras. 2017. Schema Evolution Survival Guide for Tables: Avoid Rigid Childhood and You're en Route to a Quiet Life. Journal on Data Semantics 6(4), 2017
- P. Vassiliadis, A. V. Zarras, I. Skoulis. 2017. Gravitating to Rigidity: Patterns of Schema Evolution - and its Absence - in the Lives of Tables. Information Systems 63, 2017

DS evolution

➲ **DS changes**
- **add column**
- **drop column**
- **change column datatype**
- **change column size**
- **create table**
- **drop table**
- **rename column**
- **rename table**
- **split table**
- **merge tables**

Impact on ETL

- ➲ **Deployed ETL process (workflow) may no longer be executed → needs to be repaired**
- ➲ **Pharma and banks**
  - **# data sources integrated → from dozens to over 200**
  - **# deployed workflows → from thousands to hundreds of thousands**

How to repair ETL?

- ➲ **Goal → (semi-)automatic**
- ➲ **ETL tools**
- ➲ **Business**
- ➲ **Research approaches**

ETL tools

- Open source
  - Talend Open Studio
  - Pentaho Data Integration
  - CloverETL
  - Apache NiFi
- Commercial
  - IBM InfoSphere DataStage
  - Informatica
  - ABinitio
  - Oracle Data Integrator
  - Microsoft Integration Services
- Do not support semi-automatic repair
  - only impact analysis is supported

Business

- Writing generic ETL
  - input to a generic ETL: tables and attributes
- Srceening DS changes
  - kind of views
- Need manual repairs

Research approaches

- Metrics
- Hecateus
- EIDE - Hecateus
- E-ETL
- E3TL
- MAIME

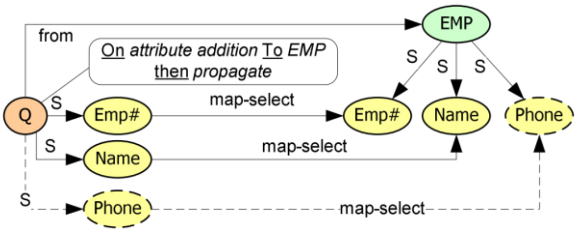
Metrics

- For assessing the design quality of an ETL process wrt its vulnerability to DS changes
- Observations
  - the more tables and attributes ETL processes the more vulnerable to changes it is
  - an ETL process with steps that reduce a number of processed attributes as early as possible is preferred

- G. Papastefanatos, P. Vassiliadis, A. Simitsis, Y. Vassiliou: Design Metrics for Data Warehouse Evolution. ER, 2008
- G. Papastefanatos, P. Vassiliadis, A. Simitsis, Y. Vassiliou: Metrics for the prediction of evolution impact in ETL ecosystems: A case study. J. Data Semantics, 1(2), 2012

Hecateus

- ETL process is represented as a graph
  - nodes: relations, attributes, queries, conditions, views, functions, DSs
  - edges: relationships between nodes



- G. Papastefanatos, P. Vassiliadis, A. Simitsis, Y. Vassiliou: Policy-Regulated Management of ETL Evolution. J. Data Semantics, 5530, 2009
- G. Papastefanatos, P. Vassiliadis, A. Simitsis, T. Sellis, Y. Vassiliou: Rule-based Management of Schema Changes at ETL sources. ADBIS, 2010

Hecateus

- The graph is annotated with policies that define the behavior of an ETL process in response to a certain DS change event
  - propagate the event, i.e. modify the graph according to a predefined policy
  - prompt an administrator
  - block the event propagation
- Can handle
  - attribute changes: rename, type change, lenght change, deletion
  - table changes: rename, deletion

Hecateus

- Drawbacks
  - policies must be explicitly defined for each graph element
  - a user must determine a policy in advance, before an evolution event occurs
  - limited to steps expressed by SQL
  - cannot handle
    - column: addition
    - table: addition, split, merge

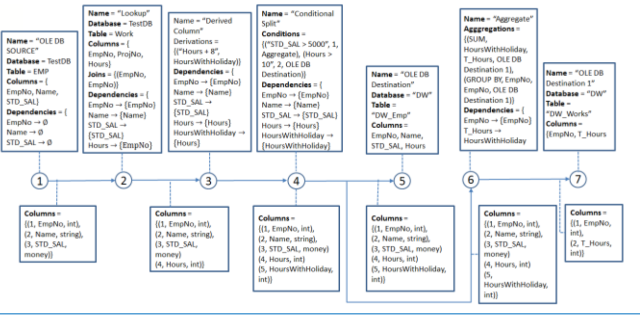
EDIE

- Built on Hecateus
- Idea: to maintain alternative variants (old versions) of data sources and ETL processes
- ETL steps annotated with policies that instruct whether they should be adapted to an evolved DS or should use an old version of the DS
- Unrealistic → DSs typically do not maintain versions

- P. Manousis, P. Vassiliadis, G. Papastefanatos: Automating the Adaptation of Evolving Data-Intensive Ecosystems. ER, 2013
- P. Manousis, P. Vassiliadis, G. Papastefanatos: Impact Analysis and Policy-Conforming Rewriting of Evolving Data-Intensive Ecosystems. Journal on Data Semantics, 4(4), 2015

MAIME

- ➲ **ETL is represented as a property graph**
- ➲ **Propagation policy for each DS change and vertex**
  - **propagate, block, prompt (like in Hecateus)**



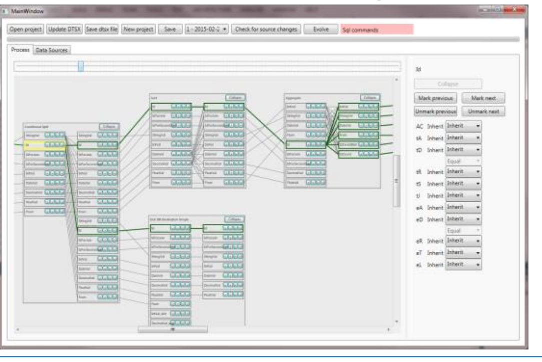
- D. Butkevicius, P.D. Freiberger, F.M. Halberg, J.B. Hansen, S. Jensen, M. Tarp: MAIME: A Maintenance Manager for ETL Processes. EDBT/ICDT Workshops, 2017

MAIME

- ➲ **Detecting DS changes: by analyzing consecutive metadata snapshots**
- ➲ **Graph altering algorithm provided**
- ➲ **Limited number of DS changes**
  - **column: add, delete, rename, change type**
- ➲ **Limited number of ETL steps supported**
  - **source, destination, aggregate, split, data conversion, derived column, lookup, sort, union all**

E-ELT

- Applies Case-Base Reasoning
- ETL process is represented as a graph



- A. Wojciechowski: ETL workflow reparation by means of case-based reasoning. Information Systems Frontiers 20(1), 2018
- A. Wojciechowski: E-ETL Framework: ETL Process Reparation Algorithms Using Case-Based Reasoning. ADBIS Workshops, 2015

E-ETL

- API to MSIS
  - download an MSIS design
  - repair it in E-ETL
  - upload the repaired design into MSIS
- ETL process reparing uses cases
  - a library of cases
  - an algorithm for searching the best case for a given repair problem
  - case similarity measure

E-ETL

- **Drawbacks**
  - **a library of cases is needed**
  - **do all ETL tools support the same ETL steps?**
    - **can a case in the library coming from one ETL tool can be uploaded into another ETL tool?**
  - **the correctness of a proposed repair cannot be formally checked**

E3TL

- **Predefined rules for evolving ETL workflows**
- **Case-based repair**
- **Rule learning from cases**

- J. Awiti, A. A. Vaisman, E. Zimányi: From Conceptual to Logical ETL Design Using BPMN and Relational Algebra. DaWaK , 2019
- J. Aviti: Algorithms and Architecture for Managing Evolving ETL Workflows. Proc. of ADBIS Workshops (CCIS), Springer, 2019
- J. Aviti, E. Zimányi: An XML Interchange Format for ETL Models. In Proc. of ADBIS Workshops (CCIS), Springer, 2019

Summary

- ➲ **Problem known for dozens of years → only partially solved**
- ➲ **None of the commercial tools supports ETL process repair**
- ➲ **UDFs make the problem more difficult**
- ➲ **Big data make the problem more difficult**

Part 2

Optimizing ETL workflows

Motivation

- # data sources integrated → from dozens to over 200
- # deployed workflows → from thousands to hundreds of thousands
- Magnetic disk
  - throughput: 200MB/s
  - read 1TB from DS → write 1TB to DW: 160 minutes
  - add ETL processing → n * ~160 minutes
  - if n=10 → processing time ~ 28 hours
- Conclusion: ETL performance optimization is vital

ETL performance optimization

- Goal: SQL-like optimization → cost based
  - ETL workflow execution plan and its optimization heuristics
- Problem 1: no algebraic optimization of separate ETL steps → no albegraic optimization of the whole ETL workflow
- Problem 2: computing statistics on a data set that is processed may be time consuming
  - processed data sets are not available in advance → no statistics
- Problem 3: ETL steps are frequently implemented as UDFs
  - a cost model of an UDF is unknown → a cost model of the whole ETL workflow is unknown

Commercial approaches

- Increasing resources (#CPU, memory, #nodes)
- Parallelizing ETL tasks → running ETL in a cluster
  - IBM InfoSphere DataStage
  - Informatica PowerCenter
  - AbInitio
  - Microsoft SQL
  - Server Integration Services
  - Oracle Data Integrator
- Moving tasks to decrease data volume asap
  - constrained to tasks expressed by SQL
  - towards DS or towards DW
    - Balanced optimization: IBM InfoSphere DataStage
    - Push-down optimization: Informatica PowerCenter

Commercial approaches

- R. Lella: Optimizing BDFS jobs using InfoSphere DataStage balanced optimization. IBM Developer Works, 2014
- IBM InfoSphere DataStage Balanced Optimization. 2008
- Introduction to InfoSphere DataStage Balanced Optimization. IBM Knowledge Center

- How to Achieve Flexible, Cost-effective Scalability and Performance through Pushdown Processing. Informatica, whitepaper, 2007

Parallel ETL processing

- How to partition a data flow?
- What to parallelize?
  - particular tasks
  - the whole workflow
  - determining where to split and merge parallel flows
- Determining an optimal number of parallel flows?
- What is an optimal amount of resources (CPU, memory, threads) for a given parallelized task?
- Which parallelization skeleton to apply?

ETL optimization: research approaches

1. **Quality metrics in an ETL design**
   - A. Simitsis, K. Wilkinson, M. Castellanos, U. Dayal: QoX-Driven ETL Design: Reducing the Cost of ETL Consulting Engagements. SIGMOD, 2009
2. **Partitioning and parallelization**
   - X. Liu, N. Iftikhar: An ETL Optimization Framework Using Partitioning and Parallelization. SAC, 2015
3. **Cost-based optimization based on statistics of ETL sub-flows (sub-expressions)**
   - R. Halasipuram, P.M.Deshpande, S. Padmanabhan: Determining Essential Statistics for Cost Based Optimization of an ETL Workflow. EDBT, 2014
4. **State-space optimization**
   - A. Simitsis, P. Vasiliadis, T. Sellis: State-Space Optimization of ETL Workflows. IEEE TKDE 17(10), 2005
5. **Logical optimization**
   - N. Kumar, P.S. Kumar: An Efficient Heuristic for Logical Optimization of ETL Workflows. BIRTE, 2010

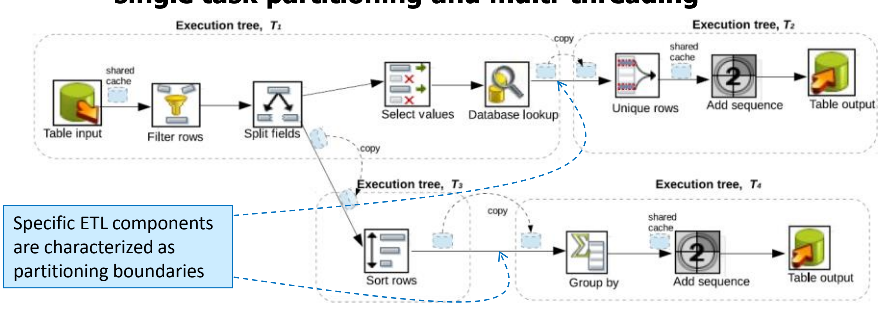
1. Quality metrics

- Proposed a layered approach for ETL design
  - layers represent: logical design, implementation, optimization, maintenance
  - at each layer some metrics are introduced
- Metrics
  - to guide each step in ETL development
  - transformations between design levels to provide certain types of optimization
    - high performance
    - recoverability
    - freshness
    - maintainability

- A. Simitsis, K. Wilkinson, M. Castellanos, U. Dayal: QoX-Driven ETL Design: Reducing the Cost of ETL Consulting Engagements. SIGMOD, 2009

2. Partitioning and parallelization

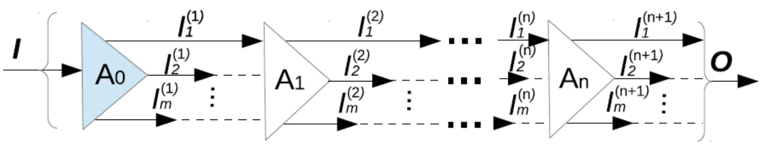
- Workload partitioning methods into so-called execution trees
  - vertical
  - horizontal
  - single task partitioning and multi-threading



- X. Liu, N. Iftikhar: An ETL Optimization Framework Using Partitioning and Parallelization. SAC, 2015

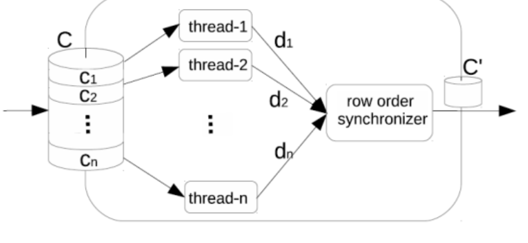
2. Partitioning and parallelization

- **Inside execution tree parallelization**
  - **input data are partitioned horizontally into n disjoint partitions (n is parameterized)**
  - **each partition is processed by a separate thread**
  - **a shared cache is used for moving data from task A_i to A_{i+1}**



2. Partitioning and parallelization

- **Internal task parallelization**
  - **for tasks with a heavy computational load**
  - **performance is increased by multi-threading parallelization**
    - **an input is divided into n splits**



3. Statistics for ETL cost based optimization

➲ **Assumption**

- **workflow optimizaiton by task reordering**
- **statistics needed for estimating workflow execution cost →cost based optimization**

- R. Halasipuram, P.M.Deshpande, S. Padmanabhan: Determining Essential Statistics for Cost Based Optimization of an ETL Workflow. EDBT, 2014

3. Statistics for ETL cost based optimization

➲ **Goal: for a given an ETL workflow, identify a set of statistics to collect**

- **the set must be used to estimate costs of all possible task reorderings**
- **the set must be minimal**
- **the cost of collecting statistics must be minimal**

➲ **Data statistics:**

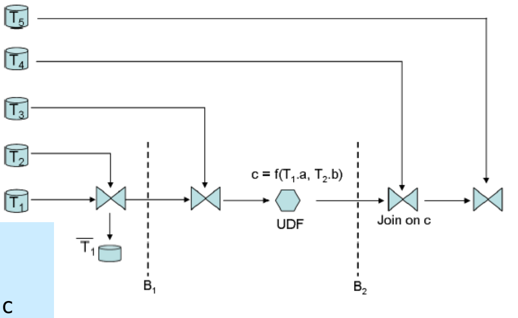
- **cardinality of table T_i**
- **attribute histograms of T_i**
- **# of distinct values of attributes**

3. Statistics for ETL cost based optimization

- **Operators**
  - **select**
  - **project**
  - **join**
  - **group-by**
  - **transform**
- **Each operator has a cost function associated, the function is based on:**
  - **data statistics**
  - **CPU and disk-access speeds**
  - **memory usage**

3. Statistics for ETL cost based optimization

- **Step 1: partition a workload**
  - **use pre-defined partitioning boundaries**
  - **optimize each partition (sub-expression - SE) independently**



Example block boundaries

- **B1: T1 is materialized**
- **B2: UDF** creates a derived attribute c that is a join attribute of the subsequent join with T_4

3. Statistics for ETL cost based optimization

- Step 2: generate sub-expressions
  - by means of operators reordering
- Step 3: generate candidate statistics set (CSS)
  - for each sub-expression determined from Step 2
- Step 4: select an optimal CSS
  - min. collecting cost + min. CSS + covers all reorderings → NP-hard problem
    - linear programming applied to solve it
- Step 5: inject into a workload a component for collecting statistics
- Step 6: run a workload and gather statistics
- Drawback
  - it does not address workload execution optimization

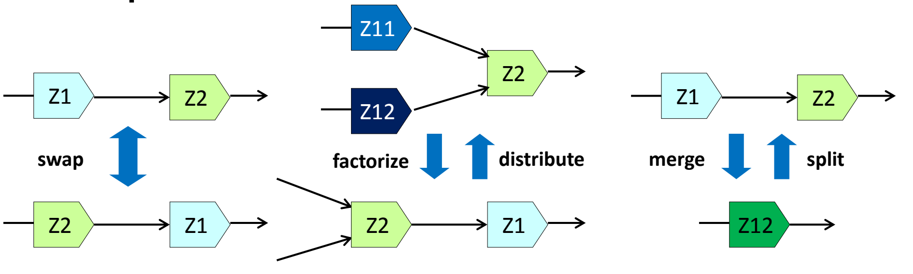
4. State-space optimization

- Performance optimization by task reordering
- Each workload gets assigned an execution cost (time, data volume)
- Searching the whole space of possible workloads is impossible (time)
- Heuristics (e.g. filter data asap) to prune the search space

- A. Simitsis, P. Vasiliadis, T. Sellis: State-Space Optimization of ETL Workflows. IEEE TKDE 17(10), 2005

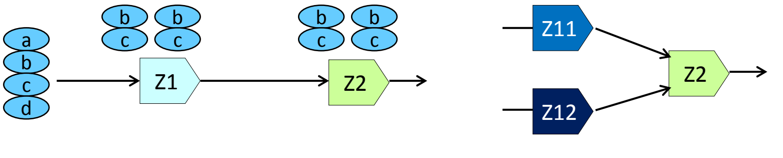
4. State-space optimization

- Workflow transformations
  - swap → change order to filter data asap
  - factorize → if Z11 i Z12 execute the same operations on different flows
  - distribute → opposite to factorize
  - merge → tasks that must be executed subsequently
  - split



4. State-space optimization

- Correctness of workflow transformation
- Swap
  - Z1 has one source, Z2 has one destination
  - compatibility of input and output schema
    - in.Z1={b,c} & out.Z1={b,c}
    - in.Z2={b,c} & out.Z2={b,c}
- Factorize/Distribute
  - Z11 i Z12 have 1 destination Z2 (set operators)
  - Z11 i Z12 do the same task on different workflows

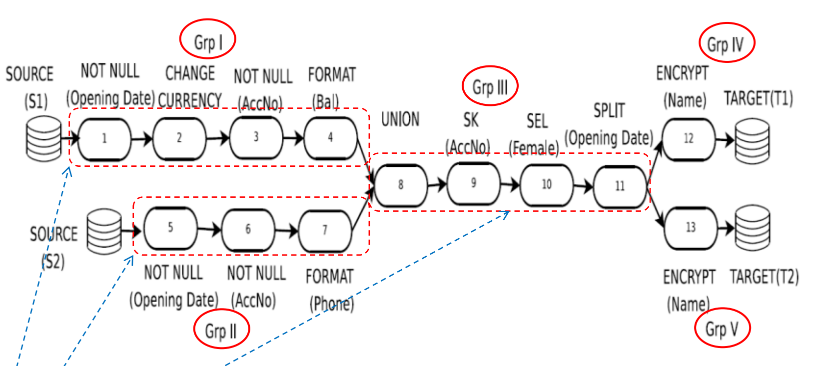


5. Logical ETL optimization

- **Based on concepts presented in:**
  - A. Simitsis, P. Vasiliadis, T. Sellis: State-Space Optimization of ETL Workflows. TKDE 17(10), 2005
- **Flow transformation techniques**
  - **swap, factorize/distribute, merge/un-merge**
  - **cost functions and selectivities of activities are used in a greedy heuristic that reorders a linear flow**
- **Optimization heuristic**
  - **task reordering**
- **Differences wrt Simitsis et.al.**
  - **focuses on optimizing linear flows only**
  - **new structure: dependency graph**

- **N. Kumar, P.S. Kumar: An Efficient Heuristic for Logical Optimization of ETL Workflows. BIRTE, 2010**

5. Logical ETL Optimization



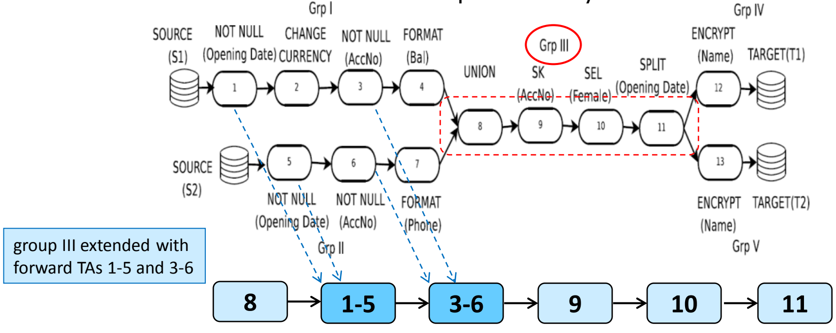
divide a complex flow into linear flows

optimize each linear flow independently

5. Logical ETL Optimization

➲ **Forward pass**

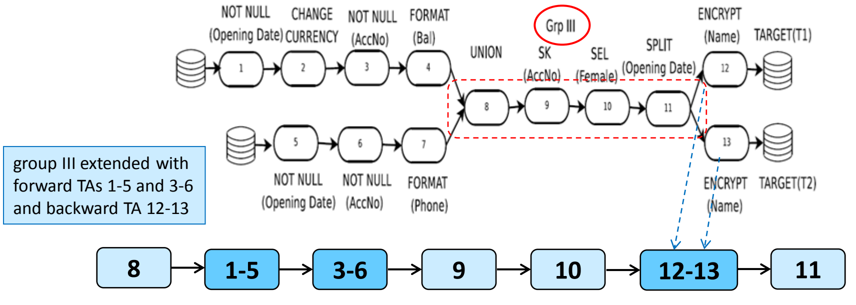
- **transferrable activities (TAs) are added to the beginning of the next group**
  - activity 1 (Grp I) and 5 (Grp II) have the same semantics on the same attribute → are represented by activity 1-5 in extended Grp III
  - activities 3 and 6 are represented by 3-6



5. Logical ETL Optimization

➲ **Backward pass**

- **execute it as the forward pass but in the direction from the end to the beginning of a linear flow**
  - **activities 12 and 13 are represented by 12-13 in Grp III**



➲ **Drawback: the forward pass may move selective activities towards the end of a flow**

5. Logical ETL Optimization

- Algorithm
  - all possible allowed combinations of TAs in linear groups are analyzed
  - extended linear groups are optimized using the same heuristic as for a normal linear group → swaping the activities and computing a cost of the linear flow

1-5 → 3-6 → 9 → 10 → 12-13

3-6 → 1-5 → 9 → 10 → 12-13

9 → 10 → 1-5 → 3-6 → 12-13

••

Summary

- Problem known for dozens of years → only partially solved
- Optimization techniques
  - parallelization (commercial, research)
  - balanced / push-down (commercial, research)
  - task reordering (research)
- Task reordering → computationally complex
- Parallel processing → feasible (promissing)
- No support for UDFs

Our focus

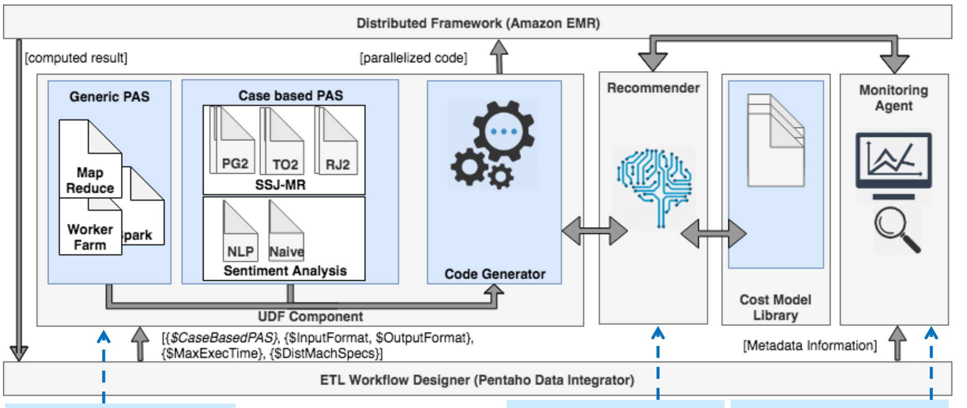
- ➲ **ETL optimization with UDFs**
  - **optimization by means of parallel processing**
  - **a cost model to determine the degree of parallelism**

- S. M. F. Ali, R. Wrembel: Towards a Cost Model to Optimize User-Defined Functions in an ETL Workflow Based on User-Defined Performance Metrics. Proc. of ADBIS, LNCS 11695, 2019
- S.M.F. Ali, J. Mey, M. Thiele: Parallelizing user-defined functions in the ETL workflow using orchestration style sheets. International Journal of Applied Mathematics and Computer Science (AMCS), 29(1), 2019
- S.M.F. Ali: Next-generation ETL Framework to Address the Challenges Posed by Big Data. DOLAP, 2018
- S.M.F. Ali, R. Wrembel: From conceptual design to performance optimization of ETL workflows: current state of research and open problems. VLDB Journal 26(6), 2017

Main problems

- ➲ **To figure out if an UDF can be parallelizable**
- ➲ **To figure out which parallel skeleton is the most appropriate for a given UDF**
- ➲ **How to apply a skeleton to a black box**

ETL optimization framework



- pre-defined code templates for execution in distributed environment

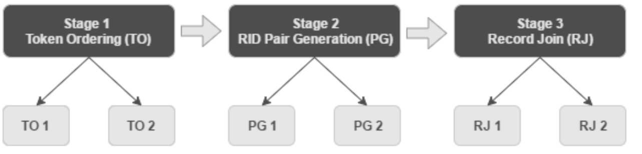
- ML algorithms to optimize a given ETL workflow, based on performance data collected during ETL executions

- monitoring ETL workflow executions
- identify performance bottlenecks
- gather performance statistics

Cost model

- **To determine an efficient configuration for distributed machines**
- **Mutliple Choice Knapsack Problem**

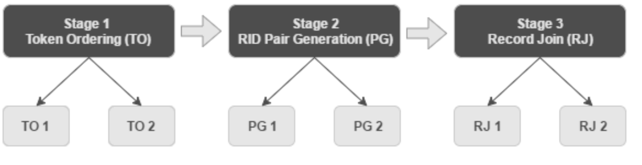
PoC



R. Vernica, M. Carey, C. Li: Efficient parallel set-similarity joins using MapReduce. SIGMOD, 2010

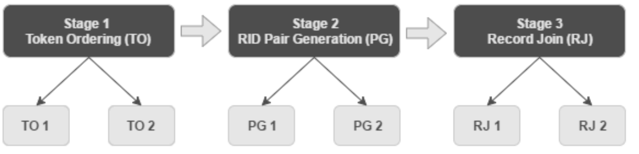
- **Set-similarity joins using MapReduce (SSJ-MR)**
- **Stage 1: token ordering**
  - **computes data statistics to generate partitioning keys, by tokenizing incoming records into a wordset**
- **Stage 2: RID pair generation**
  - **extracts a record ID (RID) and join-attribute value for each record**
  - **computes the similarity of the join-attribute values**
- **Stage 3: record join**
  - **generates pairs of joined records using RIDs of similar records**

PoC



- **Each stage can be implemented by two variants of an algorithm**
  - **TO1 or TO2 → for Token Ordering**
  - **PG1 or PG2 → for Pair Generation**
  - **RJ1 or RJ2 → for Record Join**
- **Execution environment: Amazon EMR Cluster**
- **Each stage of SSJ-MR may be executed in a differently configured Amazon EMR cluster**

PoC



		#Nodes [exec cost/h]			
Stage	Algorithm	2 [0.4$/h]	4 [0.8$/h]	8 [1.6$/h]	10 [2.0$/h]
1	BTO	191.98	125.51	91.85	84.02
	OPTO	175.39	115.36	94.82	92.80
2	BK	753.39	371.08	198.70	164.57
	PK	682.51	330.47	178.88	145.01
3	BRJ	255.35	162.53	107.28	101.54
	OPRJ	97.11	74.32	58.35	58.11

- **Execution times on Amazon EMR Cluster**

  - R. Vernica, M. Carey, C. Li: Efficient parallel set-similarity joins using MapReduce. SIGMOD, 2010

PoC

- **Particular problem**
  - **to find the best configurations of an EMR cluster for each stage separately**
- **General problem**
  - **to find the best configurations of hardware for the whole ETL workflow w.r.t. execution time and monetary cost → Multiple Choice Knapsack Problem**
- **Mutliple Choice Knapsack Problem**
  - **m classes in KP → m stages of ETL execution**
  - **W weight constraint → B monetary budget constraint**
  - **w_{ij} cost of variant j of item of class i → c_{ij} cost of variant j of a program at stage i**
  - **p_{ij} profit of variant j of item of class i → t_{ij} execution time of variant j at stage i**

Implementation

- The problem was solved by Mixed Integer Linear Programming solver → the lp_solve library (Java)
  - the implementation of the cost model https://github.com/fawadali/MCKPCostModel

Open issues

➲ ML for ETL optimization

➲ Dynamic ETL optimization