

The History, Present, and Future of ETL Technology

[Test-of-Time Award - Invited Talk]

Alkis Simitsis
Athena Research Center
Athens, Greece
alkis@athenarc.gr

Spiros Skiadopoulos
University of the Peloponnese
Tripolis, Greece
spiros@uop.gr

Panos Vassiliadis
University of Ioannina
Ioannina, Greece
pvassil@cs.uoi.gr

ABSTRACT

There is an abundance of data, but a large volume of it is unusable. Data may be noisy, unstructured, stored in incompatible for direct analysis medium or format, and often expensive to access. In most practical cases, the data needs to be processed before it can be used to extract valuable business insights. We refer to the non-trivial, end-to-end operation of extracting intelligence from raw data as an ETL process. In this paper, we review how the ETL technology has been evolved in the last 25 years, from a rather neglected engineering challenge to a first-class citizen in analytics and data processing. We present a brief historical overview of ETL, discuss its various applications and incarnations in modern data processing environments, and argue about exciting, feasible or wishful, potential future directions.

KEYWORDS

ETL, Analytics, Business Intelligence, Data Science, Data Engineering, Data Warehouses, Data lakes, User-Defined Functions

1 THE HISTORY

With the advent of database technology in the '80s and '90s, enterprises employed online transactional processing systems (OLTP) that offered efficient ways to storing, querying, and updating transactional and operational data. In most practical cases, a relational database management system (RDBMS) was used to manage OLTP. OLTP systems have been designed for application-oriented collection and recording of data, and for keeping the most current state of the enterprise. Therefore, they were optimized for multiple, concurrent, and fast writes and reads, ensuring the four primary and essential properties of a transaction: atomicity, consistency, isolation, and durability (a.k.a. ACID properties).

Having solved the data housekeeping and data logistics problem, the inevitable following necessity for enterprises was to put their data into a profitable use by getting insights into their data and operations. To this end, a new family of data systems designed to handle analytical processes started becoming popular around the mid '90s. These systems were called online analytical processing systems (OLAP) and have been the cornerstone of business intelligence and decision support making. Under the hood, OLAP employs a data warehouse (DW) or enterprise data warehouse (EDW) specifically designed to store and manipulate the data for analytical processes. The data warehouse serves as an information system that maintains historical and commutative data, essentially comprising the full analytical history of an enterprise. It is generally optimized for read-only, complex queries

typically performing heavy operations such as user-defined functions, aggregates, and multiple complicated joins, aiming at what-if scenarios and business analysis.

Table 1 lists core characteristics of OLTP and OLAP systems. Clearly, the two classes of systems share little commonalities. Still, the data needs to flow from OLTP (the recent state of the enterprise) to OLAP (the history of the enterprise); a task that is neither trivial nor obvious. This design gap is filled by dedicated processes that govern the data movement from one system to the other. These processes are collectively known as ETL processes.

OLTP	OLAP
Recent operational data	Historical data
Size in GBs, TBs	Size in TBs, PBs
Simple queries	Complex queries
Low latency	High latency
Read/write operations	Read operations
Row-store	Column-store
Day-to-day operations	Analytics, decision-making

Table 1: OLTP vs. OLAP

1.1 What is ETL?

In their original form, which is still inherent until today, Extraction-Transformation-Loading (ETL) processes describe (a) the identification and extraction of (relevant) data from various operational sources, (b) the transformations needed to cleanse and customize this data, and finally, (c) the loading of the data into a data warehouse. Figure 1 illustrates an abstract ETL process. The process starts with data collection from data providers, such as relational databases and files, and extracts either complete snapshots or differentials of the data sources. This data is propagated next to a Data Staging Area (DSA), where the data transformation (e.g., customization, integration, computation of new values and/or records, isolation of problematic records) and cleansing takes place, before it is loaded to a target data warehouse that comprises fact and dimension tables.

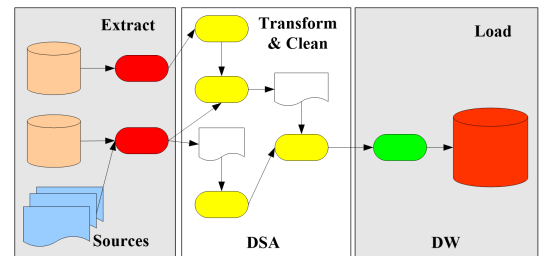


Figure 1: An abstract ETL process (source: [124])

1.2 Challenges

In the early days, this conceptually straightforward process came with several research and engineering challenges.

Schema mapping. At the schema level, we need to map one or more source schemas to one or more target schemas. This problem has been studied extensively in various flavors. From a formal point of view, schema mappings are specifications, typically expressed as a logical formalism, describing at a high level the relationships among schemas, without dealing with implementation details at the physical level. Schema mappings are building blocks for data integration and data exchange. Data integration focuses on the problem of querying across autonomous and heterogeneous data sources and aims at synthesizing data from different sources into a unified view under a global schema (mediated schema). The initial works are classified as (a) Local-as-View, where a source is described as a view expression over a mediated schema, (b) Global-as-View, where a mediated schema is described as a view over the data sources, and (c) as various combinations of these (see also [56, 92]). Data exchange aims at transforming a source instance to a target instance (materialized instance) in a way that satisfies the specifications of the schema mapping and accurately represents the source data [41, 42, 81].

Data cleansing and quality. Data quality is a major concern for ETL. Dirty or erroneous data could lead to suboptimal decisions and unreliable analysis. There has been a significant amount of work on this topic, which still remains an open problem. Example approaches proposed for data cleaning include rule-based techniques where integrity constraints are used to express data quality rules, qualitative techniques that employ machine learning to improve accuracy and efficiency of cleaning tasks, duplicate detection, automated or user-based error detection, and so on. See also [27, 40, 66, 85, 86] for more details, and [48, 104, 114] for a few early data cleaning tools.

Complex transformations. Such schema changing and data cleansing transformations typically cannot be expressed with traditional relational operators and require extra care. Similarly, other frequently used ETL transformations include data pivoting [29, 167], one-to-many mappings [26], data lineage [28], generating surrogate keys and slowly changing dimensions [80], schema changes such as removing fields, splitting a record across multiple tables, deriving entirely new data from existing values (e.g., ranking items based on the frequency of their values, computing the interval between a given moment and a timestamp withing the data), etc. Although researchers have investigated such transformations in isolation focusing mainly on efficiency and semantics enrichment, dealing with pipelines of such transformations increases significantly the complexity of the problem.

Quality of ETL processes. ETL design lacks a rigorous methodology and standardization. Typically, ETL processes are designed based on experience, good practices, and common sense. Several techniques have been proposed to measure the quality of the design by looking into metrics such as data freshness and consistency, resilience to failures, maintainability (e.g., complexity, modularity, coupling), and so on [102, 131, 149, 153, 171].

Engineering aspects. There are also several research challenges at the physical level concerning practical issues such as orchestrating, scheduling, optimizing, and tuning ETL processes. A non-exhaustive list includes: (a) Deciding cadence: the ETL process runs periodically as it may affect the efficiency of the operational sources and also is generally computationally expensive. (b) Data types: there may be a multiplicity of types at the source side ranging from structured or semi-structured to completely unstructured, e.g., relational, flexible schema (xml, avro, json), graph, text, spatial, images, etc. (c) Variety of sources: they can be heterogeneous, federated, distributed, etc. (d) Configuration: an

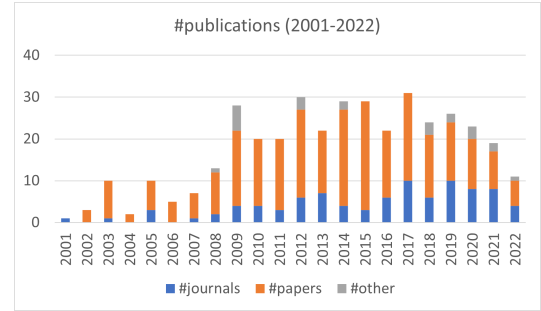


Figure 2: Papers containing ETL in their titles [source: DBLP]

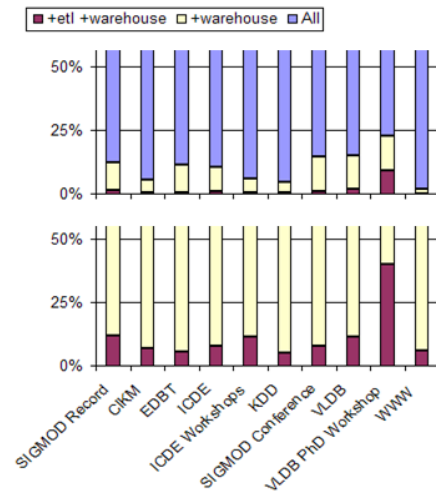


Figure 3: ETL and DW papers vs. all (top) and ETL vs. DW papers (bottom) - reference period 1999-2007 [source: DBLP]

ETL process may involve multiple systems and thus, its tuning may have many knobs. (e) Programming heterogeneity: an ETL process may be implemented in parts or as a whole either as custom-made scripts in a scripting language, or may employ SQL constructs such as COPY statements or dedicated connectors, user-defined functions, lambda functionality, or may be designed using one of the many workflow/ETL tools. (f) Determining batch size: an ETL process may work with either batches of data or data in a finer granularity (e.g., micro-batches, streaming, real-time).

1.3 Early solutions

Typically, production follows research –but this did not happen in the case of ETL. ETL tools in one form (e.g., custom scripts) or in another (e.g., specialized tools) exist in production for over 40 years. For example, as early as in 2003 there were 200+ ETL tools in the market [123]. Interestingly, there was a much slower adoption of the problem by the research community, where the first research approaches appeared in the early '00s. Figure 2 illustrates the distribution of peer-reviewed research papers related to ETL over the course of the last 20 years. Our research shows that in the period 1999-2007, on average, ETL papers comprised approximately the 10% of the total papers related to the data warehouse technology and less than 1% of the total papers in the same reference period (see also Figure 3 for publications in popular venues of the data management research community).

ETL design. A significant number of the initial works focused on ETL design, mostly at the conceptual and logical (or physiological) levels. This is a direct consequence of a very practical problem that enterprises and practitioners had to face in a data warehouse project, especially at its early stages. In most cases, the schemas of the OLTP and the OLAP systems were already in place and the task at hand was to develop an efficient and scalable design to propagate data from the former to the latter. Beside the inherent complexity of the task, an extra challenge was imposed by the lack of any kind of methodology, formalism, standard, or even recorded collective experience in designing and developing ETL processes. The majority of the solutions were ad hoc, in-house built solutions that naturally were hard to maintain and more importantly, difficult to reuse.

The first research attempts to methodological ETL design appeared in the context of conceptual ETL design. The first such work we know of, introduced a graph-theoretical approach to ETL conceptual modeling, representing transformations as ETL activities and formally defining the inter-attribute relationships among them [170]. It also introduced a palette of frequently used ETL activities –the first ever attempt to classify ETL transformations– argued (correctly, as the history has shown) that defining a closed set of transformations is both infeasible and impractical, and instead proposed a customizable, extensible, and principled method to allow designers enrich it further with custom-made, re-occurring ETL activities. Later attempts to conceptual modeling covered a large variety of aspects, including stricter formalism through: (a) UML modeling enabling nested processes, swimlanes, and multiple levels of zooming [97, 161]; (b) BPMN and BPEL modeling [2, 3, 14]; (c) business process models [33, 175]; (d) hypercubes [174]; (e) semantic web technologies employing ontologies to automate the construction of ETL design [144, 146, 147, 160]; and (e) web services technology to describe the ETL design [120]. Other approaches studied methods to ETL design driven by functional/non-functional requirements [68–70, 118, 145, 150]. Graph-based modeling has also been proposed to capture the data flow at the logical level from source schemas to a target data warehouse, including the metadata involved, direct and derived relationships among the various data flow constructs, and template mechanisms to enable reusable ETL macros [134, 168–170, 172]. There is also work describing the automated mapping from conceptual to logical models [13, 125, 130].

Optimization. Another line of research work has dealt with the optimization of ETL processes, focusing either on optimizing the end-to-end ETL process [121], or on optimizing challenging ETL transformations and functionalities.

A first approach to ETL optimization modeled the problem as a state-space problem, where given an ETL process the optimizer returned a semantically equivalent process having the best execution time possible [132, 133]. Each state is an ETL process represented as an acyclic directed graph with relations and operations as nodes and data flow relationships as edges. A transition applied to a state produces a new state. Transitions considered include: swap, factorize, distribute, merge and split. Later approaches studied the data flow optimization problem from several angles, including: (a) optimizing data flows for fault-tolerance [138, 176]; (b) identifying strategies for intermediate results materialization [101]; (c) parallelization and partition-based workload scheduling [6, 126, 154, 158]; (d) physical design and scheduling [74, 163]; (e) cost-based optimization with missing or incomplete statistics [55]; (f) optimization of data flow programs

with MapReduce-style UDFs [62, 63, 117]; (g) multi-objective data flow optimization, considering for example in tandem objectives such as performance, maintainability, fault-tolerance, and so on [32, 33, 135]; and (h) multi-engine data flow optimization, following an engine-agnostic approach [36, 71, 72, 136, 137, 139, 148]. More details on data flow optimization can be found in research surveys [5, 61, 87, 165].

Example approaches to optimizing individual ETL related tasks include snapshot difference as an efficient extraction of delta values [88, 90], data mappers [26], pivot/unpivot [29], efficient data cubes [142, 143], lineage of data transformations [28], data cleansing [48, 104, 114], efficient resumption of interrupted data flows [89], change-table techniques for incremental view maintenance [53], cardinality estimation [155], ETL tasks in the context of Map-Reduce [94, 95], and real-time processing of ETL operations [23, 67, 75, 98, 110, 111, 127, 159, 166].

ETL lifecycle. Several research approaches investigate administration aspects in ETL technology, such as monitoring and testing through regression tests [156], explaining ETL processes/tasks with natural language descriptions [128, 129, 156], and managing evolution in ETL processes [68, 105–107]. Researchers have also worked on standardization efforts, either through the identification of frequent ETL patterns [152, 167] or the specification of ETL related benchmarks [21, 131]. In addition to these efforts, a handful of research ETL prototypes has appeared in the last 20+ years [e.g., 9, 48, 70, 104, 114, 140, 151, 157, 172]. There is also a curated list of popular ETL frameworks, libraries, and software [24].

1.4 Impact

As the research thread started well beyond the time the market has been awash with commercial ETL tools, it was challenging for the research results to influence significantly the existing industry of *traditional* ETL offerings. With the exception of numerous consulting engagements [135] where several of the early ETL design approaches have been put to the test, only about a dozen leading commercial ETL offerings have adopted several of the optimization techniques described earlier. (Although the actual number of tech transfers could indeed be larger, as the majority of the commercial tools does not disclose how they operate.) Nonetheless, research has undoubtedly helped shape the next generation of ETL technology, which comprises *the Present*.

2 THE PRESENT

2.1 Evolution of the ETL architecture

Early approaches revolve around the traditional ETL architecture shown in Figure 1. In the pre-cloud era, when resources were expensive and scarce, this architecture was the only option as the three phases, extract, transform, and load, were run in concert as a pipeline, and a breakdown in one would frequently impact the others. Traditional ETL has been realized mainly in two ways: custom ETL and batch ETL. Custom ETL is built in-house using a combination of SQL and scripts in Python, Hive, etc. for bespoke target data warehouse settings. It provides great compatibility and usability at the cost of being time-consuming, labor-intensive, and error-prone. Batch ETL comes with the following characteristics: data processing in batches, periodic execution (once per day or per week), high latency, disk-based transformations, initially designed for databases, multiple copies of the data from sources to DSA then to DW. This is still a great solution for an enterprise affording to wait to collect and analyze data. However, as

nowadays this is not the most popular scenario, later efforts have relaxed all these parameters either individually or in tandem.

To satisfy the need of collecting and processing the data as soon as data changes happen, a family of real-time data integration solutions was formed. The first approach was the so-called change data capture (CDC), which is also known as logical replication. CDC detects (usually via database logs) and moves in real-time only the changed data, as opposed to moving complete data snapshots. This solution offers a near real-time data move, has low latency, but it has limited transformation capabilities and it supports mainly database sources. Because of the rather simple transformations typically met in CDC scenarios, CDC could also be characterized as a type of an ELT solution.

The ELT approach¹ moves the transformation phase from the integration platform, which now would simply collect and deliver the data, to the target data platform. Hence, ELT loads the data directly into the eventual host system and performs the transformations in-situ. In many practical cases, 'EL' now effectively means data replication and the challenge is to perform it efficiently, securely, and with high fidelity. ELT has become increasingly popular due to a number of factors. Data is being generated in ever-larger volumes, often without human input. Storage's cost is getting cheaper either on-prem or in cloud. Compute's cost has decreased over time with the plurality of open source tools (e.g., Apache Spark, Apache Hadoop, Apache Beam) and cloud offerings (e.g., AWS, Microsoft Azure, and Google Cloud). Modern cloud data platforms offer low cost solutions to analyze heterogeneous, remote, and distributed data sources in a single environment. In combination with real-time data integration that allows transforming and processing streaming data in-flight, the data can be ready for analysis the moment it arrives to the target platform. A definite shift to ELT technology has happened when enterprises started moving from on-prem data warehouses built on relational databases to Map-Reduce deployments (with Hadoop being the most popular at the time), NoSQL environments, and streaming data platforms (e.g., Kafka, Apache Flink, Apache Storm), and especially when all these increased their footprint in the cloud.

2.2 Trends in ETL processing

Based on the evolution of the ETL technology, new types of ETL (in its various forms, ETL/ELT etc.) solutions have emerged.

Streaming ETL. Modern ETL systems need to handle data generated by various types of sources at high speeds and at increasingly larger volumes [22, 166]. As an example, a recent application collects streaming data from a blockchain according to the lambda architecture and prepares it for further analysis [49]. At the same time, data sources and data consumers should be able to register to or disconnect from the processing system fast (horizontal scaling), without interrupting the continuity of the ETL process. Such requirements have led to the rise of streaming data integration systems or simply, streaming ETL [22, 23, 34, 51, 67, 75, 110, 111, 126, 166]. The data transformations carry exactly-once semantics, and are performed in-flight and usually, in-memory using state-of-the-art (distributed) stream processing techniques. However, this is exactly one of the current limitations: not all traditional ETL transformations can be performed in a streaming fashion due to their pipeline-blocking

nature. Enriching streaming ETL capabilities with additional functionality is an open research challenge.

Cloud ELT. The proliferation of device connectivity (e.g., edge-computing, internet-of-things), the cheaper storage, the many and faster connectors have increased the enterprises' demand for data to retain their competitive advantage. As the data sets and the number of available data sources grow larger, the concept of centralized data centers is aging out. Recent studies corroborate this, indicating that 81% of all enterprises have a multi-cloud strategy already laid out or in the works [20]. Hence, ETL/ELT processes are increasingly involving storage and/or processing systems hosted partially or entirely in the cloud [50, 96, 179, 180]. Cloud ELT (or cloud native ELT) aims at leveraging the full potential of the cloud technology: elastic scalability, massively parallel processing jobs, ability of routinely spin up or tear down jobs fast, horizontal/vertical autoscaling, run serverless pipelines, dynamic work rebalancing, flexible resource scheduling, and so on. Although several modern commercial offerings support some of these to some extent, how to effectively integrate such features into Cloud ELT is an excellent research direction for future work.

Reverse ETL. As the analytics technology has progressed, enterprises unlock insights from their data that were not available before. Consequently, a recent trend indicates to treat the data warehouse as another operational data store that pushes the data generated by those insights back into the operational sources the ETL process depends on [30]. For example, customer-related business data could be used to launch a targeted email marketing campaign or business insights on poor-performing advertising data could be sent back to operational sources to help the marketing team form an alternative customer engaging strategy. The rationale of reverse ETL is simple: each operational source has an isolated view of its domain, whereas the data warehouse has a global view and acts as the central repository for the entire enterprise. Hence, reverse ETL operationalizes business data by pushing the global view back to each individual operational source where it can be used in day-to-day business process. In a sense, reverse ETL closes the loop ELT creates by updating the systems it collected data in the first place. Still, further research is required for several challenges, such as schema validation, efficient sync of business and operational data, reducing the performance overhead imposed to the sources, optimizing pipeline performance, ensuring the accuracy and consistency of the data being transferred to the operational sources, and ensuring that data privacy is not violated.

2.3 Trends in ETL infrastructure

The evolution of ETL technology concurs with new directions that have opened up in the ETL/DW infrastructure.

Data lakes. A data warehouse, typically, employs a relational database to capture and store data, which implies the necessity of a schema (tabular format) to enable SQL queries, and a process, the ETL process, to implement the required schema mappings. A significant mindset change that perfectly matches with ELT is the loose interpretation of a schema and the respective schema mappings. Several applications such as analytics, machine learning, data science, full text search could also work on less structured data. Hence, another form of data repository has gained momentum in the recent years, namely the *data lake* (DL). A data lake is a centralized repository for all data, including structured, semi-structured, and unstructured. Data lakes have emerged to handle raw data on cheap storage, primarily, for data science

¹The ELT can also be met in various incarnations such as ELTL, ETLT, etc.; the main concept however does not change for the purposes of our analysis here.

	Data warehouse	Data lake
Schema	Fixed, predefined, schema-on-write	Defined at the analysis, schema-on-read
Data	Structured, processed	All data
Data quality	Curated, ground truth	Raw data
Accessibility	Changes: complicated and expensive	Highly accessible, fast to update
Applications	BI, reporting, dashboards, visualizations	ML, data discovery, data exploration, operational analytics
Users	Business analysts, developers	Business analysts, data scientists and engineers, developers, architects

Table 2: Data warehouse vs. Data lake

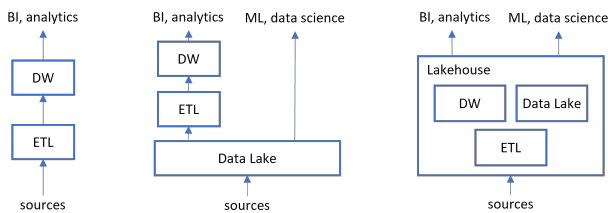


Figure 4: The BI analytics and ML evolving landscape, from left to right: (a) traditional ETL/DW architecture, (b) two-tier DW/DL architecture, (c) lakehouse architecture

and machine learning (ML). They follow a schema-on-read architecture, which allows storing data at low cost, with popular storage choices being HDFS in the first years and more recently, the cloud data lakes (e.g., S3, ADLS, GCS) that nowadays offer cheaper and more reliable archival storage.

On the flip side, generally DL does not offer typical database features such as data quality, support for transactions, consistency and isolation, whereas proper governance is essential to avoid data lakes becoming data swamps. Research has dealt with the problem of enriching data lakes with organizational functionality to assist user navigation and data exploration in various ways, including (a) optimization over a graph-based organizational structure to maximize the expected likelihood of discovering tables by navigating [103], or (b) search and management solutions for data science platform operating on schema-agnostic data repositories such as data lakes [181, 182].

Table 2 lists notable differences between data warehouses and data lakes. Enterprises typically build a two-tier data architecture that combines both systems, to enable BI on the data warehouse and ML on the data lake (see also Figure 4). However, challenges include increased storage needs for multiple data copies, extra infrastructure and operational costs, lack of trail of past analytics on the data, security and access control, and reliability, as keeping DL and DW consistent is troublesome and expensive.

Data lakehouse. Recently, several efforts have attempted to bring database and data warehouse capabilities to the data lake storage, by developing open-source storage layers that work seamlessly with popular compute engines such as Apache Spark, Presto, Flink, Hive, etc. Notable approaches include Databricks’ Delta Lake [35], Netflix’s Apache Iceberg [65], and Uber’s Apache Hudi [60]. Such efforts have led to the most recent architectural shift, namely the *data lakehouse*, (LH) [11].

The lakehouse architecture aims at overcoming the challenges of the two-tier, DW and DL design and providing the best of both

in a way that would be less complex to handle for users, reliable through simpler ETL/ELT, enabling ML and advanced analytics, and SQL-like performant (see Figure 4-right). Although the most popular commercial cloud providers already host lakehouse capabilities, the landscape of lakehouses is yet developing. Still, the various offerings converge to a clear set of characteristics. A data lakehouse is a data management system that implements the functionality of structured DW on top of unstructured DL. It supports powerful management and query optimization on par with DW, and operates on low-cost storage in an open format (e.g., Apache Parquet) accessible by the various systems typically operating in DL. This setup is an excellent fit for cloud with compute and storage separation. For example, BI applications would run on a compute cluster and ML applications would run on a GPU cluster, while both directly accessing the same storage (i.e., data, metadata, catalog, indexing, caching), either on cloud or on-premise storage systems such as HDFS. Early research work presents blueprints for a lakehouse architecture [178] and Photon, a vectorized query engine with Apache Spark API designed for lakehouses [17, 18].

Currently, early lakehouse offerings merely provide hooks to ETL tools to enable moving data to their storage layer (e.g., Databricks’ Delta Lake). However, there is more work to be done as ETL tasks need to be first-class citizens in a lakehouse architecture to orchestrate and govern data synchronization, metadata population, propagation of data to the applications through appropriate APIs (e.g., SQL, DataFrame, etc.), index and view maintenance, and so on. Example future directions include: (a) integrate ETL within a lakehouse platform, (b) devise strategies to opportunistically offload physical level transformations to ETL, (c) investigate real-time, streaming ETL capabilities and reverse ETL logic for lakehouse, and (d) provide a holistic view to tasks such as data replication, data discovery, event streaming, workflow management, and so on, as part of a generalized ETL process.

Multi-engine environment. With the plethora of databases marketed either as relational databases, NoSQL, SQL-on-Hadoop, or NewSQL, enterprises tend to employ multiple data systems, inevitably creating multiple compute and storage silos within an organization with each potentially having its own data model, query syntax, metadata, catalog, and so on. And often BI and advanced analytics require integrated access to all such data engines of the enterprise. However, given the variety of available systems, developing optimized software to query and manage data across heterogeneous systems is complex and error-prone. Although the general problem has been studied before in the context of federated databases [e.g., 31, 38, 119] and mediators [e.g., 57], in recent years, it has gained a renewed interest due to the unprecedented complexity imposed by the numerous database offerings routinely used in production.

The so-called polystore systems have been proposed to deal with this problem. Although their architecture and functionality differ, they all provide a means for mediation (either through an intermediate language or inherently) to cope with the system heterogeneity and also hooks to optimization and performant execution. Example such systems proposed include (listed in chronological order): HFMS [139], MISO [91], BigDAWG [39], IReS [37], Musketeer [52], CloudMdsQL [82, 84], RHEEM [1], Muses [73], Hybrid.Poly [108], and Polypheny-DB [173]. There have also been proposed polystore benchmarks [76, 83] and recently, a database operating system specifically designed for polystores (DBOS) [25].

Most of the so far work has focused on query usability and performance. There is little to none work regarding how to populate polystores efficiently and keep them in sync. Previous work has shown that optimizing data flows spanning multiple engines is not a trivial task, as the optimal assignment of (sub)flows to engines involves decisions on operations such as flow composition/decomposition, function shipping vs. data shipping, parallel flow execution, flow workload management, and so on [e.g., 136]. Hence, a future direction would be to study ETL processes in the context of multi-engine, multi-storage execution environments.

2.4 Alternatives to ETL

ETL technology is not the only means to enable extracting business insights from operational data. Alternative emerging directions include HTAP systems and in-situ processing.

Hybrid systems. Recent advances in research and industry related to OLTP and OLAP technologies, such as scalable transactional management and scalable analytics, and also to in-memory and cloud native database technologies, have enabled running transactional processing and analytics on the same database. This emerging architecture is collectively called HTAP or hybrid transactional and analytical processing. In principle, HTAP reduces latency in analytics by eliminating the needs for multiple copies of the same data and for moving data from operational databases to data warehouses via ETL processes. The main challenge in HTAP is to efficiently accommodate two very different workloads, the operational (many small transactions, high fraction of updates) and analytical (complex, long running, resource demanding queries) on the same database system without the execution of the one interfering with the execution of the other.

Initial efforts approached the problem with an orchestration layer on top of two carefully inter-connected OLTP and OLAP engines and showed the potential of the hybrid architecture (e.g., see Vertex [7] and CuteDB [8]). Parallel efforts include Hyper, which relies on optimistic multi-version concurrency control (MVCC) to mediate access among transactional and analytical queries [46, 78, 79]; Caldera, a prototype HTAP engine that leverages emerging hardware to handle segregated workloads and in particular, it stores data in shared memory and exploits GPGPUs to boost analytical workloads and CPUs for transactional workloads [10]; WiSer, that performs lazy resolution of read-write conflicts and aggregation constrain violations in the serialized data [16]; and commercial HTAP systems that approach the problem by employing MVCC for snapshot isolation and/or combining row-store with column-store capabilities (DB2 [113], SAP HANA [43], TiDB [59]). Recent studies investigate techniques such as elastic resource scheduling toward adaptive HTAP [115] and present a performance characterization of HTAP workloads [141]. More details can be found in a recent tutorial [93].

In-situ processing. Another direction is to perform in-situ query processing to avoid the ETL process all together and thus, reduce the data-to-query time. That is to avoid data loading while still maintaining the whole feature set of a modern database system. Toward this end, NoDB is a prototype that considers raw data files a first-class citizen, fully integrated with the query engine [4]. The core limitations of this approach include the repeated parsing, the tokenizing overhead, and the expensive data type conversion. In-situ approaches utilize adaptive indexing that maintains positional information to provide efficient access to raw data files, together with a flexible caching structure [4, 19, 99, 100].

3 THE FUTURE

One's angle could be that the evolution of ETL technology seems to have made a full circle from 'a tedious engineering task' to 'a challenging research design and optimization problem' and back to 'it has been absorbed by the data ecosystem evolution'. Another view could be that 'any data move from point A to point B that involves a transformation constitutes an ETL process' and in that sense, ETL is and will always be an integral part of data technology. Our take is that the ETL technology remains relevant as long as it adapts to the modern business needs and data technology advancements. From a research point of view, there are several potential directions for future work.

New ETL pipelines due to architectural shifts. The evolving data ecosystem we briefly described in 'The Present' has created new challenges for the ETL technology. We already mentioned several potential improvements and current limitations of emerging ETL flavors, namely streaming ETL, cloud native ELT, and reverse ETL, along with the challenges imposed by the new architectures, such as the lakehouse architecture and the multi-engine (polystore) environment (see Section 2). In particular, the lakehouse architecture opens up engaging opportunities for a variety of ETL use cases, which would leverage the combined functionality of the analytics and the operational systems to feed business intelligence, multimodal processing, and AI/ML ETL pipelines. The first pipeline resembles traditional ETL and powers reports, dashboards, and SQL-like analysis of structured data. The second pipeline targets an emerging use case involving complex, potentially streaming, analytics employing ML operations on multiplicity of data types (text, images) using a variety of programming languages and frameworks. The third pipeline brings full-blown ML pipelines within the scope of ETL providing a single pane of glass for data scientists and analysts to perform their analytics tasks. Although admittedly a non trivial task, we believe that adapting to the challenges of the new data stack will constitute the immediate next directions for the ETL technology.

UDF-fueled in-engine ETL. Beside data connectors and a handful of SQL-like operations, most ETL functionality takes place outside data engines, as the large majority of ETL operations implement complex procedural logic that cannot be expressed by means of declarative SQL. Most data engines support user-defined functions (UDFs), which in principle could implement many operations frequently encountered in ETL processes. However, UDF performance until recently was subpar due to the impedance mismatch between UDF execution and SQL processing. Recent research efforts and systems have presented significant, game-changing performance improvements in UDF execution in modern data engines, exploiting the various low-level hooks and optimizations such engines support, including vectorization, JIT/LLVM compilation, tuple/vector-at-a-time execution, function inlining, in/out-process execution, parallelization, and so on [44]. Example research prototypes and systems supporting performant UDFs include Froid [112], Mutable [54], ReSQL [47], UDO [122], and YeSQL [45].

Having performant UDFs, a valid proposal would be to move the ETL processing inside the data engine and exploit all the nice properties and features it offers, such as query optimization, ACID properties, out-of-the-box parallelization, optimal storage layouts, etc. Still, UDFs should not necessarily be seen as a competitor to ETL. On the contrary, they arise as a powerful companion to ETL and especially, ELT approaches, as the entire ETL process or part of it can now execute within the database. This

enables additional optimization opportunities as the database query optimizer can participate in the ETL process and optimize (part of) it. Optimization strategies such as operator fusion or pushing down/up operators between ETL and the database form interesting future directions. Another intriguing challenge would be to determine how to split an ETL process to ETL and UDF operations; for example, control flow logic and orchestration would likely remain outside the data engine.

Learning ETL. Managing, tuning, and optimizing ETL processes is an extremely complex task due to the plethora of ETL operations, the plurality of inter-connected compute and storage systems, the variety of processing modes (batch, streaming, on-demand), the often conflicting objectives (e.g., performance vs. maintainability vs. fault-tolerance), and the variability of the size, structure, and complexity of ETL processes. Calibrating these factors to generate ‘good’ (as in not-too-bad) ETL designs is tough –if not infeasible– to be performed manually. Early solutions to ETL optimization have provided reasonable solutions, as long as the assumed cost models are accurate (see Section 1.3); but, unfortunately this is often not the case in practical applications.

Recent advances in learning query optimization aim at complementing or even replacing a cost-based query optimizer with a learning optimizer, which aims at learning the behavior of query operators and query patterns over time and tends to learn also from previous decisions (i.e., execution plans) [162]. The concept could be explored in ETL optimization as well [58]. The various factors that affect ETL execution, such as operators, configuration parameters, data flow patterns, data sources, metadata, past executions, etc. could be encoded appropriately and then used to train a machine learning model. Techniques such as reinforcement learning or a supervised approach that have been tested in learning query optimization could be a good starting point toward learning, self-managed ETL.

Private ETL. A long overdue and rather disregarded affair in ETL technology is the issue of data privacy. To the best of our knowledge, research has not dealt with this topic so far. ETL tools have focused mostly on data protection and data security due to the worldwide restrictive legislation such as GDPR, HIPAA, CCPA, etc. Leading ETL tools provide hooks, such as encryption and hashing personally identifying information (PII), to assist ETL designers protect sensitive data and abide by regulatory compliance. Cloud-based solutions allow designers choose where their data is processed and the hosting provider to use in order to meet regulatory or data residency requirements. Most ETL tools provide also functionality to ensure data security, including the usual practices such as monitoring https traffic, TLS 1.2+ external connections, multi-factor authentication, compliance and regulatory assurances via SOC, PCI DSS, ISO/IEC 27001, etc.

However, providing private ETL computations is imperative in privacy sensitive applications, e.g., medical or financial applications. A timely and practical, future research direction is to explore privacy preserving ETL operations employing techniques such as anonymization, differential privacy, homomorphic encryption, secure multi-party computation, etc. typically used in decentralized data and federated machine learning scenarios.

Self-service data preparation. From a usability perspective, users need a single pane of glass and thus, they should be able to instantly spin up customized dashboards to observe key metrics in real-time, through on-demand reporting, dynamic graphical interfaces, and low-code or no-code environments (e.g., with natural language descriptions processed by text-to-SQL systems [77]), without depending on central analysis, a full-blown ETL process,

or complex programming. This form of *personal ETL* should process data already transformed into an appropriate, standardized format and operate at a significantly smaller scale than traditional ETL. Challenges include *reliably* converting a data analysts’ data to a standardized format, s.t. it would be joinable with their other data, ideally in-situ. From a technical standpoint, personal ETL could borrow techniques from incremental ETL or on-demand ETL based on probabilistic query processing [15, 177].

It’s a new world –again. The operational and analytics ecosystems, although evolving, they both continue to thrive and play an integral role to the data stack. However, new applications are gaining momentum and at a large extent drive the changes in the data technology. Data and business teams use modern tools to seamlessly generate insights and value from data, such as data workspaces that provide an end-to-end view of the core analytics workflow, reverse ETL that pushes insights back to the operational sources and helps automate the insight-to-action process, DataOps/MLOps to increase automation [12, 116, 164], and various ML application frameworks. Along these lines, new tasks have emerged as critical components to support key data processes and workflows, such as data preparation, data exploration, data discovery, data validation, data enrichment, feature engineering, observability, ML model auditing [109]. Modern ETL should be influenced by such tasks and provide functionality beyond schema mappings, transformation, and cleansing.

For example, a 2010’s data analyst used ETL to map two tables `company[name, address]` and `sales[product, amt]` from an operational source to a DW fact table to later perform sales forecasting. A 2023’s data scientist expects an ML-ETL tool, given appropriate input, to (a) identify the relevant csv or json files in a data lake containing lines as ‘Hewlett Pack|ProLiant DL380 Gen10|https://buy.hpe.com|2,723.54|95054’, (b) recognize that the files contain information about a *rich data type*, namely company, (c) clean erroneous entries ‘Hewlett Pack’ to ‘Hewlett Packard Enterprise’, (d) identify key features in the data set, (e) generate a model to forecast sales, and (f) present the *right* visuals to the user. Each of these tasks could be modeled as an operation in a modern ETL tool that encapsulates functionality similar to (or better, in synergy with) automated ML techniques [64].

4 CONCLUSIONS

The ETL technology and data integration in general has been the cornerstone of business intelligence, decision making, and data analytics for over 25 years. ETL thrives while at the same time it evolves along with shifting business needs and data technology advancements. As researchers and practitioners alike are exploring ways to extract value from large collections of raw data, ETL is the connecting glue to make this happen. In this paper, we presented a brief overview of the ETL history, described recent trends in the end-to-end data stack, and discussed some interesting, in our opinion, future directions that will most likely impact the next generation of ETL and data integration technology. The past 20+ years have been educating, enjoyable, and productive in devising and realizing efficient and effective ways to tame data intricacies and peculiarities blending a multiplicity of technologies and applying them in the real world. We look forward to the next 20 that will be even more exciting and fruitful.

Acknowledgements. We heartily thank the DOLAP 2023 Test-Of-Time Award Committee for the honor and our colleagues with whom we shared a thrilling 20-year journey in the ETL-land and beyond. This work has been partially supported by EU Horizon 2020 programme INODE (grant agreement No 863410).

REFERENCES

- [1] Divy Agrawal, Sanjay Chawla, Bertty Contreras-Rojas, Ahmed K. Elmagarmid, Yasser Idris, Zoi Kaoudi, Sebastian Kruse, Ji Lucas, Essam Mansour, Mourad Ouazzani, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Nan Tang, Saravanan Thirumuruganathan, and Anis Troudi. 2018. RHEEM: Enabling Cross-Platform Data Processing - May The Big Data Be With You! -. *Proc. VLDB Endow.* 11, 11 (2018), 1414–1427.
- [2] Zineb El Akkaoui and Esteban Zimányi. 2009. Defining ETL workflows using BPMN and BPEL. In *ACM DOLAP*. 41–48.
- [3] Zineb El Akkaoui, Esteban Zimányi, Jose-Norberto Mazón, and Juan Trujillo. 2013. A BPMN-Based Design and Maintenance Framework for ETL Processes. *Int. J. Data Warehous. Min.* 9, 3 (2013), 46–72.
- [4] Ioannis Alagiannis, Renata Borovica-Gajic, Miguel Branco, Stratos Idreos, and Anastasia Ailamaki. 2015. NoDB: efficient query execution on raw data files. *Commun. ACM* 58, 12 (2015), 112–121.
- [5] Syed Muhammad Fawad Ali and Robert Wrembel. 2017. From conceptual design to performance optimization of ETL workflows: current state of research and open problems. *VLDB J.* 26, 6 (2017), 777–801.
- [6] Syed Muhammad Fawad Ali and Robert Wrembel. 2019. Towards a Cost Model to Optimize User-Defined Functions in an ETL Workflow Based on User-Defined Performance Metrics. In *ADBIS (LNCS)*, Vol. 11695. 441–456.
- [7] Kevin Wilkinson Alkis Simitis. 2016. *Vertex: A Hybrid Database System for Mixed Workloads*. Technical Report HPE-2016-91. Hewlett Packard Labs. <https://www.labs.hpe.com/techreports/2016/HPE-2016-91.pdf>.
- [8] Olga Poppe Alkis Simitis, Kevin Wilkinson. 2016. *Fast Analytics in Real-time Operations Management*. Technical Report HPE-2016-93. Hewlett Packard Labs. <https://www.labs.hpe.com/techreports/2016/HPE-2016-93.pdf>.
- [9] Ove Andersen, Christian Thomsen, and Kristian Torp. 2018. SimpleETL: ETL Processing by Simple Specifications. In *DOLAP*, Vol. 2062. CEUR-WS.org.
- [10] Raja Appuswamy, Manos Karpathiotakis, Danica Porobic, and Anastasia Ailamaki. 2017. The Case For Heterogeneous HTAP. In *CIDR*.
- [11] Michael Armbrust, Tathagata Das, Sameer Paranjpye, Reynold Xin, Shixiong Zhu, Ali Ghodsi, Burak Yavuz, Mukul Murthy, Joseph Torres, Liwen Sun, Peter A. Boncz, Mostafa Mokhtar, Herman Van Hovell, Adrian Ionescu, Alicja Luszczak, Michal Switkowski, Takuya Ueshin, Xiao Li, Michal Szafranski, Pieter Senster, and Matei Zaharia. 2020. Delta Lake: High-Performance ACID Table Storage over Cloud Object Stores. *PVLDB* 13, 12 (2020), 3411–3424.
- [12] Anastasios Arvanitis, Shrivath Babu, Eric Chu, Adrian Popescu, Alkis Simitis, and Kevin Wilkinson. 2019. Automated Performance Management for the Big Data Stack. In *CIDR*.
- [13] Judith Awiti, Alejandro A. Vaisman, and Esteban Zimányi. 2019. From Conceptual to Logical ETL Design Using BPMN and Relational Algebra. In *DaWaK (LNCS)*, Vol. 11708. 299–309.
- [14] Judith Awiti, Alejandro A. Vaisman, and Esteban Zimányi. 2020. Design and implementation of ETL processes using BPMN and relational algebra. *Data Knowl. Eng.* 129 (2020), 101837.
- [15] Lorenzo Baldacci, Matteo Golfarelli, Simone Graziani, and Stefano Rizzi. 2017. QETL: An approach to on-demand ETL from non-owned data sources. *Data Knowl. Eng.* 112 (2017), 17–37.
- [16] Ronald Barber, Adam J. Storm, Yuanyuan Tian, Pinar Tözün, Yingjun Wu, Christian Garcia-Arellano, Ronen Grosman, Guy M. Lohman, C. Mohan, René Müller, Hamid Pirahesh, Vijayshankar Raman, and Richard Sidle. 2019. WiSer: A Highly Available HTAP DBMS for IoT Applications. In *IEEE ICDE*. 268–277.
- [17] Alexander Behm and Shoumik Palkar. 2022. Photon: A High-Performance Query Engine for the Lakehouse. In *CIDR*.
- [18] Alexander Behm, Shoumik Palkar, Utkarsh Agarwal, Timothy Armstrong, David Cashman, Ankur Dave, Todd Greenstein, Shant Hovsepian, Ryan Johnson, Arvind Sai Krishnan, Paul Leventis, Ala Luszczak, Prashanth Menon, Mostafa Mokhtar, Gene Pang, Sameer Paranjpye, Greg Rahn, Bart Samwel, Tom van Bussel, Herman Van Hovell, Maryann Xue, Reynold Xin, and Matei Zaharia. 2022. Photon: A Fast Query Engine for Lakehouse Systems. In *ACM SIGMOD*. 2326–2339.
- [19] Nikos Bikakis, Stavros Maroulis, George Papastefanatos, and Panos Vassiliadis. 2021. In-situ visual exploration over big raw data. *Inf. Syst.* 95 (2021).
- [20] Bill Sorenson. 2020. 10 Cloud Computing Statistics You Need to Know. Available at: <https://netgaincloud.com/10-cloud-computing-statistics-you-need-to-know>.
- [21] Christoph Boden, Andrea Spina, Tilmann Rabl, and Volker Markl. 2017. Benchmarking Data Flow Systems for Scalable Machine Learning. In *ACM SIGMOD*. 5:1–5:10.
- [22] Michal Bodziony, Szymon Roszyk, and Robert Wrembel. 2020. On Evaluating Performance of Balanced Optimization of ETL Processes for Streaming Data Sources. In *DOLAP*, Vol. 2572. 74–78.
- [23] Mihaela A. Bornea, Antonios Deligiannakis, Yannis Kotidis, and Vasilis Vassalos. 2011. Semi-Streamed Index Join for near-real time execution of ETL transformations. In *IEEE ICDE*. 159–170.
- [24] Paul Brown. 2023. awesome-etl - list of ETL tools (circa 2023). Available at: <https://github.com/pawl/awesome-etl>.
- [25] Michael J. Cafarella, David J. DeWitt, Vijay Gadepally, Jeremy Kepner, Christos Kozyrakis, Tim Kraska, Michael Stonebraker, and Matei Zaharia. 2020. A Polystore Based Database Operating System (DBOS). In *DMAH@PVLDB (LNCS)*, Vol. 12633. 3–24.
- [26] Paulo Carreira, Helena Galhardas, Antónia Lopes, and João Pereira. 2007. One-to-many data transformations through data mappers. *Data Knowl. Eng.* 62, 3 (2007), 483–503.
- [27] Xu Chu, Ihab F. Ilyas, Sanjay Krishnan, and Jiannan Wang. 2016. Data Cleaning: Overview and Emerging Challenges. In *ACM SIGMOD*. 2201–2206.
- [28] Yingwei Cui and Jennifer Widom. 2003. Lineage tracing for general data warehouse transformations. *VLDB J.* 12, 1 (2003), 41–58.
- [29] Conor Cunningham, Goetz Graefe, and César A. Galindo-Legaria. 2004. PIVOT and UNPIVOT: Optimization and Execution Strategies in an RDBMS. In *VLDB*. 998–1009.
- [30] Bibhu Dash and Swati Swayamsiddha. 2022. Reverse ETL for Improved Scalability, Observability, and Performance of Modern Operational Analytics - A Comparative Review. In *IEEE OCIT*. 491–494.
- [31] Umeshwar Dayal. 1983. Processing Queries Over Generalization Hierarchies in a Multidatabase System. In *VLDB*. 342–353.
- [32] Umeshwar Dayal, Malú Castellanos, Alkis Simitis, and Kevin Wilkinson. 2009. Data integration flows for business intelligence. In *EDBT (ACM International Conference Proceeding Series)*, Vol. 360. 1–11.
- [33] Umeshwar Dayal, Kevin Wilkinson, Alkis Simitis, and Malú Castellanos. 2009. Business Processes Meet Operational Business Intelligence. *IEEE Data Eng. Bull.* 32, 3 (2009), 35–41.
- [34] Antonios Deligiannakis, Nikos Giatrakos, Yannis Kotidis, Vasilis Samoladas, and Alkis Simitis. 2021. Extreme-Scale Interactive Cross-Platform Streaming Analytics - The INFORE Approach. In *SEA-Data@VLDB*, Vol. 2929. 7–13.
- [35] Delta Lake. 2023. Delta Lake. Available at: <https://github.com/delta-io/delta>.
- [36] David J. DeWitt, Alan Halverson, Rimma V. Nehme, Srinath Shankar, Josep Aguilar-Saborit, Artin Avanes, Miro Flaszka, and Jim Gramling. 2013. Split query processing in polybase. In *ACM SIGMOD*. 1255–1266.
- [37] Katerina Doka, Nikolaos Papailiou, Dimitrios Tsoumakos, Christos Mantas, and Nectarios Koziris. 2015. IReS: Intelligent, Multi-Engine Resource Scheduler for Big Data Analytics Workflows. In *ACM SIGMOD*. 1451–1456.
- [38] Weimin Du, Ravi Krishnamurthy, and Ming-Chien Shan. 1992. Query Optimization in a Heterogeneous DBMS. In *VLDB*. 277–291.
- [39] Jennie Duggan, Aaron J. Elmore, Michael Stonebraker, Magdalena Balazinska, Bill Howe, Jeremy Kepner, Sam Madden, David Maier, Tim Mattson, and Stanley B. Zdonik. 2015. The BigDAWG Polystore System. *SIGMOD Rec.* 44, 2 (2015), 11–16.
- [40] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vasilios S. Verykios. 2007. Duplicate Record Detection: A Survey. *IEEE TKDE* 19, 1 (2007), 1–16.
- [41] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. 2005. Data exchange: semantics and query answering. *Theor. Comput. Sci.* 336, 1 (2005), 89–124.
- [42] Ronald Fagin, Phokion G. Kolaitis, Lucian Popa, and Wang Chiew Tan. 2004. Composing Schema Mappings: Second-Order Dependencies to the Rescue. In *ACM SIGMOD*. 83–94.
- [43] Franz Färber, Norman May, Wolfgang Lehner, Philipp Große, Ingo Müller, Hannes Rauhe, and Jonathan Dees. 2012. The SAP HANA Database – An Architecture Overview. *IEEE Data Eng. Bull.* 35, 1 (2012), 28–33.
- [44] Yannis Foufoulas and Alkis Simitis. 2023. User-Defined Functions in Modern Data Engines. In *IEEE ICDE*.
- [45] Yannis E. Foufoulas, Alkis Simitis, Eleftherios Stamatiogiannakis, and Yannis E. Ioannidis. 2022. YeSQL: "You extend SQL" with Rich and Highly Performant User-Defined Functions in Relational Databases. *Proc. VLDB Endow.* 15, 10 (2022), 2270–2283.
- [46] Florian Funke, Alfons Kemper, and Thomas Neumann. 2012. Compacting Transactional Data in Hybrid OLTP & OLAP Databases. *Proc. VLDB Endow.* 5, 11 (2012), 1424–1435.
- [47] Henning Funke, Jan Mühlig, and Jens Teubner. 2022. Low-latency query compilation. *VLDB J.* 31, 6 (2022), 1171–1184.
- [48] Helena Galhardas, Daniela Florescu, Dennis E. Shasha, and Eric Simon. 2000. AJAX: An Extensible Data Cleaning Tool. In *ACM SIGMOD*. 590.
- [49] Roberta Galici, Laura Ordile, Michele Marchesi, Andrea Pinna, and Roberto Tonelli. 2020. Applying the ETL Process to Blockchain Data. Prospect and Findings. *Inf.* 11, 4 (2020), 204.
- [50] Victor Giannakouris, Alejandro Fernandez, Alkis Simitis, and Shrivath Babu. 2019. Cost-Effective, Workload-Adaptive Migration of Big Data Applications to the Cloud. In *ACM SIGMOD*. 1909–1912.
- [51] Nikos Giatrakos, David Arnu, Theodoros Bitsakis, Antonios Deligiannakis, Minos N. Garofalakis, Ralf Klinkenberg, Aris Konidaris, Antonis Kontaxakis, Yannis Kotidis, Vasilis Samoladas, Alkis Simitis, George Stamatakis, Fabian Temme, Mate Torok, Edwin Yaqub, Arnau Montagud, Miguel Ponce de Leon, Holger Arndt, and Stefan Burkard. 2020. INFORE: Interactive Cross-platform Analytics for Everyone. In *ACM CIKM*. 3389–3392.
- [52] Ionel Gog, Malte Schwarzkopf, Natacha Crooks, Matthew P. Grosvenor, Allen Clement, and Steven Hand. 2015. Musketeer: all for one, one for all in data processing systems. In *EuroSys*. 2:1–2:16.
- [53] Himanshu Gupta and Inderpal Singh Mumick. 2006. Incremental maintenance of aggregate and outerjoin expressions. *Inf. Syst.* 31, 6 (2006), 435–464.
- [54] Immanuel Haffner and Jens Dittrich. 2023. A simplified Architecture for Fast, Adaptive Compilation and Execution of SQL Queries. In *EDBT*. 1–13.
- [55] Ramanujam Halasipuram, Prasad M. Deshpande, and Sriram Padmanabhan. 2014. Determining Essential Statistics for Cost Based Optimization of an ETL Workflow. In *EDBT*. 307–318.
- [56] Alon Y. Halevy, Anand Rajaraman, and Joann J. Ordille. 2006. Data Integration: The Teenage Years. In *VLDB*. 9–16.

- [57] Joachim Hammer, Hector Garcia-Molina, Kelly Ireland, Yannis Papakonstantinou, Jeffrey D. Ullman, and Jennifer Widom. 1995. Information Translation, Mediation, and Mosaic-Based Browsing in the TSIMMIS System. In *ACM SIGMOD*. 483.
- [58] Ian Henry. 2018. Machine Learning Prototype – Automating ETL. Available at: <https://blogs.sap.com/2018/11/09/machine-learning-prototype-automating-etl>.
- [59] Dongxu Huang, Qi Liu, Qiu Cui, Zhuhe Fang, Xiaoyu Ma, Fei Xu, Li Shen, Liu Tang, Yuxing Zhou, Menglong Huang, Wan Wei, Cong Liu, Jian Zhang, Jianjun Li, Xuelian Wu, Lingyu Song, Ruoxi Sun, Shuaipeng Yu, Lei Zhao, Nicholas Cameron, Liquan Pei, and Xin Tang. 2020. TiDB: A Raft-based HTAP Database. *Proc. VLDB Endow.* 13, 12 (2020), 3072–3084.
- [60] Hudi. 2023. Apache Hudi. Available at: <https://hudi.apache.org>.
- [61] Fabian Hueske and Volker Markl. 2014. Optimization of Massively Parallel Data Flows. In *Large-Scale Data Analytics*. 41–74.
- [62] Fabian Hueske, Mathias Peters, Aljoscha Krettek, Matthias Ringwald, Kostas Tzoumas, Volker Markl, and Johann-Christoph Freytag. 2013. Peeking into the optimization of data flow programs with MapReduce-style UDFs. In *IEEE ICDE*. 1292–1295.
- [63] Fabian Hueske, Mathias Peters, Matthias Sax, Astrid Rheinländer, Rico Bergmann, Aljoscha Krettek, and Kostas Tzoumas. 2012. Opening the Black Boxes in Data Flow Optimization. *Proc. VLDB Endow.* 5, 11 (2012), 1256–1267.
- [64] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren (Eds.). 2019. *Automated Machine Learning - Methods, Systems, Challenges*. Springer.
- [65] Iceberg. 2023. Apache Iceberg. Available at: <https://iceberg.apache.org>.
- [66] Theodore Johnson and Tamraparni Dasu. 2003. Data Quality and Data Cleaning: An Overview. In *ACM SIGMOD*. 681.
- [67] Thomas Jörg and Stefan Dessloch. 2009. Near Real-Time Data Warehousing Using State-of-the-Art ETL Tools. In *BIRTE*, Vol. 41. 100–117.
- [68] Petar Jovanovic, Oscar Romero, Alkis Simitsis, and Alberto Abelló. 2012. Integrating ETL Processes from Information Requirements. In *DaWaK (LNCS)*, Vol. 7448. 65–80.
- [69] Petar Jovanovic, Oscar Romero, Alkis Simitsis, and Alberto Abelló. 2016. Incremental Consolidation of Data-Intensive Multi-Flows. *IEEE TKDE* 28, 5 (2016), 1203–1216.
- [70] Petar Jovanovic, Oscar Romero, Alkis Simitsis, Alberto Abelló, Héctor Candón, and Sergi Nadal. 2015. Quarry: Digging Up the Gems of Your Data Treasury. In *EDBT*. 549–552.
- [71] Petar Jovanovic, Alkis Simitsis, and Kevin Wilkinson. 2014. BabbleFlow: a translator for analytic data flow programs. In *ACM SIGMOD*. ACM, 713–716.
- [72] Petar Jovanovic, Alkis Simitsis, and Kevin Wilkinson. 2014. Engine independence for logical analytic flows. In *IEEE ICDE*. 1060–1071.
- [73] Abdulrahman Kaitoua, Tilmann Rabl, Asterios Katsifodimos, and Volker Markl. 2019. Muses: Distributed Data Migration System for Polystores. In *IEEE ICDE*. 1602–1605.
- [74] Anastasios Karagiannis, Panos Vassiliadis, and Alkis Simitsis. 2013. Scheduling strategies for efficient ETL execution. *Inf. Syst.* 38, 6 (2013), 927–945.
- [75] Alexandros Karakasidis, Panos Vassiliadis, and Evangelia Pitoura. 2005. ETL queues for active data warehousing. In *IQIS@ACM SIGMOD*. ACM, 28–39.
- [76] Jeyhun Karimov, Tilmann Rabl, and Volker Markl. 2018. PolyBench: The First Benchmark for Polystores. In *TPCTC (LNCS)*, Vol. 11135. 24–41.
- [77] George Katsogiannis-Meimarakis and Georgia Koutrika. 2023. A survey on deep learning approaches for text-to-SQL. *The VLDB Journal* (2023).
- [78] Alfons Kemper and Thomas Neumann. 2011. HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots. In *IEEE ICDE*. 195–206.
- [79] Alfons Kemper, Thomas Neumann, Florian Funke, Viktor Leis, and Henrik Mühle. 2012. HyPer: Adapting Columnar Main-Memory Data Management for Transactional AND Query Processing. *IEEE Data Eng. Bull.* 35, 1 (2012), 46–51.
- [80] Ralph Kimball, Margy Ross, Warren Thornthwaite, Joy Mundy, and Bob Becker. 2008. *The Data Warehouse Lifecycle Toolkit: Practical Techniques for Building Data Warehouse and Business Intelligence Systems*. Wiley.
- [81] Phokion G. Kolaitis. 2005. Schema mappings, data exchange, and metadata management. In *ACM SIGMOD*. 61–75.
- [82] Boyan Kolev, Carlyna Bondiombouy, Patrick Valduriez, Ricardo Jiménez-Peris, Raquel Pau, and José Pereira. 2016. The CloudMdsQL Multistore System. In *ACM SIGMOD*. 2113–2116.
- [83] Boyan Kolev, Raquel Pau, Aleksandra Levchenko, Patrick Valduriez, Ricardo Jiménez-Peris, and José Pereira. 2016. Benchmarking polystores: The CloudMdsQL experience. In *IEEE ICDE*. 2574–2579.
- [84] Boyan Kolev, Patrick Valduriez, Carlyna Bondiombouy, Ricardo Jiménez-Peris, Raquel Pau, and José Pereira. 2016. CloudMdsQL: querying heterogeneous cloud data stores with a common language. *Distributed Parallel Databases* 34, 4 (2016), 463–503.
- [85] Nick Koudas, Sunita Sarawagi, and Divesh Srivastava. 2006. Record linkage: similarity measures and algorithms. In *ACM SIGMOD*. ACM, 802–803.
- [86] Nick Koudas and Divesh Srivastava. 2005. Approximate Joins: Concepts and Techniques. In *VLDB*. ACM, 1363.
- [87] Georgia Kougka, Anastasios Gounaris, and Alkis Simitsis. 2018. The many faces of data-centric workflow optimization: a survey. *Int. J. Data Sci. Anal.* 6, 2 (2018), 81–107.
- [88] Wilburt Labio and Hector Garcia-Molina. 1996. Efficient Snapshot Differential Algorithms for Data Warehousing. In *VLDB*. 63–74.
- [89] Wilburt Labio, Janet L. Wiener, Hector Garcia-Molina, and Vlad Gorelik. 2000. Efficient Resumption of Interrupted Warehouse Loads. In *ACM SIGMOD*. 46.
- [90] Wilburt Labio, Jun Yang, Yingwei Cui, Hector Garcia-Molina, and Jennifer Widom. 2000. Performance Issues in Incremental Warehouse Maintenance. In *VLDB*. 461–472.
- [91] Jeff LeFevre, Jagan Sankaranarayanan, Hakan Hacigümüs, Jun’ichi Tatemura, Neoklis Polyzotis, and Michael J. Carey. 2014. MISO: soup up big data query processing with a multistore system. In *ACM SIGMOD*. 1591–1602.
- [92] Maurizio Lenzerini. 2002. Data Integration: A Theoretical Perspective. In *ACM SIGMOD*. 233–246.
- [93] Guoliang Li and Chao Zhang. 2022. HTAP Databases: What is New and What is Next. In *ACM SIGMOD*. 2483–2488.
- [94] Xiufeng Liu, Christian Thomsen, and Torben Bach Pedersen. 2012. MapReduce-based Dimensional ETL Made Easy. *Proc. VLDB Endow.* 5, 12 (2012), 1882–1885.
- [95] Xiufeng Liu, Christian Thomsen, and Torben Bach Pedersen. 2013. ETLMR: A Highly Scalable Dimensional ETL Framework Based on MapReduce. *Trans. Large Scale Data Knowl. Centered Syst.* 8 (2013), 1–31.
- [96] Xiufeng Liu, Christian Thomsen, and Torben Bach Pedersen. 2014. CloudETL: scalable dimensional ETL for hive. In *IDEAS*. 195–206.
- [97] Sergio Luján-Mora, Panos Vassiliadis, and Juan Trujillo. 2004. Data Mapping Diagrams for Data Warehouse Design with UML. In *ER (LNCS)*, Vol. 3288. 191–204.
- [98] Gang Luo, Jeffrey F. Naughton, Curt J. Ellmann, and Michael Watzke. 2006. Transaction Reordering and Grouping for Continuous Data Loading. In *BIRTE (LNCS)*, Vol. 4365. 34–49.
- [99] Stavros Maroulis, Nikos Bikakis, George Papastefanatos, Panos Vassiliadis, and Yannis Vassiliou. 2021. RawVis: A System for Efficient In-situ Visual Analytics. In *ACM SIGMOD*. 2760–2764.
- [100] Stavros Maroulis, Nikos Bikakis, George Papastefanatos, Panos Vassiliadis, and Yannis Vassiliou. 2023. Resource-aware adaptive indexing for in situ visual exploration and analytics. *VLDB J.* 32, 1 (2023), 199–227.
- [101] Rana Faisal Munir, Sergi Nadal, Oscar Romero, Alberto Abelló, Petar Jovanovic, Maik Thiele, and Wolfgang Lehner. 2018. Intermediate Results Materialization Selection and Format for Data-Intensive Flows. *Fundam. Informaticae* 163, 2 (2018), 111–138.
- [102] Emona Nakuçi, Vasileios Theodorou, Petar Jovanovic, and Alberto Abelló. 2014. Bijoux: Data Generator for Evaluating ETL Process Quality. In *ACM DOLAP*. 23–32.
- [103] Fatemeh Nargesian, Ken Q. Pu, Bahar Ghadiri Bashardoost, Erkang Zhu, and Renée J. Miller. 2023. Data Lake Organization. *IEEE TKDE* 35, 1 (2023), 237–250.
- [104] Felix Naumann, Alexander Bilke, Jens Bleiholder, and Melanie Weis. 2006. Data Fusion in Three Steps: Resolving Schema, Tuple, and Value Inconsistencies. *IEEE Data Eng. Bull.* 29, 2 (2006), 21–31.
- [105] George Papastefanatos, Panos Vassiliadis, Alkis Simitsis, Timos K. Sellis, and Yannis Vassiliou. 2009. Rule-Based Management of Schema Changes at ETL Sources. In *ADBIS (LNCS)*, Vol. 5968. 55–62.
- [106] George Papastefanatos, Panos Vassiliadis, Alkis Simitsis, and Yannis Vassiliou. 2009. Policy-Regulated Management of ETL Evolution. *J. Data Semant.* 13 (2009), 147–177.
- [107] George Papastefanatos, Panos Vassiliadis, Alkis Simitsis, and Yannis Vassiliou. 2012. Metrics for the Prediction of Evolution Impact in ETL Ecosystems: A Case Study. *J. Data Semant.* 1, 2 (2012), 75–97.
- [108] Maksim Podkorytov and Michael N. Gubanov. 2019. HybridPoly: A Consolidated Interactive Analytical Polystore System. In *IEEE ICDE*. 1996–1999.
- [109] Neoklis Polyzotis, Sudip Roy, Steven Euijong Whang, and Martin Zinkevich. 2018. Data Lifecycle Challenges in Production Machine Learning: A Survey. *SIGMOD Rec.* 47, 2 (2018), 17–28.
- [110] Neoklis Polyzotis, Spiros Skiadopoulos, Panos Vassiliadis, Alkis Simitsis, and Nils-Erik Frantzell. 2007. Supporting Streaming Updates in an Active Data Warehouse. In *IEEE ICDE*. 476–485.
- [111] Neoklis Polyzotis, Spiros Skiadopoulos, Panos Vassiliadis, Alkis Simitsis, and Nils-Erik Frantzell. 2008. Meshing Streaming Updates with Persistent Data in an Active Data Warehouse. *IEEE TKDE* 20, 7 (2008), 976–991.
- [112] Karthik Ramachandra, Kwanghyun Park, K. Venkatesh Emani, Alan Halverson, César A. Galindo-Legaria, and Conor Cunningham. 2017. Froid: Optimization of Imperative Programs in a Relational Database. *Proc. VLDB Endow.* 11, 4 (2017), 432–444.
- [113] Vijayshankar Raman, Gopi K. Attaluri, Ronald Barber, Naresh Chainani, David Kalmuk, Vincent KulandaiSamy, Jens Leenstra, Sam Lightstone, Shaorong Liu, Guy M. Lohman, Tim Malkemus, René Müller, Ippokratis Pandis, Berni Schiefer, David Sharpe, Richard Sidle, Adam J. Storm, and Liping Zhang. 2013. DB2 with BLU Acceleration: So Much More than Just a Column Store. *Proc. VLDB Endow.* 6, 11 (2013), 1080–1091.
- [114] Vijayshankar Raman and Joseph M. Hellerstein. 2001. Potter’s Wheel: An Interactive Data Cleaning System. In *VLDB*. 381–390.
- [115] Aunn Raza, Periklis Chrysogelos, Angelos-Christos G. Anadiotis, and Anastasia Ailamaki. 2020. Adaptive HTAP through Elastic Resource Scheduling. In *ACM SIGMOD*. 2043–2054.
- [116] Cédric Renggli, Luka Rimanic, Nezihe Merve Gürel, Bojan Karlas, Wentao Wu, and Ce Zhang. 2021. A Data Quality-Driven View of MLOps. *IEEE Data Eng. Bull.* 44, 1 (2021), 11–23.
- [117] Astrid Rheinländer, Arvid Heise, Fabian Hueske, Ulf Leser, and Felix Naumann. 2015. SOFA: An extensible logical optimizer for UDF-heavy data flows. *Inf. Syst.* 52 (2015), 96–125.

- [118] Oscar Romero, Alkis Simitsis, and Alberto Abelló. 2011. GEM: Requirement-Driven Generation of ETL and Multidimensional Conceptual Designs. In *DaWaK (LNCS)*, Vol. 6862. 80–95.
- [119] Mary Tork Roth, Manish Arya, Laura M. Haas, Michael J. Carey, William F. Cody, Ronald Fagin, Peter M. Schwarz, Joachim Thomas, and Edward L. Wimmers. 1996. The Garlic Project. In *ACM SIGMOD*. 557.
- [120] Lutz Schlesinger, Florian Irmert, and Wolfgang Lehner. 2005. Supporting the ETL-process by Web Service technologies. *Int. J. Web Grid Serv.* 1, 1 (2005), 31–47.
- [121] Timos K. Sellis and Alkis Simitsis. 2007. ETL Workflows: From Formal Specification to Optimization. In *ADBIS (LNCS)*, Vol. 4690. 1–11.
- [122] Moritz Sichert and Thomas Neumann. 2022. User-Defined Operators: Efficiently Integrating Custom Algorithms into Modern Databases. *Proc. VLDB Endow.* 15, 5 (2022), 1119–1131.
- [123] Alkis Simitsis. 2003. List of ETL tools (circa 2003). Available at: <https://web.imsi.athenarc.gr/~alkis/publications/ETLTools.htm>.
- [124] Alkis Simitsis. 2004. *Modeling and Optimization of Extraction-Transformation-Loading (ETL) Processes in Data Warehouse Environments*. Ph.D. Dissertation.
- [125] Alkis Simitsis. 2005. Mapping conceptual to logical models for ETL processes. In *ACM DOLAP*. ACM, 67–76.
- [126] Alkis Simitsis, Chetan Gupta, Song Wang, and Umeshwar Dayal. 2010. Partitioning real-time ETL workflows. In *IEEE ICDE Workshops*. 159–162.
- [127] Alkis Simitsis, Chetan Gupta, Kevin Wilkinson, and Umeshwar Dayal. 2012. Optimizing Flows for Real Time Operations Management. In *SSDBM (LNCS)*, Vol. 7338. 607–612.
- [128] Alkis Simitsis, Dimitrios Skoutas, and Malú Castellanos. 2008. Natural language reporting for ETL processes. In *ACM DOLAP*. 65–72.
- [129] Alkis Simitsis, Dimitrios Skoutas, and Malú Castellanos. 2010. Representation of conceptual ETL designs in natural language using Semantic Web technology. *Data Knowl. Eng.* 69, 1 (2010), 96–115.
- [130] Alkis Simitsis and Panos Vassiliadis. 2008. A method for the mapping of conceptual designs to logical blueprints for ETL processes. *Decis. Support Syst.* 45, 1 (2008), 22–40.
- [131] Alkis Simitsis, Panos Vassiliadis, Umeshwar Dayal, Anastasios Karagiannis, and Vasiliki Tziouvara. 2009. Benchmarking ETL Workflows. In *TPCTC (LNCS)*, Vol. 5895. 199–220.
- [132] Alkis Simitsis, Panos Vassiliadis, and Timos K. Sellis. 2005. Optimizing ETL Processes in Data Warehouses. In *IEEE ICDE*. 564–575.
- [133] Alkis Simitsis, Panos Vassiliadis, and Timos K. Sellis. 2005. State-Space Optimization of ETL Workflows. *IEEE TKDE* 17, 10 (2005), 1404–1419.
- [134] Alkis Simitsis, Panos Vassiliadis, Manolis Terrovitis, and Spiros Skiadopoulos. 2005. Graph-Based Modeling of ETL Activities with Multi-level Transformations and Updates. In *DaWaK (LNCS)*, Vol. 3589. 43–52.
- [135] Alkis Simitsis, Kevin Wilkinson, Malú Castellanos, and Umeshwar Dayal. 2009. QoX-driven ETL design: reducing the cost of ETL consulting engagements. In *ACM SIGMOD*. 953–960.
- [136] Alkis Simitsis, Kevin Wilkinson, Malú Castellanos, and Umeshwar Dayal. 2012. Optimizing analytic data flows for multiple execution engines. In *ACM SIGMOD*. 829–840.
- [137] Alkis Simitsis, Kevin Wilkinson, and Umeshwar Dayal. 2013. Hybrid Analytic Flows - the Case for Optimization. *Fundam. Informaticae* 128, 3 (2013), 303–335.
- [138] Alkis Simitsis, Kevin Wilkinson, Umeshwar Dayal, and Malú Castellanos. 2010. Optimizing ETL workflows for fault-tolerance. In *IEEE ICDE*. 385–396.
- [139] Alkis Simitsis, Kevin Wilkinson, Umeshwar Dayal, and Meichun Hsu. 2013. HFMS: Managing the lifecycle and complexity of hybrid analytic data flows. In *ICDE IEEE*. 1174–1185.
- [140] Alkis Simitsis, Kevin Wilkinson, and Petar Jovanovic. 2013. xPAD: a platform for analytic data flows. In *ACM SIGMOD*. 1109–1112.
- [141] Utku Sirin, Sandhya Dwarkadas, and Anastasia Ailamaki. 2021. Performance Characterization of HTAP Workloads. In *IEEE ICDE*. 1829–1834.
- [142] Yannis Sismanis, Antonios Deligiannakis, Yannis Kotidis, and Nick Rousopoulos. 2003. Hierarchical dwarfs for the rollup cube. In *ACM DOLAP*. 17–24.
- [143] Yannis Sismanis, Antonios Deligiannakis, Nick Roussopoulos, and Yannis Kotidis. 2002. Dwarf: shrinking the PetaCube. In *ACM SIGMOD*, Michael J. Franklin, Bongki Moon, and Anastasia Ailamaki (Eds.). ACM, 464–475.
- [144] Dimitrios Skoutas and Alkis Simitsis. 2006. Designing ETL processes using semantic web technologies. In *ACM DOLAP*. 67–74.
- [145] Dimitrios Skoutas and Alkis Simitsis. 2007. Flexible and Customizable NL Representation of Requirements for ETL processes. In *NLDB (LNCS)*, Vol. 4592. 433–439.
- [146] Dimitrios Skoutas and Alkis Simitsis. 2007. Ontology-Based Conceptual Design of ETL Processes for Both Structured and Semi-Structured Data. *Int. J. Semantic Web Inf. Syst.* 3, 4 (2007), 1–24.
- [147] Dimitrios Skoutas, Alkis Simitsis, and Timos K. Sellis. 2009. Ontology-Driven Conceptual Design of ETL Processes Using Graph Transformations. *J. Data Semant.* 13 (2009), 120–146.
- [148] George Stamatakis, Antonis Kontaxakis, Alkis Simitsis, Nikos Giatrakis, and Antonios Deligiannakis. 2022. SheerMP: Optimized Streaming Analytics-as-a-Service over Multi-site and Multi-platform Settings. In *EDBT*. 2:558–2:561.
- [149] Vasileios Theodorou, Alberto Abelló, Wolfgang Lehner, and Maik Thiele. 2016. Quality measures for ETL processes: from goals to implementation. *Concurr. Comput. Pract. Exp.* 28, 15 (2016), 3969–3993.
- [150] Vasileios Theodorou, Alberto Abelló, Maik Thiele, and Wolfgang Lehner. 2014. A Framework for User-Centered Declarative ETL. In *DOLAP*. 67–70.
- [151] Vasileios Theodorou, Alberto Abelló, Maik Thiele, and Wolfgang Lehner. 2015. POIESIS: a Tool for Quality-aware ETL Process Redesign. In *EDBT*. 545–548.
- [152] Vasileios Theodorou, Alberto Abelló, Maik Thiele, and Wolfgang Lehner. 2017. Frequent patterns in ETL workflows: An empirical approach. *Data Knowl. Eng.* 112 (2017), 1–16.
- [153] Vasileios Theodorou, Petar Jovanovic, Alberto Abelló, and Emona Nakuçi. 2017. Data generator for evaluating ETL process quality. *Inf. Syst.* 63 (2017), 80–100.
- [154] Maik Thiele, Ulrike Fischer, and Wolfgang Lehner. 2009. Partition-based workload scheduling in living data warehouse environments. *Inf. Syst.* 34, 4-5 (2009), 382–399.
- [155] Maik Thiele, Tim Kiefer, and Wolfgang Lehner. 2009. Cardinality estimation in ETL processes. In *ACM DOLAP*. 57–64.
- [156] Christian Thomsen and Torben Bach Pedersen. 2006. ETLDiff: A Semi-automatic Framework for Regression Test of ETL Software. In *DaWaK (LNCS)*, Vol. 4081. 1–12.
- [157] Christian Thomsen and Torben Bach Pedersen. 2009. pygrametl: a powerful programming framework for extract-transform-load programmers. In *ACM DOLAP*. 49–56.
- [158] Christian Thomsen and Torben Bach Pedersen. 2011. Easy and effective parallel programmable ETL. In *ACM DOLAP*. 37–44.
- [159] Christian Thomsen, Torben Bach Pedersen, and Wolfgang Lehner. 2008. RiTE: Providing On-Demand Data for Right-Time Data Warehousing. In *IEEE ICDE*. 456–465.
- [160] Rizkallah Touma, Oscar Romero, and Petar Jovanovic. 2015. Supporting Data Integration Tasks with Semi-Automatic Ontology Construction. In *ACM DOLAP*. 89–98.
- [161] Juan Trujillo and Sergio Luján-Mora. 2003. A UML Based Approach for Modeling ETL Processes in Data Warehouses. In *ER (LNCS)*, Vol. 2813. 307.
- [162] Dimitris Tsemmis and Alkis Simitsis. 2022. Database Optimizers in the Era of Learning. In *IEEE ICDE*. 3213–3216.
- [163] Vasiliki Tziouvara, Panos Vassiliadis, and Alkis Simitsis. 2007. Deciding the physical implementation of ETL workflows. In *ACM DOLAP*. 49–56.
- [164] Manasi Vartak. 2021. From ML Models to Intelligent Applications: The Rise of MLOps. *Proc. VLDB Endow.* 14, 13 (2021), 3419.
- [165] Panos Vassiliadis. 2011. A Survey of Extract-Transform-Load Technology. In *Integrations of Data Warehousing, Data Mining and Database Technologies - Innovative Approaches*. 171–199.
- [166] Panos Vassiliadis and Alkis Simitsis. 2009. Near Real Time ETL. In *New Trends in Data Warehousing and Data Analysis*. Annals of Information Systems, Vol. 3. Springer, 1–31.
- [167] Panos Vassiliadis, Alkis Simitsis, and Eftychia Baikousi. 2009. A taxonomy of ETL activities. In *ACM DOLAP*. 25–32.
- [168] Panos Vassiliadis, Alkis Simitsis, Panos Georgantas, and Manolis Terrovitis. 2003. A Framework for the Design of ETL Scenarios. In *CAiSE (LNCS)*, Vol. 2681. 520–535.
- [169] Panos Vassiliadis, Alkis Simitsis, Panos Georgantas, Manolis Terrovitis, and Spiros Skiadopoulos. 2005. A generic and customizable framework for the design of ETL scenarios. *Inf. Syst.* 30, 7 (2005), 492–525.
- [170] Panos Vassiliadis, Alkis Simitsis, and Spiros Skiadopoulos. 2002. Conceptual modeling for ETL processes. In *DOLAP*. 14–21.
- [171] Panos Vassiliadis, Alkis Simitsis, Manolis Terrovitis, and Spiros Skiadopoulos. 2005. Blueprints and Measures for ETL Workflows. In *ER*, Vol. 3716. 385–400.
- [172] Panos Vassiliadis, Zografoula Vagena, Spiros Skiadopoulos, Nikos Karayannis, and Timos K. Sellis. 2001. ARKTOS: towards the modeling, design, control and execution of ETL processes. *Inf. Syst.* 26, 8 (2001), 537–561.
- [173] Marco Vogt, Nils Hansen, Jan Schönholz, David Lengweiler, Isabel Geissmann, Sebastian Philipp, Alexander Stierner, and Heiko Schultdt. 2020. Polypheny-DB: Towards Bridging the Gap Between Polystores and HTAP Systems. In *DMAH@PVLDB (LNCS)*, Vol. 12633. 25–36.
- [174] Kevin Wilkinson and Alkis Simitsis. 2011. Designing integration flows using hypercubes. In *EDBT*. 503–508.
- [175] Kevin Wilkinson, Alkis Simitsis, Malú Castellanos, and Umeshwar Dayal. 2010. Leveraging Business Process Models for ETL Design. In *ER (LNCS)*, Vol. 6412. 15–30.
- [176] Chen Xu, Markus Holzemer, Manohar Kaul, and Volker Markl. 2016. Efficient fault-tolerance for iterative graph processing on distributed dataflow systems. In *IEEE ICDE*. 613–624.
- [177] Ying Yang, Niccolò Meneghetti, Ronny Fehling, Zhen Hua Liu, and Oliver Kennedy. 2015. Lenses: An On-Demand Approach to ETL. *Proc. VLDB Endow.* 8, 12 (2015), 1578–1589.
- [178] Matei Zaharia, Ali Ghodsi, Reynold Xin, and Michael Armbrust. 2021. Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics. In *CIDR*.
- [179] Eftim Zdravevski, Cas Apanowicz, Krzysztof Stencel, and Dominik Slezak. 2019. Scalable Cloud-based ETL for Self-serving Analytics. In *ICDM*. 317–331.
- [180] Eftim Zdravevski, Petre Lameski, Ace Dimitrievski, Marek Grzegorowski, and Cas Apanowicz. 2019. Cluster-size optimization within a cloud-based ETL framework for Big Data. In *IEEE BigData*. 3754–3763.
- [181] Yi Zhang and Zachary G. Ives. 2019. Juneau: Data Lake Management for Jupyter. *Proc. VLDB Endow.* 12, 12 (2019), 1902–1905.
- [182] Yi Zhang and Zachary G. Ives. 2020. Finding Related Tables in Data Lakes for Interactive Data Science. 1951–1966.