# Report on the Workshop on Factorized Databases

Dan Olteanu
Department of Informatics
University of Zurich

## 1 Overview

The workshop took place in Zurich and online from August 2 to 4, 2022. It was attended by researchers from 17 academic institutions and industry labs, including Microsoft Gray Systems Lab, Omics Data Automation, Oracle Labs Zurich, RelationalAI, and TigerGraph. It featured 18 talks and plenty of opportunities for discussions. The vast majority of participants attended in person.

The talks and the list of participants are available at the workshop's webpage: `https://fdbresearch.github.io/FDBworkshop2022.html`.

## 2 Factorized Databases in Brief

Factorized representations are relational algebra expressions built using the union operator, the Cartesian product operator and data values. Succinctness is achieved by using the distributivity of product over union. Factorization is complementary to value-based compression, such as run-length encoding; the two can be combined to achieve an even greater compression factor.

Factorized databases build on two observations:

First, the tabular representation of the answers to conjunctive queries over relational databases entail redundancy that can be avoided by a lossless, yet more succinct *factorized* representation.

Second, for any conjunctive query, factorized representations of the query answer can be computed directly from the input database and in time proportional to their sizes and the input database size.

Since their introduction a decade ago (Olteanu and Závodný, ICDT 2012; Bakibayev, Olteanu, and Závodný, PVLDB 2012), there has been great progress on the theory, systems and applications of factorized databases. What makes them interesting to study and use in practice is their proper balance between succinctness and efficiency of subsequent processing. An increasing body of follow-up work shows that diverse processing can be executed directly on such factorized representations, so without the need to de-factorize. The work showcased at the workshop can be broadly classified as follows:

1. Foundations of factorized databases;
2. Enumeration and ranking of query answers;
3. Succinct query provenance;
4. Graph databases;
5. Factorized in-database analytics;
6. Factorized machine learning.

## 3 Technical Program

This workshop offered a brief look at progress on the above six areas of research and a great opportunity to look ahead. Further reading on this progress is available at `https://fdbresearch.github.io/impact.html`.

### 3.1 Foundations of Factorized Databases

Factorized databases exploit the structure of the conjunctive query to avoid the full tabular materialization of the query answer. Yet what happens in case there is no query to guide the construction of the factorization?

Dan Suciu (University of Washington) proposed an information-theoretic view of the problem of factorizing a noisy input relation. Reformulated, the problem is to synthesize a join over smaller relations such that its answer approximates well the input relation. He put forward an approach to compute a good approximate factorization that exploits approximate dependencies in the data. In the language of information theory, approximate functional dependencies become constraints on conditional entropy, approximate multi-valued dependencies become constraints on mutual information, while the traditional axioms for such dependencies become simple Shannon inequalities.

## 3.2 Query Enumeration and Ranking

An important property of the factorized representation of a relation is that, despite its succinctness, the tuples in the relation can be enumerated with constant delay, which is as efficient as from a tabular representation.

Nofar Carmeli (ENS Paris) presented a dichotomy for the problems of quantile query evaluation and of direct access to the $k$-th answer of a conjunctive query according to a user-specified order: There is a syntactic characterization of all conjunctive queries that admit logarithmic delay after linear-time preprocessing needed to construct a succinct representation of the query answer. An equivalent syntactic characterization was previously given for the class of group-by aggregate and order-by queries whose answer tuples can be enumerated with constant delay over factorized databases.

Paris Koutris (University of Wisconsin-Madison) considered the trade-off between join materialization and answer enumeration for a class of acyclic queries. He also discussed the answer enumeration problem for join queries and a given ranking function. Deep further analyzed such trade-offs for several classes of queries with projection, such as $k$-path reachability.

Antoine Amarilli (Télécom Paris-Tech) extended the constant-delay enumeration result from factorized databases to general circuits in decomposable deterministic negation normal form. This result has applications to the enumeration problem for: the matches of document spanners on words; the answers to monadic second-order queries on trees with support for efficient update operations; and the extraction results for enumeration grammars.

## 3.3 Query Provenance

Factorization can be also applied to the provenance polynomial of a query. Conjunctive queries can be classified based on how well their provenance polynomials factorize, regardless of the input database: Hierarchical queries are precisely the conjunctive queries whose provenance polynomials can be factorized such that each variable (representing an input tuple) only occurs a constant number of times; if the hierarchical query has no self-join, then there is a factorization of the provenance polynomial where each variable occurs at most once. This has applications to probabilistic databases, where such variables are random and carry probability distributions. In this setting, the hierarchical queries (without self-joins) can be computed in polynomial time in the size of the database.

Daniel Deutch (Tel Aviv University) presented the problem of computing Shapley values in query answering and explained how this relates to the problem of query evaluation over probabilistic databases. Here as well, the hierarchical queries are precisely the conjunctive queries for which the Shapley value of any input tuple can be computed in polynomial time. Two effective solutions for computing Shapley values of the input tuples using tools from probabilistic query evaluation were also presented.

The factorization of the provenance polynomial is a space-efficient alternative to its representation in disjunctive normal form and allows for efficient computation of expressive provenance-related queries used for query explanation.

Boris Glavic (Illinois Institute of Technology) introduced the PUG system (Provenance Unification through Graphs) for why and why-not provenance. PUG takes a Datalog query and a provenance question as input and generates a Datalog program that computes an explanation, which is the part of the provenance that is relevant to answer the question. This system demonstrates how a desirable factorization of provenance can be achieved by rewriting the input query. This is akin to factorization of provenance polynomials in the semi-ring model. Recent results on how provenance factorization by circuits impacts the performance of queries over bag semantics probabilistic databases were also presented.

## 3.4 Graph Databases

Factorization is a natural fit for the representation and processing of graph data, since graph traversals call for many-to-many joins with large outputs that are representable succinctly in factorized form.

Amine Mhedhbi (University of Waterloo) revisited column-oriented storage and query processing techniques in the context of graph database management systems. It is argued that a core decision in the design of such a graph database management system is to adopt a factorized tuple-set representation scheme to avoid repetitions of values. A block-based processor that factorizes the intermediate answers of joins over graph data was also presented.

Ainur Smagulova (TigerGraph) presented an approach for parallel execution of SQL queries on top of a graph processing engine. The nodes in the distributed system receive factorized representations of join results and compute aggregates directly over such factorizations.

## 3.5 Factorized In-Database Analytics

A wide range of analytics can be computed directly on factorized databases so without the materialization of the database joins. The workshop showcased

examples of such analytics, from SQL aggregates to linear programs and linear algebra expressions.

Zachary Huang (University of Columbia) introduced the problem of iterative analytics over joins. The proposed solution avoids the materialization of the join answer and shares the computation of a bulk of aggregates across the iterations and also within each iteration. This is supported by a succinct representation of the join called Calibrated Junction Hypertree. This problem is relevant in a variety of applications such as OLAP, query explanation, streaming data, and data augmentation for machine learning.

Immanuel Trummer (Cornell University) considered the problem of adaptive query evaluation and introduced a new query engine that combines worst-case optimal factorized evaluation with intra-query learning. It breaks down the execution time into time slices in which different execution plans can be used. By judiciously balancing exploration of new execution plans versus exploitation of promising, previously used execution plans, it places upper bounds on the expected gap between optimal and actual execution cost.

Florent Capelli (University of Lille) discussed the problem of optimizing linear programs whose variables are tuples in the answers to conjunctive queries over relational databases. The insight is that for any such linear program there are equivalent linear programs whose variables range over values from a factorized representation of the query answer. The reduction in the number of variables in the equivalent linear program is on par with the reduction from the size of a table representation to that of a factorized representation of the query answer.

Amir Shaikhha (University of Edinburgh) presented a declarative language that can express relational algebra with aggregations, linear algebra, and functional collections over data such as relations and matrices using nested semi-ring dictionaries. Such dictionaries can express multisets, arrays, matrices and restricted factorized representations. Thanks to the algebraic structure of semi-ring dictionaries, this language unifies a wide range of optimizations that are commonly confined to either database or linear algebra systems.

Nils Vortmeier (Ruhr-University Bochum) introduced the FiGaRo algorithm for the factorized computation of QR decomposition over matrices defined by relational database joins. FiGaRo avoids the materialization of the join and pushes the QR decomposition past the join. For acyclic joins, it takes time linear in the database size and independent of the join size and incurs far less rounding errors than the classical QR algorithms.

David Justo (Microsoft) considered the problem of compiling factorized linear algebra expressions to different programming language targets. The proposed solution leverages and extends the Oracle's GraalVM industrial polyglot compiler and runtime.

### 3.6 Factorized Machine Learning

The efficient computation of aggregates over factorized databases enables more complex analytics such as training machine learning models.

Arun Kumar (University of California San Diego) reflected on lessons learned on the benefits and limits of factorized machine learning, as well as the roadblocks and open challenges in translating this paradigm to practice. The talk overviewed works that applied the factorized machine learning paradigm to generalized linear models, clustering methods, a unified formal framework based on linear algebra, and models with feature interactions.

Mahmoud Abo Khamis (RelationalAI) reported on the rAD native in-database automatic differentiation framework built at RelationalAI to support machine learning, data analytics, and mathematical optimization. The input to rAD is a program in Rel, a declarative generalization of Datalog with aggregation and function symbols. This input program computes a multivalued function. The output to rAD is another Rel program that computes the derivatives of the input program with respect to some given input relations. Performing automatic differentiation in a high-level database language like Rel enables the evaluation of the derivatives while enjoying many features offered by the underlying database engine like factorization, query optimization and compilation, as well as support for higher order derivatives.

Milos Nikolic (University of Edinburgh) presented the problem of data imputation in normalized relational databases. An interesting insight is that the state-of-the-art Multiple Imputation by Chained Equations method, which requires the training of multiple regression models, can be reformulated such that the computationally most intensive parts of the training process are shared across the models and executed entirely on the normalized data within the database system.

## 4 Conclusion

Factorized databases are a rich field of study that offers technical challenges and solutions for both theoretical and systems research. They proved useful at the interface of databases and other fields such

as optimization, linear algebra, and machine learning, as they help effectively reduce the asymptotic complexity and improve the runtime performance for problems computed over database joins.

This workshop is the first effort to put together researchers who work on the theory, systems and applications of factorized databases. The participants considered it a success. The social activities in the beautiful summer days in Zurich also helped bring the on-site participants together.

An exciting topic that was not represented at the workshop is the use of factorization for incremental maintenance of query results and machine learning models trained over database queries under updates to the input database.

There are open problems in the six main topics represented at the workshop.

Does relational data admit more succinct representations than factorizations yet still allow for desirable properties such as constant-delay tuple enumeration and linear-time sum-product aggregates? Which further popular machine learning models, matrix decomposition techniques and iterative linear algebra methods over relational data can benefit from factorization? Beyond linear programs, can (non-)convex optimization programs over databases be rewritten to much smaller programs following a factorized data structure?

On the systems side, recent and promising efforts combine techniques such as vectorization, compression, and block-based access with factorization. Prototypes that exploit factorization for efficient maintenance of conjunctive queries under updates and for training machine learning already exist in industry (the RelationalAI engine) and also in the public domain (the LMFAO and F-IVM engines).

## 5 Acknowledgements

This workshop celebrated the end of Dan Olteanu's ERC consolidator grant "Foundations of Factorized Data Management Systems" and the start of his chair for Data Systems and Theory at the University of Zurich. The author would like to acknowledge the help of: Denise Gloor on logistics; Haozhe Zhang on the workshop web page; and Ahmet Kara and Qing Chen on technical support. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 682588.