

Just-in-time Data Integration

Martin Hentschel
ETH Zurich, Switzerland

Laura Haas
IBM Almaden Research Center, USA

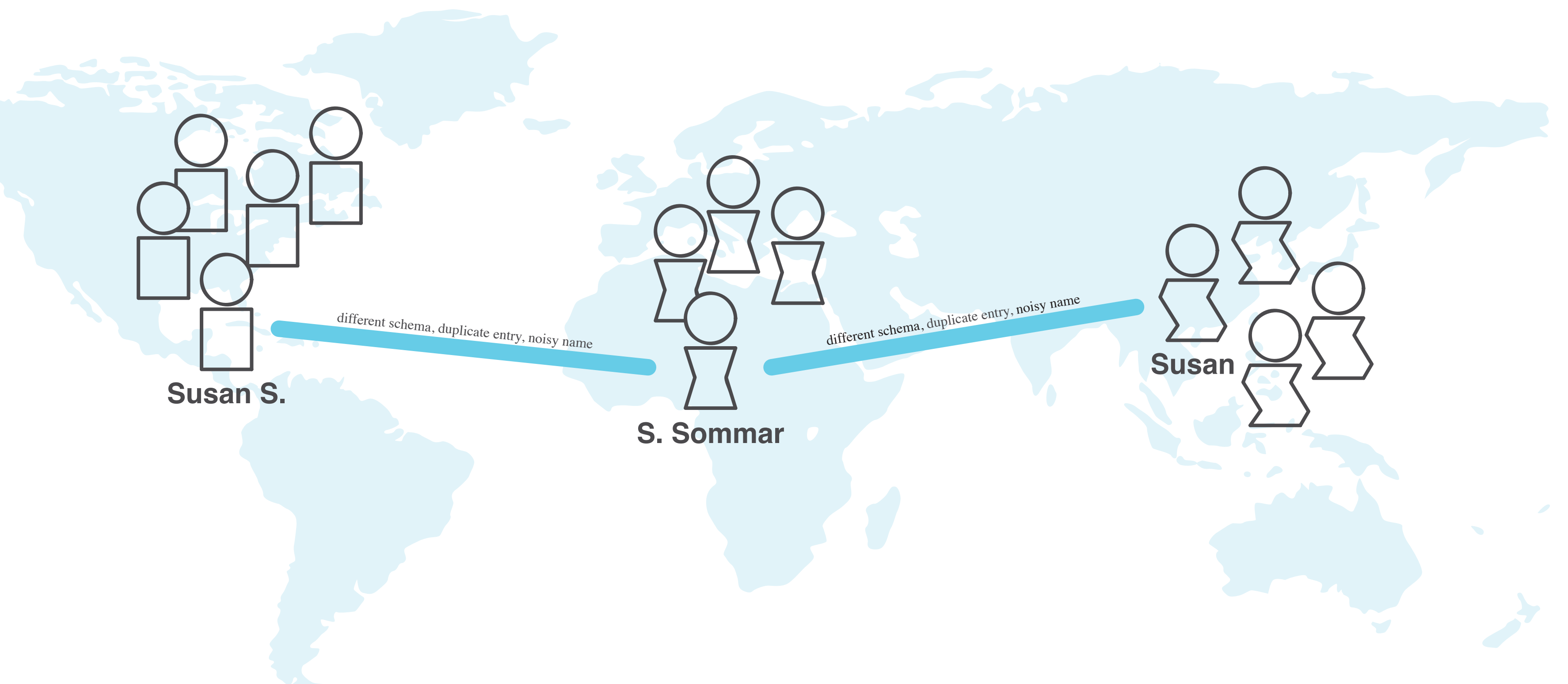
Renée J. Miller
University of Toronto, Canada

State of the world

In today's global enterprises **data is distributed** among subsidiaries. Because of local differences **schemas are diverse** and data tends to be **noisy** and may for example contain duplicates.

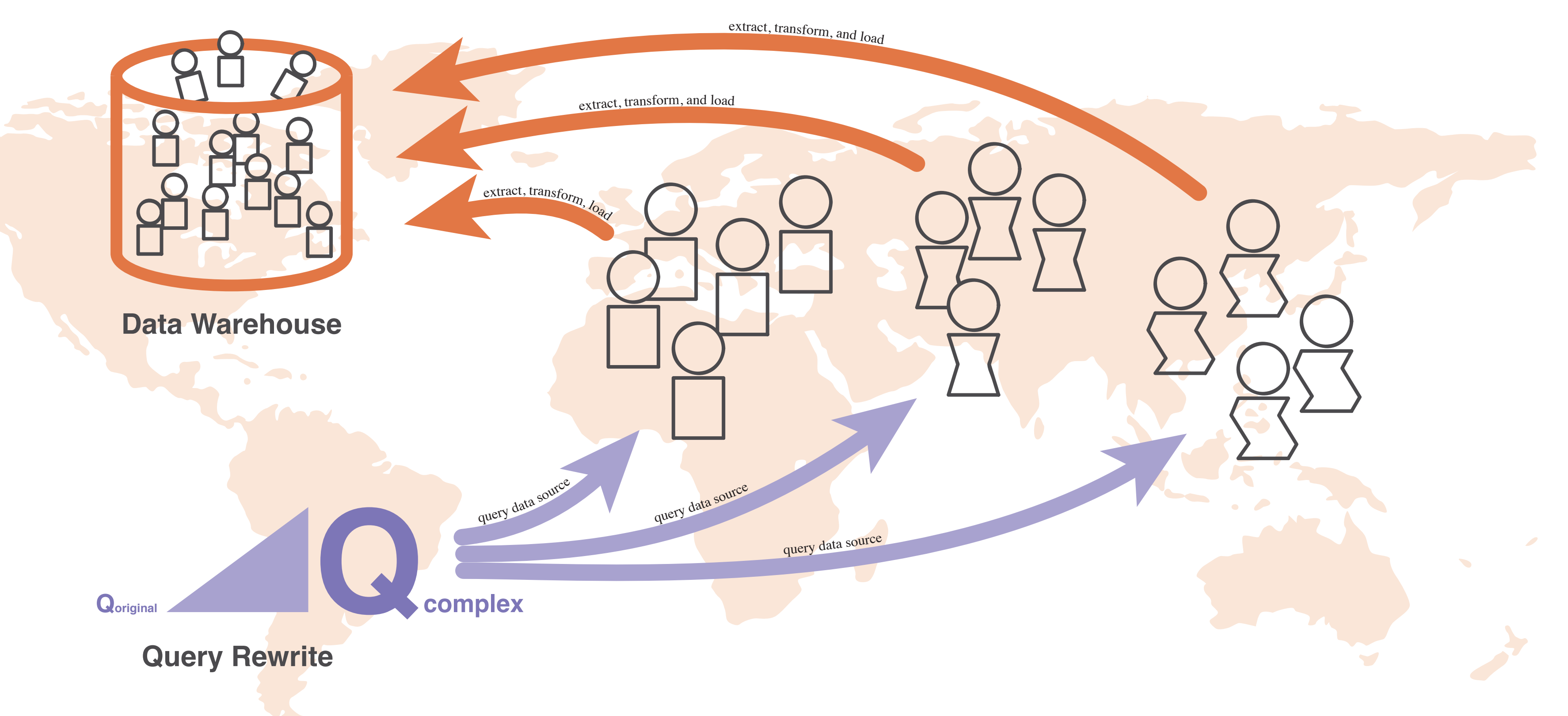
There is a growing demand for many small application that are quickly built and serve a single purpose, instead of one big application that fits all. This allows companies to react to changing markets.

To scale to massive amounts of data, storage systems are being transformed from pull-on-demand models to push-on-change models. The data flows through the system.



Data warehousing

Today, data warehousing is the most common approach to integrate data. It extracts the data from the data sources, transforms them, and loads the transformed data into a separate database. In the data warehouse, the integrated data can then be queried efficiently. The **set-up** of a data warehouse is **time-consuming** and **expensive**. Companies wanting to build a small application fear the high start-up cost.



Federation

The other common approach, federation, rewrites the original query to match the data sources. The data is queried directly from the data sources. Federation is not suited for environments where data characteristics are unpredictable, for example data flows of schema-less data or data with mixed schemas.

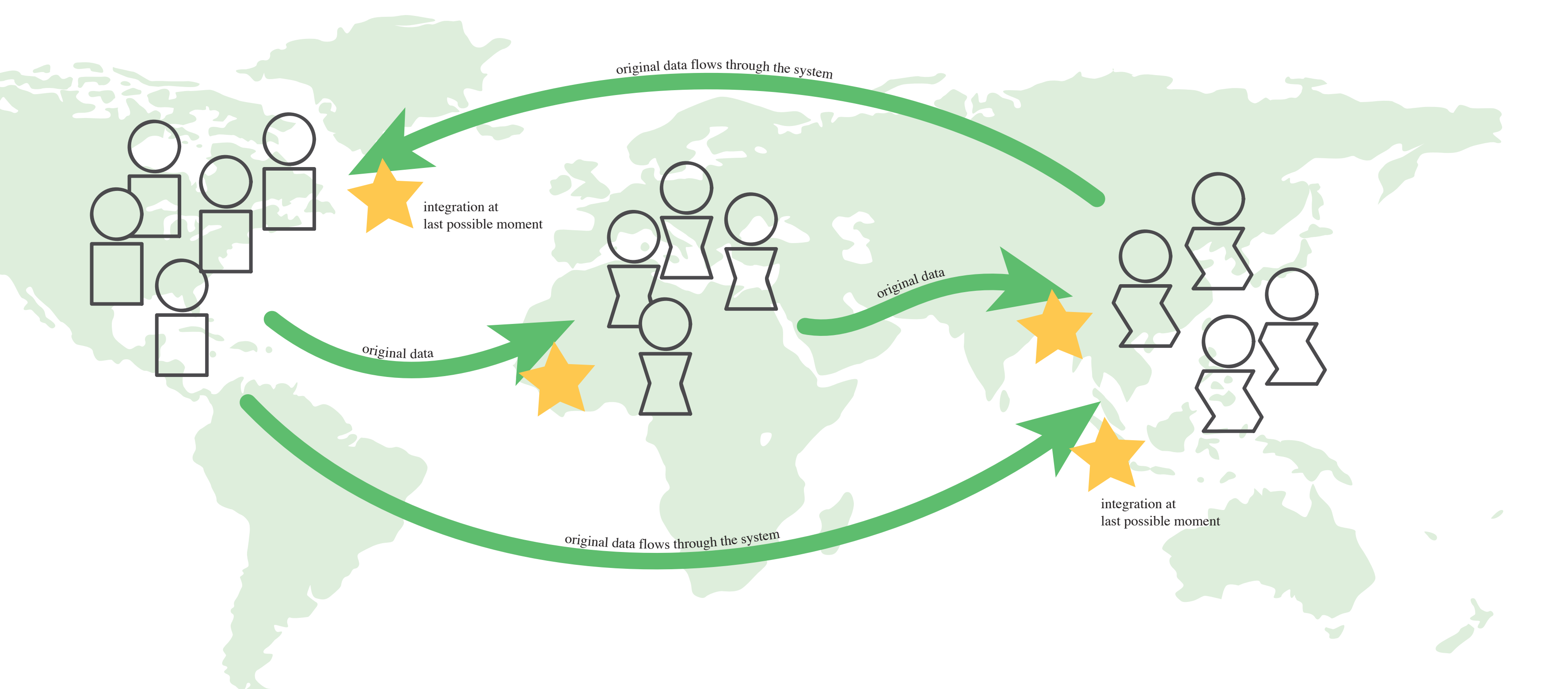
In such scenarios federation generates **inefficient complex queries**, potentially leading to exponential runtime behavior. Federation has lower start-up costs when building small applications.

Just-in-time data integration

We present a **novel** approach to integrate data that offers great **flexibility**. Data is integrated in an **incremental** and **iterative** fashion. It has low start-up cost and is great for building small applications.

Our approach is novel because data integration happens at the last possible moment, during **runtime** of a query. During runtime the data is annotated with additional information. We are able to integrate data at the schema level and instance level (e.g. merge duplicates).

Our approach **scales well** with the number of source schemas and ultimately outperforms state-of-the-art approaches in terms of throughput. It is well suited for data flow architectures.



How does it work

The basic idea is that when a query is executed, each data item touched is annotated. The annotations provide new access paths through the data. To merge duplicate data instances the data is annotated with a new composite node that contains the original nodes. The new composite node is handled by the query processor in the same way as any original node.

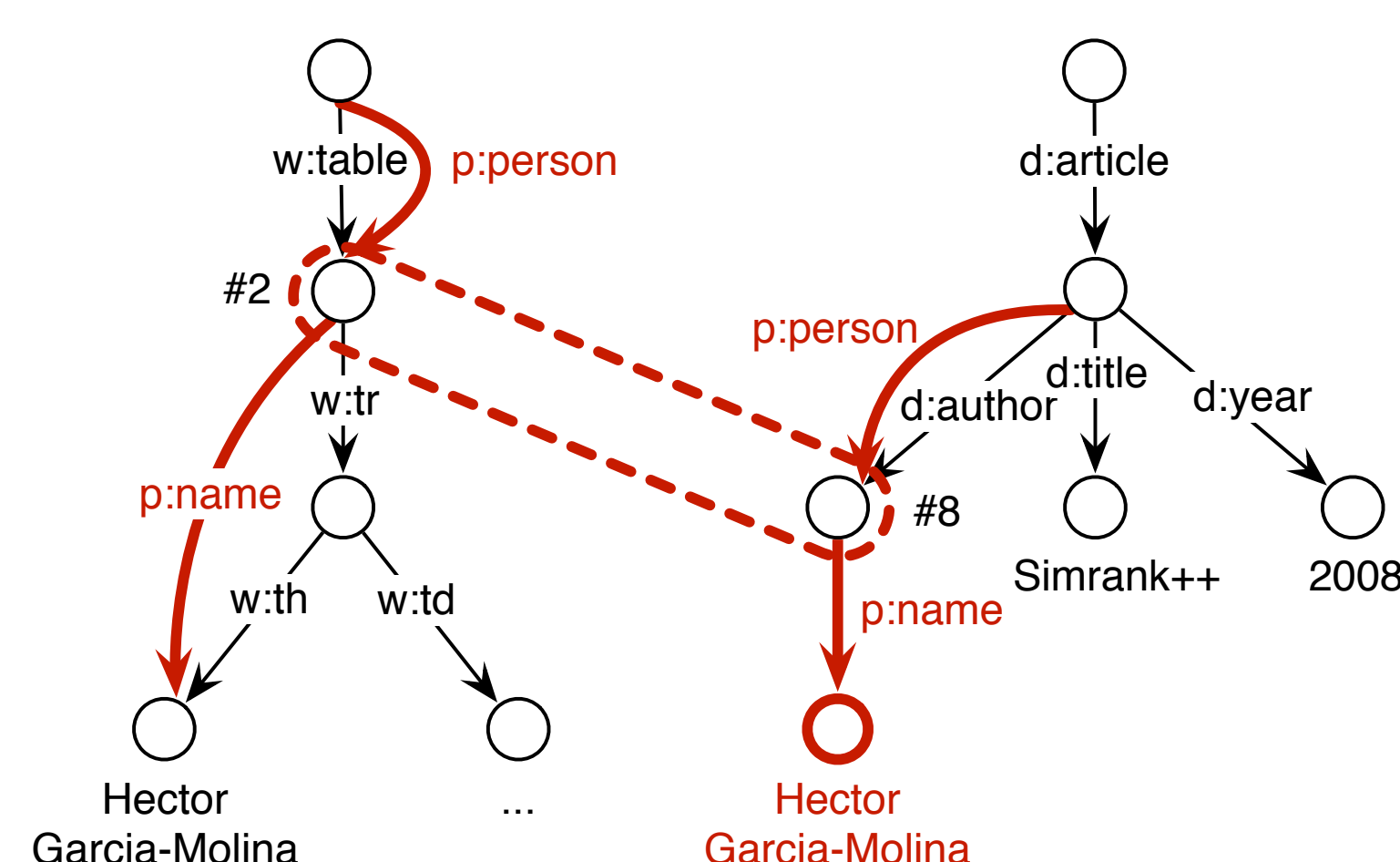


Figure: Data snippets from Wikipedia and DBLP annotated to match the schema of People People People, our demo application. All schemas and mapping rules, which trigger the annotations, can be found in our paper *Just-in-time Data Integration in Action*, VLDB, 2010.

How does it compare

The state-of-the-art approaches to data integration are federation and transformation. Federation engines rewrite a query to match the different data sources. The rewrite happens at compile-time without knowledge of the data. It has to provide for all possible input data, leading to query complexity. Transformation (data warehousing) transforms all data without knowledge of the query. All data is transformed independent of whether the integrated data is ever used. Just-in-time data integration takes effect at runtime of a query. It has knowledge of both data and query. It only integrates those parts of the data necessary to produce the query result.

Approach	Query	Document
Federation	$(p:person a:author w:table)[1] / (p:name text() w:tr/w:tr)$	
Transformation	$p:person[1]/p:name$	
Just-in-time	$p:person[1]/p:name$	

Figure: Different approaches to execute query $p:person[1]/p:name$ (get the name of the first person). The parts of the query and data that are changed by the respective approaches are shown in red.

How does it perform

Our implementation of just-in-time data integration shows best performance for small messages that flow through the system – such as streaming systems, pub-sub systems, and data flow architectures. Ultimately, it outperforms state-of-the-art approaches in terms of throughput by orders of magnitude.

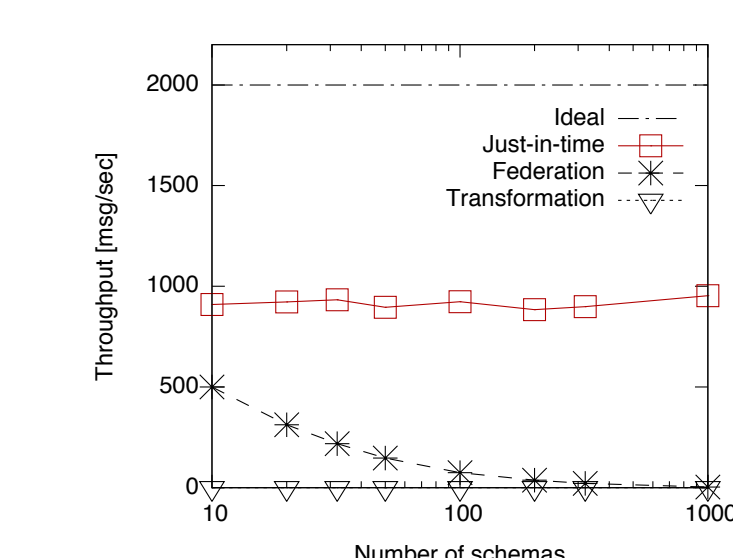


Figure: Throughput vs. number of source schemas. XMark Query 4.

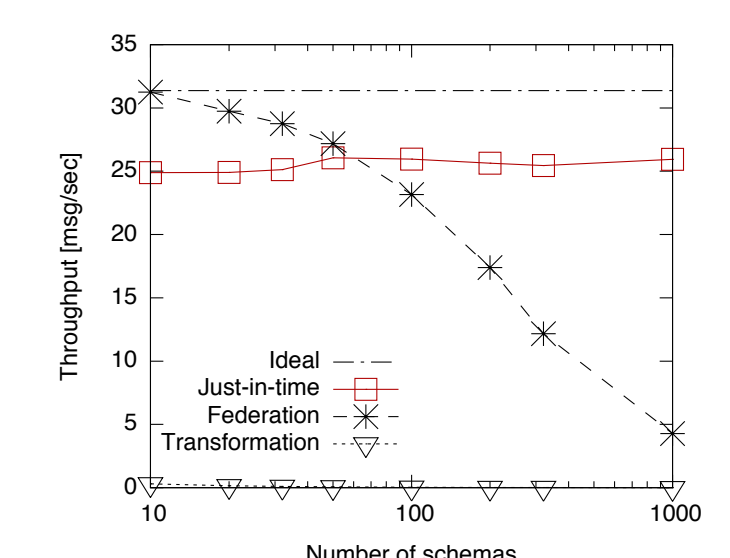


Figure: Throughput vs. number of source schemas. XMark Query 11.

A complete description of these benchmarks is available in our technical report *Scalable Data Integration by Mapping Data to Queries*, ETH Zurich, 2009.