Relational AI Systems: In Pursuit of Simplicity

Dan Olteanu

April 24, 2023

What is this Talk about?

My research agenda:

- Investigate the principles behind computational challenges for data processing
- Design simple and scalable solutions towards these challenges in both academia and industry

This talk: Two ideas in relational AI

What is this Talk about?

My research agenda:

- Investigate the principles behind computational challenges for data processing
- Design simple and scalable solutions towards these challenges in both academia and industry

This talk: Two ideas in relational AI

- But first: Why relational?

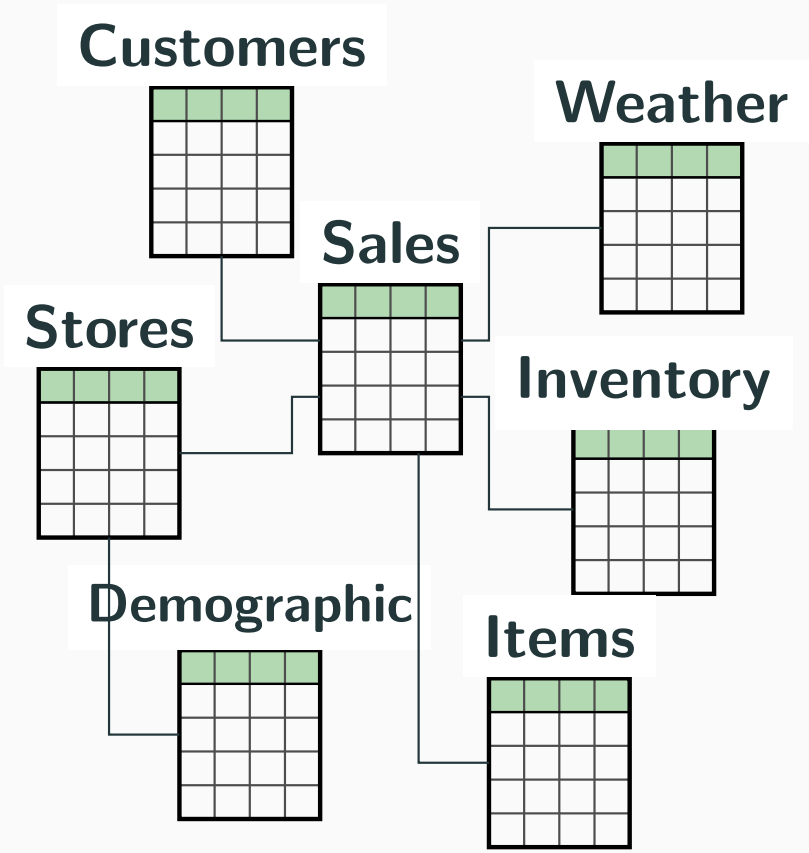
Relational Model: Jewel in the Data Management Crown

Simple model rooted in logic, invented by Codd at IBM in 1969

- *The* most widely deployed software paradigm of any type

  Similar reach: zip, libpng, libjpeg

- $>$ trillion SQLite active instances:
  - Android/iPhone/iOS devices
  - Mac/Windows10 machines
  - Firefox/Chrome/Safari browsers
  - Skype, iTunes, PhP, Python
  - smart TV sets
  - automotive multimedia systems



Why is the Relational Paradigm Ubiquitous?

It is not for lack of trying something else..

- **Transactional** databases were initially navigational
- Relational took over:

  Oracle
  (née Relational Software)
  Current Market Cap: $250.4B

  Ingres
  (née Relational Technology)

  Informix
  (née Relational Databases)

Why is the Relational Paradigm Ubiquitous?

It is not for lack of trying something else..

- **Analytic** databases were initially multidimensional arrays (tensors)
- Relational took over:

  Tableau Software

  Market Cap at sale: $11.6B

Why is the Relational Paradigm Ubiquitous?

It is not for lack of trying something else..

- **Big Data** systems were initially MapReduce/Spark
- Relational took over:

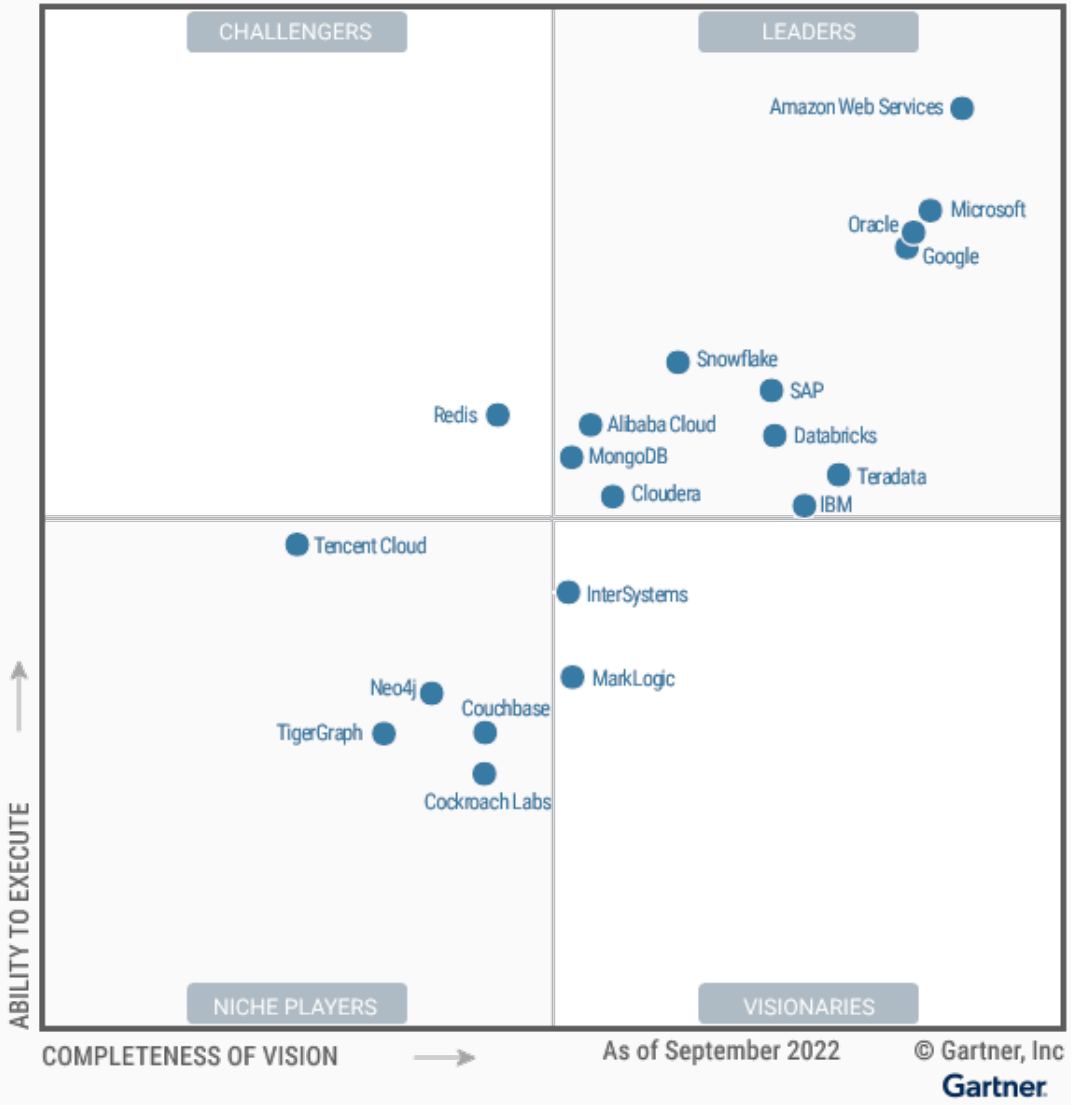
  Snowflake Software

  Current Market Cap: $97.5B

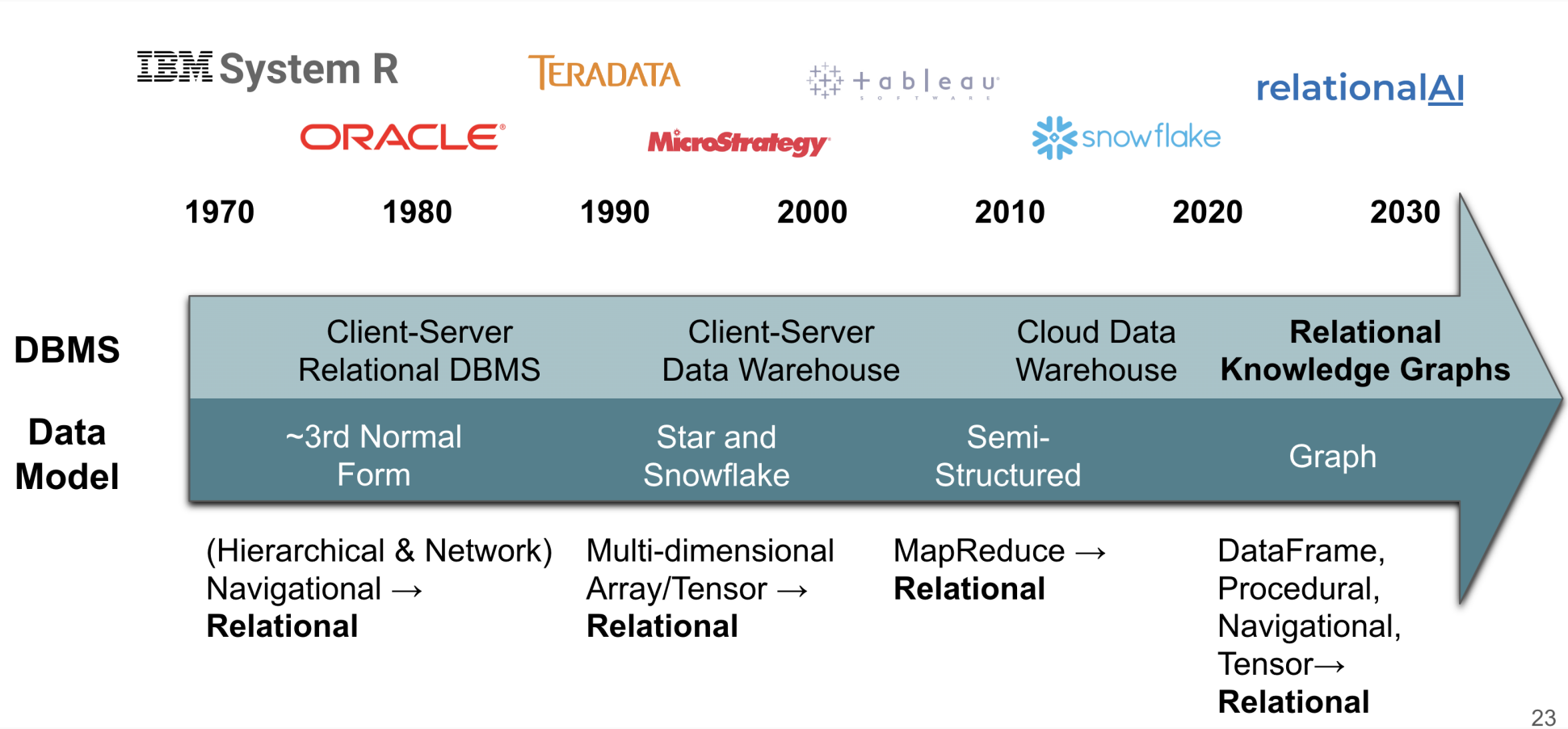
  Spark turned relational

  Google BigQuery

  AWS Cloud Databases

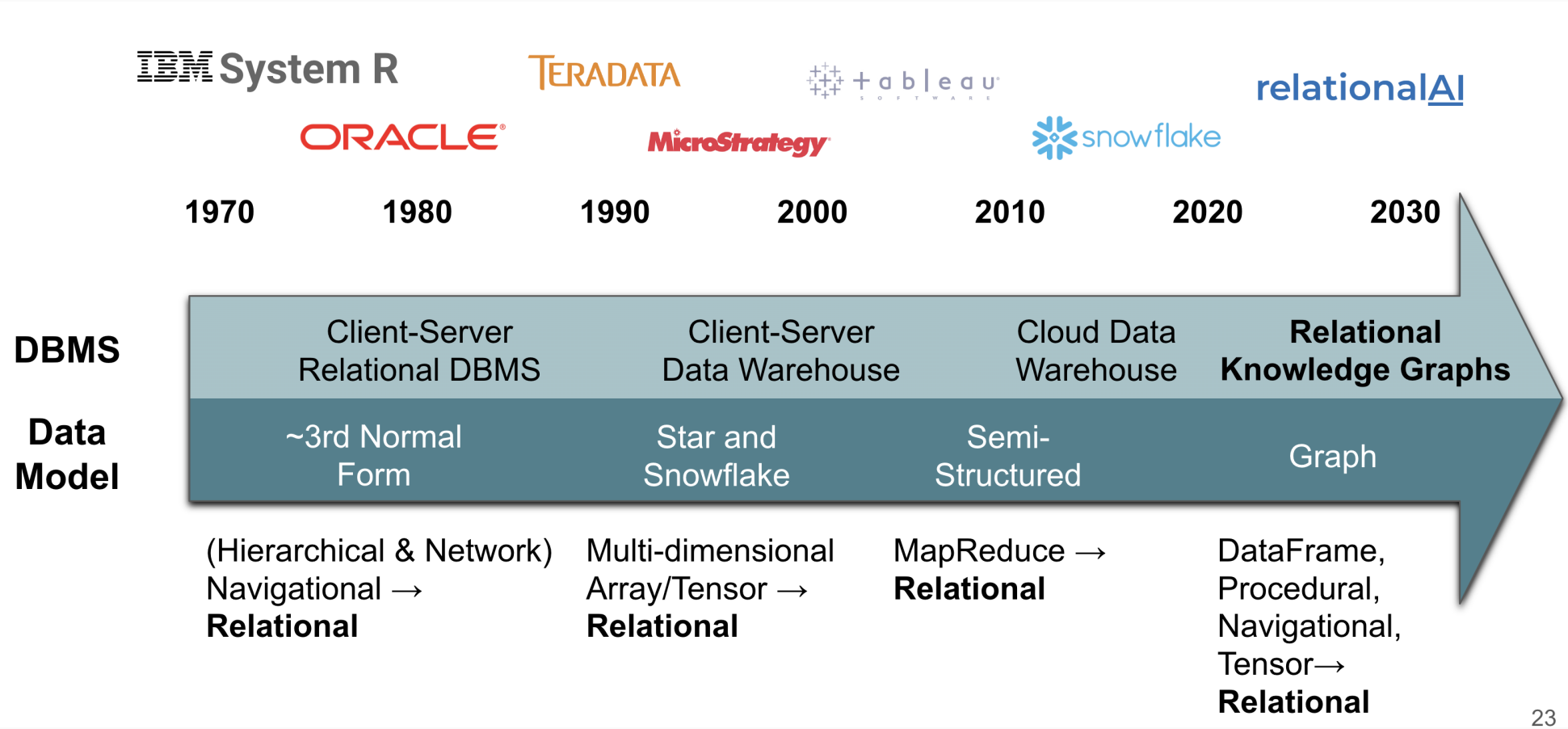


Relational Always Wins!



(graphic courtesy of Molham Aref, RelationalAI CEO)

Relational Always Wins!



"Making the simple complicated is commonplace; making the complicated simple, that's creativity." – C. Mingus

(graphic courtesy of Molham Aref, RelationalAI CEO)

But Really Why is the Relational Paradigm Ubiquitous?

- Very simple data model rooted in logic

  First principles then implementation

But Really Why is the Relational Paradigm Ubiquitous?

- Very simple data model rooted in logic

  First principles then implementation

- Separation of What from How

  Relational systems are declarative

But Really Why is the Relational Paradigm Ubiquitous?

- Very simple data model rooted in logic

  First principles then implementation

- Separation of What from How

  Relational systems are declarative

- Automatic Programming

  query optimization, memory mgt, parallelization, incrementalization

But Really Why is the Relational Paradigm Ubiquitous?

- Very simple data model rooted in logic

  First principles then implementation

- Separation of What from How

  Relational systems are declarative

- Automatic Programming

  query optimization, memory mgt, parallelization, incrementalization

- Easy to understand by domain experts

  Domain experts are cheaper and more plentiful than programmers

But Really Why is the Relational Paradigm Ubiquitous?

- Very simple data model rooted in logic

  First principles then implementation

- Separation of What from How

  Relational systems are declarative

- Automatic Programming

  query optimization, memory mgt, parallelization, incrementalization

- Easy to understand by domain experts

  Domain experts are cheaper and more plentiful than programmers

- Easy to implement in practice

  Tables have rows, all row have the same columns :)

- ...

Relational Data Processing Poses Technical Challenges

Achilles heel: Rigid data format that encourages redundancy

Redundancy in data begets redundancy in computation

- Redundancy hides the true computational complexity
- Key reason for lack of efficiency and scalability

Relational Data Processing Poses Technical Challenges

Achilles heel: Rigid data format that encourages redundancy

Redundancy in data begets redundancy in computation

- Redundancy hides the true computational complexity
- Key reason for lack of efficiency and scalability

This talk looks at redundancy when:

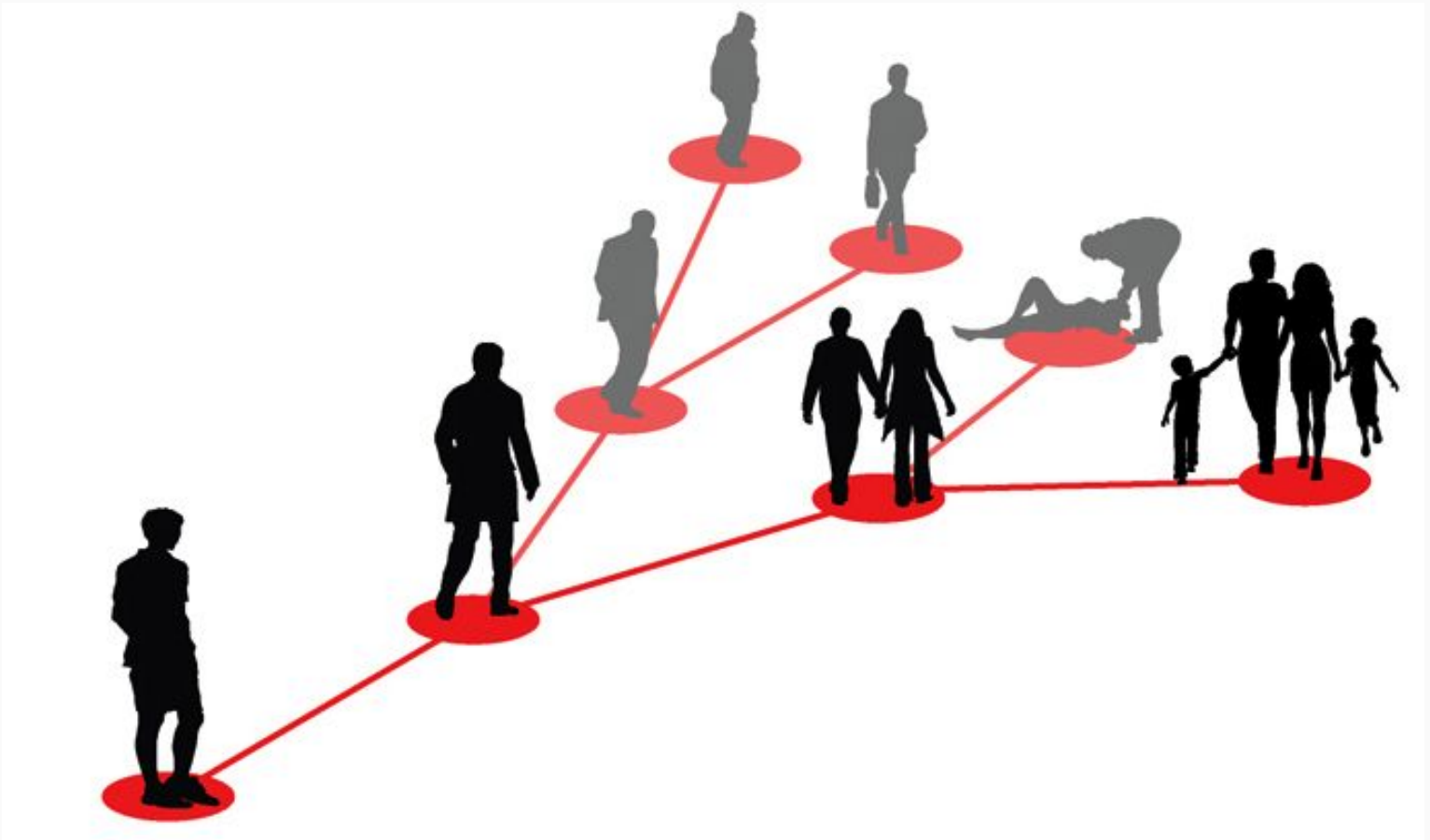
- Reasoning under uncertainty
- Training machine learning models over relational data

Probabilistic Databases

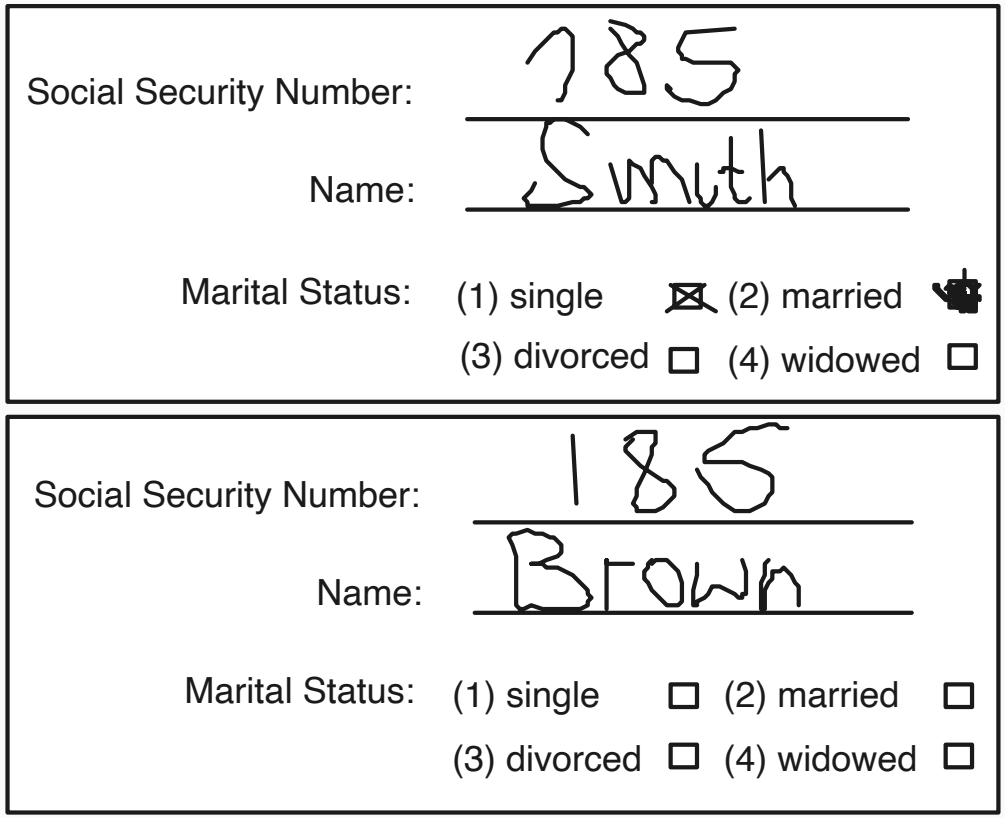
Many Worlds Interpretation

Data may admit many interpretations or *possible worlds*

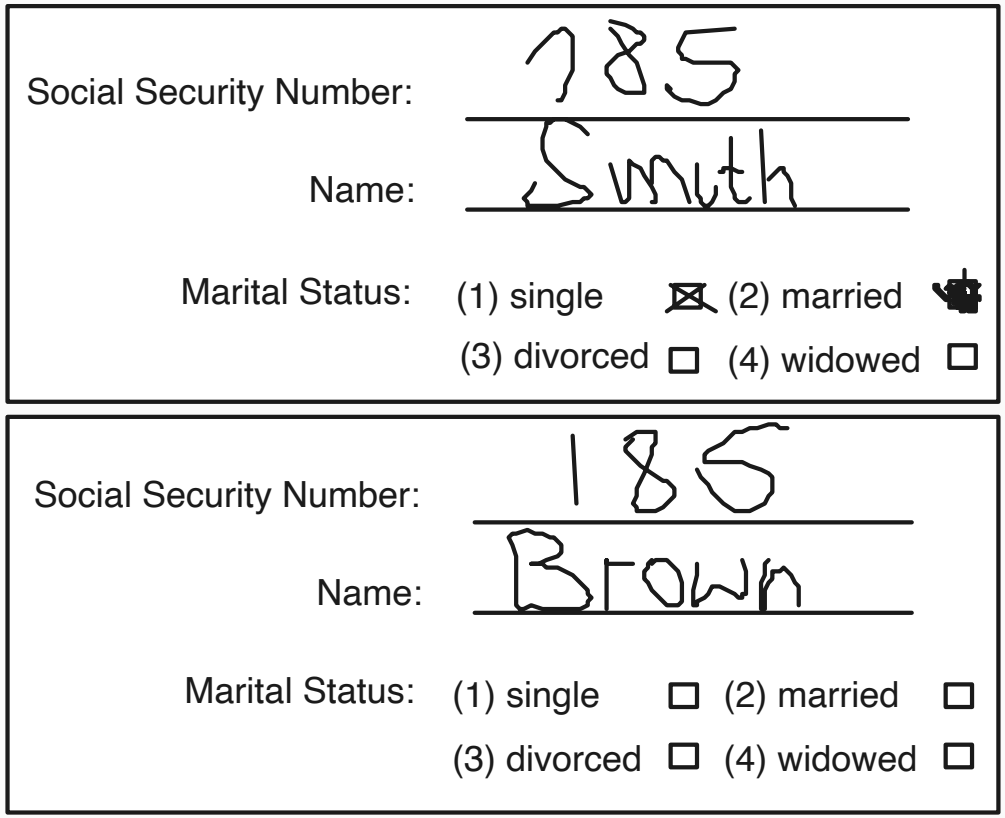
- Different runs of scientific and social experiments may have (slightly) different outcomes



Example: Manually Completed Census Forms



Example: Manually Completed Census Forms



Several interpretations of the above simple forms are possible

- What is the marital status of Smith or Brown?
- What are their social security numbers? 185? 186? 785?
- Some interpretations more likely (probable) than others

Interpretations of the Manually Completed Census Forms

SSN	Name	Status	Prob
185	Smith	Single	0.2
185	Brown	Single	0.2

Interpretations of the Manually Completed Census Forms

SSN	Name	Status	Prob
185	Smith	Single	0.2
185	Brown	Single	0.2

SSN	Name	Status	Prob
785	Smith	Single	0.3
185	Brown	Single	0.2

Interpretations of the Manually Completed Census Forms

SSN	Name	Status	Prob
185	Smith	Single	0.2
185	Brown	Single	0.2

SSN	Name	Status	Prob
785	Smith	Single	0.3
185	Brown	Single	0.2

SSN	Name	Status	Prob
185	Smith	Married	0.2
185	Brown	Single	0.2

SSN	Name	Status	Prob
785	Smith	Married	0.3
185	Brown	Single	0.2

Interpretations of the Manually Completed Census Forms

SSN	Name	Status	Prob
185	Smith	Single	0.2
185	Brown	Single	0.2

SSN	Name	Status	Prob
785	Smith	Single	0.3
185	Brown	Single	0.2

SSN	Name	Status	Prob
185	Smith	Married	0.2
185	Brown	Single	0.2

SSN	Name	Status	Prob
785	Smith	Married	0.3
185	Brown	Single	0.2

. . . . . .

for each interpretation for Smith, each possible interpretation for Brown

Total interpretations = 32: 4 (for Smith) $\times$ 8 (for Brown)

Computational Challenges in Probabilistic Databases

How to represent compactly the very many worlds?

Computational Challenges in Probabilistic Databases

How to represent compactly the very many worlds?

- Very many $\approx 10^{10^6}$ worlds (in our experiments)
- Each world needs ≈ 1 Gigabyte (678,000 book pages)
- Each world has a likelihood (probability) for being true

Computational Challenges in Probabilistic Databases

How to represent compactly the very many worlds?

- Very many $\approx 10^{10^6}$ worlds (in our experiments)
- Each world needs ≈ 1 Gigabyte (678,000 book pages)
- Each world has a likelihood (probability) for being true

Answer: Avoid redundancy in the representation

Computational Challenges in Probabilistic Databases

How to represent compactly the very many worlds?

- Very many $\approx 10^{10^6}$ worlds (in our experiments)
- Each world needs ≈ 1 Gigabyte (678,000 book pages)
- Each world has a likelihood (probability) for being true

Answer: Avoid redundancy in the representation

- 10^{10^6} worlds need ≈ 6 Gigabytes (in our experiments)

Computational Challenges in Probabilistic Databases

How to efficiently query all the worlds?

Computational Challenges in Probabilistic Databases

How to efficiently query all the worlds?

- Efficient $\neq$ Query one world at a time
- Ideal: Time to query all worlds $\approx$ time to query one world

Computational Challenges in Probabilistic Databases

How to efficiently query all the worlds?

- Efficient $\neq$ Query one world at a time
- Ideal: Time to query all worlds $\approx$ time to query one world

Distinguish fast queries from slow queries

- Syntactic characterization of queries by their computational complexity $\Rightarrow$ Dichotomy for query answering

Dichotomy for Query Answering

Dichotomy for Query Answering

Queries are

either easy and can be solved efficiently

or hard and cannot be solved efficiently

Dichotomies sound simple yet are very challenging to prove.

"Simple can be harder than complex." –Steve Jobs

Dichotomy for Query Answering

Queries are

either easy and can be solved efficiently

- Exact computation feasible

or hard and cannot be solved efficiently

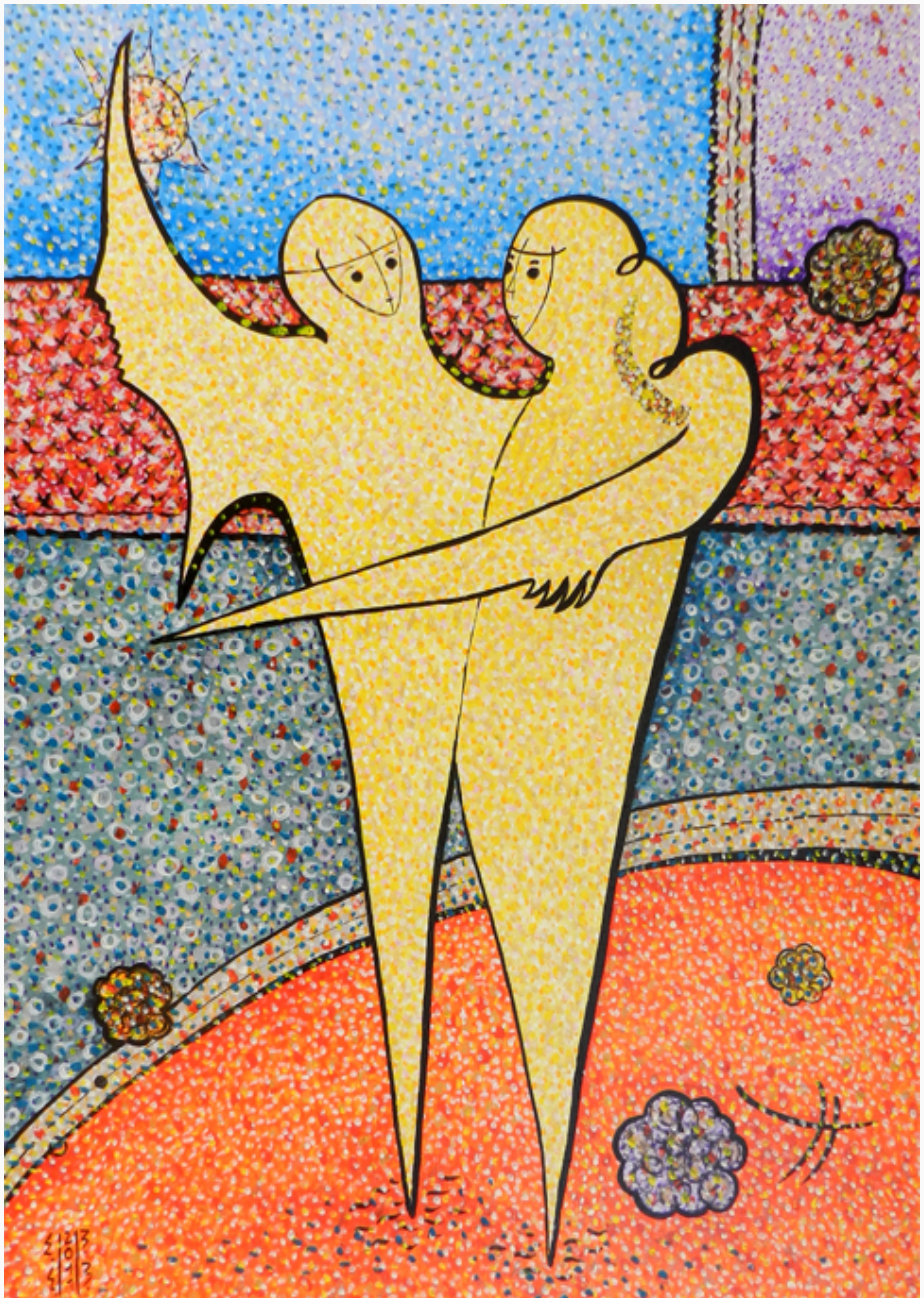
- Approximate computation feasible

Dichotomies sound simple yet are very challenging to prove.

"Simple can be harder than complex." –Steve Jobs

How Do Hard Queries Look Like?

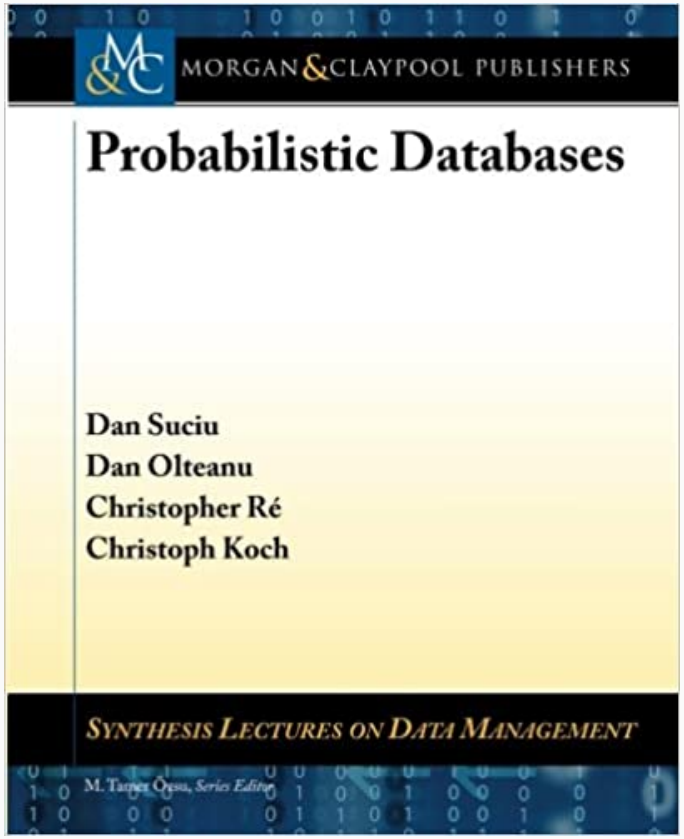
How Do Easy Queries Look Like?



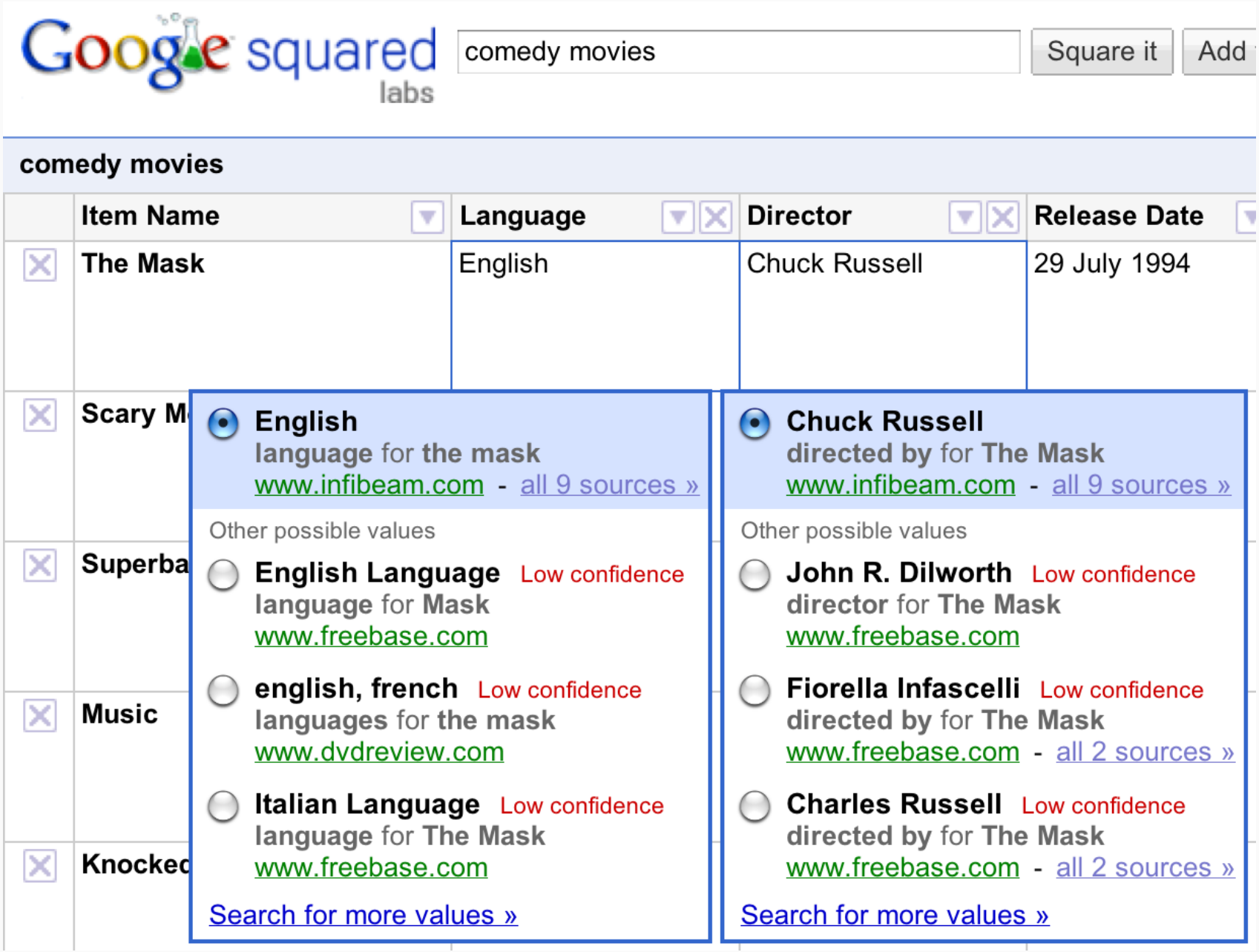
Systems and Theory for Probabilistic Databases

Systems and Theory for Probabilistic Databases

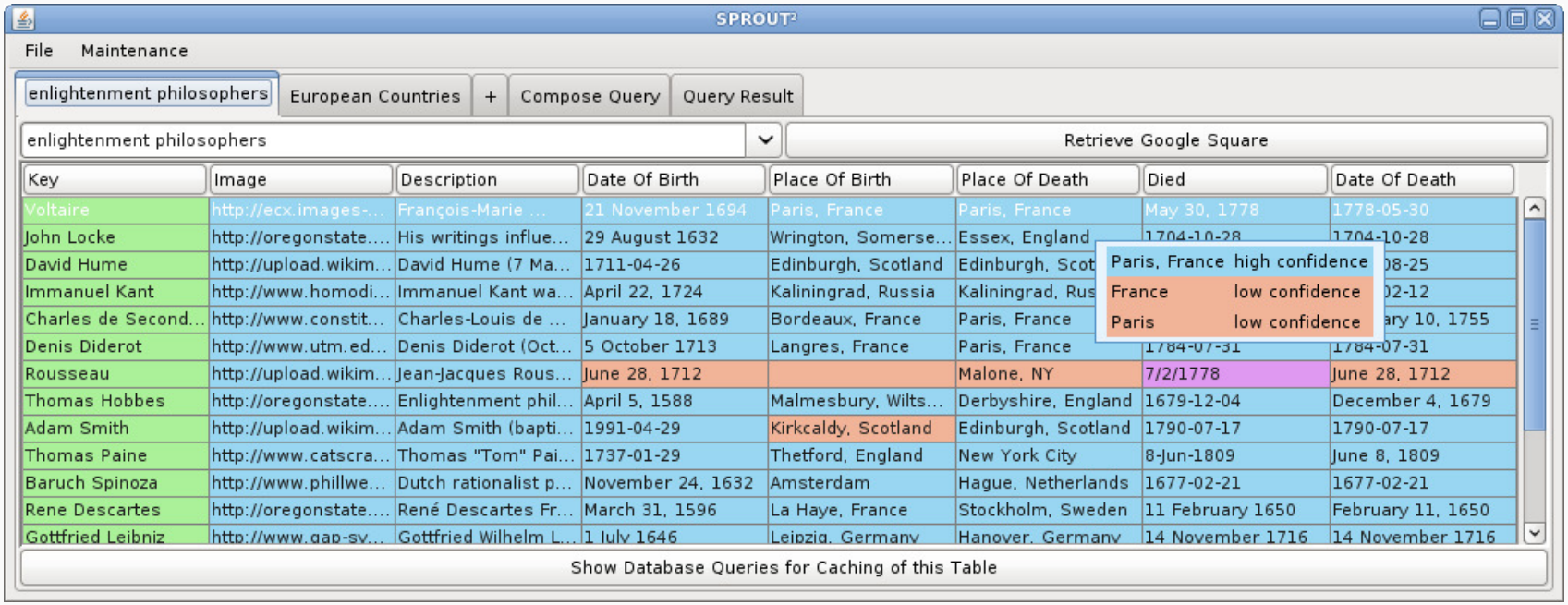
Systems and Theory for Probabilistic Databases

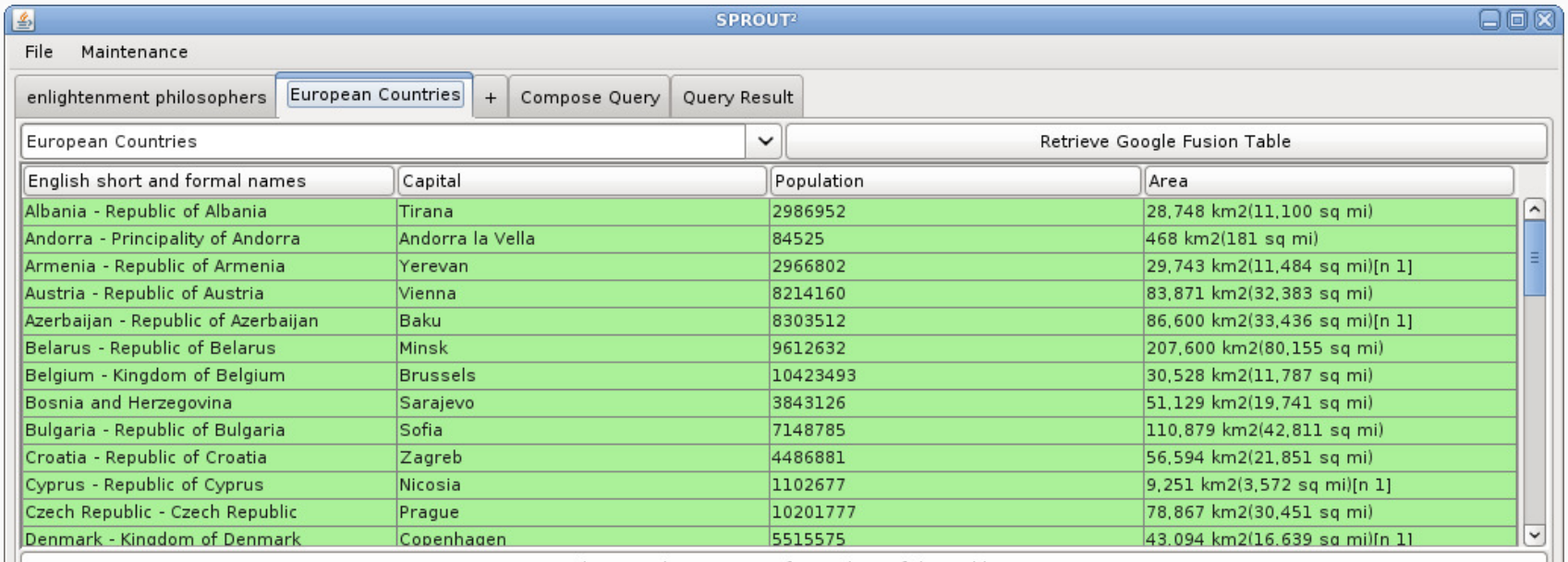


Application: Probabilistic Google Search

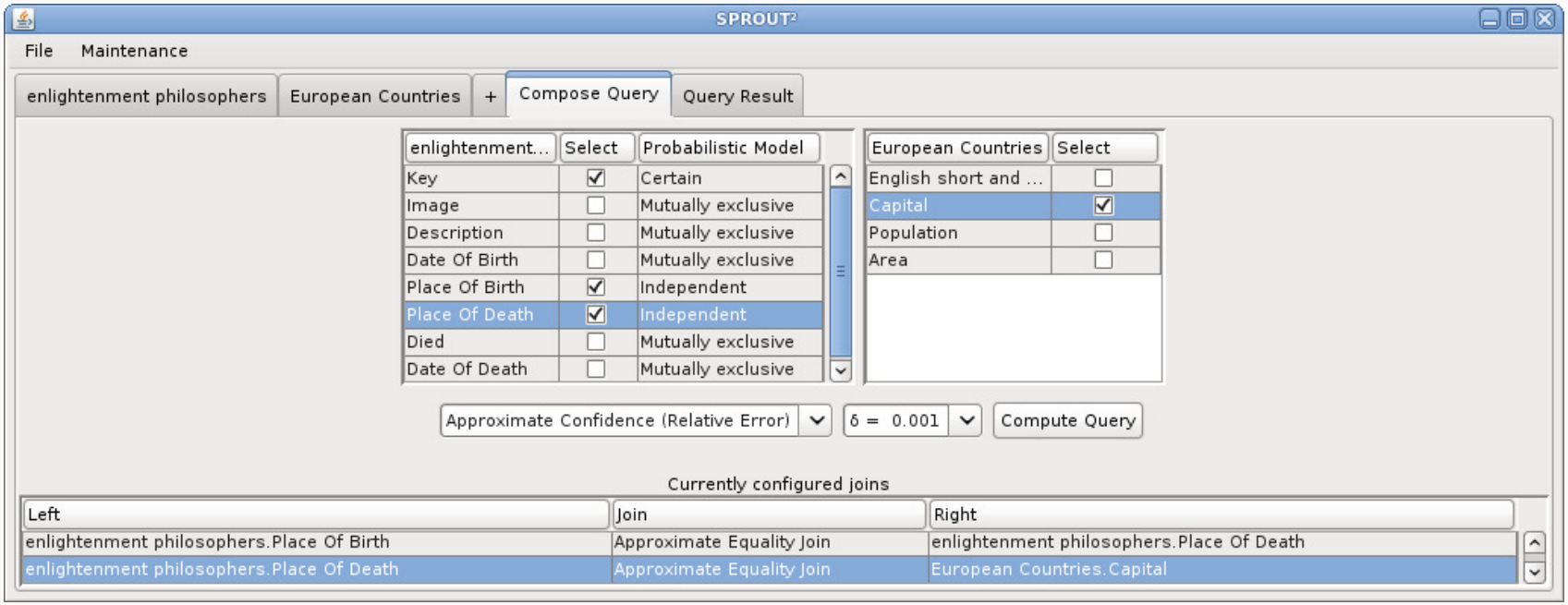


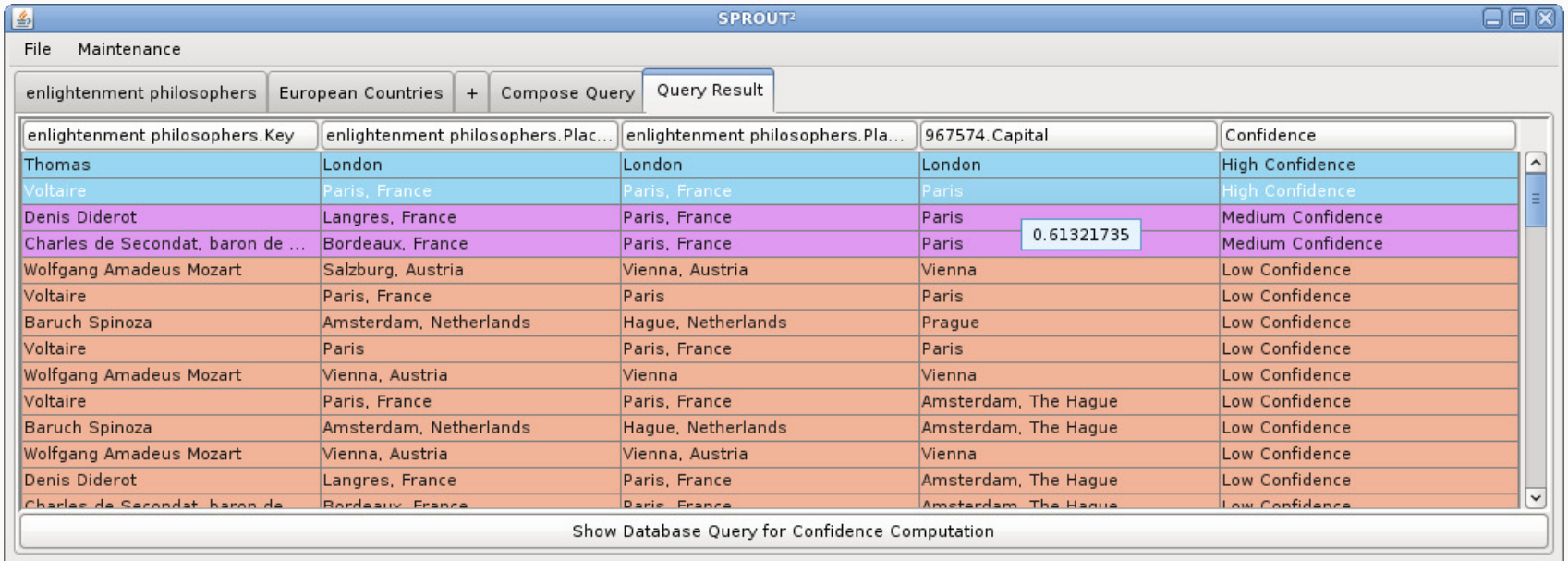
Probabilistic Google Search with SPROUT^2





Probabilistic Google Search with SPROUT^2





Factorized Databases

Factorization in Arithmetic [Primary School Maths]

$$(2 * 100) + (3 * 100)$$

Factorization in Arithmetic [Primary School Maths]

$$(2 * 100) + (3 * 100) = (2+ 3) * 100$$

Factorization in Boolean Algebra [Secondary School Maths]

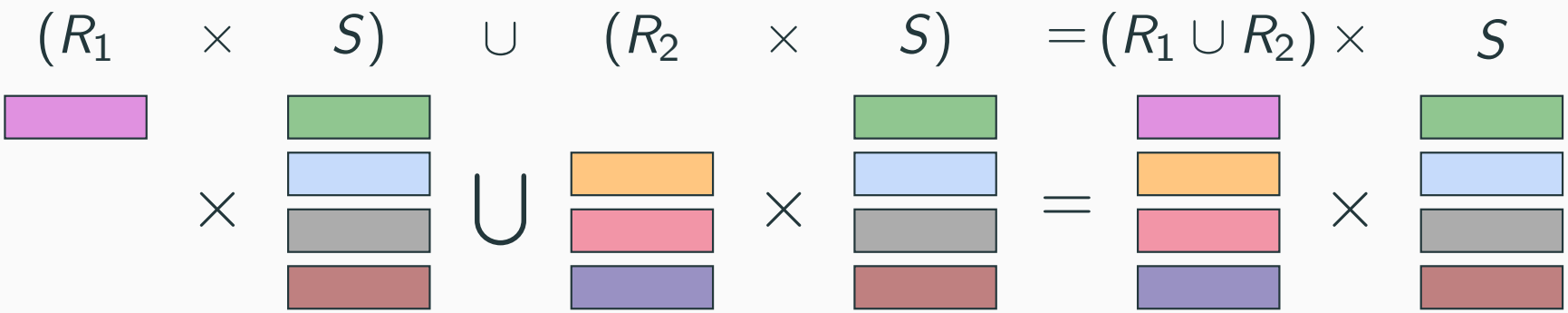
$$(x \text{ and } y) \text{ or } (z \text{ and } y) = (x \text{ or } z) \text{ and } y$$

where x, y, z are Boolean variables

$$(R_1 \times S) \cup (R_2 \times S) = (R_1 \cup R_2) \times S$$

where $\times$ is Cartesian product and $\bigcup$ is union; R_1, R_2, S are relations

Factorization in Relational Algebra [BSc DB Course]



where $\times$ is Cartesian product and $\bigcup$ is union; R_1, R_2, S are relations

The Distributivity Law of Rings [MSc/PhD course]

All previous identities are instances of the same distributivity law of an algebraic structure called the ring with sum-product operations:

Identity	Sum	Product	Domain
$(a * b) + (c * b) = (a + c) * b$	$+$	$*$	Reals
$(x \text{ and } y) \text{ or } (z \text{ and } y) = (x \text{ or } z) \text{ and } y$	$\vee$	$\wedge$	Booleans
$(R_1 \times S) \cup (R_2 \times S) = (R_1 \cup R_2) \times S$	$\cup$	$\times$	Relations

Why Factorize?

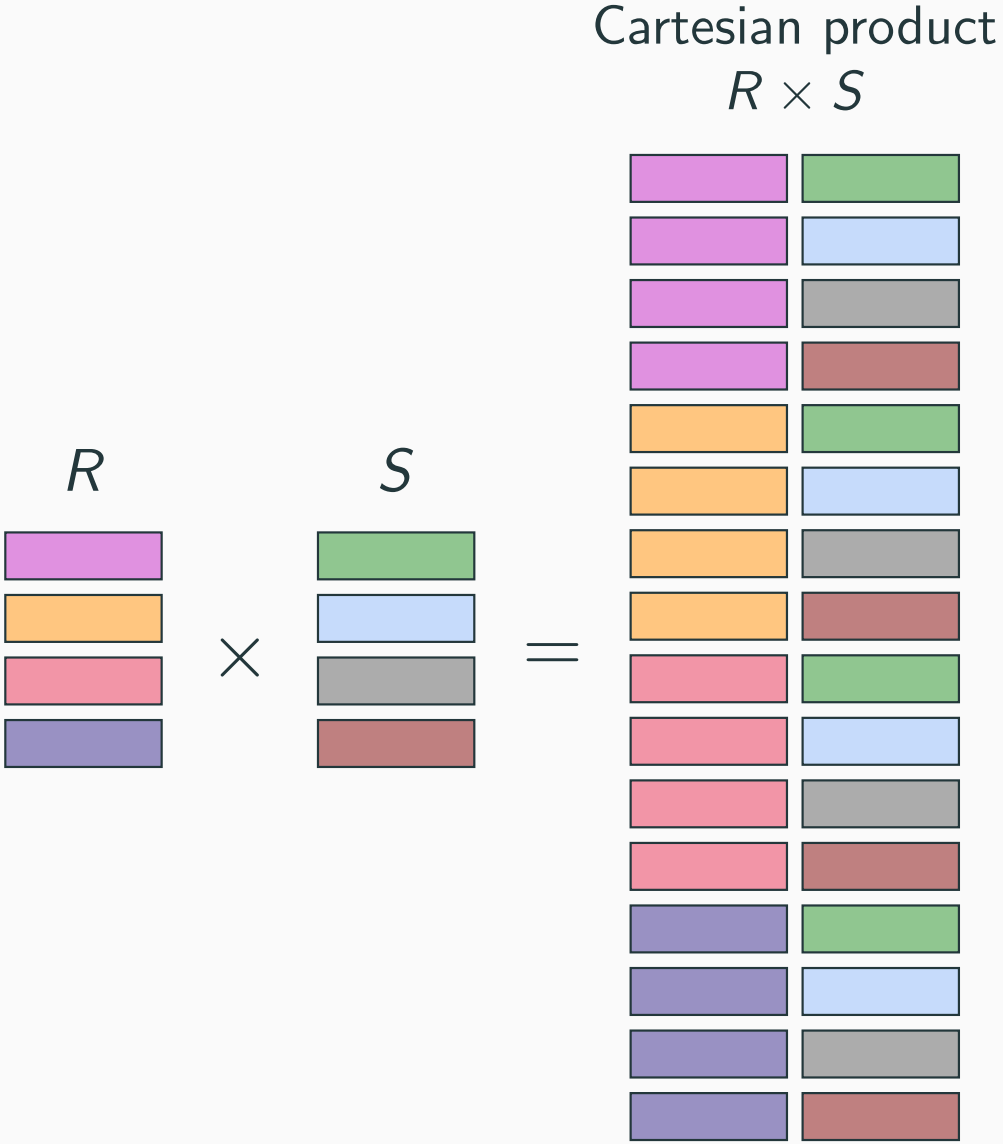
Factorization reduces redundant computation

Why Factorize?

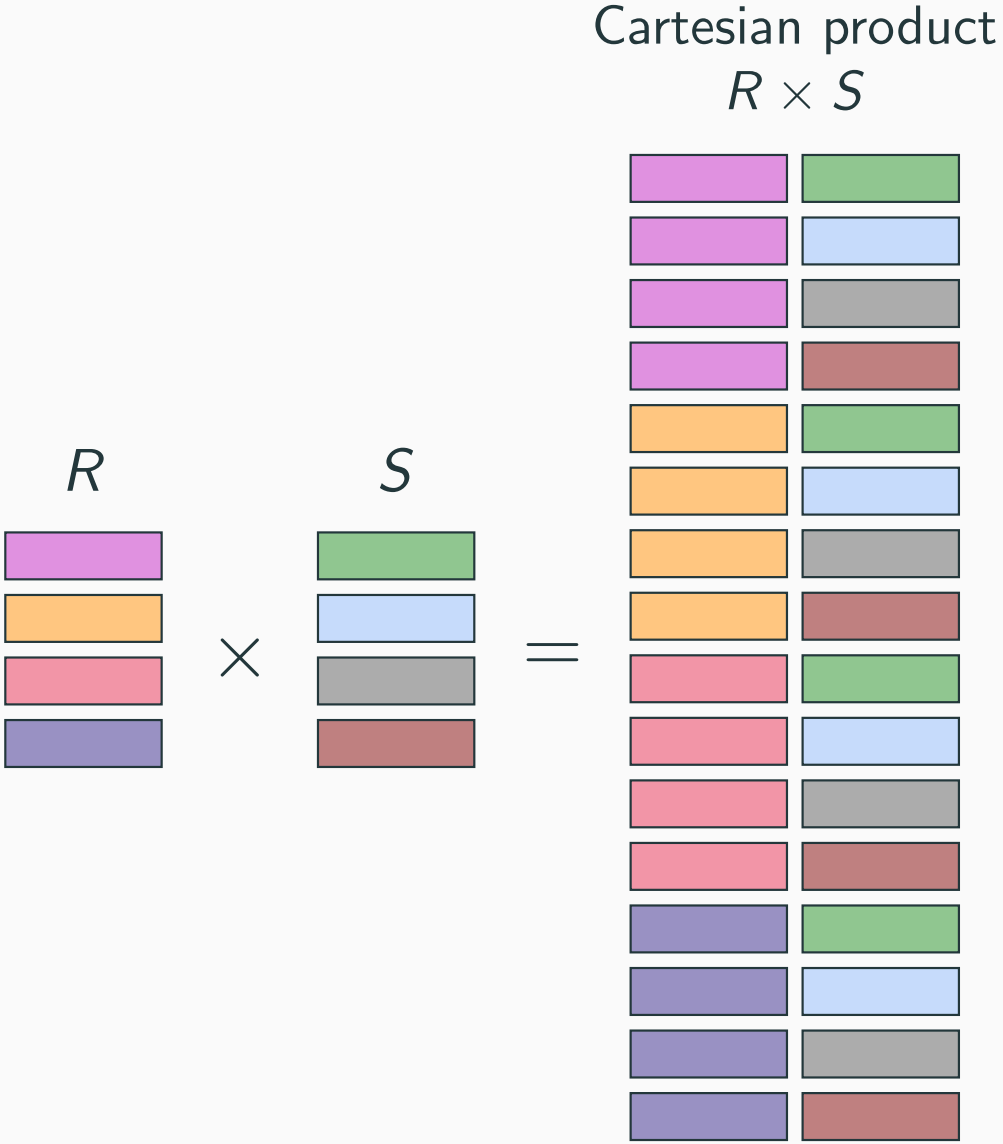
Factorization reduces redundant computation

"The ability to simplify means to eliminate the unnecessary so that the necessary may speak." – Hans Hofmann

Key Advantage of Factorization



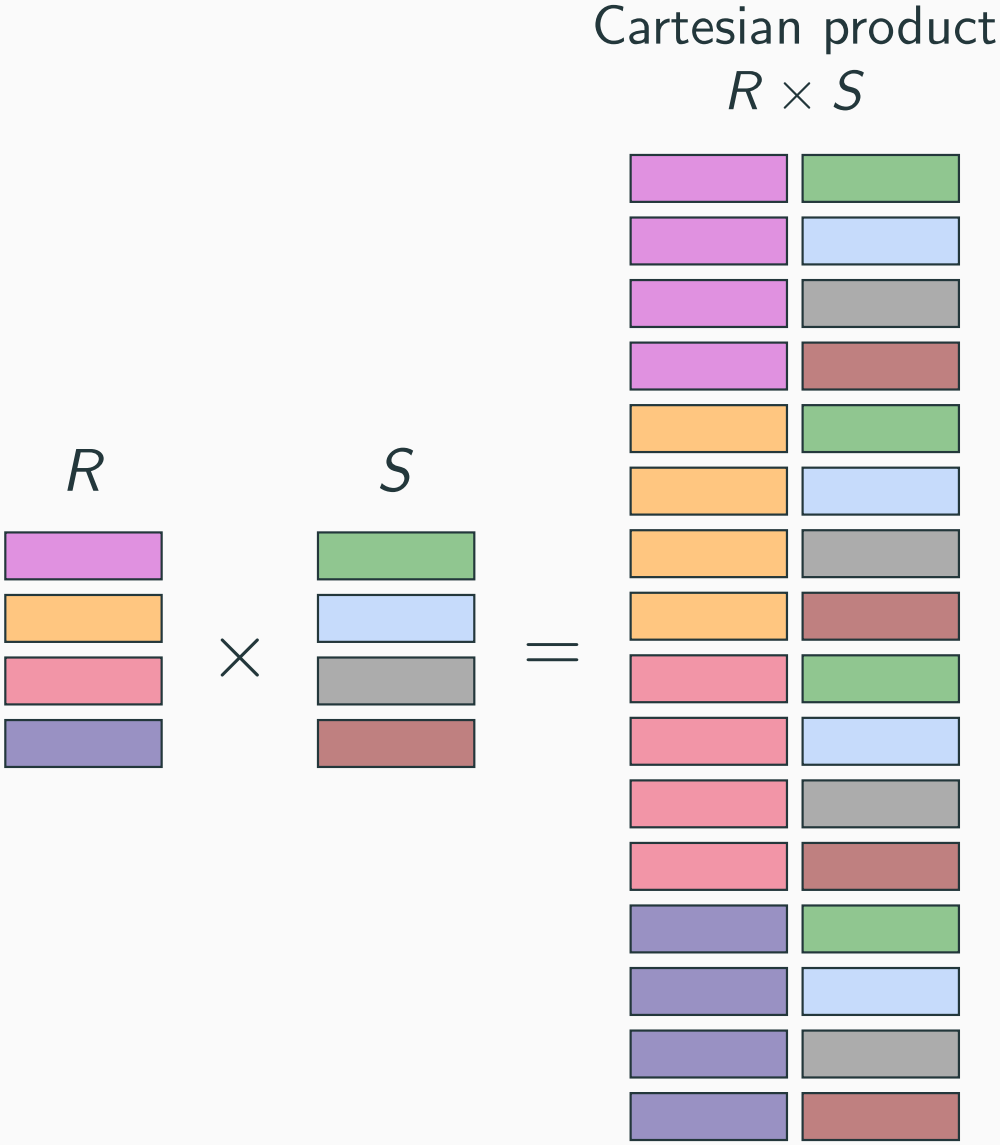
Key Advantage of Factorization



Factorized form (left):

- Lossless representation
- More compact
- Supports computation:
  - Database queries
  - Matrix computation
  - Model training

Key Advantage of Factorization



Factorized form (left):

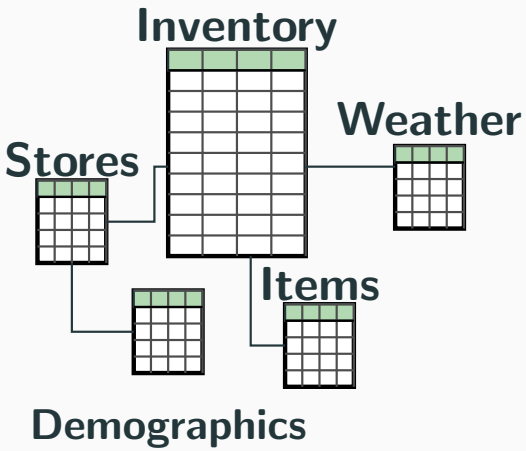
- Lossless representation
- More compact
- Supports computation:
  - Database queries
  - Matrix computation
  - Model training

Example:

Compute Count($R \times S$)

as Count(R) * Count(S)

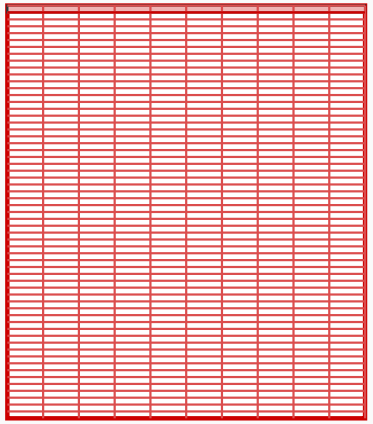
State of Affairs in Learning over Relational Data



Relational Data

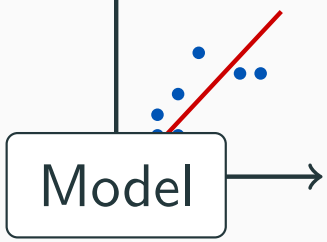
Feature Extraction

10,000s of Features

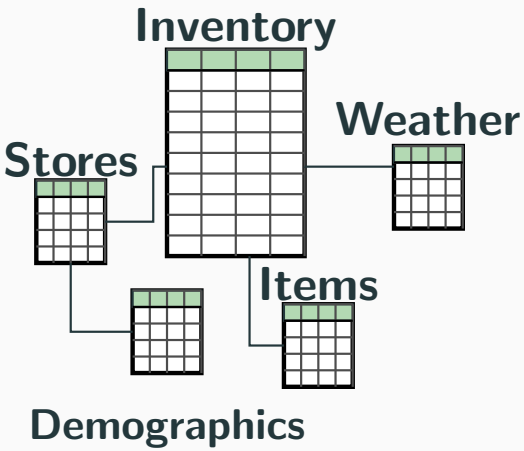


Training Dataset

ML Tool

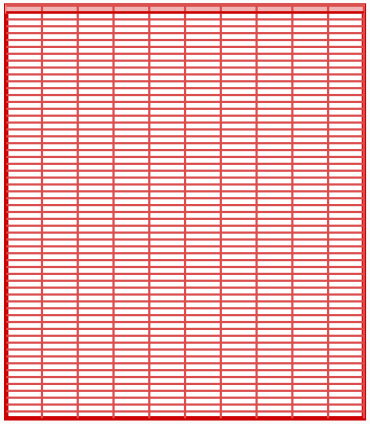


Factorized Learning over Relational Data



Feature Extraction

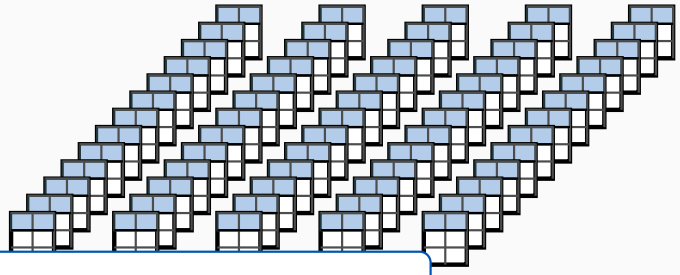
10,000s of Features



Relational Data

Training Dataset

ML Tool



Batch of Queries

Factorization

Optimization

Model

Factorized computation

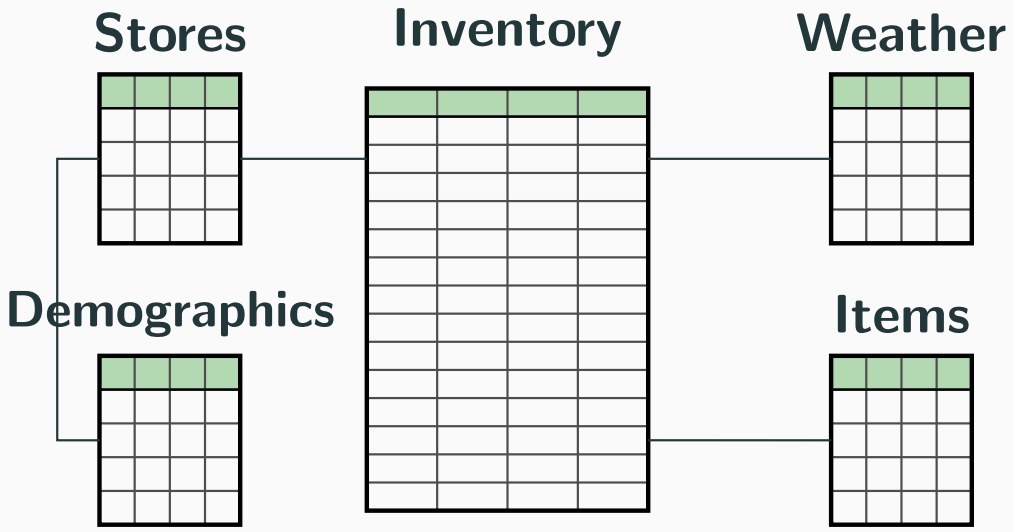
drastically improves

the **time and accuracy**

of model training

over relational data

Factorization can Achieve 1000x Speedup



Relation	Size on Disk (CSV)
Inventory	2 GB
Items	129 KB
Stores	139 KB
Demographics	161 KB
Weather	33 MB
Join	23GB

Factorization can Lead to 1000x Faster Training

Train a linear regression model to predict *inventory* given all features

	PostgreSQL+TensorFlow	
	Time	**Size**
Database	–	2.1 GB
Join Relations	152.06 secs	23 GB
Export Data	351.76 secs	23 GB
Query batch	–	–
Learn	12,738.31 secs	–
Total time	13,242.13 secs	

Factorization can Lead to 1000x Faster Training

Train a linear regression model to predict *inventory* given all features

	PostgreSQL+TensorFlow		Our system	
	Time	**Size**	**Time**	**Size**
Database	–	2.1 GB	–	2.1 GB
Join Relations	152.06 secs	23 GB	–	–
Export Data	351.76 secs	23 GB	–	–
Query batch	–	–	6.08 secs	37 KB
Learn	12,738.31 secs	–	0.05 secs	–
Total time	13,242.13 secs		6.13 secs	

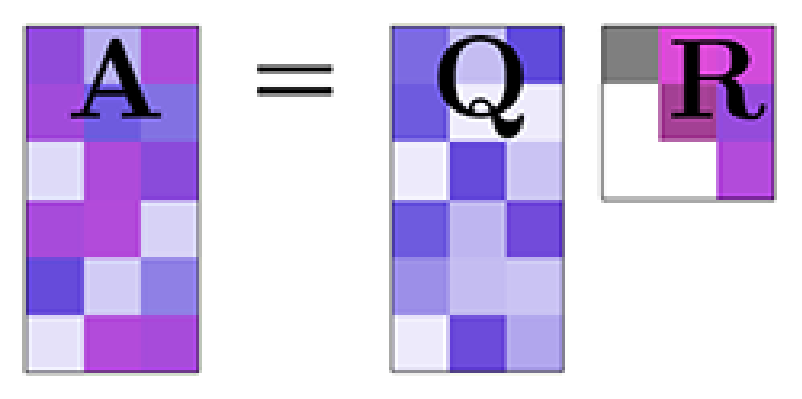
$2,160\times$ faster while being more accurate (RMSE on 2% test data)

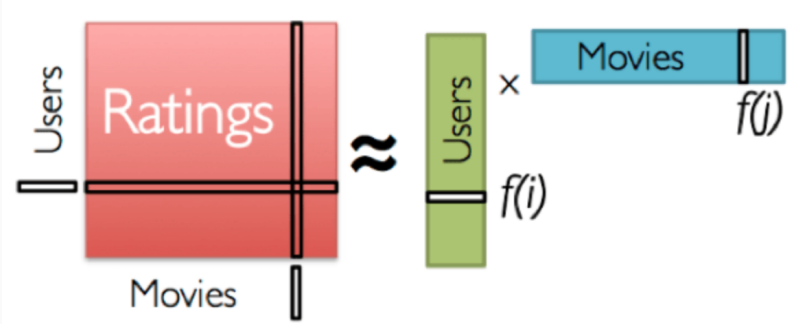
Similar Speedups Observed for other Datasets & Models

Factorization can lead to 1000x Better Numerical Accuracy

Problem: Decompose large matrices defined by relational data

- QR decomposition
- Singular Value Decomposition
- Principal Component Analysis
- Low-rank matrix decomposition





Factorization ⇒ less (redundant) computation

- fewer square roots, divisions, and multiplications

Why are Speedups & Numerical Accuracy Useful?

- Less energy to achieve the same task as competing systems
- Commodity machines can now perform the task previously done on more powerful machines or many more machines
- We can train more models within the same time budget
- Maintain prediction models fresh on a second/minute/hour basis instead of every day/week
- Numerically unstable algorithms are of no use for critical tasks that require precise computation
- ...

Systems and Theory for Factorized Computation

- Publicly available, open-source systems: LMFAO & F-IVM
  - Influenced the design of commercial system RelationalAI
- Impact in database theory: test-of-time award
  - We answered questions on the optimality and computational complexity of factorization
  - Influenced graph database design, static and dynamic query evaluation, provenance, factorized machine learning
- Summer of 2022: Workshop in Zurich dedicated to factorized computation

Going More Succinct than Factorization

- Subject of on-going work by several research groups
- More succinct $\Rightarrow$ subsequent computation not efficient

Going More Succinct than Factorization

- Subject of on-going work by several research groups
- More succinct $\Rightarrow$ subsequent computation not efficient

"Everything should be made as simple as possible,
but not simpler." – *Sessions paraphrasing Einstein*

Acknowledgments

FDB team, in particular:


Ahmet


Amir


Dorde


Haozhe


Jakub


Max


Milos

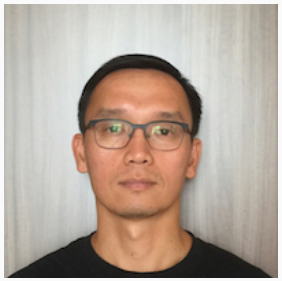

Nils

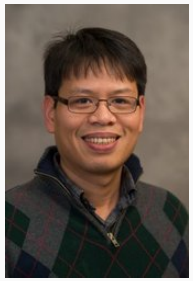

Robert

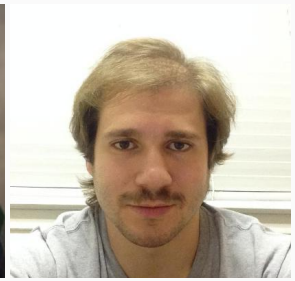
LogicBlox & RelationalAI teams, in particular:


ElSeidy


Henrik


Hung


Long


Mahmoud


Molham


Niko

Thank you!