Givens QR Decomposition over Relational Databases

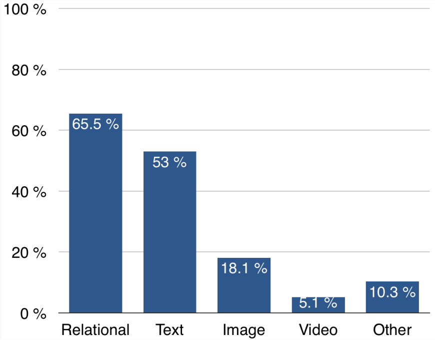
Dan Olteanu | Nils Vortmeier | Đorđe Živanović

University of Zurich UZH | Ruhr Universität Bochum RUB | University of Oxford

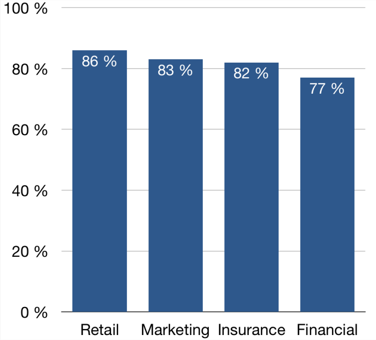
Workshop Factorized Databases, 4.8.2022

Project Context: Machine Learning for Relational Data

Kaggle Survey: Most Data Scientists use Relational Data at Work!



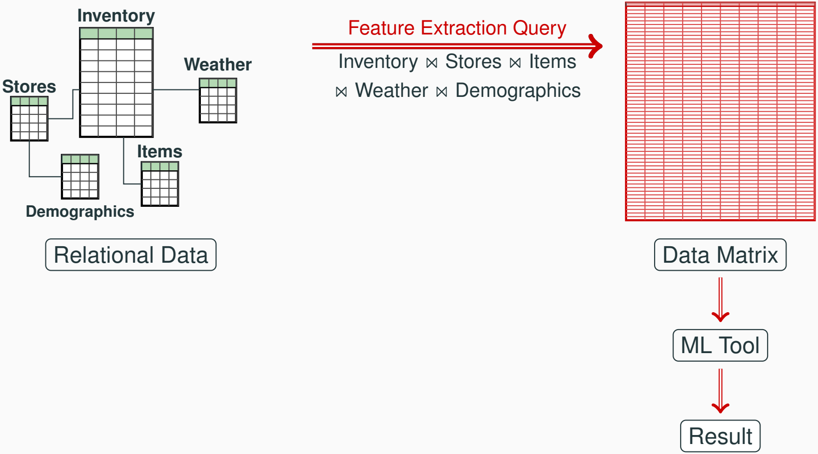
Overall



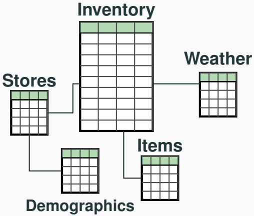
By Industry

Source: The State of Data Science & Machine Learning 2017, Kaggle, October 2017
(based on 2017 Kaggle survey of 16,000 ML practitioners)

Adapting the Machine Learning Pipeline for Relational Data



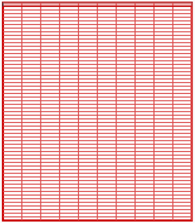
Adapting the Machine Learning Pipeline for Relational Data



Relational Data

Feature Extraction Query

Inventory $\bowtie$ Stores $\bowtie$ Items $\bowtie$ Weather $\bowtie$ Demographics



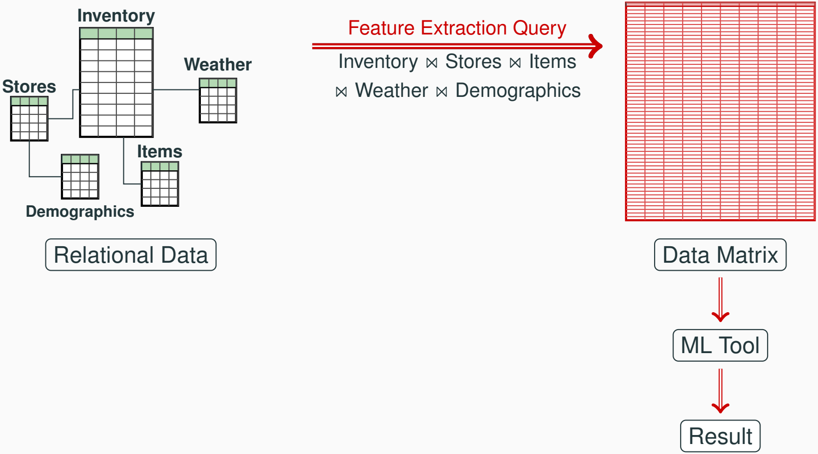
Data Matrix

ML Tool

Result

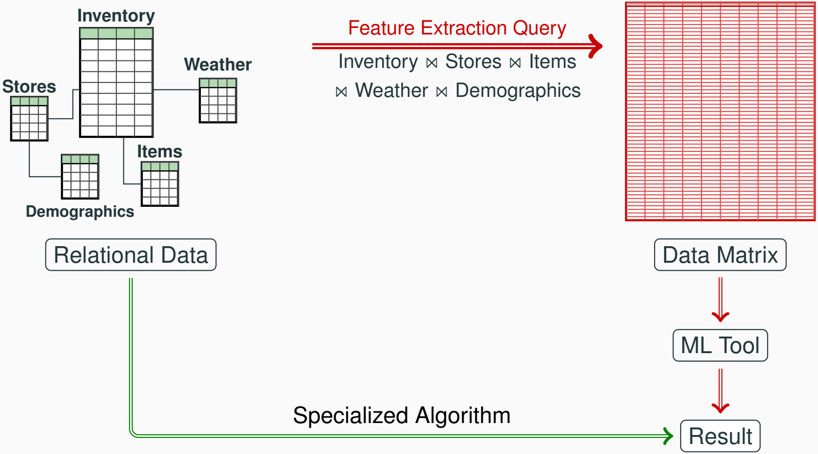
- Data matrix can be orders of magnitude larger than initial relations

Adapting the Machine Learning Pipeline for Relational Data



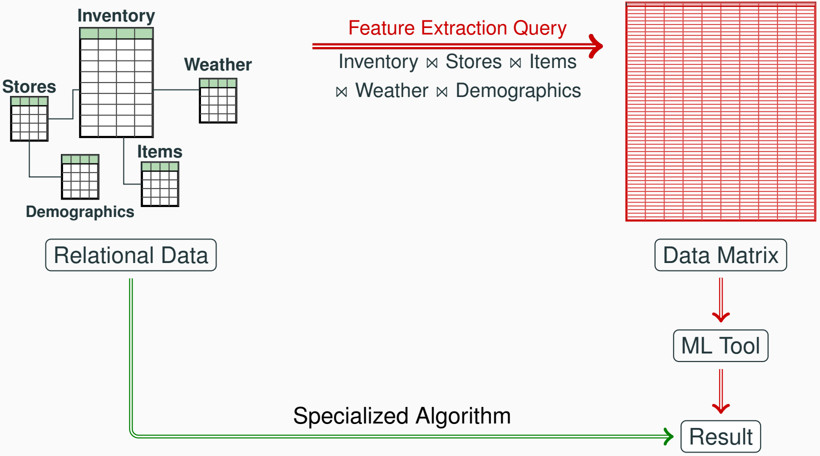
- Data matrix can be orders of magnitude larger than initial relations
- We want to avoid materializing and exporting the data matrix

Adapting the Machine Learning Pipeline for Relational Data



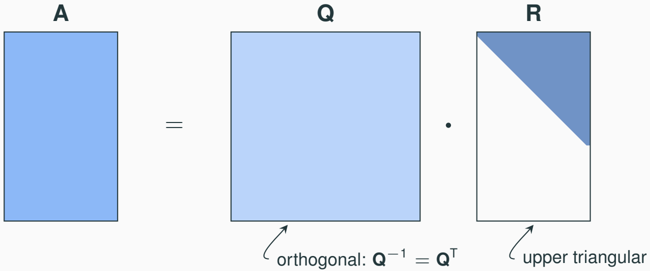
- Data matrix can be orders of magnitude larger than initial relations
- We want to avoid materializing and exporting the data matrix
- Goal: Devise algorithm that runs in linear time wrt. the input database

Adapting the Machine Learning Pipeline for Relational Data

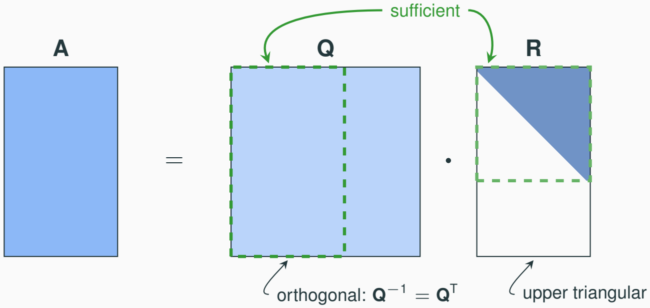


- Data matrix can be orders of magnitude larger than initial relations
- We want to avoid materializing and exporting the data matrix
- Goal: Devise algorithm that runs in linear time wrt. the input database
- Assumption: Join is acyclic, i.e., admits a join tree

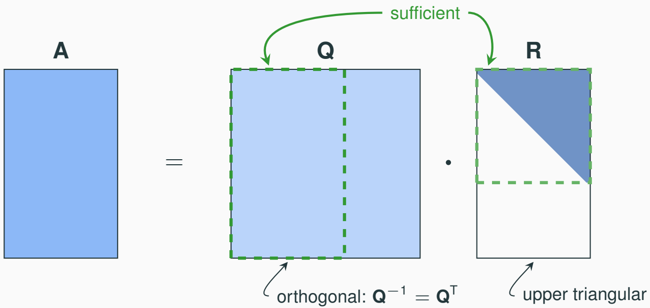
QR Decomposition: Building Block of Machine Learning Algorithms



QR Decomposition: Building Block of Machine Learning Algorithms



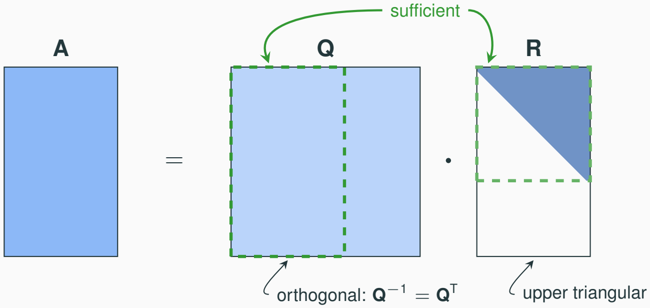
QR Decomposition: Building Block of Machine Learning Algorithms



QR decompositions are used for

- computing the least squares estimator $\hat{\mathbf{x}}$ of $\mathbf{A}\mathbf{x} = \mathbf{b}$
- obtaining a Cholesky decomposition of $\mathbf{A}^{\mathsf{T}}\mathbf{A}$
- computing a singular value decomposition of $\mathbf{A}$
- ...

QR Decomposition: Building Block of Machine Learning Algorithms

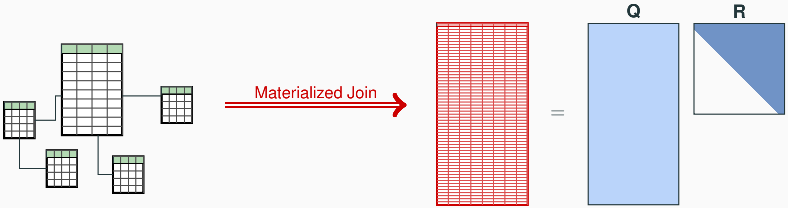


QR decompositions are used for

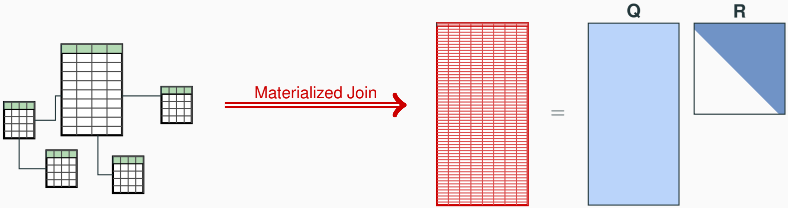
- computing the least squares estimator $\hat{\mathbf{x}}$ of $\mathbf{A}\mathbf{x} = \mathbf{b}$
- obtaining a Cholesky decomposition of $\mathbf{A}^\mathsf{T}\mathbf{A}$
- computing a singular value decomposition of $\mathbf{A}$
- ...

For many applications, only $\mathbf{R}$ is necessary

Main Contribution: The FiGaRo Algorithm

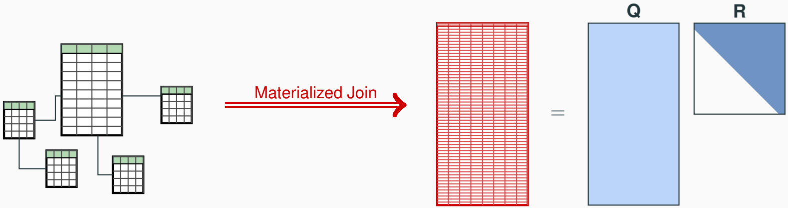


Main Contribution: The FiGaRo Algorithm



We propose FiGaRo: **F**actorized **Gi**vens **Ro**tations

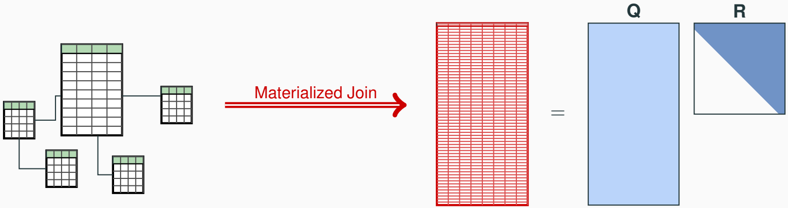
Main Contribution: The FiGaRo Algorithm



We propose FiGaRo: **F**ACTORIZED **G**IVENS **R**OTATIONS

+ Pushes the computation of **R** past the joins to the input database

Main Contribution: The FiGaRo Algorithm

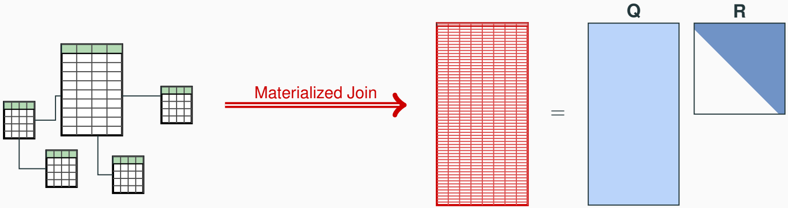


We propose FiGaRo: **F**ACTORIZED **G**IVENS **R**OTATIONS

+ Pushes the computation of **R** past the joins to the input database

+ Takes time linear in the number of tuples **in the input database**

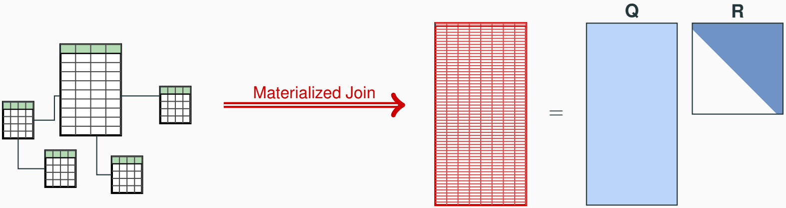
Main Contribution: The FIGARO Algorithm



We propose FIGARO: **F**ACTORIZED **G**IVENS **R**OTATIONS

+ Pushes the computation of **R** past the joins to the input database
+ Takes time linear in the number of tuples **in the input database**
+ Is equivalent to the application of Givens rotations to the materialized join

Main Contribution: The FiGaRo Algorithm



We propose FiGaRo: **F**ACTORIZED **G**IVENS **R**OTATIONS

- Pushes the computation of **R** past the joins to the input database
- Takes time linear in the number of tuples **in the input database**
- Is equivalent to the application of Givens rotations to the materialized join
- If needed, **Q** can be obtained efficiently from **R** and the input database

Standard Algorithm for Computing the QR Decomposition: Givens rotations

$$\begin{bmatrix} 2 & 5 & 1 & 4 \\ 9 & 8 & 4 & 5 \\ 3 & 7 & 2 & 9 \\ 4 & 3 & 6 & 6 \\ 1 & 8 & 5 & 2 \end{bmatrix}$$

Standard Algorithm for Computing the QR Decomposition: Givens rotations

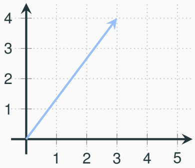
$$\begin{bmatrix} 2 & 5 & 1 & 4 \\ 9 & 8 & 4 & 5 \\ 3 & 7 & 2 & 9 \\ 4 & 3 & 6 & 6 \\ 1 & 8 & 5 & 2 \end{bmatrix}$$

Standard Algorithm for Computing the QR Decomposition: Givens rotations

$$\begin{bmatrix} 2 & 5 & 1 & 4 \\ 9 & 8 & 4 & 5 \\ 3 & 7 & 2 & 9 \\ 4 & 3 & 6 & 6 \\ 1 & 8 & 5 & 2 \end{bmatrix}$$

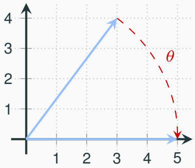
Standard Algorithm for Computing the QR Decomposition: Givens rotations

$$\begin{bmatrix} 2 & 5 & 1 & 4 \\ 9 & 8 & 4 & 5 \\ 3 & 7 & 2 & 9 \\ 4 & 3 & 6 & 6 \\ 1 & 8 & 5 & 2 \end{bmatrix}$$



Standard Algorithm for Computing the QR Decomposition: Givens rotations

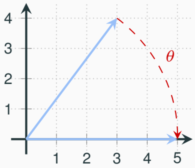
$$\begin{bmatrix} 2 & 5 & 1 & 4 \\ 9 & 8 & 4 & 5 \\ 3 & 7 & 2 & 9 \\ 4 & 3 & 6 & 6 \\ 1 & 8 & 5 & 2 \end{bmatrix}$$



$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$$

Standard Algorithm for Computing the QR Decomposition: Givens rotations

$$\begin{bmatrix} 2 & 5 & 1 & 4 \\ 9 & 8 & 4 & 5 \\ 3 & 7 & 2 & 9 \\ 4 & 3 & 6 & 6 \\ 1 & 8 & 5 & 2 \end{bmatrix}$$

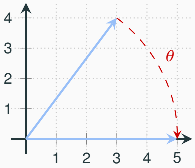


$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$$

$$\theta \quad \begin{bmatrix} 2 & 5 & 1 & 4 \\ 9 & 8 & 4 & 5 \\ 3 & 7 & 2 & 9 \\ 4 & 3 & 6 & 6 \\ 1 & 8 & 5 & 2 \end{bmatrix}$$

Standard Algorithm for Computing the QR Decomposition: Givens rotations

$$\begin{bmatrix} 2 & 5 & 1 & 4 \\ 9 & 8 & 4 & 5 \\ 3 & 7 & 2 & 9 \\ 4 & 3 & 6 & 6 \\ 1 & 8 & 5 & 2 \end{bmatrix}$$

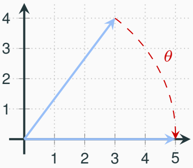


$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$$

$$\begin{array}{c} \\ \\ \theta \\ \\ \\ \end{array} \begin{bmatrix} 2 & 5 & 1 & 4 \\ 9 & 8 & 4 & 5 \\ 3 & 7 & 2 & 9 \\ 4 & 3 & 6 & 6 \\ 1 & 8 & 5 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \cos\theta & \sin\theta & 0 \\ 0 & 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 5 & 1 & 4 \\ 9 & 8 & 4 & 5 \\ 3 & 7 & 2 & 9 \\ 4 & 3 & 6 & 6 \\ 1 & 8 & 5 & 2 \end{bmatrix}$$

Standard Algorithm for Computing the QR Decomposition: Givens rotations

$$\begin{bmatrix} 2 & 5 & 1 & 4 \\ 9 & 8 & 4 & 5 \\ 3 & 7 & 2 & 9 \\ 4 & 3 & 6 & 6 \\ 1 & 8 & 5 & 2 \end{bmatrix}$$

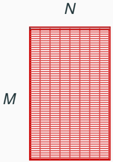


$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 5 & 1 & 4 \\ 9 & 8 & 4 & 5 \\ 3 & 7 & 2 & 9 \\ 4 & 3 & 6 & 6 \\ 1 & 8 & 5 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \cos\theta & \sin\theta & 0 \\ 0 & 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 5 & 1 & 4 \\ 9 & 8 & 4 & 5 \\ 3 & 7 & 2 & 9 \\ 4 & 3 & 6 & 6 \\ 1 & 8 & 5 & 2 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 5 & 1 & 4 \\ 9 & 8 & 4 & 5 \\ 5 & & & \\ 0 & & & \\ 1 & 8 & 5 & 2 \end{bmatrix}$$

Runtime of FiGaRo versus Prior Work

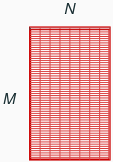


Prior work: Givens rotations on the materialized join

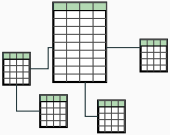
- Number of necessary rotations: $\mathcal{O}(MN)$
- Runtime linear in M, quadratic in N

Runtime of FIGARO versus Prior Work

Prior work: Givens rotations on the materialized join



- Number of necessary rotations: $\mathcal{O}(MN)$
- Runtime linear in M, quadratic in N



FIGARO on the input database with K tuples

- Runtime linear in K, quadratic in N
- If $M \gg K$:
  - FIGARO is **faster** than the standard algorithms
  - FIGARO is **experimentally more accurate** than standard implementations
- Even if $M \approx K$, FIGARO may be faster than the standard algorithms

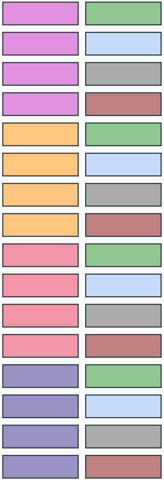
Example: FiGaRo on a Cartesian Product

Two relations

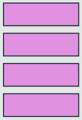
Example: FiGaRo on a Cartesian Product

Two relations

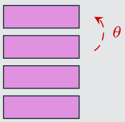
Cartesian product



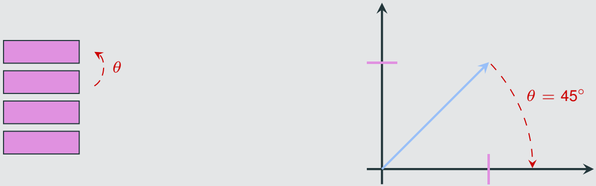
Example: FIGARO on a Cartesian Product



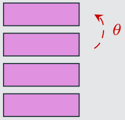
Example: FiGaRo on a Cartesian Product

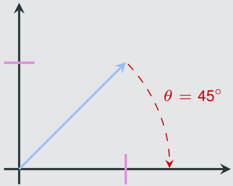


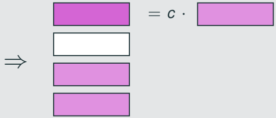
Example: FIGARO on a Cartesian Product



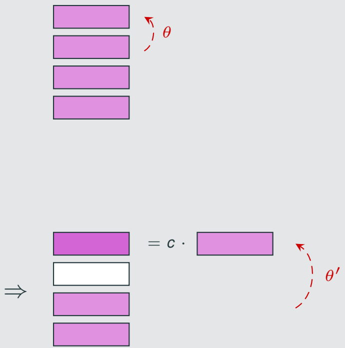
Example: FIGARO on a Cartesian Product

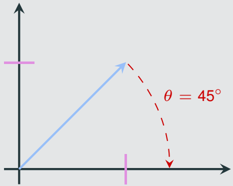




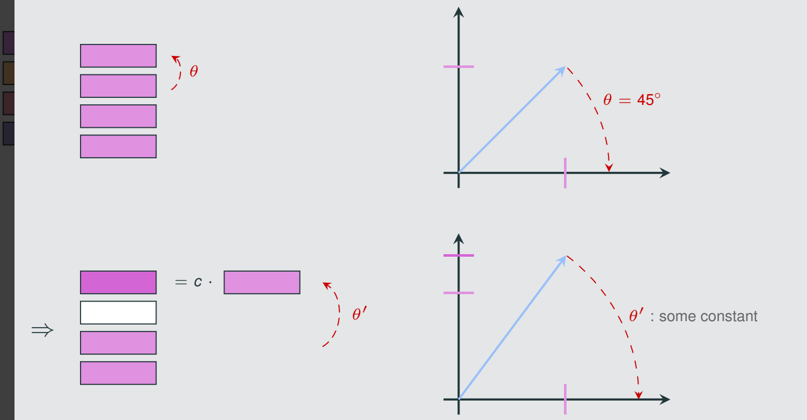


Example: FiGaRo on a Cartesian Product

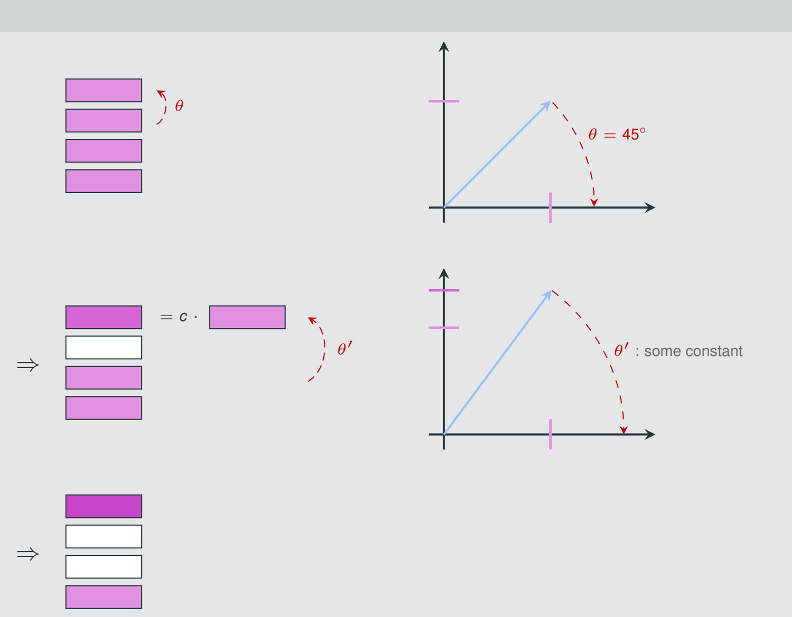




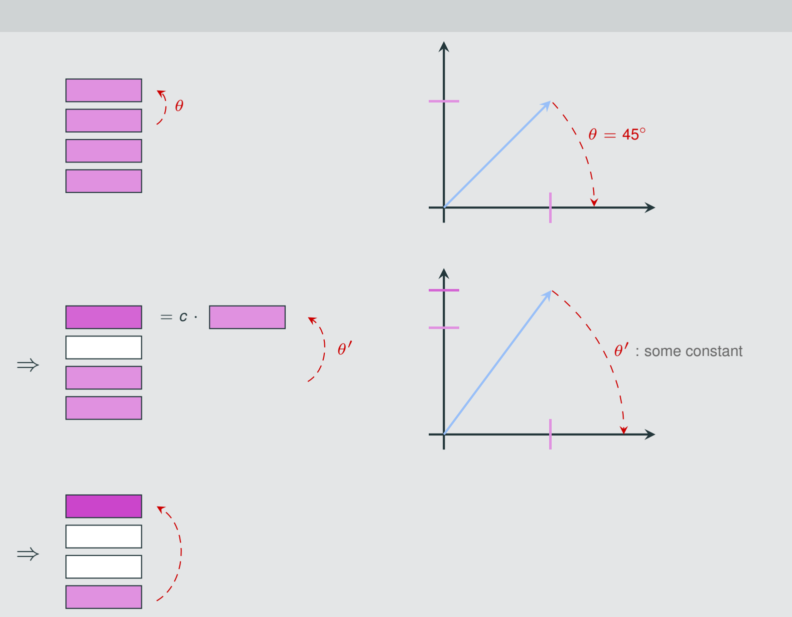
Example: FIGARO on a Cartesian Product



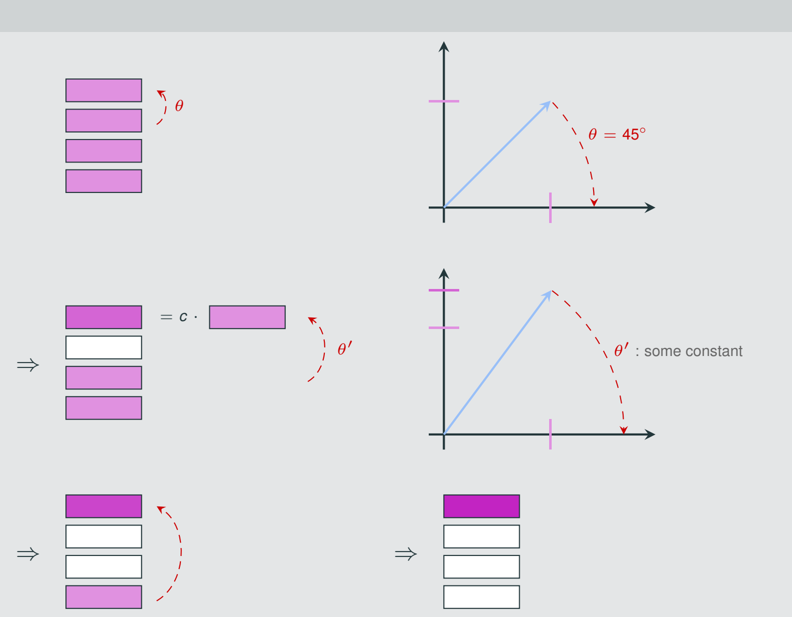
Example: FIGARO on a Cartesian Product



Example: FIGARO on a Cartesian Product



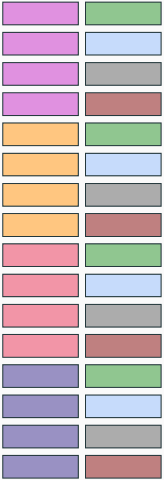
Example: FiGaRo on a Cartesian Product



Example: FiGaRo on a Cartesian Product

Two relations

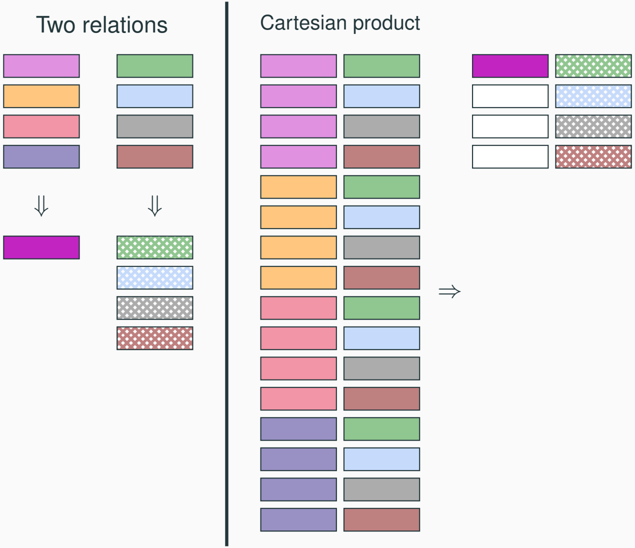
Cartesian product



Example: FiGaRo on a Cartesian Product

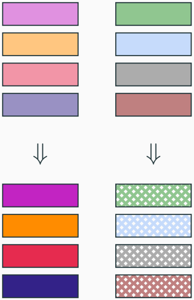
Two relations

Cartesian product

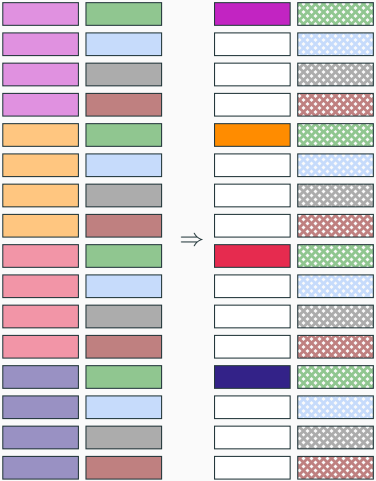


Example: FiGaRo on a Cartesian Product

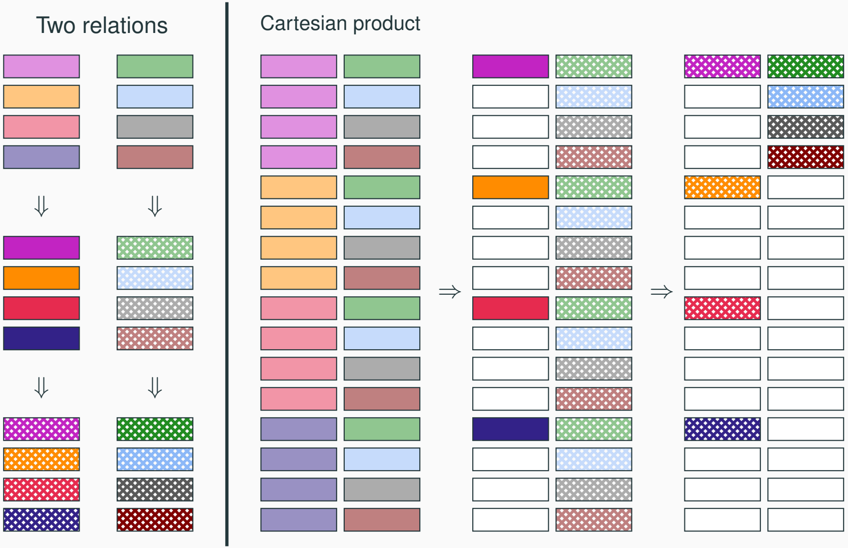
Two relations



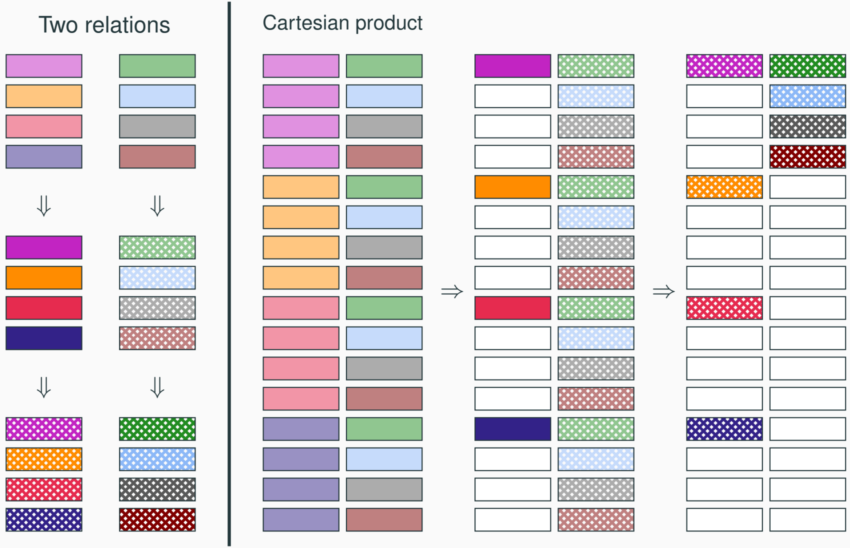
Cartesian product



Example: FiGaRo on a Cartesian Product



Example: FIGARO on a Cartesian Product



- Instead of applying rotations to Cartesian product:
  compute non-zero rows from input database in linear time

Towards the general case: Cartesian Product

Combine all rotations in orthogonal matrix **M**

$$\mathbf{M} = \quad \cdots$$

Towards the general case: Cartesian Product

Combine all rotations in orthogonal matrix **M**

$$\mathbf{M} = \cdots$$

For any matrix **A** one can compute **MA** in linear time

Towards the general case: Cartesian Product

Combine all rotations in orthogonal matrix **M**

$$\mathbf{M} = \quad \cdots$$

For any matrix **A** one can compute **MA** in linear time

$$\mathbf{MA} = \begin{bmatrix} \mathcal{H}(\mathbf{A}) \\ \mathcal{T}(\mathbf{A}) \end{bmatrix}$$

Head: first row of **MA**

Tail: remaining rows of **MA**

Towards the general case: Cartesian Product

Combine all rotations in orthogonal matrix **M**

$$\mathbf{M} =$$

For any matrix **A** one can compute **MA** in linear time

$$\mathbf{MA} = \begin{bmatrix} \mathcal{H}(\mathbf{A}) \\ \mathcal{T}(\mathbf{A}) \end{bmatrix}$$

Head: first row of **MA**

Tail: remaining rows of **MA**

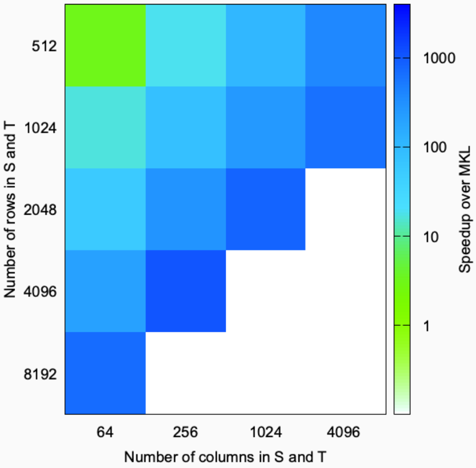
QR decomposition for Cartesian Product of matrices **S** and **T** with p resp. q rows:

$$\mathbf{S} \times \mathbf{T} = \mathbf{Q} \cdot \begin{bmatrix} \sqrt{q}\,\mathcal{H}(\mathbf{S}) & \sqrt{p}\,\mathcal{H}(\mathbf{T}) \\ \sqrt{q}\,\mathcal{T}(\mathbf{S}) & \mathbf{0} \\ \mathbf{0} & \sqrt{p}\,\mathcal{T}(\mathbf{T}) \end{bmatrix}$$

Experimental Evaluation

Runtime Performance on Cartesian Products

- Input: Matrices **S** and **T** with different numbers of rows and columns
- Intel MKL: on the materialized Cartesian product **S** $\times$ **T**
- FIGARO: on **S** and **T** directly



Accuracy Experiments: Setup

- Usually, accuracy of algorithms for QR decomposition is measured in terms of orthogonality of $\mathbf{Q}$

Accuracy Experiments: Setup

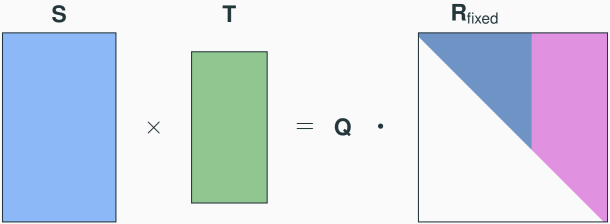
- Usually, accuracy of algorithms for QR decomposition is measured in terms of orthogonality of $\mathbf{Q}$
- We propose a new accuracy based on $\mathbf{R}$

Given:

- Upper triangular matrix $\mathbf{R}_{\text{fixed}}$, serving as ground truth

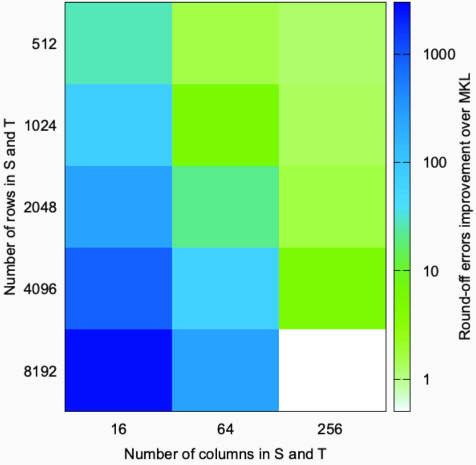
- Dimensions of target matrices $\mathbf{S}, \mathbf{T}$

Compute: Matrices $\mathbf{S}, \mathbf{T}$ such that



Accuracy for QR Decomposition of Cartesian Products

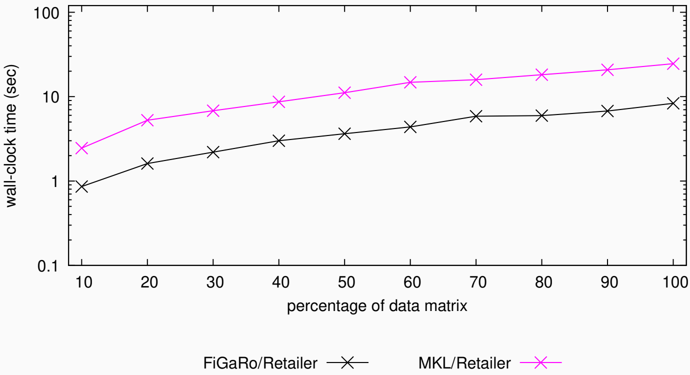
- Input: Matrices $\mathbf{S}, \mathbf{T}$ with varying numbers of rows and columns
- Relative error is measured using Frobenius norm of obtained $\hat{\mathbf{R}}_{\text{fixed}}$ vs the ground truth $\mathbf{R}_{\text{fixed}}$



Runtime Performance as Function of the Size of the Materialized Join (1/3)

Retailer dataset

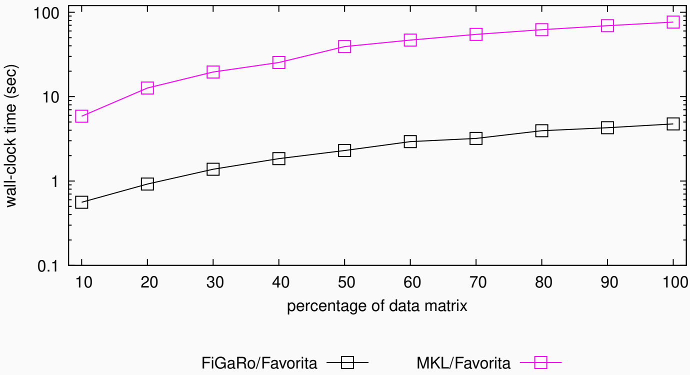
- # tuples in the input database vs materialized join: 86M vs 84M
- # data columns in the materialized join: 43
- # values in the materialized join vs input database (relative): 10x



Runtime Performance as Function of the Size of the Materialized Join (2/3)

Favorita dataset

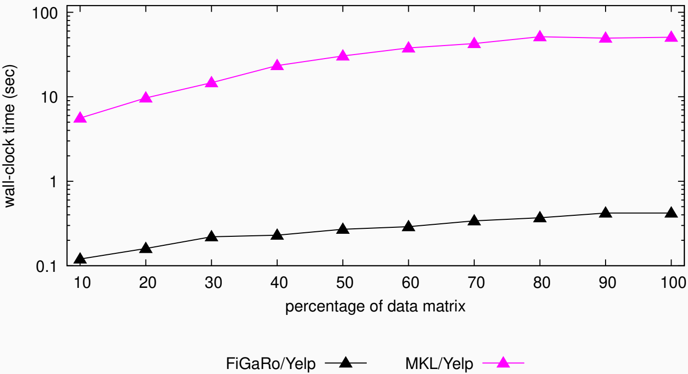
- # tuples in the input database vs materialized join: 125M vs 127M
- # data columns in the materialized join: 30
- # values in the materialized join vs input database (relative): 6x



Runtime Performance as Function of the Size of the Materialized Join (3/3)

Yelp dataset

- # tuples in the input database vs materialized join: 2M vs 150M
- # data columns in the materialized join: 50
- # values in the materialized join vs input database (relative): 332x



Conclusion and Future Work

Conclusion and Future Work

- FIGARO pushes the computation of the QR decomposition past the join
- It significantly reduces the computation time in experiments on synthetic and real-world relational data
- It can be numerically more accurate than standard implementations
- The techniques can be transferred to other matrix decompositions, e.g. LU decomposition