Lightweight Correlation-Aware Table Compression

Mihail Stoian, Alexander van Renen, Jan Kobiolka, Ping-Lin Kuo, Josif Grabocka, Andreas Kipf

University of Technology Nuremberg

Virtual: Spoiler

Compression

```python
import pandas as pd
import virtual

# Read your data.
df = pd.read_csv('file.csv')

# Your operations.
df = ...

# Virtualize + save to Parquet.
virtual
  .to_format(df, 'file.parquet')
```

Virtual: Spoiler

Compression

```python
import pandas as pd
import virtual

# Read your data.
df = pd.read_csv('file.csv')

# Your operations.
df = ...

# Virtualize + save to Parquet.
virtual
  .to_format(df, 'file.parquet')
```

Query

```python
import virtual

virtual.query('''
  select avg(price)
  from read_parquet(
    "file.parquet"
  ) where year >= 2024''',
  engine = 'duckdb'
)
```

Table Compression: Status Quo

Columnar Encoding Schemes

- Frame-of-Reference (FOR), Run-Length-Encoding (RLE) etc.
- Pretty lightweight ⇒ Fast decompression ✓
- File sizes: Could be better.. 🙄

Table Compression: Status Quo

Columnar Encoding Schemes

- Frame-of-Reference (FOR), Run-Length-Encoding (RLE) etc.
- Pretty lightweight ⇒ Fast decompression ✓
- File sizes: Could be better.. 🙄

Open File Formats

- Recent surge: Apache Parquet, ORC, etc.
- Research prototypes: BtrBlocks, FastLanes.

Table Compression: Status Quo

Columnar Encoding Schemes

- Frame-of-Reference (FOR), Run-Length-Encoding (RLE) etc.
- Pretty lightweight ⇒ Fast decompression ✓
- File sizes: Could be better.. 🙄

Open File Formats

- Recent surge: Apache Parquet, ORC, etc.
- Research prototypes: BtrBlocks, FastLanes.
- *Still using the standard encoding schemes.*
  ⇒ They have reached a plateau.

Exploiting Hidden Functions

Property Total	Burglary	Larceny	Motor Vehicle Theft
5583	1884	3264	435
6368	1988	3878	502
6641	2246	3858	537

$$\texttt{Property Total} = \texttt{Burglary} + \texttt{Larceny} + \texttt{Motor Vehicle Theft}$$

Exploiting Hidden Functions

Property Total	Burglary	Larceny	Motor Vehicle Theft
5583	1884	3264	435
6368	1988	3878	502
6641	2246	3858	537

$$\texttt{Property Total} = \texttt{Burglary} + \texttt{Larceny} + \texttt{Motor Vehicle Theft}$$

Property Total_offset	Burglary	Larceny	Motor Vehicle Theft
0	1884	3264	435
0	1988	3878	502
0	2246	3858	537

Functions

Requirements

(a) Make target column redundant ⇒ Zero storage fingerprint.

(b) Touch as few columns as possible ⇒ Fast table scans.

(c) Allow multiple functions ⇒ Even better compression.

Approach

Sparse Linear Regression

- Linear regression ⇒ linear functions.

Approach

Sparse Linear Regression

- Linear regression ⇒ linear functions.
- However: We have to enforce (b).

Approach

Sparse Linear Regression

- Linear regression ⇒ linear functions.
- However: We have to enforce (b).
  ⇒ Optimize via $\mathcal{L}_0$ penalty.

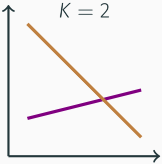
Approach

Sparse Linear Regression

- Linear regression ⇒ linear functions.
- However: We have to enforce (b).
  ⇒ Optimize via $\mathcal{L}_0$ penalty.

K-Regression

- Train *multiple* linear (sparse) regressors.



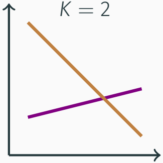
Approach

Sparse Linear Regression

- Linear regression ⇒ linear functions.
- However: We have to enforce (b).
  ⇒ Optimize via $\mathcal{L}_0$ penalty.

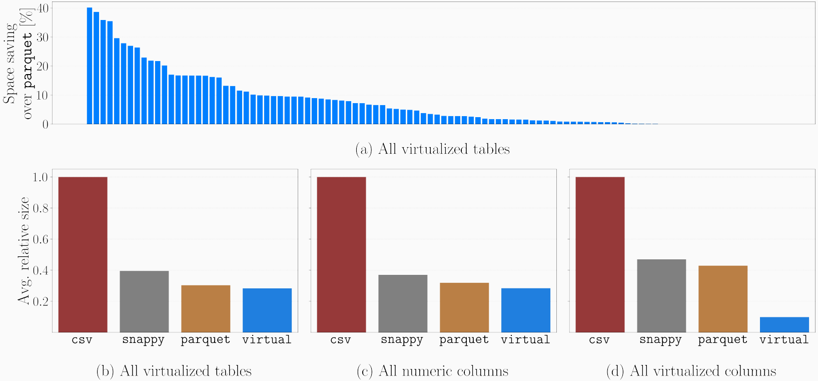
K-Regression

- Train *multiple* linear (sparse) regressors.



- Encode which regression we select via an auxiliary column.

Evaluation: File Sizes



(a) All virtualized tables

(b) All virtualized tables

(c) All numeric columns

(d) All virtualized columns

Figure 1: Comparison to Parquet+Snappy (`parquet`) on 103 `data.gov` tables

Evaluation: Query Latency

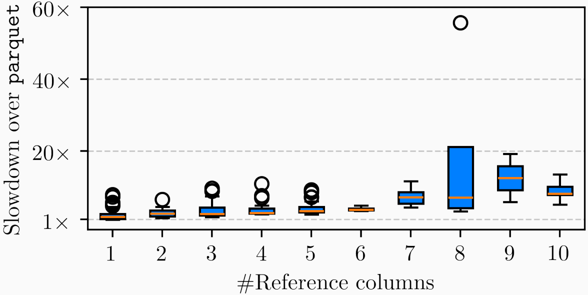


Figure 2: Linear column scan slowdown

Summary

Lightweight Correlation-Aware Compression

- Learn multiple sparse linear regressors.

Summary

Lightweight Correlation-Aware Compression

- Learn multiple sparse linear regressors.
- Exploit them in compression and query execution.

`github.com/utndatasystems/virtual`