Parquet

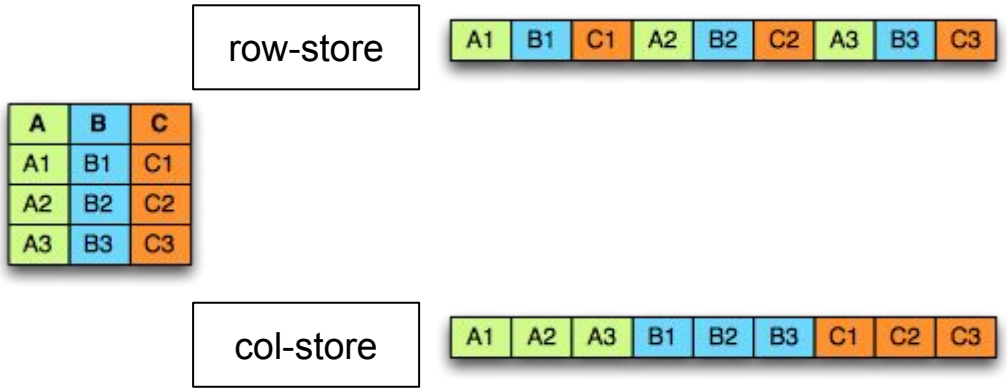
Distributed Information Systems Workshop, 17. 5. 2017

Silvan Heller

What is Parquet

- A File Storage Format for Databases
- Competitors: JSON, CSV and others
- Parquet is ***Column-oriented***
- Use Parquet to store Relational & Non-relational Data

Recap: Row vs Column-Storage



- For column-based queries better performance
- Better Compression (more homogenous data)
- Datatype is the same in a column which enables encoding optimization for processors

https://blog.twitter.com/2013/dremel-made-simple-with-parquet

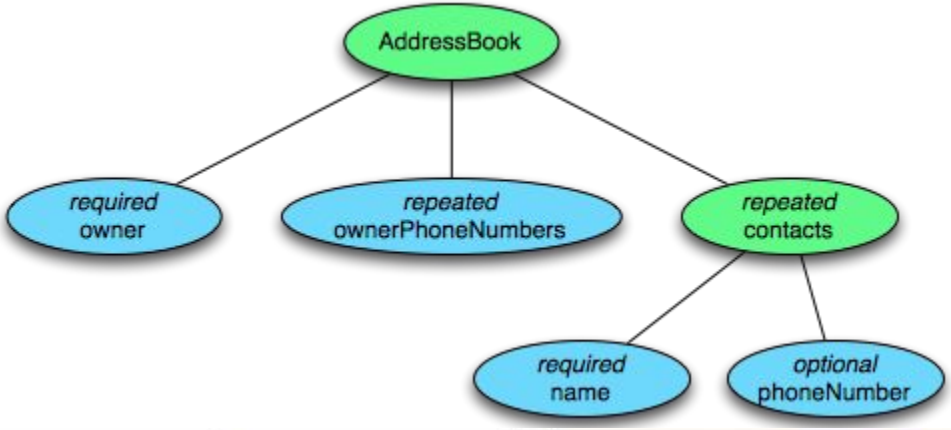
Storing Nested Objects

Data Model

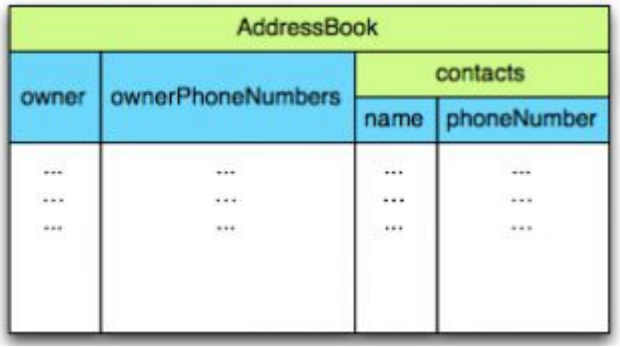
- Closely related to Protocol Buffer Syntax
- A field is either *repeated* (*), *optional* (?), or *required*

```
message AddressBook {
        required string owner;
        repeated string ownerPhoneNumbers;
        repeated group contacts {
                required string name;
                optional int32 phoneNumber;
        }
}
```

Data Model - Visualized



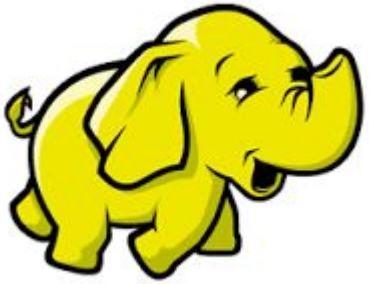
Column	Type
owner	string
ownerPhoneNumbers	string
contacts.name	string
contacts.phoneNumber	string



https://blog.twitter.com/2013/dremel-made-simple-with-parquet

Benchmarking Parquet

Technology



Scenario I

ID	String-data 1	String-data 2	...	String-data #cols	Number
1-#rows	“...”	“...”	...	“...”	1-10

Meet the players

CSV

```
id,0c
0,>nda=
1,8#bfQ
2,?$s+6
3,PY7#s
4,",oXS+"
```

JSON

```
{"id":0,"0c":"+9Oe9"}
{"id":1,"0c":"KMy0Z"}
{"id":2,"0c":"Hl[cv"}
{"id":3,"0c":"y?(j7"}
{"id":4,"0c":"~F<&["}
```

ORC

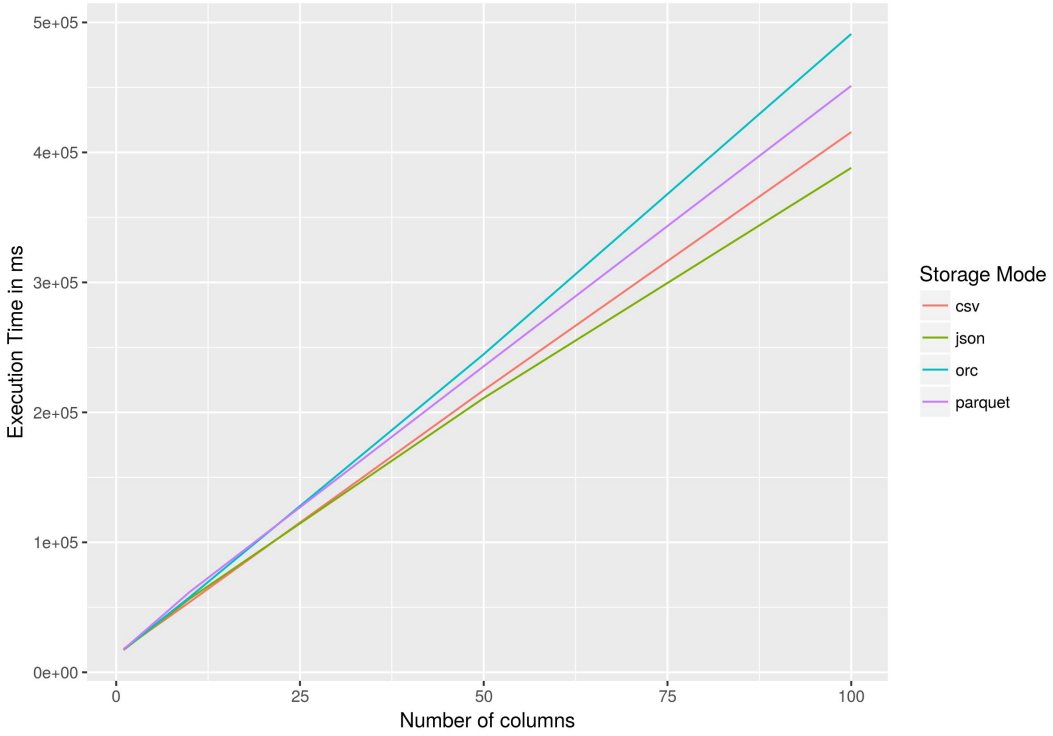
Compressed Encoding

Apache orc™

Parquet

Compressed Encoding

Write Overhead (10^6 rows, 100 string-length)



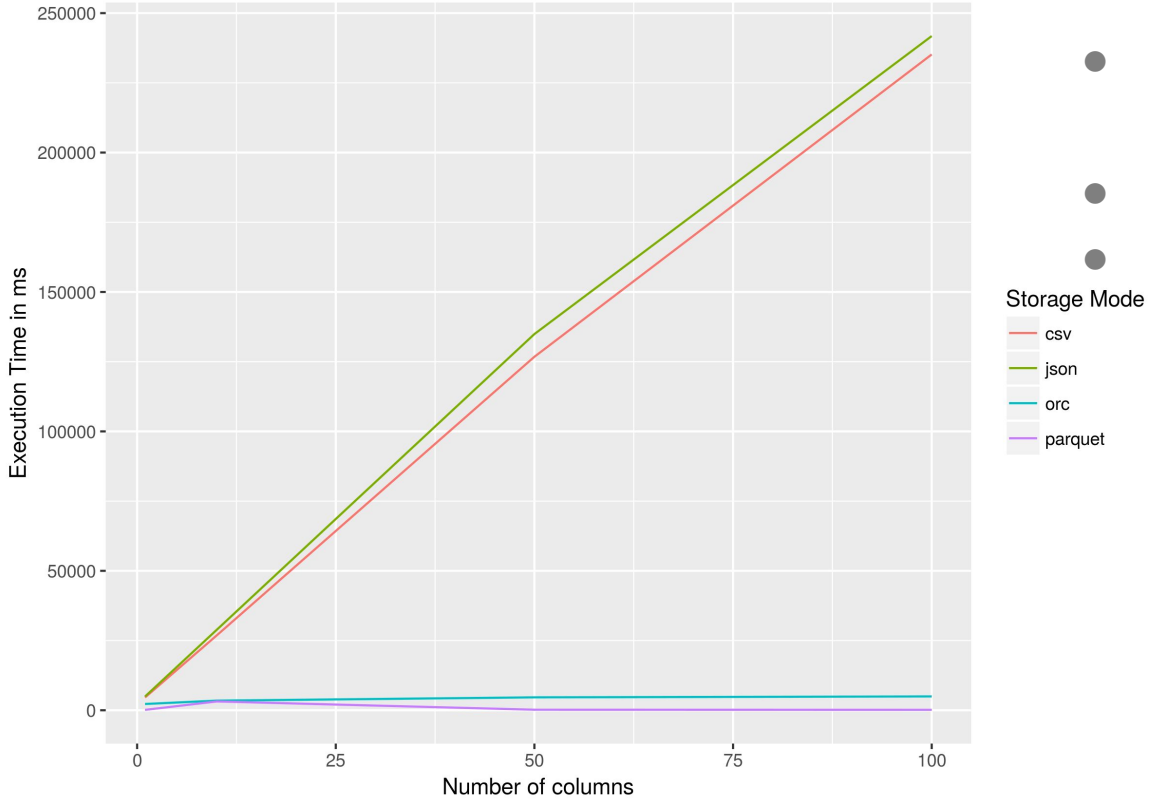
- Grows linearly with number of columns
- Also expected
- As expected, there is a (small) penalty associated with special file formats

File Size for 10^6 Rows, 100 columns

- CSV: 9.9 GB
- JSON: 11 GB
- ORC: 9.5 GB
- Parquet: 9.8 GB

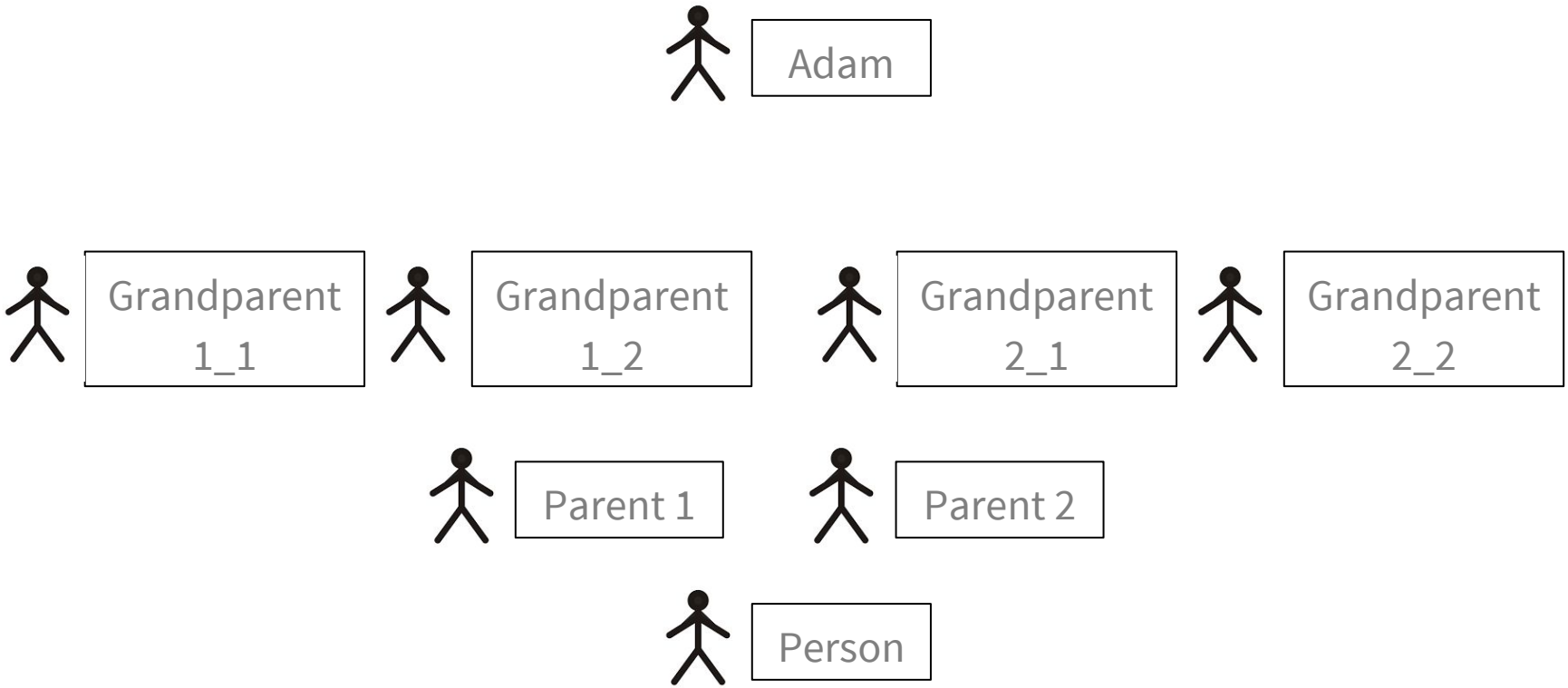
- As Expected, JSON pays a heavy price for its schema storage method

AVG(id): 10^6 Rows, 100 String Length



- Constant execution time as number of columns grows
- ORC also performs fine
- This is what we expected

Scenario II / Nested Objects



You've met the players

CSV

id,0c
0,>I da=
1,8#bfQ
2,?$s+S
3,PY7#s
4,",oXS+"

JSON

{"id":0,"0c":"+9Oe9"}
{"id":1,"0c":"KMy0Z"}
{"id":2,"0c":"Hl[cv"}
{"id":3,"0c":"y?(j7"}
{"id":4,"0c":"~F<&["}

ORC

Compressed Encoding

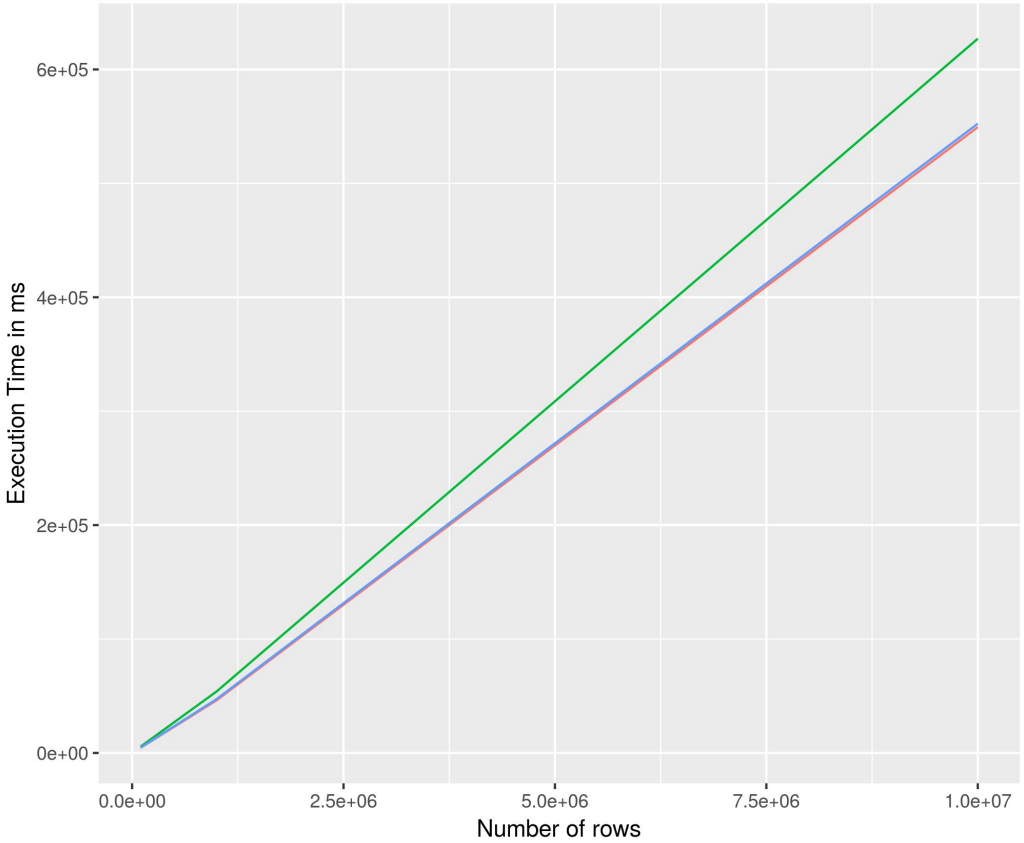
Apache orc™

Parquet

Compressed Encoding

Spark-CSV does not support
nested Objects

Writing Nested Objects



- ORC performs slightly worse
- We attribute this to String compression

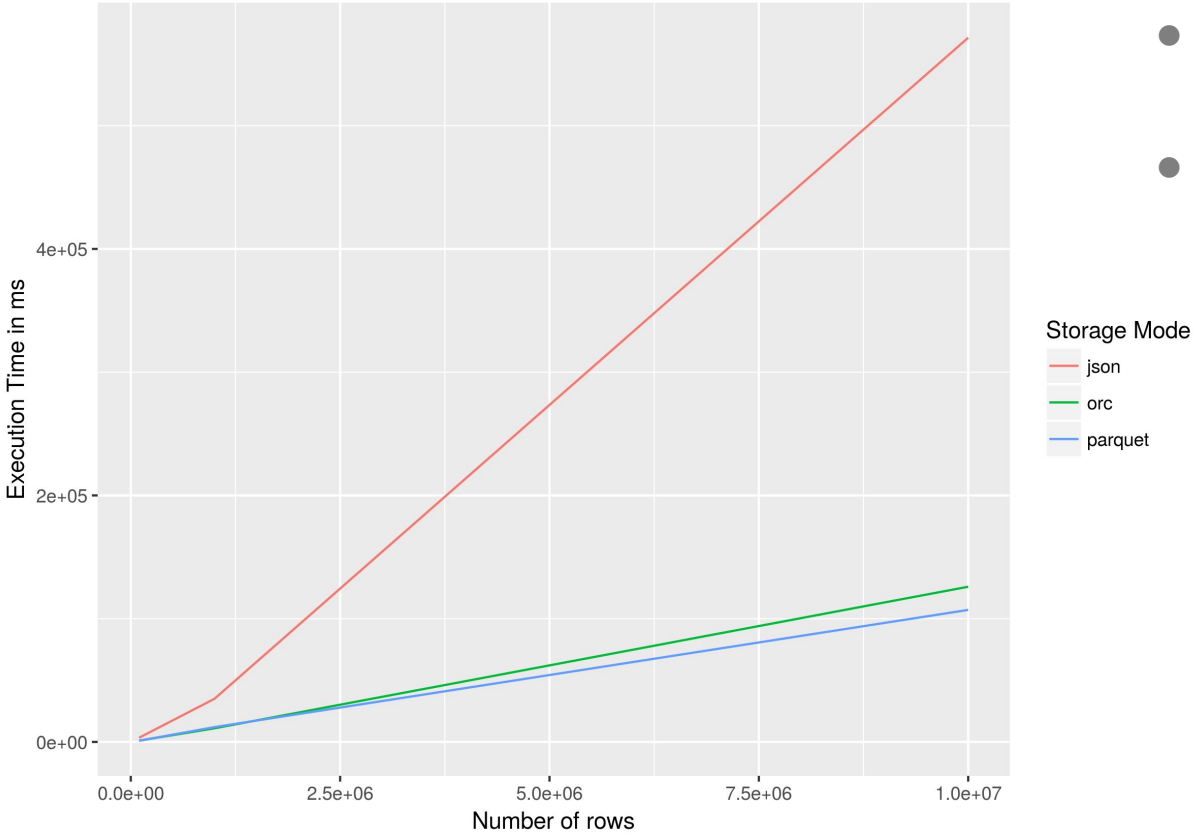
File Size, 10^7 People, Name-length = 100

- JSON: 11GB
- ORC: 6.7 GB
- Parquet: 7.0 GB

The picture is even bleaker for JSON when we set name-length = 5

- JSON: 3.7 GB
- ORC: 416 MB(!)
- Parquet: 540 MB(!)

Query Time for avg(grandparent1_1.age)



- Again, Parquet clearly beats JSON
- Growth is linear as expected since we perform a tablescan

Conclusion

Lessons Learned

- Parquet is Plug-and-Play with Apache Spark
  - `dataframe.write.parquet(filename)`
- You should only use JSON if you care about interoperability
- But: you can always convert your fetched data to JSON
- ORC is also Plug-and-Play with Spark, has performed very well in the benchmarks
- Full data set size: 150GB, Benchmark Time ~ 5 hours

Get the Code:
https://github.com/silvanheller/parquet-demo

Questions

References

DIS Slides, FS17, Heiko Schuldt (Row v Column Storage)

https://blog.twitter.com/2013/dremel-made-simple-with-parquet

https://parquet.apache.org/documentation/latest/

Dremel: Interactive Analysis of Web-Scale Datasets, Melnik et. al

https://github.com/vitrivr/ADAMpro (Code snippets)

Brief History

- Google Dremel
- Twitter and Cloudera create *Parquet*
- *Parquet* becomes *Apache Parquet*

cloudera

Recap: Row vs Column-Storage

Row Stores

- Relational databases use "Row Stores"
- Store entire tuples on database pages
- Optimized for OLTP applications (write or read complete tuples)

Customers

ID	Name	City	Balance	Discount
01	Legrand	Geneva	-1'080,00	0.10
02	Marty	Basel	-8'00,00	0.20
03	Frei	Basel	0,00	0,10
04	Janvier	Geneva	0,00	0,10
05	Rossi	Lugano	0,00	0,05
06	Meier	Zurich	-3'800,00	0,05
07	Hürlimann	Lucerne	-100,00	0,05
08	Schmid	Lausanne	-2'235,00	0,10
09	McAllen	Zurich	-550,00	0,00
10	Lacroix	Geneva	-31'000,00	0,20
…	…	…	…	…

Column Store Database Systems

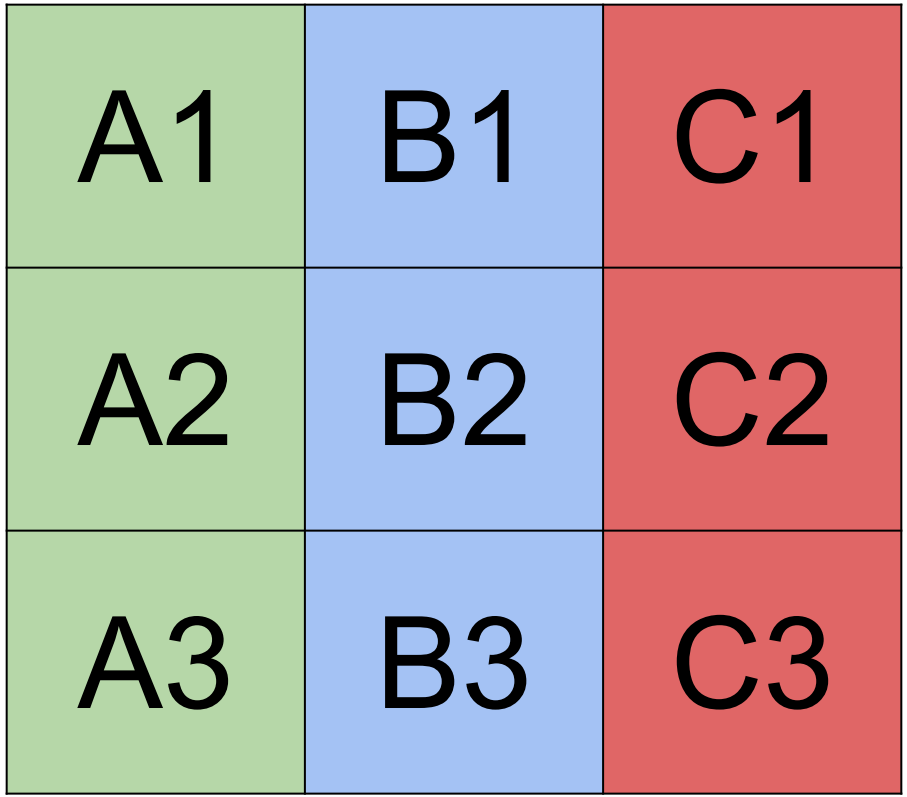
- Columns stores are optimized for applications that are characterized by …
- … long, complex read transactions that do not request full tuples (OLAP)
- … and rather few update operations

Customers

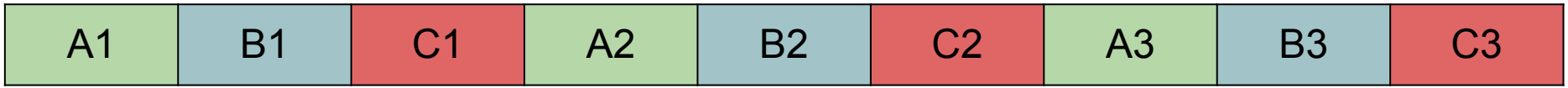
ID	Name	City	Balance	Discount
01	Legrand	Geneva	-1'080,00	0.10
02	Marty	Basel	-8'00,00	0.20
03	Frei	Basel	0,00	0,10
04	Janvier	Geneva	0,00	0,10
05	Rossi	Lugano	0,00	0,05
06	Meier	Zurich	-3'800,00	0,05
07	Hürlimann	Lucerne	-100,00	0,05
08	Schmid	Lausanne	-2'235,00	0,10
09	McAllen	Zurich	-550,00	0,00
10	Lacroix	Geneva	-31'000,00	0,20
…	…	…	…	…

database pages

Custom Tables



Row-Major



Column-Major