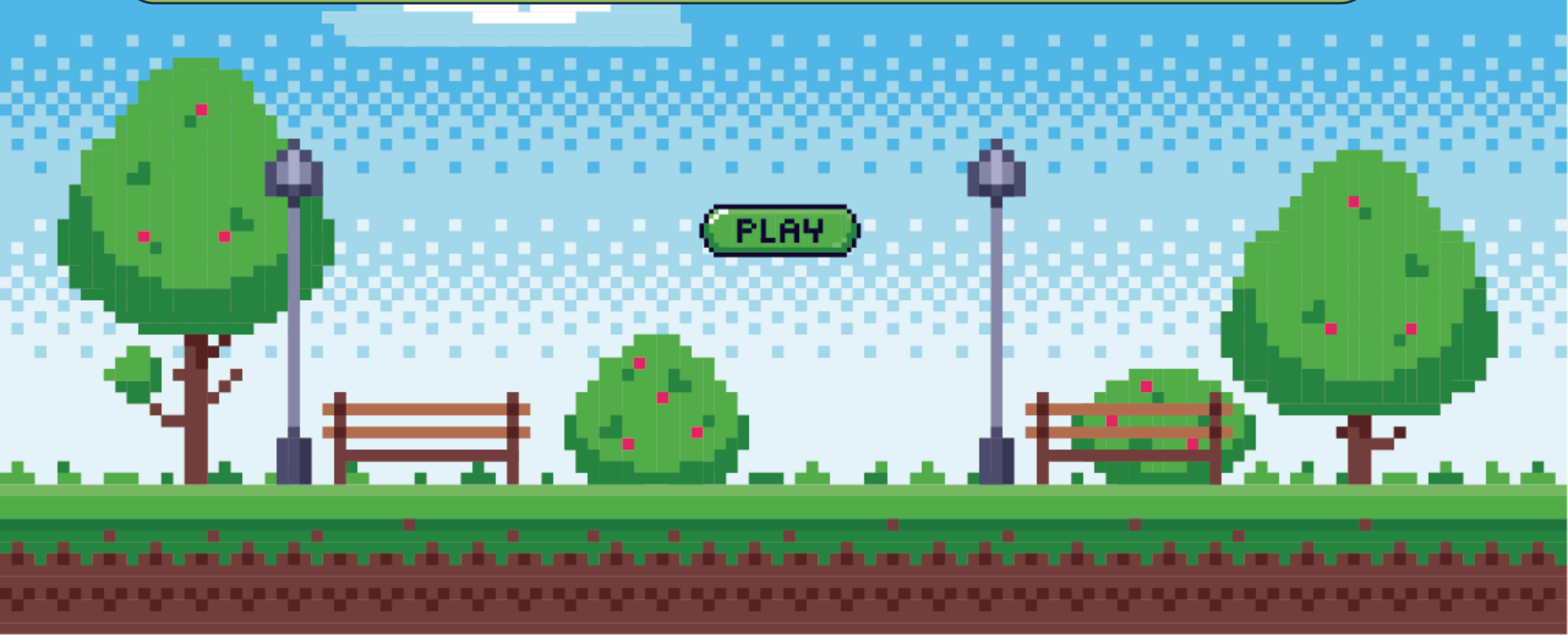
SQL Saturday 1082 Portland/Vancouver

November 2nd 2024

Demystifying Delta Parquet

The Foundation of the Lakehouse

Jarid McKenzie
Analytics Architect




Microsoft
AMD
PURESTORAGE®
redgate
rev.io
three tree tech
OMNIDATA
NORTH BANK INNOVATIONS
INTERJECT
SQL Saturday
2024
Sponsors

Jarid McKenzie

Analytics Architect

SCAN ME

LinkedIn: jarid-mckenzie

Foundatum

GitHub (Foundatum)

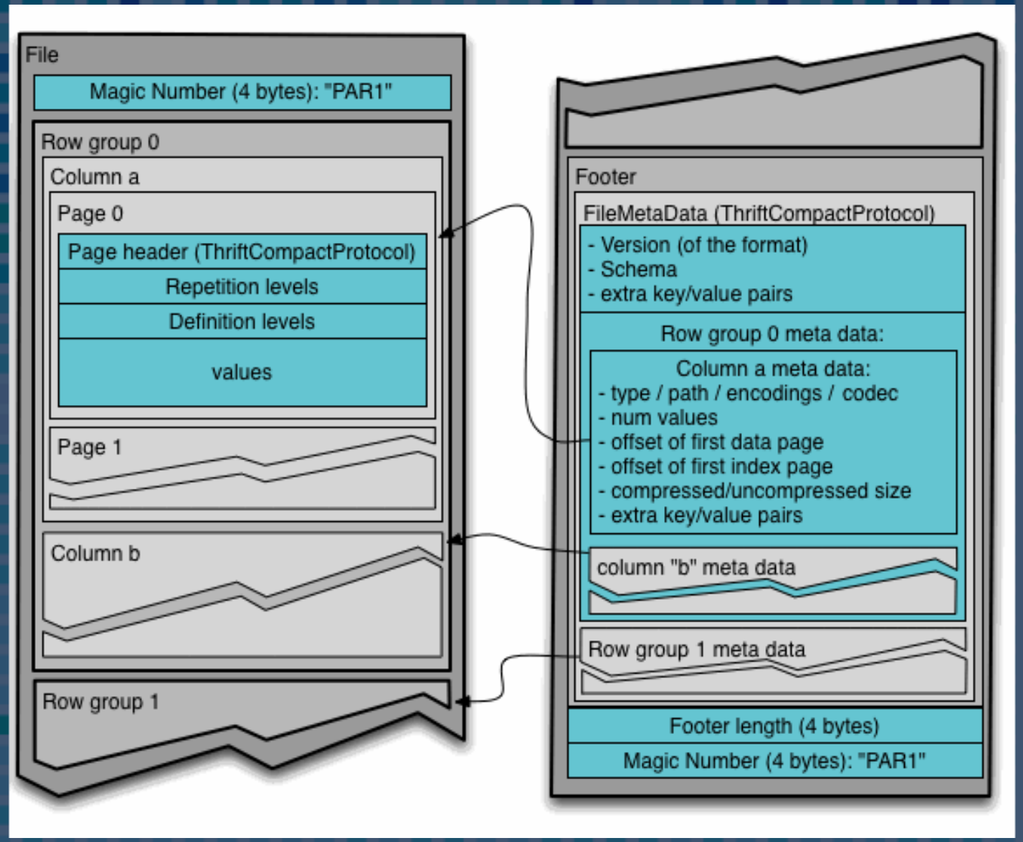
- Lead Analytics Architect
- Post Secondary Instructor
- Nerd

Parquet

The untranslatable texts

Parquet is built from the ground up with complex nested data structures in mind,and uses the record shredding and assembly algorithm described in the Dremel paper.

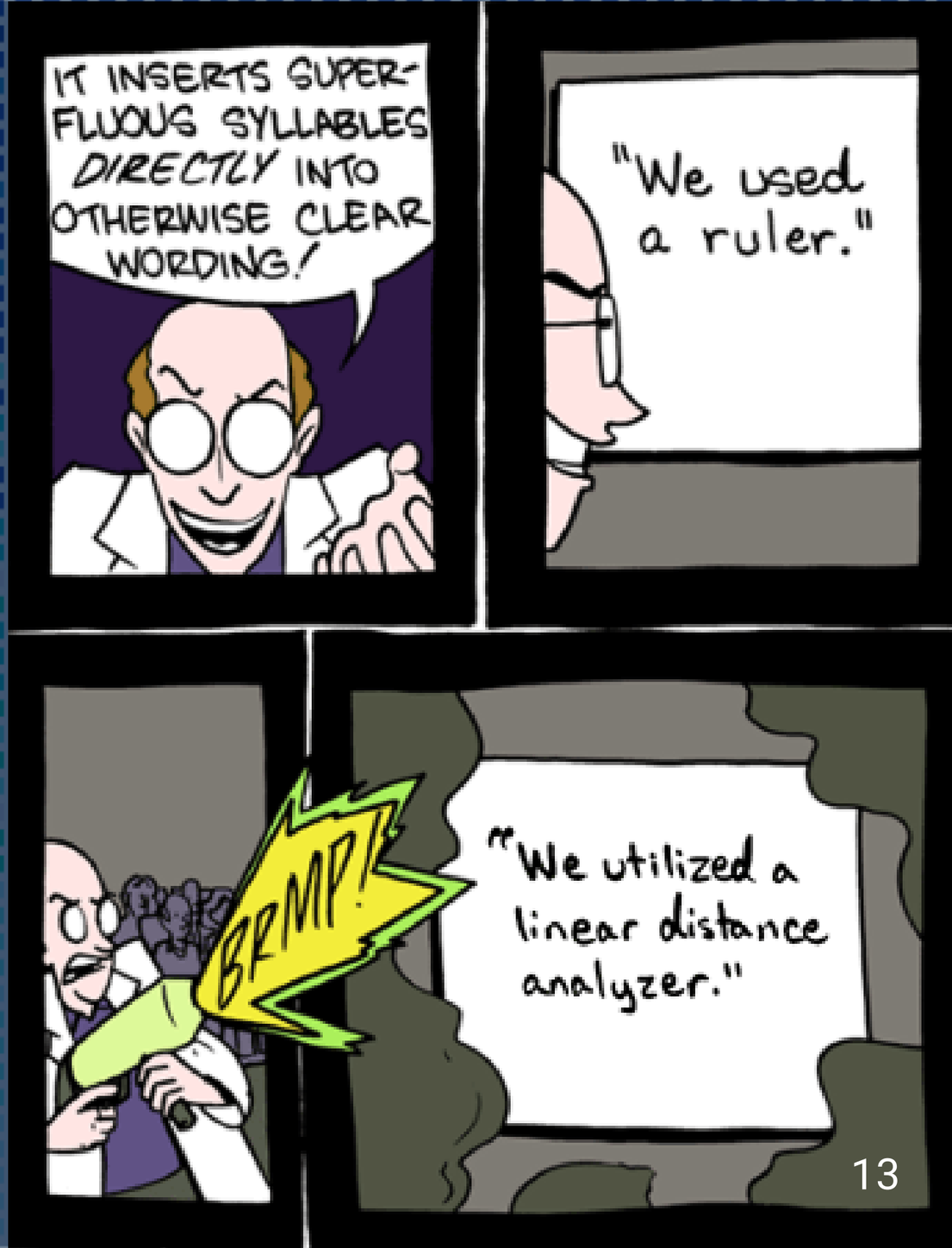
We believe this approach is superior to simple flattening of nested name spaces.



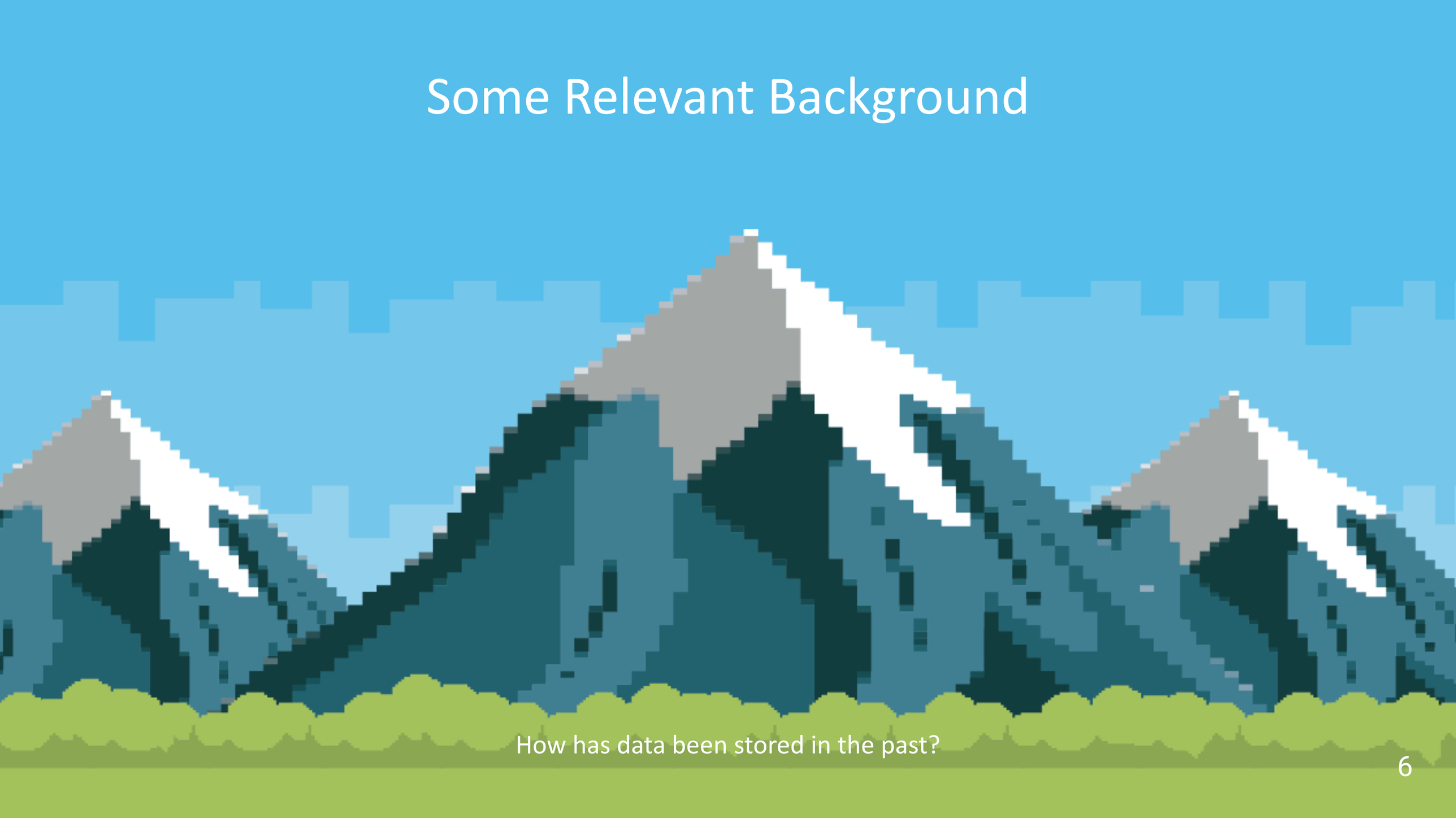
Sesquipedalian

Loquaciousness

"Smart" people, using long words to prove their "intelligence"

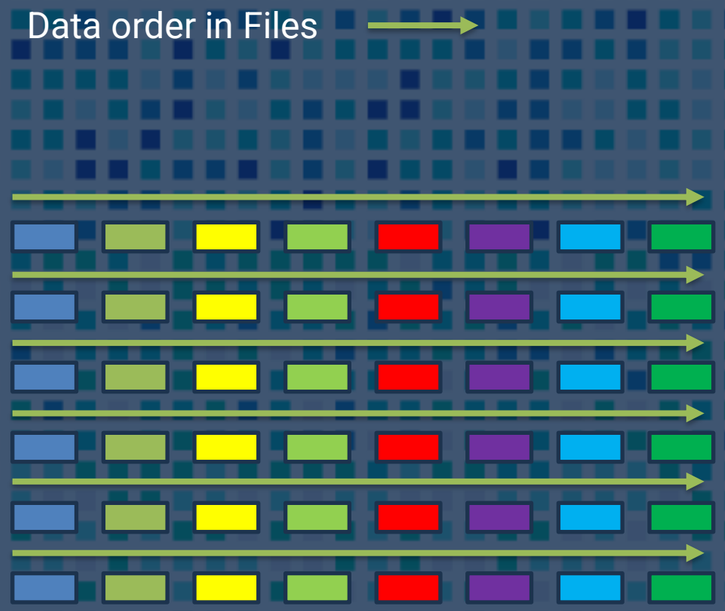


Some Relevant Background



How has data been stored in the past?

Row Order



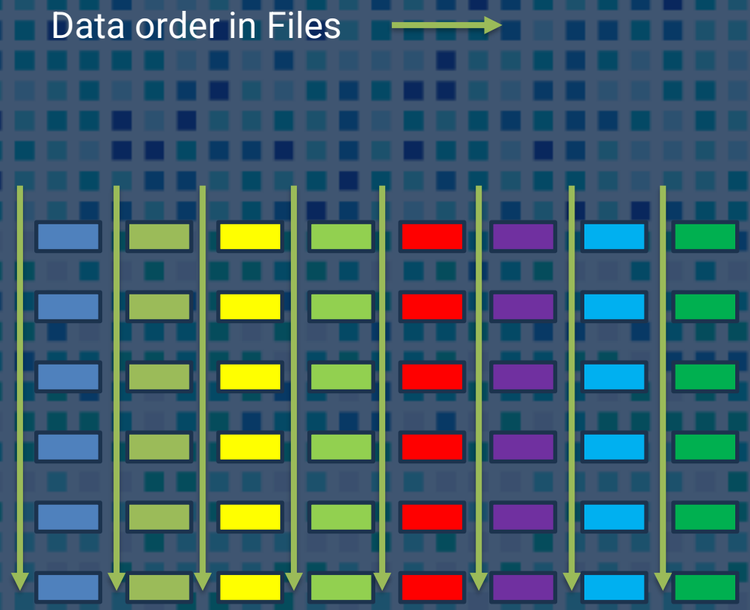
Where is this found?

- OLTP Databases

What is the goal?

- CRUD operations on related data
- Significant number of writers
- Small transactions

Row Order



Where is this found?

- OLAP databases (Analysis Services, Power BI)

What is the goal?

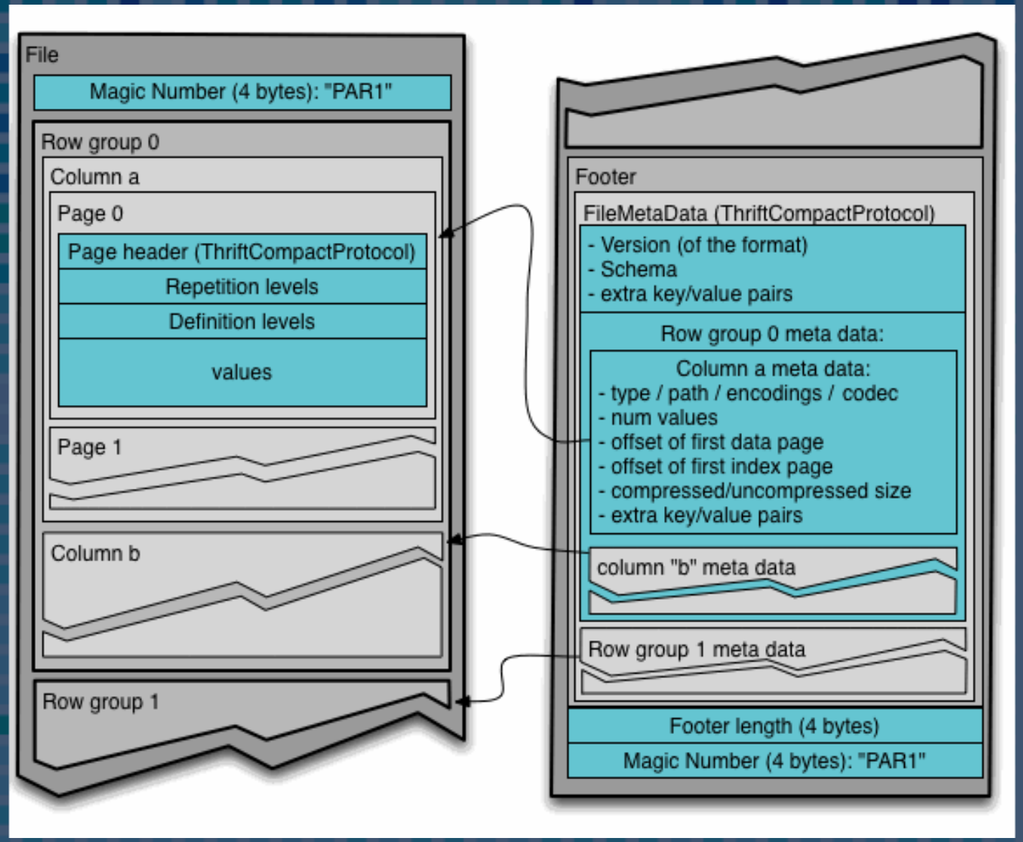
- Data Compression
- Fast Filtering and Aggregation
- Large Transactions

Parquet

The untranslatable texts

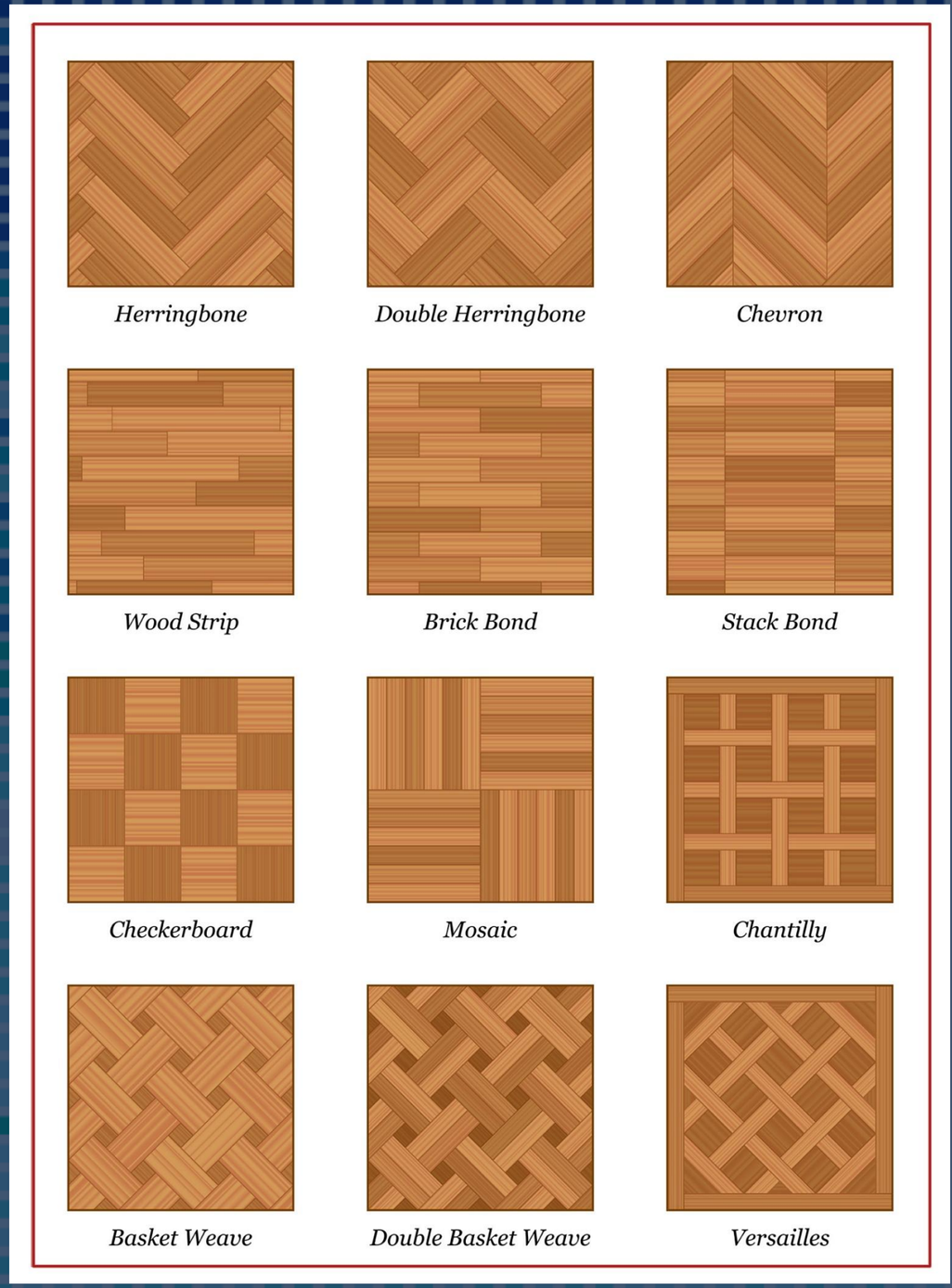
Parquet is built from the ground up with complex nested data structures in mind,and uses the record shredding and assembly algorithm described in the Dremel paper.

We believe this approach is superior to simple flattening of nested name spaces.



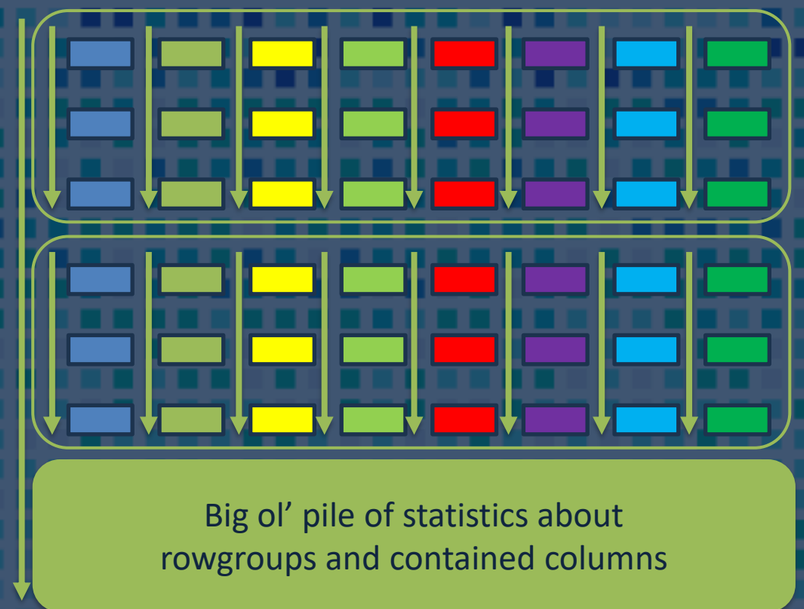
Parquet
Files

1. Row Groups are created (default size of 1Gb)
2. Each column within the row groups are created as a columnar format
3. The columns are written as data pages (8Kb)
4. Metadata and statistics are created and written as a footer to the file
5. Files are meant to be immutable



Parquet Order

Data order in Files →



Where is this found?

- It's the definition of the file format

What is the goal?

- Nested Data Structures
- Immutability
- Concurrent Reads
- Data Compression
- Fast Filtering and Aggregation



Row Order

ID, phone, name, postalZip, country
1,(643) 571-2303,Denton Thompson,Y6K 8Y5,New Zealand
2,1-448-338-9384,Troy Flores,A8N 1L8,New Zealand
3,1-860-372-7484,Kameko Dejesus,67P 8R2,Nigeria
4,1-516-879-7836,Fiona Silva,75T 3T2,Peru
5,1-181-836-3494,Velma Suarez,R5T 8B4,Sweden
6,1-748-245-1827,Martina Orr,H7L 7B0,Poland

Column Order

ID 1,2,3,4,5,6

Phone (643) 571-2303, 1-448-338-9384, 1-860-372-7484, 1-516-879-7836, 1-181-836-3494, 1-748-245-1827

Name Denton Thompson, Troy Flores, Kameko Dejesus, Fiona Silva, Velma Suarez, Martina Orr

PostalZip Y6K 8Y5, A8N 1L8, 67P 8R2, 75T 3T2, R5T 8B4, H7L 7B0

Country New Zealand, New Zealand, Nigeria, Peru, Sweden, Poland

Parquet

ID 1,2,3
Phone (643) 571-2303, 1-448-338-9384, 1-860-372-7484
Name Denton Thompson, Troy Flores, Kameko Dejesus
PostalZip Y6K 8Y5, A8N 1L8, 67P 8R2
Country New Zealand, New Zealand, Nigeria

ID 4,5,6
Phone 1-516-879-7836, 1-181-836-3494, 1-748-245-1827
Name Fiona Silva, Velma Suarez, Martina Orr
PostalZip 75T 3T2, R5T 8B4, H7L 7B0
Country Peru, Sweden, Poland

Statistics

Data Types

Primitive Types:

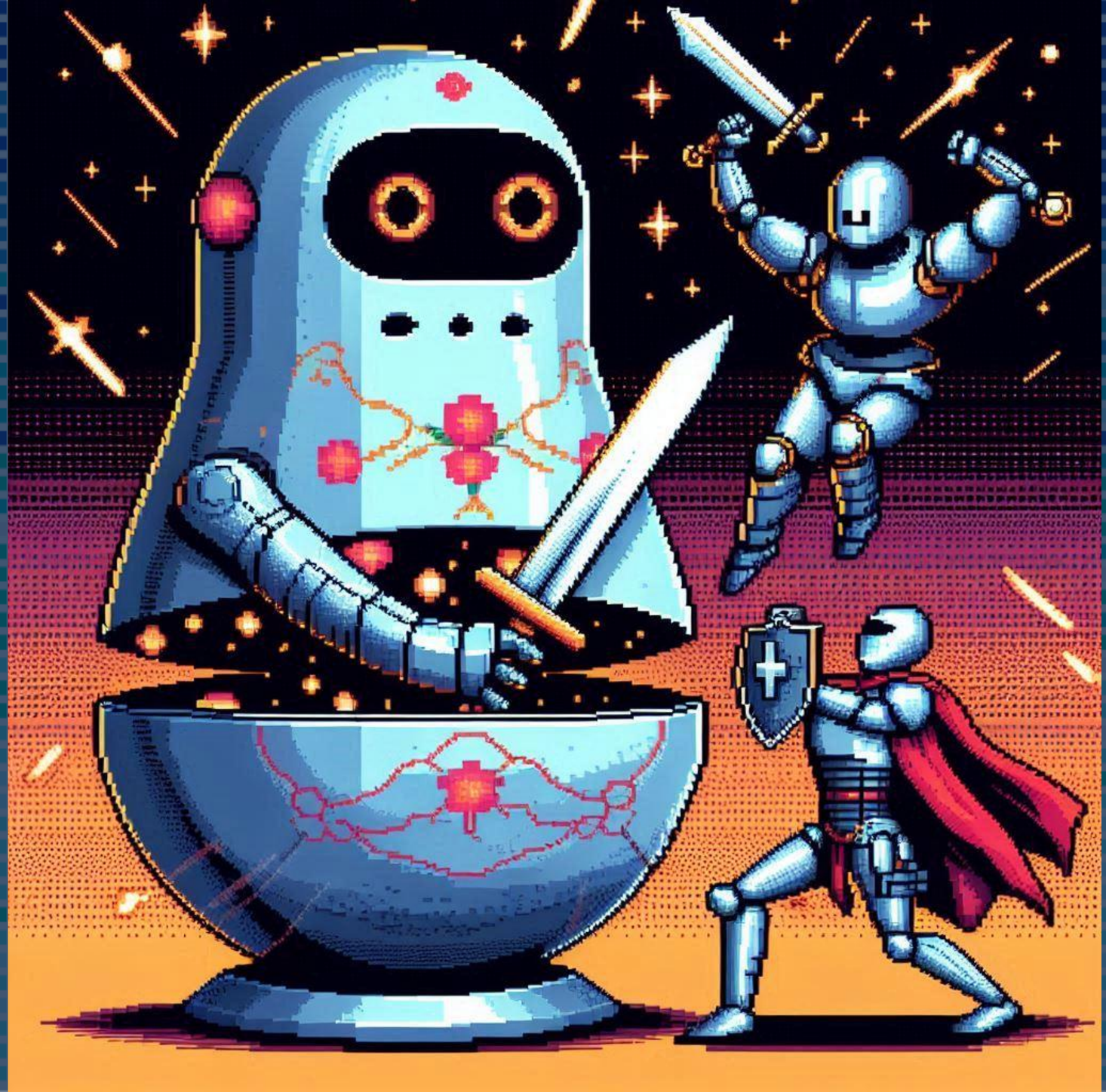
- BOOLEAN: 1 bit boolean
- INT32: 32 bit signed ints
- INT64: 64 bit signed ints
- INT96: 96 bit signed ints
- FLOAT: IEEE 32-bit floating point values
- DOUBLE: IEEE 64-bit floating point values
- BYTE_ARRAY: arbitrarily long byte arrays
- FIXED_LEN_BYTE_ARRAY: fixed length byte arrays

Logical Types:

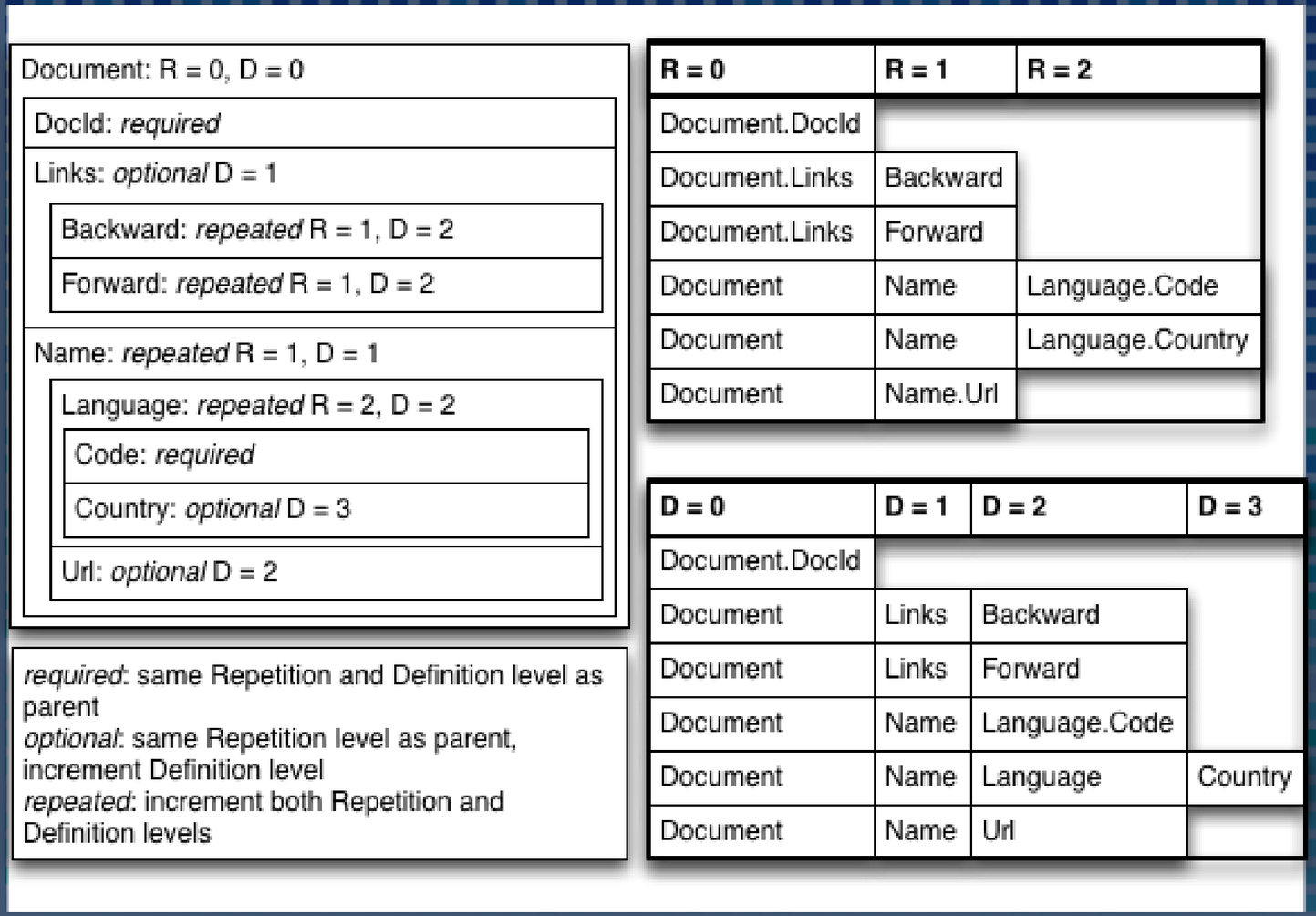
Extend primitive types by specifying how the primitive types should be interpreted.

Strings are stored as a byte array with UTF8 annotation.

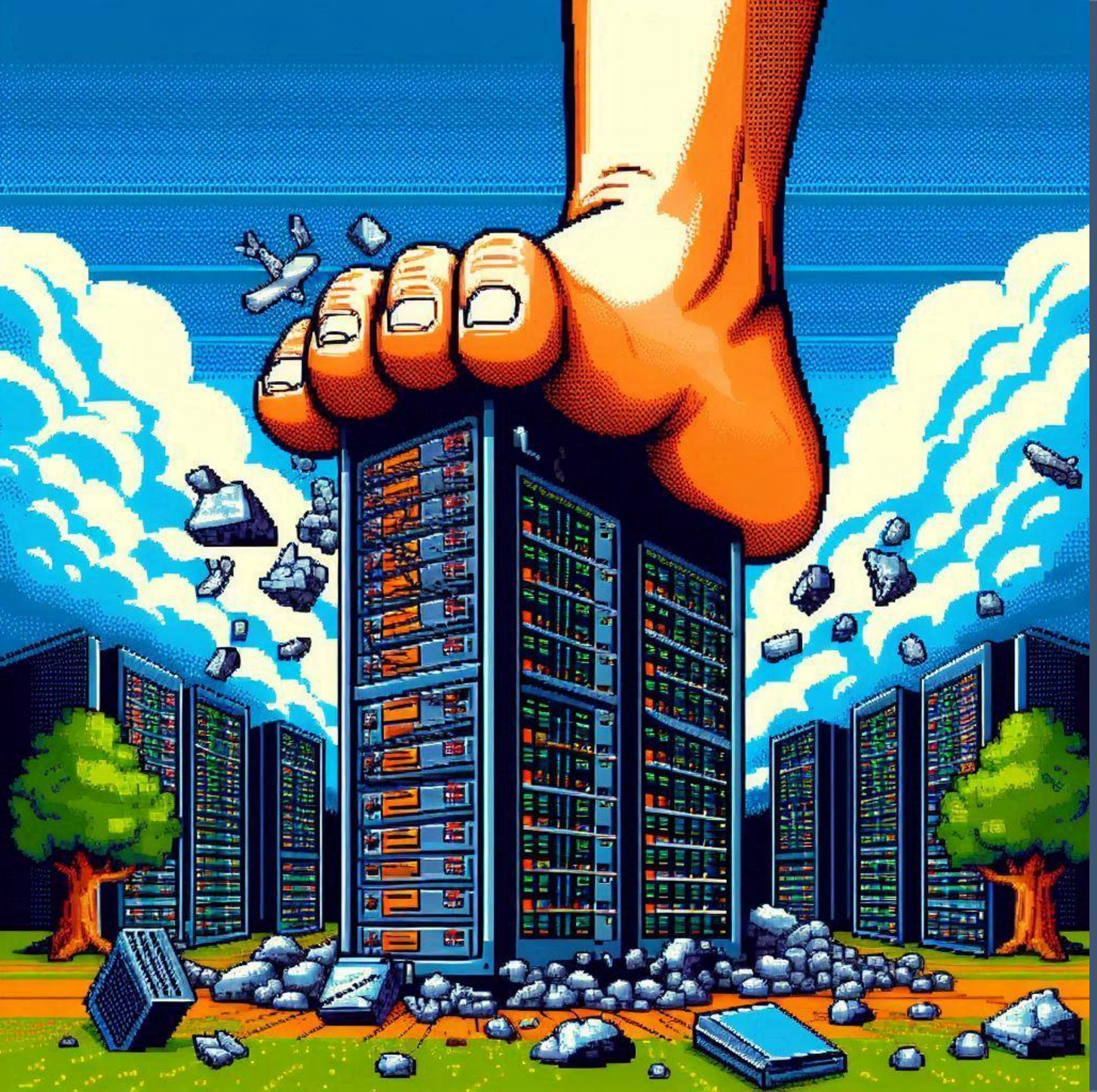
Nested Data



Nested Data



The striping and assembly algorithms from the Dremel paper · julienledem/redelm Wiki · GitHub

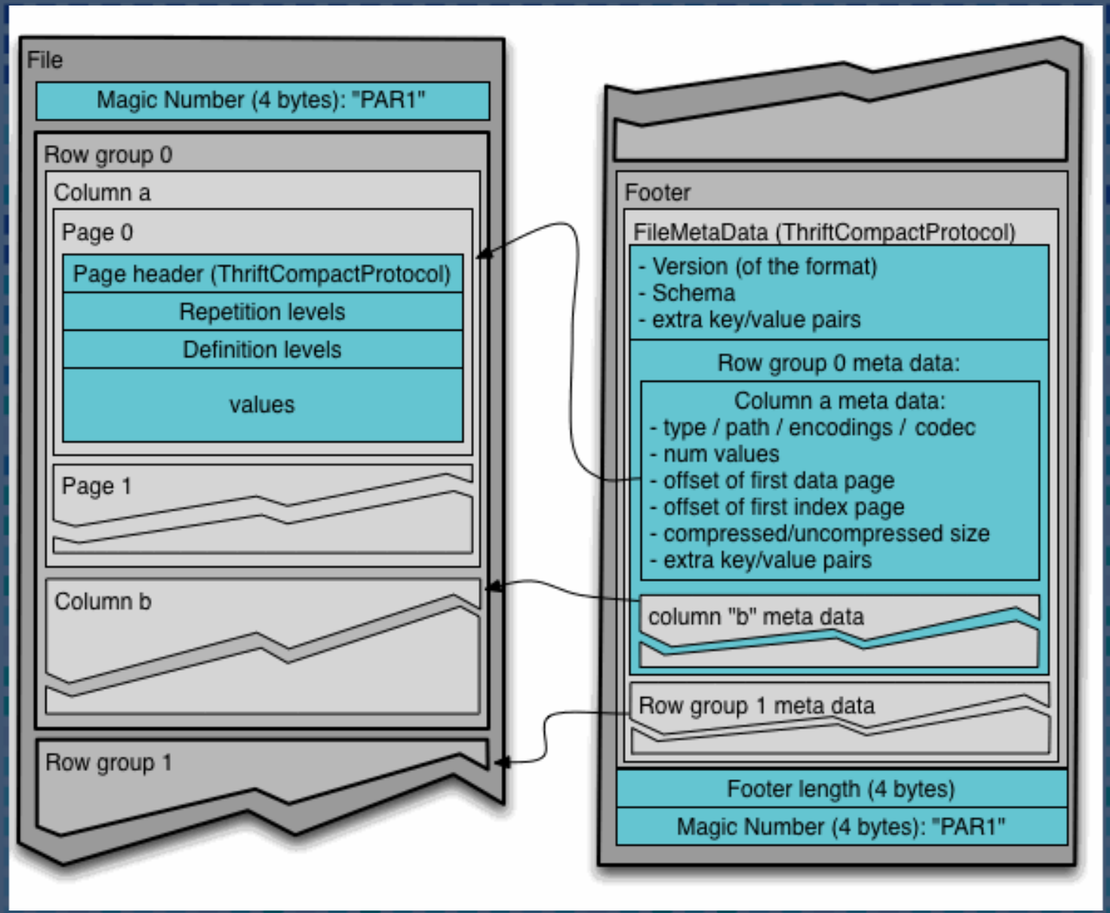


Compression & Encoding

Compression

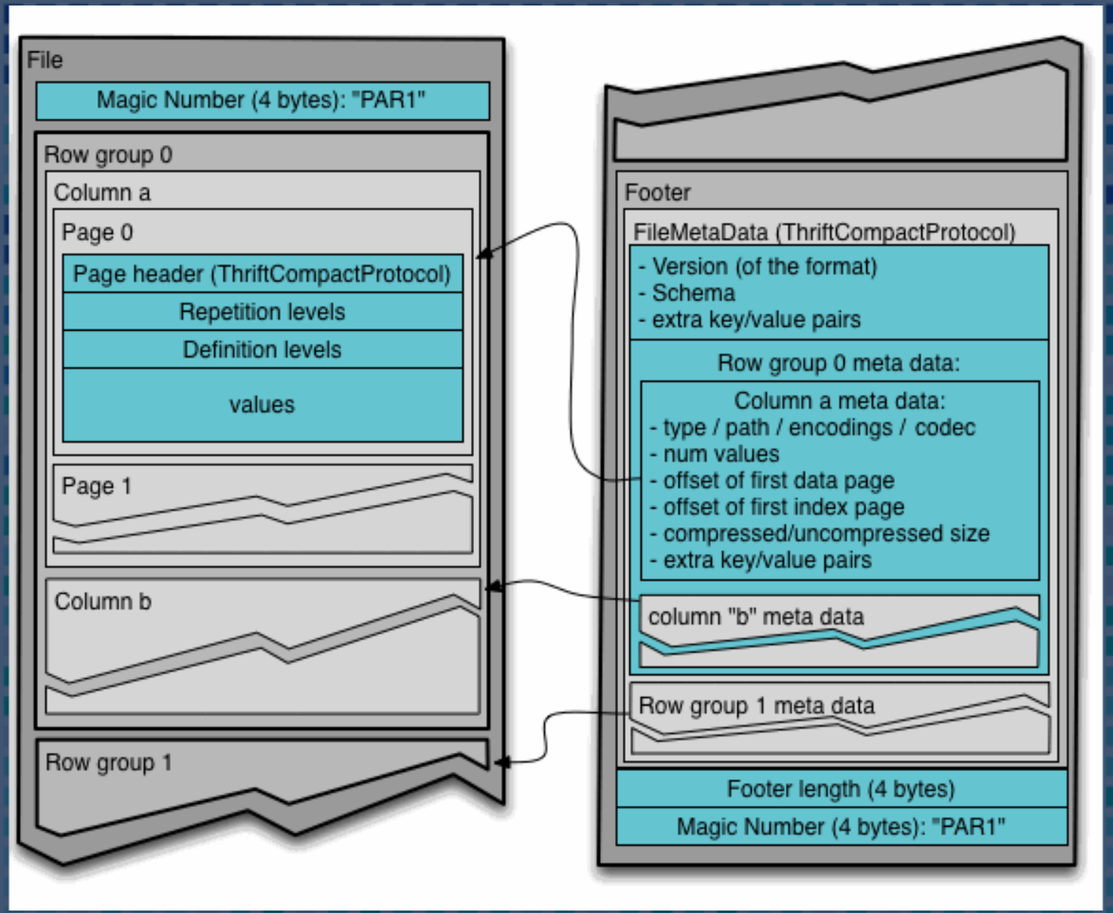
Multiple Compression Schemes

- SNAPPY, GZIP, LZO, BROTLI, ZSTD, LZ4_RAW
- Can compress either dictionary pages or data pages
- Compressed/Uncompressed Size is stored in the column meta-data



Encoding

Encodings
- Plain
- Dictionary Encoding (Plain and RLE)
- Run Length Encoding
- Bit Packing
- Delta Encoding
- A few other esoteric ones (Link)



Dictionary Encoding

Replacing datatypes with dictionary and indexes

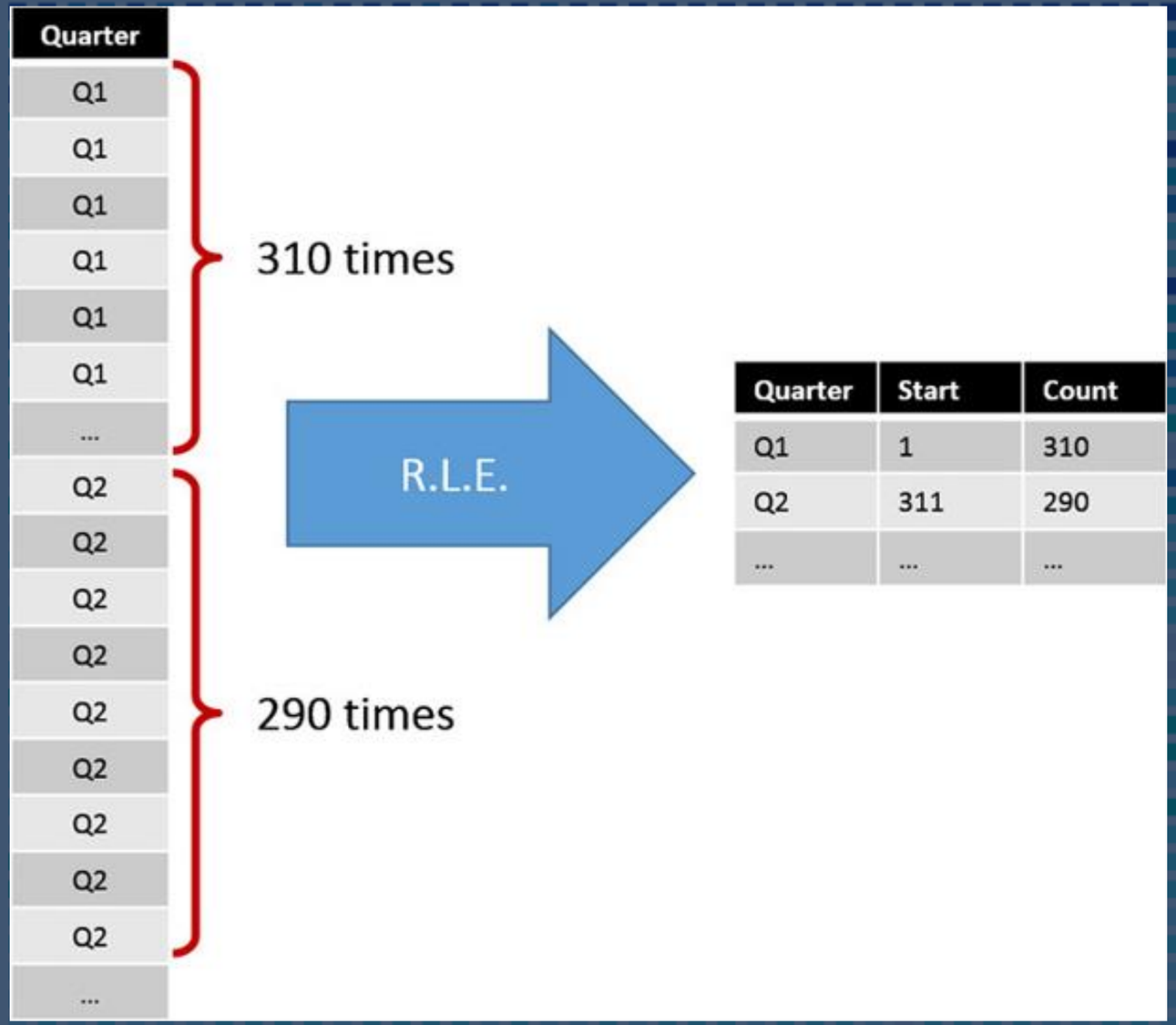
Color
Red
Red
White
Black
Blue
Red
Blue
Black
Black

Dictionary Encoding

Color Id
0
0
1
2
3
0
3
2
2

Id	Color
0	Red
1	White
2	Black
3	Blue

Dictionary Encoding



Dictionary Encoding

The numbers 1 through 7 using bit width 3:

```
dec value: 0   1   2   3   4   5   6   7
bit value: 000 001 010 011 100 101 110 111
bit label: ABC DEF GHI JKL MNO PQR STU VWX
```

would be encoded like this where spaces mark byte boundaries (3 bytes):

```
bit value: 10001000 11000110 11111010
bit label: HIDEFABC RMNOJKLG VWXSTUPQ
```

Dictionary Encoding

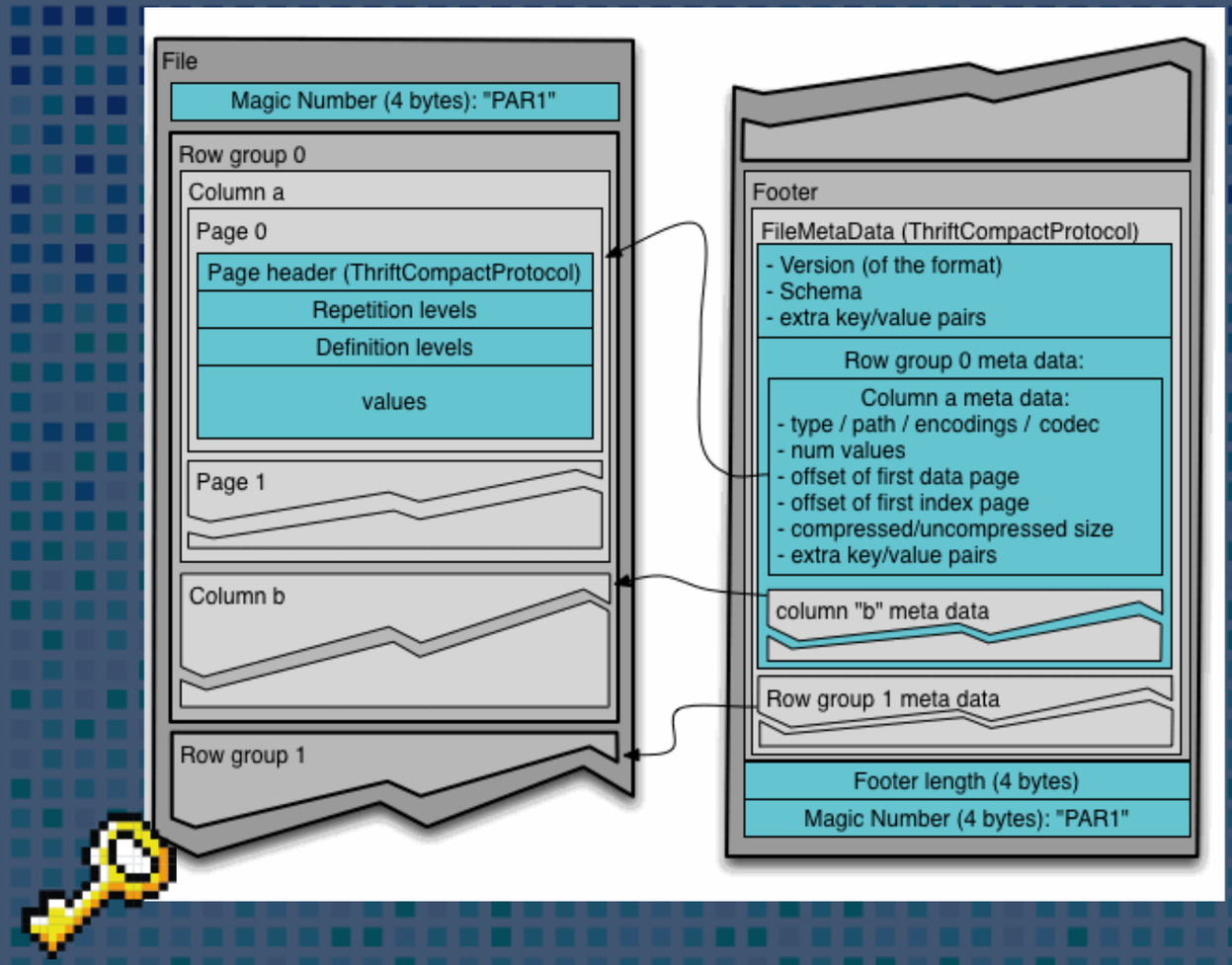
	Unix Timestamps	Deltas	MinDelta	RelativeDelta			
	1729610543034		3419				
	1729610547098	4064		645			
	1729610550992	3894		475			
	1729610555003	4011		592			
	1729610559054	4051		632			
	1729610562555	3501		82			
	1729610566347	3792		373			
	1729610569766	3419		0			
	header						
	block size	miniblock count	Value Count	First Value			
	8	1	8	1729610543034			
	Miniblock						
	Min Delta	bitwidth	Relative Deltas packed on 10 bit				
	3419	10	1010000101111011011100101000010011110000001010010010111010100000000000				

Parquet Footer

What is this section all about?

Page Index

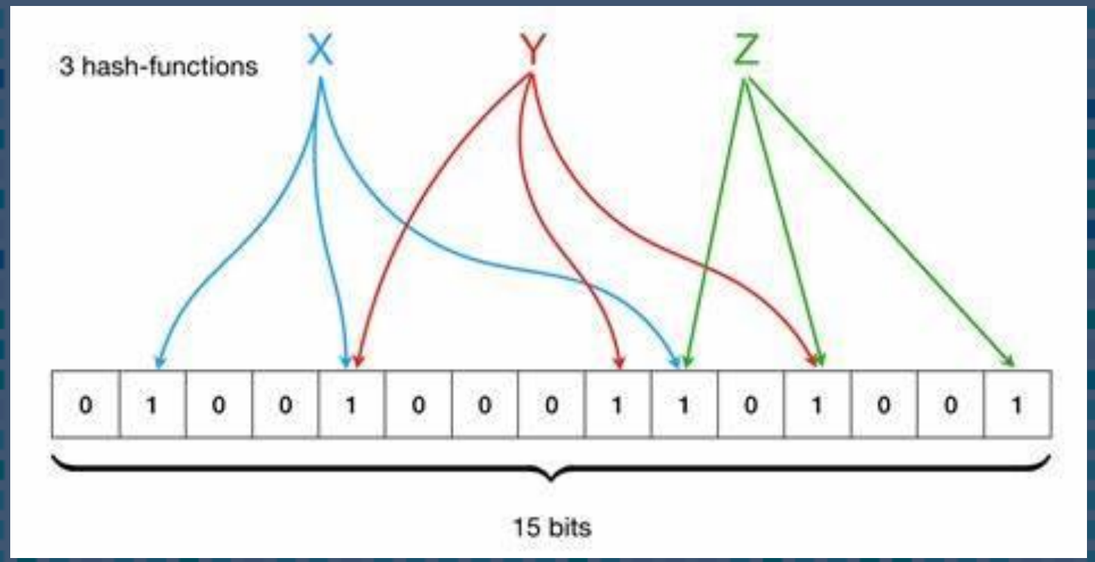
Stores offsets to specific locations within the file and includes meta data like the min and max values of the column / data page



[Page Index | Parquet (apache.org)]

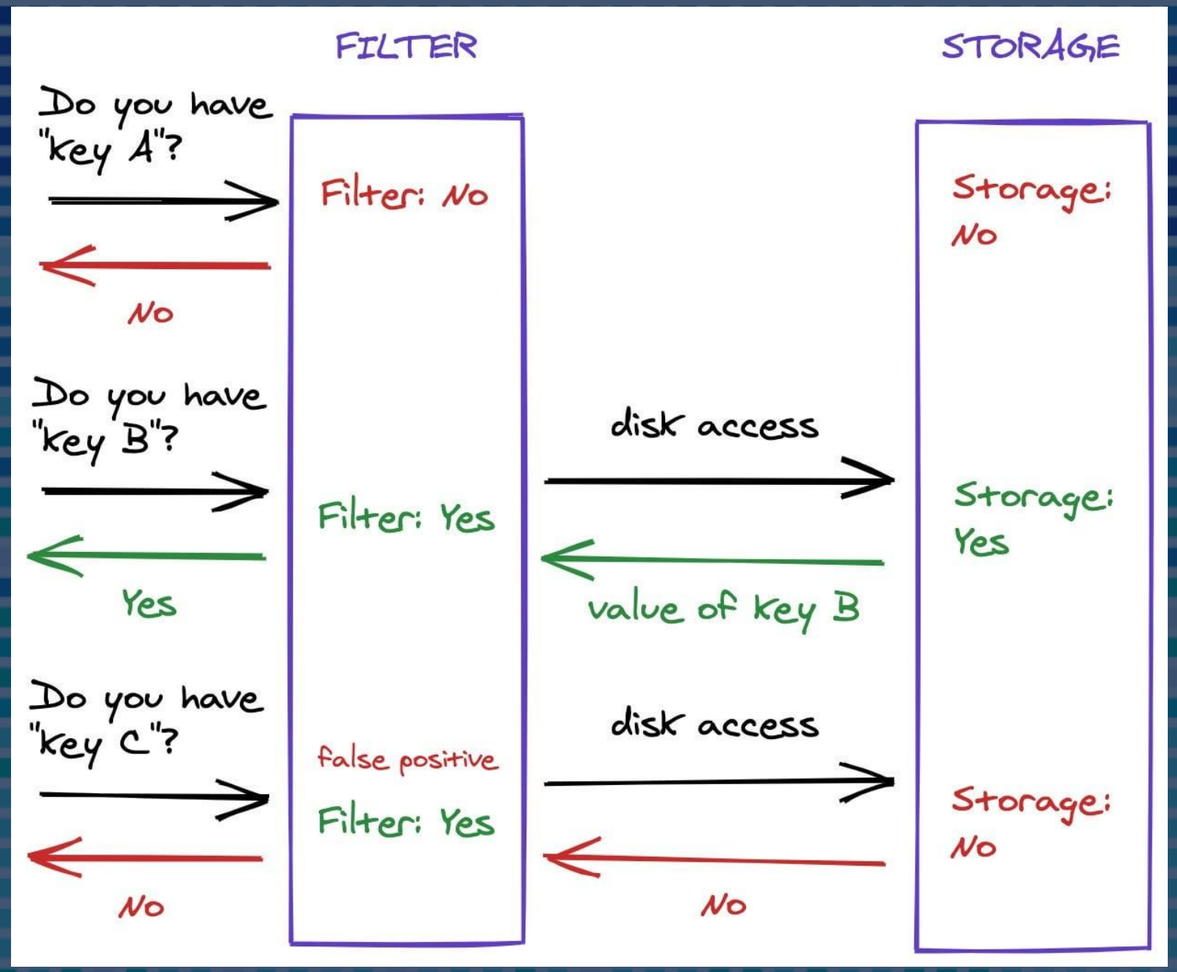
Bloom Filter

A Bloom filter is a compact data structure that overapproximates a set. It **can respond to membership queries with either “definitely no” or “probably yes”**, where the probability of false positives is configured when the filter is initialized. Bloom filters do not have false negatives.



Bloom Filter

A Bloom filter is a compact data structure that overapproximates a set. It **can respond to membership queries with either “definitely no” or “probably yes”**, where the probability of false positives is configured when the filter is initialized. Bloom filters do not have false negatives.



Delta, on top of Parquet

ACID

1. **Atomicity**: Ensures that all operations within a transaction are completed successfully. If any part of the transaction fails, the entire transaction is rolled back, leaving the database in its previous state.
2. **Consistency**: Guarantees that a transaction will bring the database from one valid state to another, maintaining database rules and constraints.
3. **Isolation**: Ensures that transactions are executed independently of one another. Intermediate states of a transaction are invisible to other transactions, preventing them from accessing data being modified by another transaction.
4. **Durability**: Once a transaction has been committed, it remains so, even in the event of a system failure. This ensures that the results of the transaction are permanently recorded in the database.

Pessimistic Concurrency

- **Assumption**: Assumes that conflicts between transactions are common.
- **Mechanism**: Locks resources at the start of a transaction to prevent other transactions from accessing them.
- **Conflict Resolution**: Conflicts are avoided by ensuring that only one transaction can access the resource at a time.
- **Performance**: Can be slower due to the overhead of acquiring and releasing locks, but ensures data consistency.
- **Use Cases**: Ideal for write-heavy applications or scenarios where data integrity is critical and conflicts are likely.

Optimistic Concurrency

- **Assumption**: Assumes that conflicts between transactions are rare.
- **Mechanism**: Transactions proceed without locking resources. Before committing, the system checks if any conflicts occurred.
- **Conflict Resolution**: If a conflict is detected, the transaction is rolled back and retried.
- **Performance**: Generally better for systems with low contention, as it avoids the overhead of locking.
- **Use Cases**: Suitable for read-heavy applications where write conflicts are infrequent.


GAME OVER

Credits

Presentation template
by Slides for Education

Illustrations and
infographics by Freepik
Photos created by Unsplash