Data Management in Large-Scale Distributed Systems

Column-oriented Storage

Thomas Ropars

thomas.ropars@univ-grenoble-alpes.fr

http://tropars.github.io/

2021

References

- Designing Data-Intensive Applications by Martin Kleppmann
  - Chapters 2: *Column-oriented storage*
  - Chapter 4: *Formats for Encoding Data*

- Presentation of Prof Michael Stonebraker @EPFL (`https://slideshot.epfl.ch/play/suri_stonebraker`): *Everything You Learned in Your DBMS Class is Wrong*

In this lecture

- OLTP vs OLAP
  - Data warehouse
  - ETL
- Representation of large data on disks
  - Data that will be queried for analysis
- Column-oriented storage
  - Column-oriented file formats

Agenda

Storage for data analytics

Column-oriented storage

File formats

OLTP vs OLAP

Online transaction processing (OLTP)

- Interactive applications
  - Interaction with end users
- Read/update a small number of records (transactions)
- Requires high availability and low latency

Online analytic processing (OLAP)

- Scan over a large number of records
- Compute statistics
  - Used internally

Data warehouse

Running data analytics on OLTP databases

Problems ?

Data warehouse

Running data analytics on OLTP databases

Problems ?

- Performance
  - Disturb the OLTP requests
  - Not adapted to serve OLAP requests
- Complexity
  - An entreprise might have multiple OLTP systems

Data warehouse

Running data analytics on OLTP databases

Problems ?

- Performance
  - Disturb the OLTP requests
  - Not adapted to serve OLAP requests
- Complexity
  - An entreprise might have multiple OLTP systems

Data warehouse

- DB dedicated to data analysis
- Contains a read-only copy of the data of all the OLTP systems in the company

ETL

Inserting data into a data warehouse

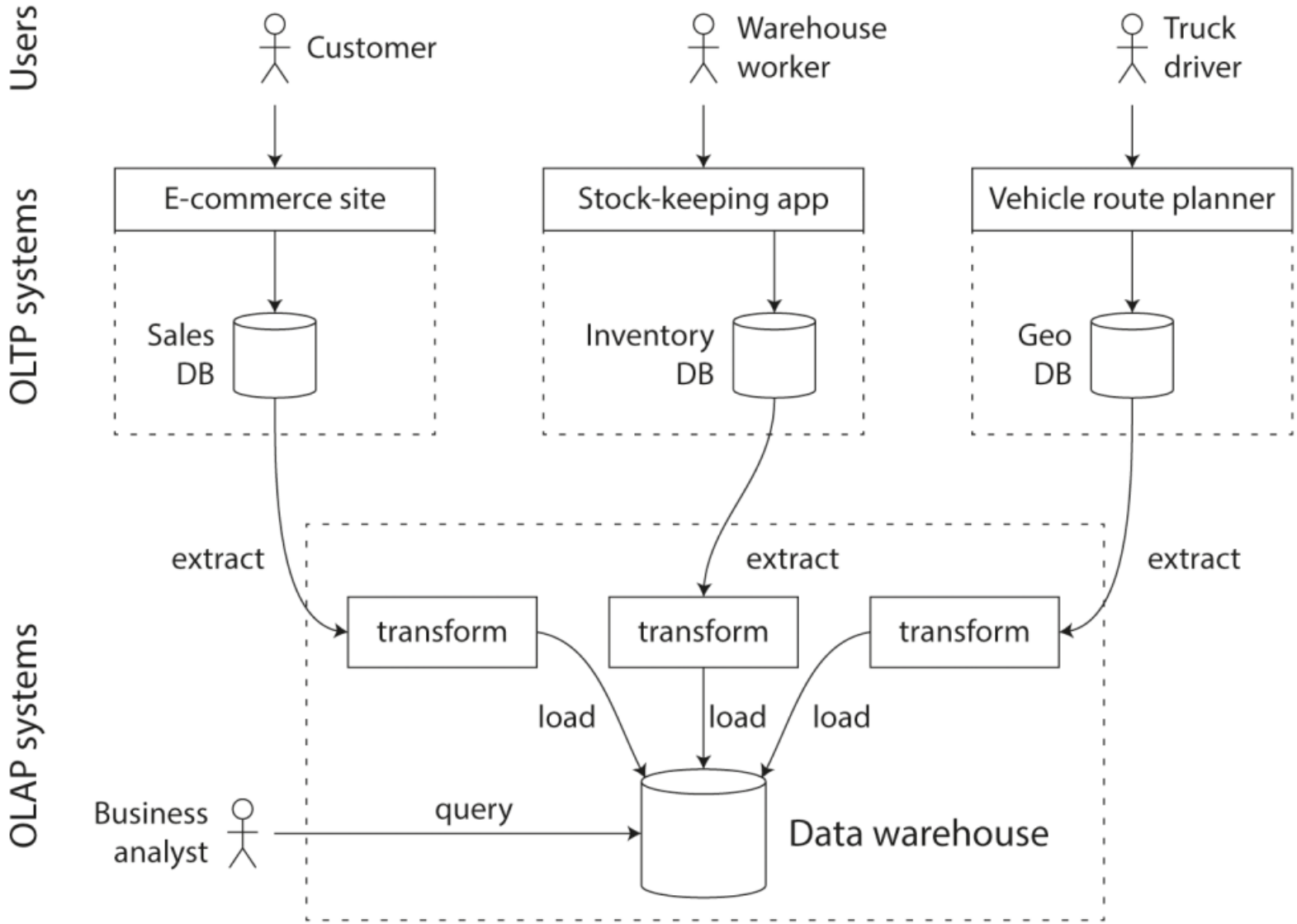
- Data transfers from the OLTP systems
  - Periodic data dumps
  - Continuous stream of updates
- Stores an history of the events over time

Extract-Transform-Load (ETL)

- Process of getting data into the warehouse
  - Extract data from OLTP databases
  - Clean up and transform data into an analysis-friendly schema
  - Load into the data warehouse

ETL process

Figure by M. Kleppmann



Summary of the comparison between OLAP and OLTP

Source: M. Kleppmann

Property	Transaction processing systems (OLTP)	Analytic systems (OLAP)
Main read pattern	Small number of records per query, fetched by key	Aggregate over large number of records
Main write pattern	Random-access, low-latency writes from user input	Bulk import (ETL) or event stream
Primarily used by	End user/customer, via web application	Internal analyst, for decision support
What data represents	Latest state of data (current point in time)	History of events that happened over time
Dataset size	Gigabytes to terabytes	Terabytes to petabytes

Agenda

Storage for data analytics

Column-oriented storage

File formats

Storing data on disks

- The representation of data on disks is in general not the same as in memory
  - Storing a pointer on disk would be meaningless
  - Random accesses on disk can be very slow

Challenges (for OLAP)

- Try to have a compact representation of the data
  - Organize the data so that they can be efficiently compressed
- Optimize the performance of read operations
  - Write once, read many

Row-oriented storage

Most OLTP systems store data in a row-oriented fashion

- All the values from one row of a table are stored next to each other

Limitations

Row-oriented storage

Most OLTP systems store data in a row-oriented fashion

- All the values from one row of a table are stored next to each other

Limitations

- Inefficient data compression
  - Data of different types are next to each other
- Inefficient read operations
  - We are often only interested in a few entries in a row
    - But we have to read the full row
  - We may want to filter elements based on a condition on one entry
    - But we have to read all the rows

Column-oriented storage

Description

- Stores all the values from each column together
  - Each column in a different file
  - Rows in the same order in each file
- Efficient compression
  - Values in one column are of the same type (e.g., integers)
  - The number of distinct values in a column is often small (not the case for rows)
- Oportunities for optimizations on read

Optimizations for read operations

Limiting the amount of data read

- Context: A request that selects a subset of columns
- Solution with columnar storage:
  - We can read only the files corresponding to these columns

Filtering based on a condition

- Context: We are interested in items corresponding to a condition
  - `SELECT * FROM Customers WHERE Country='Mexico'`
- Solution with columnar storage
  - Check the condition by reading only the corresponding column
  - Store a summary (min, max, etc) of the column at the beginning of each partition to fully skip reading when possible

Write operations

Sorting rows

- Can help improving read operations
- Based on the filtering criteria that is used the more often

How to insert new data?

- Updates *in place* would be tricky and slow
  - Columns are compressed
  - We need to keep the ordering of rows
- Use a LSM-tree approach
  - Accumulate data in memory
  - Write in bulk to the storage

Agenda

Storage for data analytics

Column-oriented storage

File formats

Storing data on disks

- Many file formats exist in the context of *Big Data*
  - CSV
  - JSON
  - Avro
  - Parquet
  - ORC
  - etc.
- Several formats are supported by:
  - Distributed file systems (eg, HDFS)
  - NoSQL databases (eg, MongoDB)
  - Data processing engines (eg, Spark)

What are the properties of each file format? Which one to choose?

Textual formats

Examples of such formats

- CSV
- JSON
- XML

Advantages

- Readable by humans

Drawbacks

- High storage footprint
- Very low read performance

Textual formats

CSV

- Comma Separated Values
- Good for storing data organized as a single table
  - The name of the columns is given by the first row (not verbose)
- No hierarchical structure

JSON – XML

- Support for hierarchical structures
- Very verbose (large footprint)

Binary encoding formats

Examples

- Avro (Hadoop)
- Thrift (Facebook)
- Protocol Buffers (Google)

Idea

- Describe the data using a schema
- Pack all fields describing an item (a row) in a binary format

Advantages

- Can lead to huge space reduction

Example

By M. Kleppmann

```
{
  "userName": "Martin",
  "favoriteNumber": 1337,
  "interests": ["daydreaming", "hacking"]
}
```

Figure: A JSON document

Storage space

- If stored as JSON text file: 81 bytes
- If stored as simple binary JSON encoding (not using a schema): 66 bytes
  - Space saved on the representation of numbers and on structure information

With Avro

```
{
  "type": "record",
  "name": "Person",
  "fields": [
    {"name": "userName",
      "type": "string"},
    {"name": "favoriteNumber", "type": ["null", "long"], "default": null},
    {"name": "interests",
      "type": {"type": "array", "items": "string"}}
  ]
}
```

Figure: Definition of the schema

With Avro

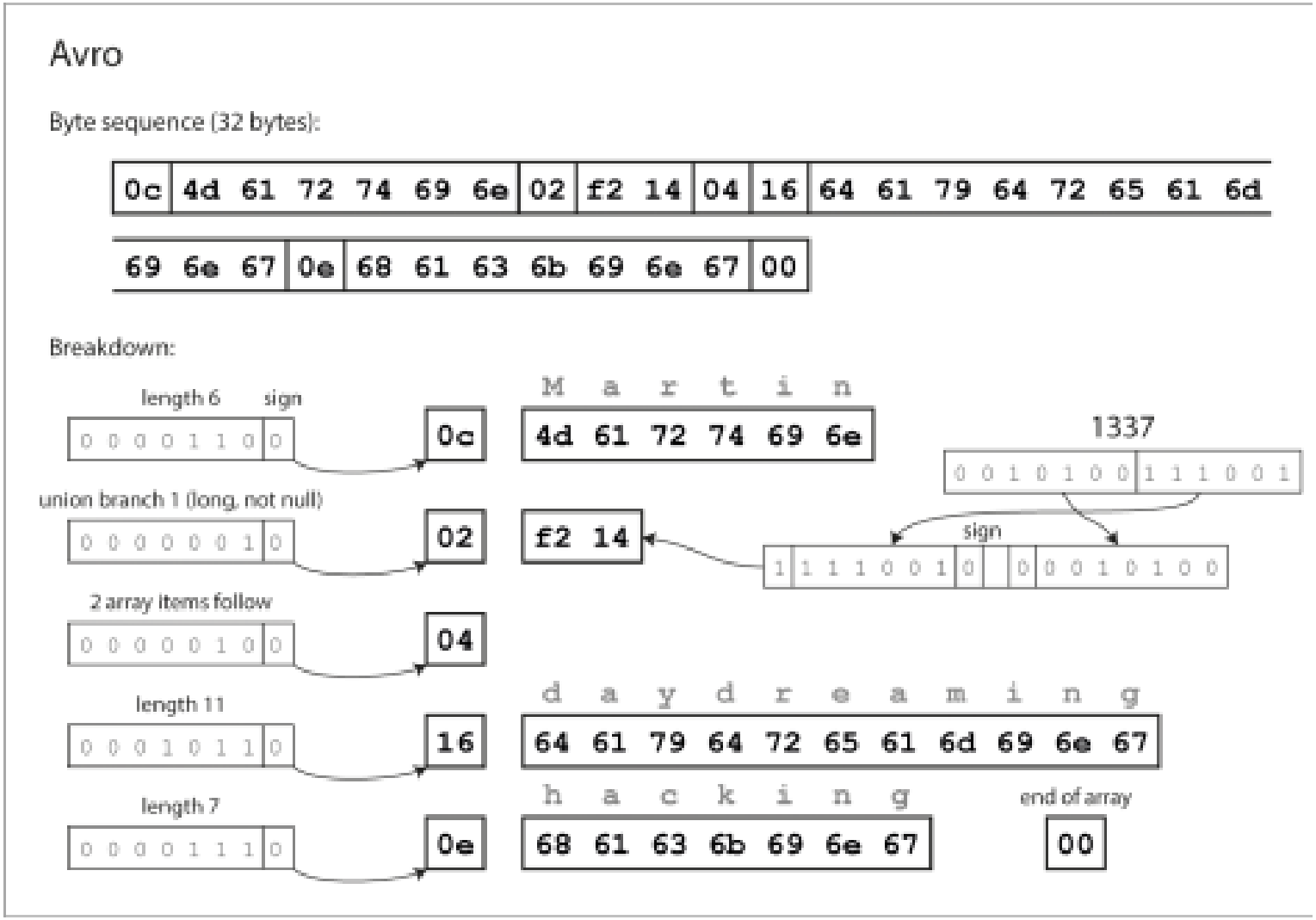


Figure: Binary representation of the item (32 bytes)

Column-oriented formats

All formats described until now are row-oriented

- All the values from one row of a table are stored next to each other
- Column-oriented file formats adopt the column-oriented approach

Examples of column-oriented formats

- Parquet (Twitter + Cloudera)
- ORC (Hadoop)

Parquet data layout

- Data are stored in files
- A file consists of one or more row groups
  - A set of rows
- A row group contains exactly one column chunk per column
  - A column chunk is contiguous in the file
- Metadata are stored at the end of the file
  - Position of each column chunk
  - Statistics about each chunk
    - Min/Max statistics for numbers
    - Dictionary filtering for other columns (as long as less than 40k different values)
- About sorting
  - Rows can be sorted based on a specific column

Example

source: https://blog.usejournal.com/sorting-and-parquet-3a382893cde5

Description of the data

- Customer table with one column being the country
- `SELECT * FROM Customers WHERE Country='Mexico'`
- Some numbers:
  - 10M rows
  - 10k rows per row group
  - 1% of the customers are from Mexico

Amount of data read to answer the query

- With a row-based format:
- With an unsorted parquet file:

- With a sorted parquet file:

Example

source: https://blog.usejournal.com/sorting-and-parquet-3a382893cde5

Description of the data

- Customer table with one column being the country
- `SELECT * FROM Customers WHERE Country='Mexico'`
- Some numbers:
  - 10M rows
  - 10k rows per row group
  - 1% of the customers are from Mexico

Amount of data read to answer the query

- With a row-based format: All data
- With an unsorted parquet file:

- With a sorted parquet file:

Example

source: https://blog.usejournal.com/sorting-and-parquet-3a382893cde5

Description of the data

- Customer table with one column being the country
- `SELECT * FROM Customers WHERE Country='Mexico'`
- Some numbers:
  - 10M rows
  - 10k rows per row group
  - 1% of the customers are from Mexico

Amount of data read to answer the query

- With a row-based format: All data
- With an unsorted parquet file:
  - Probability of a row group with no customer from Mexico: $0.99^{10000} = 2.25 \times 10^{-44}$
  - All row groups
- With a sorted parquet file:

Example

source: https://blog.usejournal.com/sorting-and-parquet-3a382893cde5

Description of the data

- Customer table with one column being the country
- `SELECT * FROM Customers WHERE Country='Mexico'`
- Some numbers:
  - 10M rows
  - 10k rows per row group
  - 1% of the customers are from Mexico

Amount of data read to answer the query

- With a row-based format: All data
- With an unsorted parquet file:
  - Probability of a row group with no customer from Mexico: $0.99^{10000} = 2.25 \times 10^{-44}$
  - All row groups
- With a sorted parquet file: 1% of the row groups (10)

Another example

source: https://www.linkedin.com/pulse/we-taking-only-half-advantage-columnar-file-format-eric-sun/

Description of the data

- A website log dataset
  - Information in one entry:
    `timestamp, user_id, cookie, page_id, http_header, ...`
- Queries filters against `page_id`

Some results

- Avro:
  - Compressed data footprint: 1.4 TB
  - Amount of data read on query: 1.4 TB
- ORC[1] unsorted/sorted:
  - Compressed data footprint: 0.9 TB / 0.5 TB
  - Amount of data read on query: 300 GB / 200 MB

[1] similar to parquet

Additional references

Suggested reading

- *Dremel: Interactive Analysis of Web-Scale Datasets.*, S. Melnik et al., VLDB, 2010.