

Форматы хранения

CSV, JSON, XML, AVRO, ORC, PARQUET и др.

ПОДХОДЫ К ХРАНЕНИЮ ДАННЫХ

ФОРМАТЫ ФАЙЛОВ

Город	Область	Население
Москва	-	12 655 050
Санкт-Петербург	-	5 384 342

ФОРМАТЫ ФАЙЛОВ

Row oriented (строковые, линейные)

- CSV
- JSON
- XML
- Map File
- Sequence (writable сериализация)
- AVRO (сериализация)

Классическая построчная запись данных

Высокая **скорость записи** на диск

Москва	-	12 655 050
--------	---	------------



Санкт-Петербург	-	5 384 342
-----------------	---	-----------

ФОРМАТЫ ФАЙЛОВ

Москва
Санкт-Петербург



-
-



12 655 050
5 384 342

Column oriented (колоночные, столбцовые)

- Parquet
- ORC
- RCFile

Сначала записываются все строки одной колонки, потом все строки другой и т.д.

Лучше поддаётся **сжатию**.

Больше **скорость чтения**, при анализе данных, при фильтрации по колонкам.

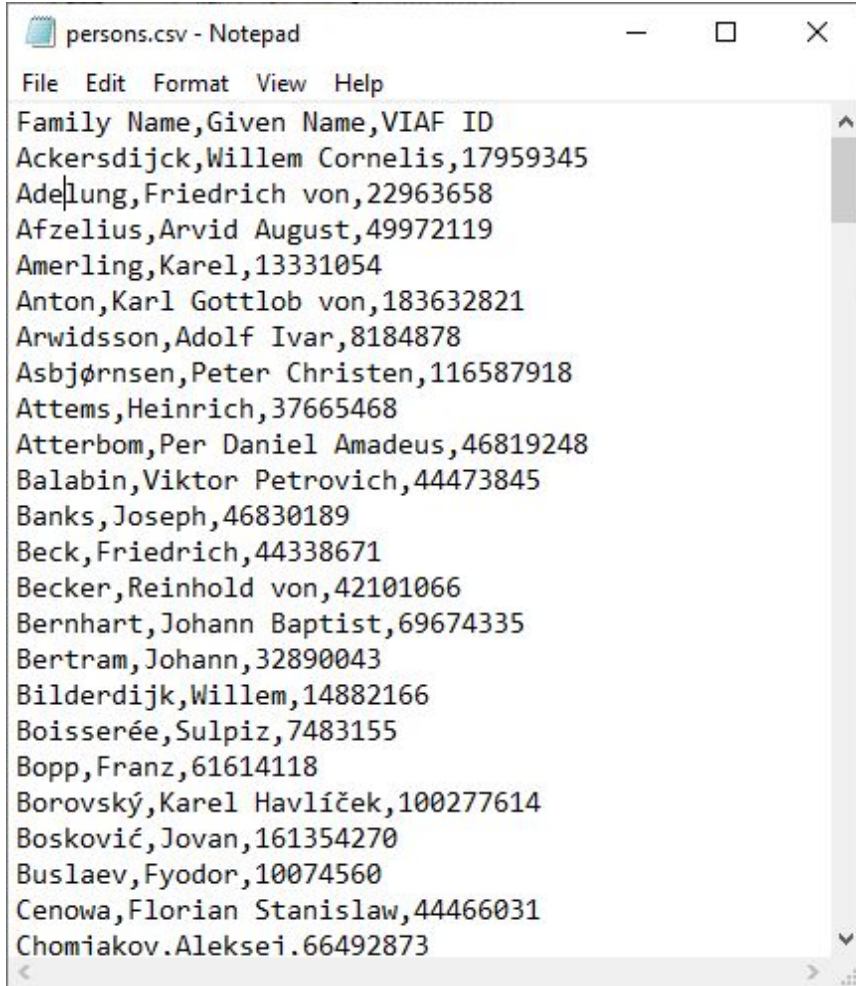
Большой расход **RAM** при записи за счет кэширования.

Что хотим получить от формата хранения

- Быструю запись
- Быстрое чтение
- Эффективное хранение (сжатие)
- Разделимость (читать не весь файл)
- Стабильную схему данных

СТРОЧНЫЕ ФОРМАТЫ (ROW ORIENTED)

CSV



```
persons.csv - Notepad
File Edit Format View Help
Family Name,Given Name,VIAF ID
Ackersdijck,Willem Cornelis,17959345
Adelung,Friedrich von,22963658
Afzelius,Arvid August,49972119
Amerling,Karel,13331054
Anton,Karl Gottlob von,183632821
Arwidsson,Adolf Ivar,8184878
Asbjørnsen,Peter Christen,116587918
Attems,Heinrich,37665468
Atterbom,Per Daniel Amadeus,46819248
Balabin,Viktor Petrovich,44473845
Banks,Joseph,46830189
Beck,Friedrich,44338671
Becker,Reinhold von,42101066
Bernhart,Johann Baptist,69674335
Bertram,Johann,32890043
Bilderdijk,Willem,14882166
Boisserée,Sulpiz,7483155
Bopp,Franz,61614118
Borovský,Karel Havlíček,100277614
Bosković,Jovan,161354270
Buslaev,Fyodor,10074560
Cenowa,Florian Stanislaw,44466031
Chomiakov,Aleksei,66492873
```

- Можно прочитать глазами
- Много инструментов для работы (даже Excel)
- Быстрая запись +
- Быстрое чтение -
- Эффективное хранение -
- Разделимость +/-
- Схема данных -

JSON

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
```

- Можно прочитать глазами
- Много инструментов для работы
- Быстрая запись +
- Быстрое чтение -
- Эффективное хранение --
- Разделимость +/-
- Схема данных +/-

XML



XML

```
<?xml version="1.0" encoding="iso-8859-8" standalone="yes" ?>
<CURRENCIES>
  <LAST_UPDATE>2004-07-29</LAST_UPDATE>
  <CURRENCY>
    <NAME>dollar</NAME>
    <UNIT>1</UNIT>
    <CURRENCYCODE>USD</CURRENCYCODE>
    <COUNTRY>USA</COUNTRY>
    <RATE>4.527</RATE>
    <CHANGE>0.044</CHANGE>
  </CURRENCY>
  <CURRENCY>
    <NAME>euro</NAME>
    <UNIT>1</UNIT>
    <CURRENCYCODE>EUR</CURRENCYCODE>
    <COUNTRY>European Monetary Union</COUNTRY>
    <RATE>5.4417</RATE>
    <CHANGE>-0.013</CHANGE>
  </CURRENCY>
</CURRENCIES>
```

- Можно прочитать глазами
- Много инструментов для работы
- Быстрая запись +
- Быстрое чтение -
- Эффективное хранение --
- Разделимость +/-
- Схема данных +, +/-

MAP FILE

Argentina	1
Australia	38
Austria	7
Bahrain	1
Belgium	8
Bermuda	1
Brazil	5
Bulgaria	1
CO	1
Canada	76
Cayman Isls	1
China	1
Costa Rica	1
Country	1
Czech Republic	3
Denmark	15
Dominican Republic	1
Finland	2
France	27
Germany	25
Greece	1
Guatemala	1
Hong Kong	1
Hungary	2

- Можно прочитать глазами
- Можно импортировать как TSV
- Быстрая запись +
- Быстрое чтение -
- Эффективное хранение +/-
- Разделимость +
- Схема данных -

SEQUENCE FILE

Sequence File Header
3 Byte (SEQ) + 1 Byte (Version) (e.g. SEQ4 or SEQ6)
Text – Key Class Name
Text – Value Class Name
Boolean - Is Compressed
Boolean – Is blockCompressed
CompressionCodec Class Name
MetaData
Sync Marker

- Нельзя прочитать глазами
- Быстрая запись +
- Быстрое чтение -
- Эффективное хранение +/-
- Разделимость ++
- Схема данных +

AVRO

legopiece.avro															
00000000	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e 0f
00000000	4f	62	6a	01	02	16	61	76	72	6f	2e	73	63	68	Obj...avro.schem
00000010	61	8e	03	7b	22	74	79	70	65	22	3a	22	72	65	až.{"type":"reco
00000020	72	64	22	2c	22	6e	61	6d	65	22	3a	22	4c	65	rd","name":"Lego
00000030	50	69	65	63	65	22	2c	22	6e	61	6d	65	73	70	Piece","namespac
00000040	65	22	3a	22	63	6f	6d	2e	65	78	61	6d	70	6c	e":"com.example.
00000050	61	76	72	6f	53	61	6d	70	6c	65	2e	6d	6f	64	avroSample.model
00000060	22	2c	22	66	69	65	6c	64	73	22	3a	5b	7b	22	", "fields": [{"na
00000070	6d	65	22	3a	22	6d	6f	64	65	6c	4e	61	6d	65	me":"modelName",
00000080	22	74	79	70	65	22	3a	22	73	74	72	69	6e	67	"type":"string"}]
00000090	2c	7b	22	6e	61	6d	65	22	3a	22	6d	6f	64	65	, {"name":"modelY
000000a0	65	61	72	22	2c	22	74	79	70	65	22	3a	22	69	ear","type":"int
000000b0	22	7d	2c	7b	22	6e	61	6d	65	22	3a	22	73	74	"}," {"name":"stub
000000c0	4e	75	6d	62	65	72	22	2c	22	74	79	70	65	22	Number","type":"
000000d0	73	74	72	69	6e	67	22	7d	5d	7d	00	17	6d	7b	string"]}]..m{ãB
000000e0	44	da	cb	3f	14	a4	53	56	af	a0	c5	02	2e	1e	DÚĚ?.»SV Å...Br
000000f0	69	63	6b	20	31	58	32	20	47	72	65	65	6e	fe	ick 1X2 Greenpÿÿ
00000100	ff	0f	02	32	17	6d	7b	e3	42	44	da	cb	3f	14	ÿ..2.m{ãBDÚĚ?.»S
00000110	56	af	a0	c5	V Å.....

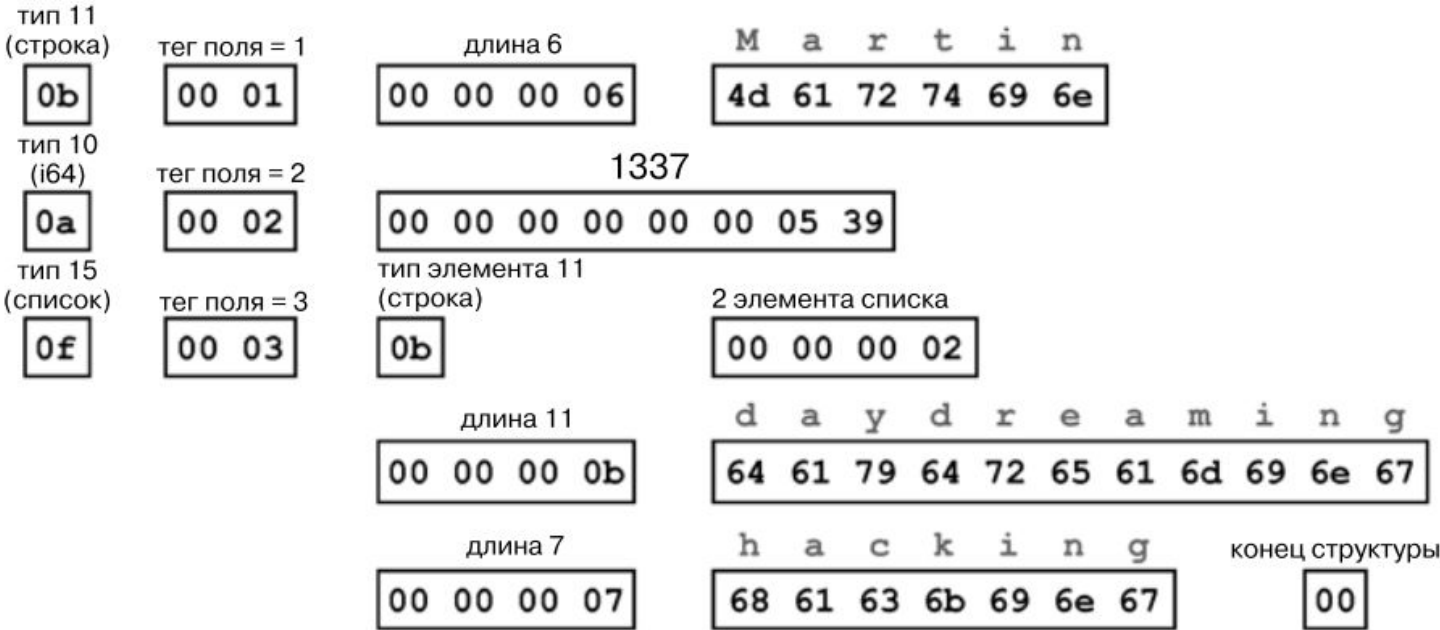
- Нельзя прочитать глазами
- Быстрая запись +
- Быстрое чтение -
- Эффективное хранение +/-
- Разделимость +
- Схема данных +

59 байт

0b	00 01	00 00 00 06	4d 61 72 74 69 6e	0a	00 02	00 00 00 00
00 00 05 39	0f	00 03	0b	00 00 00 02	00 00 00 0b	64 61 79 64
72 65 61 6d 69 6e 67	00 00 00 07	68 61 63 6b 69 6e 67	00			

Разбор:

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

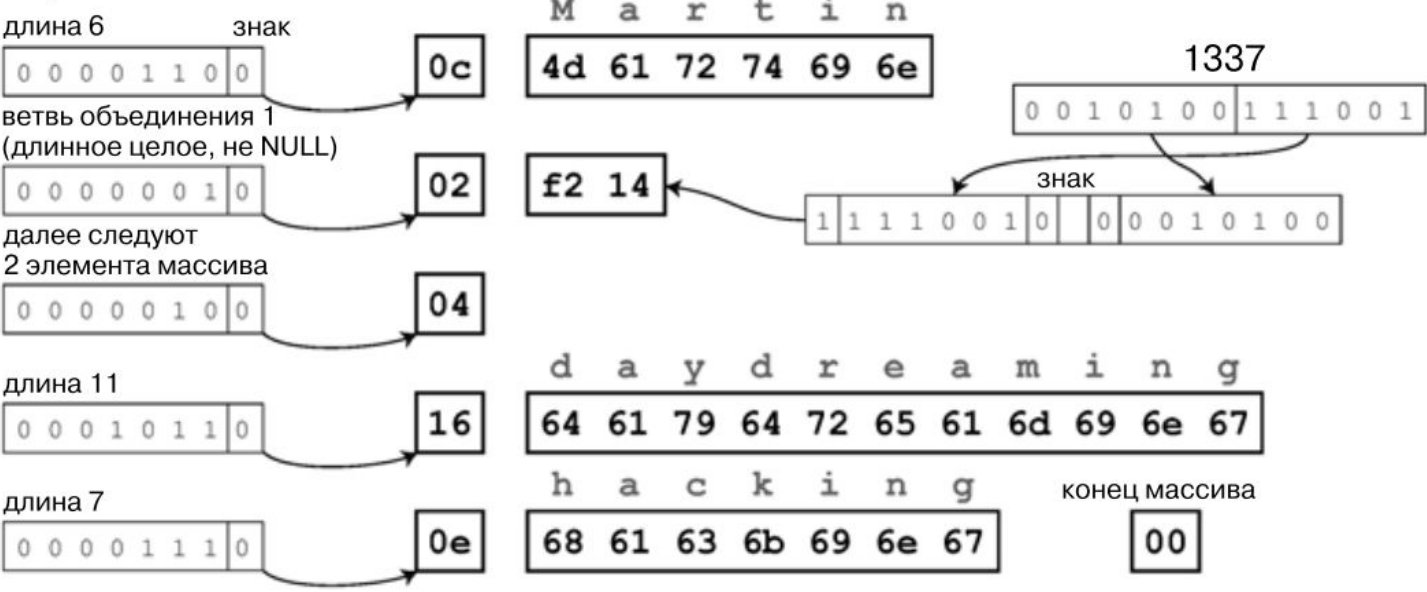


32 байта

0c	4d	61	72	74	69	6e	02	f2	14	04	16	64	61	79	64	72	65	61	6d
69	6e	67	0e	68	61	63	6b	69	6e	67	00								

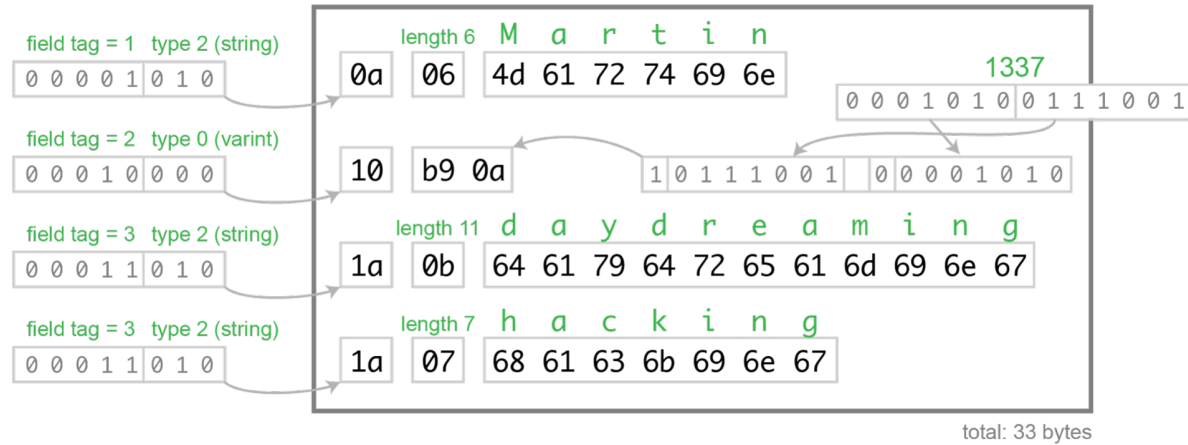
```
{
  "userName": "Martin",
  "favoriteNumber": 1337,
  "interests": ["daydreaming", "hacking"]
}
```

Разбор:



PROTOBUF

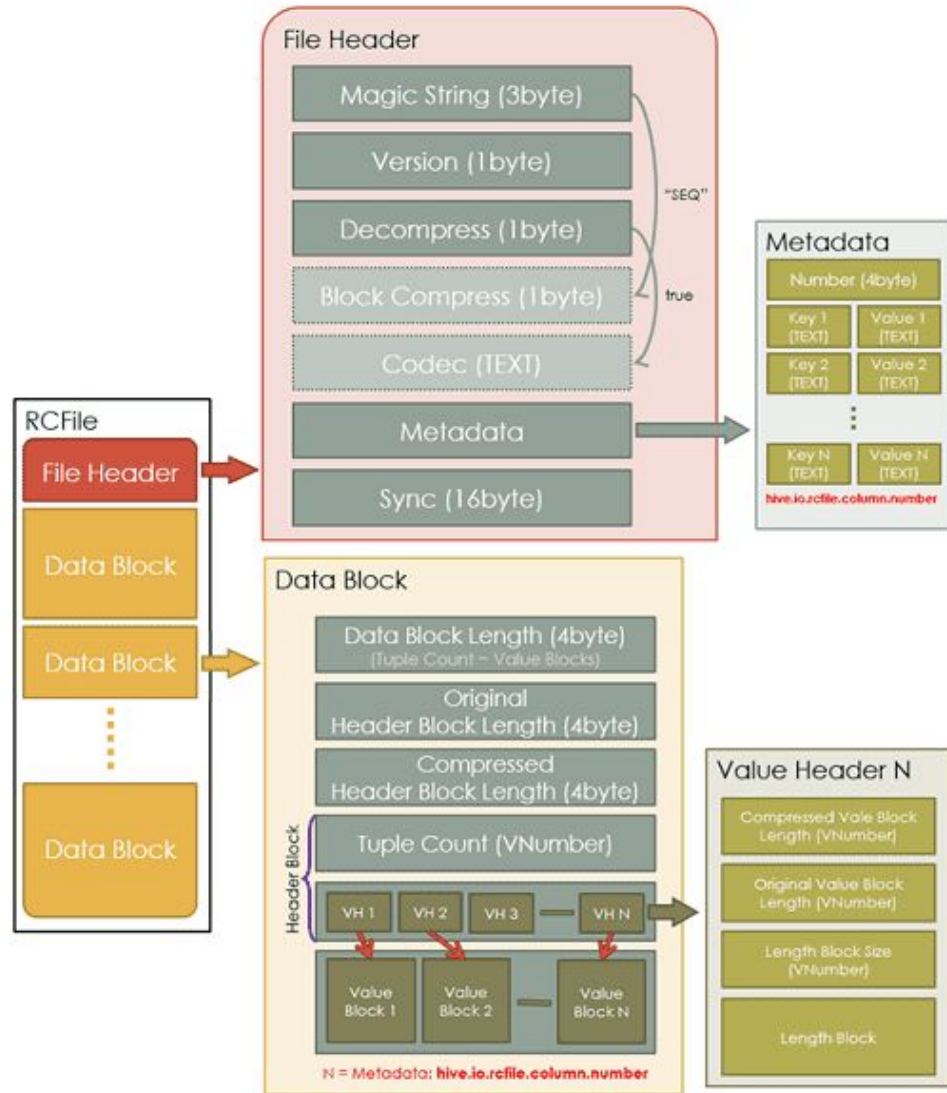
Protocol Buffers



- Нельзя прочитать глазами
- Быстрая запись +
- Быстрое чтение -
- Эффективное хранение +/-
- Разделимость +
- Схема данных +

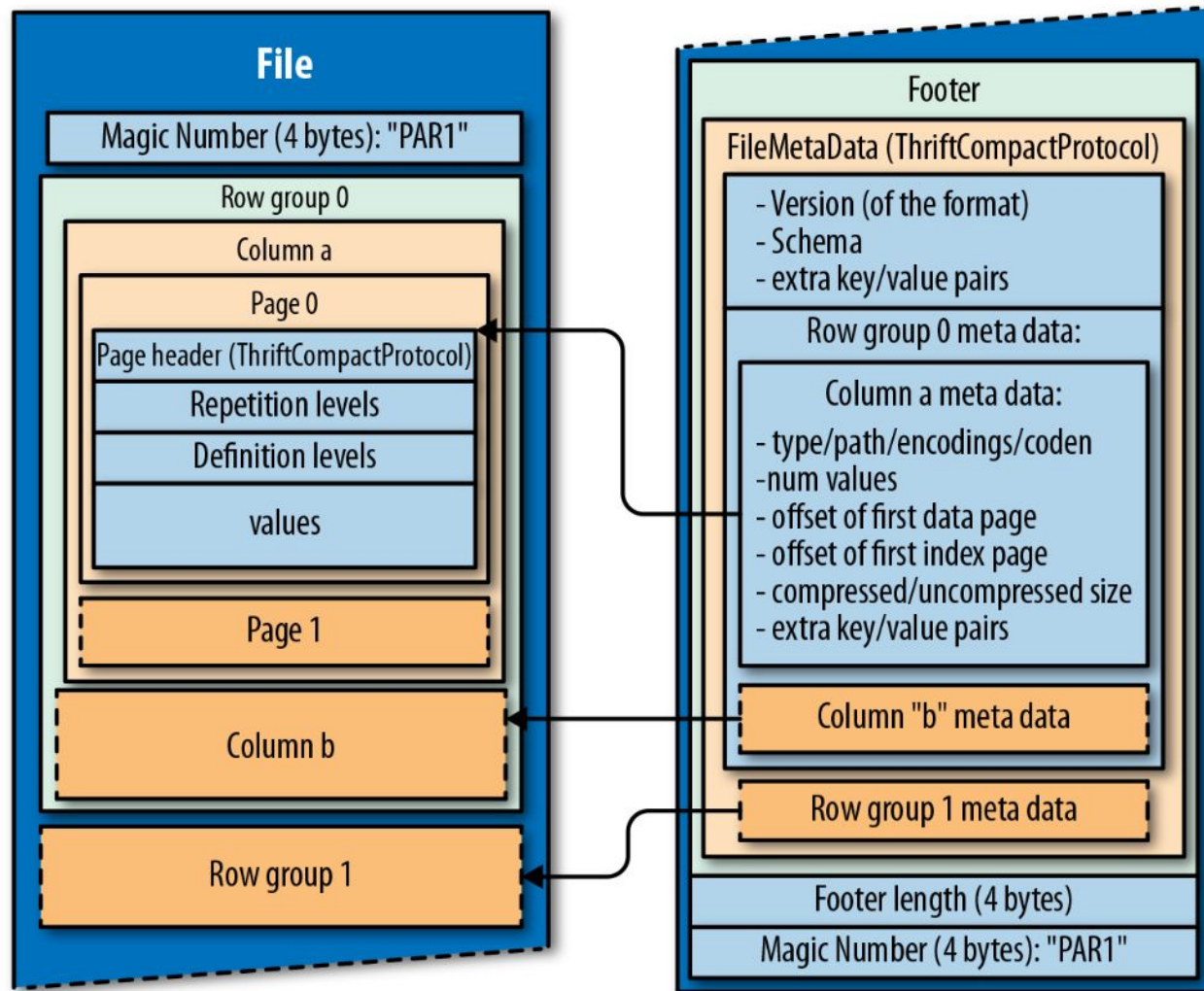
КОЛОНОЧНЫЕ ФОРМАТЫ (COLUMN ORIENTED)

RC FILE



- Нельзя прочитать глазами
- Быстрая запись -
- Быстрое чтение +
- Эффективное хранение +
- Разделимость +
- Схема данных +

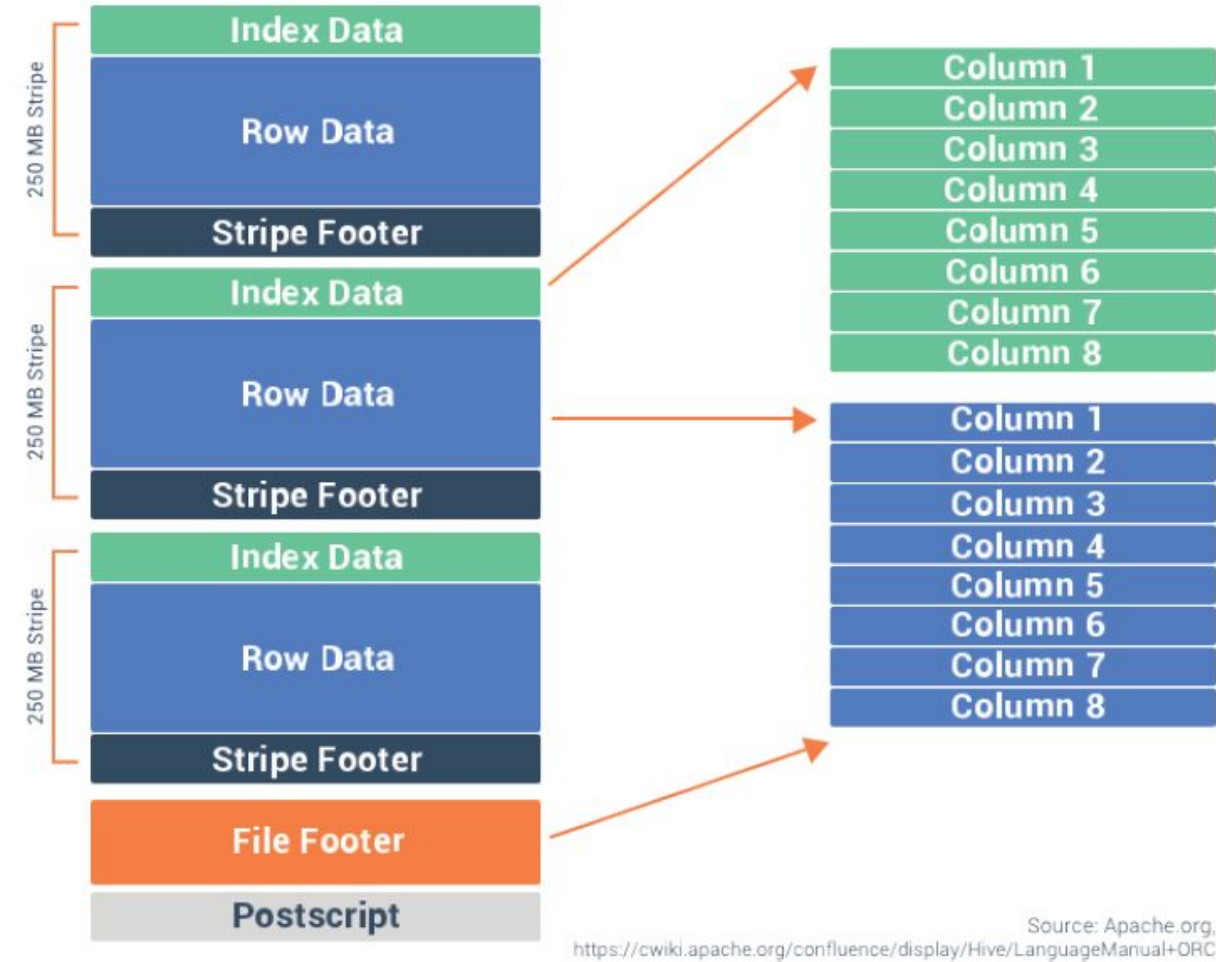
PARQUET



- Нельзя прочитать глазами
- Быстрая запись -
- Быстрое чтение +
- Эффективное хранение +
- Разделимость +
- Схема данных +

ORC

ORC FILE STRUCTURE



- Нельзя прочитать глазами
- Быстрая запись -
- Быстрое чтение +
- Эффективное хранение +
- Разделимость +
- Схема данных +

BIG DATA FORMATS COMPARISON

	Avro	Parquet	ORC
Schema Evolution Support			
Compression			
Splitability			
Most Compatible Platforms	Kafka, Druid	Impala, Arrow Drill, Spark	Hive, Presto
Row or Column	Row	Column	Column
Read or Write	Write	Read	Read

Source: Nexla analysis, April 2018

- Column-oriented для аналитики
- Row-oriented для OLTP
- AVRO – для большой нагрузки по записи, например для Apache Kafka
- Parquet – если у вас любят Parquet
- ORC – если у вас любят ORC
- CSV – если нужна простая интеграция

КОМПРЕССИЯ

Компрессия

Snappy, LZ4

- Storage Space: Medium
- CPU Usage: Low
- Splittable: Yes

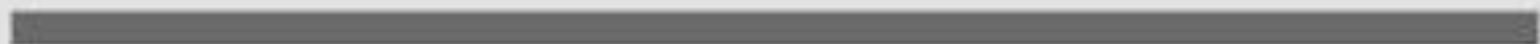
GZIP

- Storage Space: High
- CPU Usage: Medium
- Splittable: No

Подробнее: [Hadoop Compression](#)

ПЕРЕРЫВ

10:00



ПРАКТИКА

ПОДНИМАЕМ ЛОКАЛЬНЫЙ КЛАСТЕР

```
docker start -i gbhdp
```

ПРИМЕР РАБОТЫ

Поработаем с датасетом [LEGO Database](#)

Загружаем данные в контейнер:

```
$ docker cp inventory_parts.csv gbhdp:/home/hduser/
```

Помещаем их в HDFS:

```
$ hdfs dfs -put /home/hduser/inventory_parts.csv /
```

Переключаемся на терминал с хадуп сессией и включаем интерактивную оболочку:

```
$ hive
```

ПРИМЕР РАБОТЫ

Создаем таблицу для inventory_parts.csv:

```
CREATE TABLE lego_inventory_parts(inventory_id INT, part_num STRING,  
color_id INT, quantity INT, is_spare STRING)  
  COMMENT 'Information on lego inventory parts'  
  ROW FORMAT DELIMITED  
  FIELDS TERMINATED BY ','  
  STORED AS TEXTFILE  
  TBLPROPERTIES('skip.header.line.count'='1');
```

ПРИМЕР РАБОТЫ

Загружаем данные в таблицы:

```
LOAD DATA INPATH '/inventory_parts.csv' INTO TABLE lego_inventory_parts;
```

Проверяем результат:

```
SELECT * FROM lego_inventory_parts LIMIT 5;
```

ПРОВЕРЯЕМ ИМПОРТ

Проверяем, что после импорта датасета в hdfs по старому пути нет:

```
$ hdfs dfs -ls /
```

Зато он есть в hive warehouse:

```
$ hdfs dfs -ls /user/hive/warehouse/lego_inventory_parts/inventory_parts.csv
```

Посмотрим, сколько места он занимает:

```
$ hdfs dfs -du -h /user/hive/warehouse/lego_inventory_parts
```


ИМПОРТ В PARQUET

Запускаем интерактивную оболочку:

```
$ hive
```

ИМПОРТ В PARQUET

Создаем таблицу с хранением данных в формате parquet и сжатием snappy:

```
SET parquet.compression=SNAPPY;  
CREATE TABLE lego_inventory_parts_par_snappy(inventory_id INT,  
part_num STRING, color_id INT, quantity INT, is_spare STRING)  
STORED AS PARQUET;
```

Наполним её:

```
INSERT OVERWRITE TABLE lego_inventory_parts_par_snappy  
SELECT * FROM lego_inventory_parts;
```

ИМПОРТ В PARQUET

Ещё одна таблица с хранением данных в формате parquet и сжатием gzip:

```
SET parquet.compression=GZIP;  
CREATE TABLE lego_inventory_parts_par_gz(inventory_id INT, part_num  
STRING, color_id INT, quantity INT, is_spare STRING)  
STORED AS PARQUET;
```

Наполним её:

```
INSERT OVERWRITE TABLE lego_inventory_parts_par_gz  
SELECT * FROM lego_inventory_parts;
```

ИМПОРТ В PARQUET

И ещё одна таблица с хранением данных в формате parquet, но без сжатия:

```
SET parquet.compression=UNCOMPRESSED;  
CREATE TABLE lego_inventory_parts_par_raw(inventory_id INT, part_num  
STRING, color_id INT, quantity INT, is_spare STRING)  
STORED AS PARQUET;
```

Наполним её:

```
INSERT OVERWRITE TABLE lego_inventory_parts_par_raw  
SELECT * FROM lego_inventory_parts;
```

ПРОВЕРЯЕМ ИМПОРТ В PARQUET

Выходим из интерактивной оболочки:

```
exit;
```

Посмотрим, сколько места занимает каждая таблица:

```
$ hdfs dfs -du -h /user/hive/warehouse/*
```

При желании, через cat можно посмотреть содержимое каждой таблицы:

```
$ hdfs dfs -cat /user/hive/warehouse/lego_inventory_parts_par_snappy/*
```

СМОТРИМ СОДЕРЖИМОЕ PARQUET

Скачиваем parquet tools:

```
wget https://repo1.maven.org/maven2/org/apache/parquet/parquet-tools/1.11.1/parquet-tools-1.11.1.jar
```

Строка запуска:

```
$ hadoop jar parquet-tools-1.11.1.jar COMMAND [OPTIONS] PATH
```

Допустимые команды:

- cat, head — отобразить содержимое / первые N строк
- schema — напечатать схему
- meta — метаданные
- dump — вывести группы строк внутри файла и метаданные по ним
- merge — слить несколько parquet файлов в один

ИМПОРТ В ORC

Запускаем интерактивную оболочку:

```
$ hive
```

ИМПОРТ В ORC

Создаем таблицу с хранением данных в формате orc и сжатием snappy:

```
CREATE TABLE lego_inventory_parts_orc_snappy(inventory_id INT,  
part_num STRING, color_id INT, quantity INT, is_spare STRING)  
STORED AS ORC  
TBLPROPERTIES('orc.compress'='SNAPPY');
```

Наполним её:

```
INSERT OVERWRITE TABLE lego_inventory_parts_orc_snappy  
SELECT * FROM lego_inventory_parts;
```


ИМПОРТ В ORC

Ещё одна таблица с хранением данных в формате orc и без сжатия:

```
CREATE TABLE lego_inventory_parts_orc_raw(inventory_id INT, part_num  
STRING, color_id INT, quantity INT, is_spare STRING)  
STORED AS ORC  
TBLPROPERTIES('orc.compress'='NONE');
```

Наполним её:

```
INSERT OVERWRITE TABLE lego_inventory_parts_orc_raw  
SELECT * FROM lego_inventory_parts;
```

ПРОВЕРЯЕМ ИМПОРТ В ORC

Выходим из интерактивной оболочки:

```
exit;
```

Посмотрим, сколько места занимает каждая таблица:

```
$ hdfs dfs -du -h /user/hive/warehouse/*
```

При желании, через cat можно посмотреть содержимое каждой таблицы:

```
$ hdfs dfs -cat /user/hive/warehouse/lego_inventory_parts_orc_raw/*
```

СМОТРИМ СОДЕРЖИМОЕ ORC

Скачиваем avro tools:

```
wget https://repo1.maven.org/maven2/org/apache/orc/orc-tools/1.6.10/orc-tools-1.6.10-uber.jar
```

Строка запуска:

```
$ hadoop jar orc-tools-1.6.10-uber.jar COMMAND [OPTIONS] PATH
```

Допустимые команды:

- data — вывести данные файла
- scan — произвести сканирование файла
- meta — метаданные
- convert — конвертировать CSV/JSON файлы в ORC
- json-schema — определить схему в json файлах

ИМПОРТ В AVRO

Запускаем интерактивную оболочку:

```
$ hive
```

ИМПОРТ В AVRO

Создаем таблицу с хранением данных в формате avro:

```
CREATE TABLE lego_inventory_parts_avro(inventory_id INT, part_num  
STRING, color_id INT, quantity INT, is_spare STRING)  
STORED AS AVRO;
```

Наполним её:

```
INSERT OVERWRITE TABLE lego_inventory_parts_avro  
SELECT * FROM lego_inventory_parts;
```

ПРОВЕРЯЕМ ИМПОРТ В AVRO

Выходим из интерактивной оболочки:

```
exit;
```

Посмотрим, сколько места занимает каждая таблица:

```
$ hdfs dfs -du -h /user/hive/warehouse/*
```

При желании, через cat можно посмотреть содержимое каждой таблицы:

```
$ hdfs dfs -cat /user/hive/warehouse/lego_inventory_parts_avro/*
```

СМОТРИМ СОДЕРЖИМОЕ AVRO

Скачиваем avro tools:

```
wget https://repo1.maven.org/maven2/org/apache/avro/avro-tools/1.10.2/avro-tools-1.10.2.jar
```

Строка запуска:

```
$ hadoop jar avro-tools-1.10.2.jar COMMAND [OPTIONS] PATH
```

Допустимые команды:

- getschema — вывести схему данных
- getmeta — вывести метаданные
- tojson — конвертировать в json

Полный список можно посмотреть [тут](#)

ОСТАНОВКА ЛОКАЛЬНОГО КЛАСТЕРА

```
exit
```


ПРАКТИЧЕСКОЕ ЗАДАНИЕ



ПРАКТИЧЕСКОЕ ЗАДАНИЕ

1. Создайте таблицу `lego_inventory_parts_cut` без колонок типа `STRING`
2. Сравните степень сжатия (отношения несжатого файла к сжатому) таблицы в которой есть колонки типа `STRING` с таблицей без колонок этого типа
3. Чем `AVRO` отличается от `CSV`?

Спасибо!

**Каждый день
вы становитесь
лучше :)**

