
H·BASE

Особенности HBase

- Распределенная база данных:
  - Работает на кластере серверов
  - Легко горизонтально маштабируется
- NoSQL база данных:
  - Не предоставляет SQL-доступ
  - Не предоставляет реляционной модели

Особенности HBase

- Column-Oriented хранилище данных
  - нет фиксированной структуры колонок
  - произвольное число колонок для ключа
- Спроектирована для поддержки больших таблиц
  - Миллиарды строк и миллионы колонок
- Поддержка произвольных операций чтения/записи

Особенности HBase

- Основана на идеях Google BigTable
  - *http://labs.google.com/papers/bigtable.html*
- BigTable поверх GFS => HBase поверх HDFS
- Масштабируемость с помощью шардирования
- Автоматический fail-over
- Простой Java API
- Интеграция с MapReduce

Когда нужно использовать HBase

Больших объемов данных

- Для маленьких данных лучше использовать РБД

Паттерн доступа к данным:

- Выборка значений по заданному ключу
- Последовательный скан в диапазоне ключей

Когда нужно использовать HBase

Свободная схема данных

- Строки могу существенно отличаться по своей структуре
- В схеме может быть множество колонок и большинство из них будет равно *null*

Когда НЕ нужно использовать HBase

Традиционный доступ к данным в стиле РБД

- Приложения с транзакциями
- Реляционная аналитика (*'group by', 'join‘ u 'where column like‘* и т.д.)

Плохо подходит для доступ к данным на основе текстовых запросов (*LIKE %text%*)

Key	CF1:C1	CF1:C2	CF2:C1	CF2:C2
1	Some text	7	True	
2		10	False	
3	For example	1234		
4	Not empty		False	
5		11	True	
6		52		

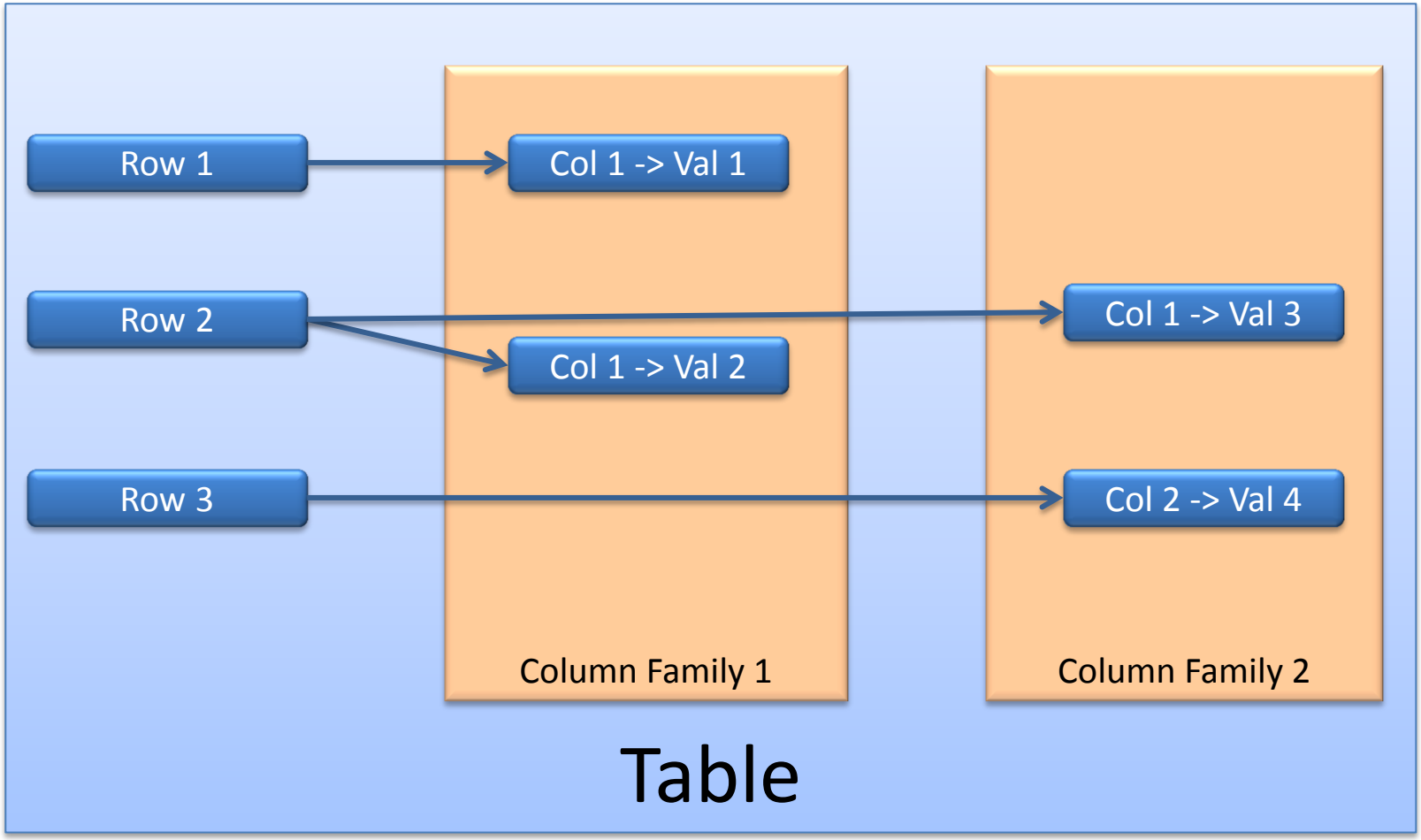
HBase Column Families

Column Family описывает общие свойства колонок:

- Сжатие
- Количество версий данных
- Время жизни (Time To Live)
- Опция хранения только в памяти (In-memory)
- Хранится в отдельном файле (HFile/StoreFile)

HBase Column Families

- Конфигурация *CF* статична
  - Задается в процессе создания таблицы
  - Количество *CF* ограничено небольшим числом
- Колонки наоборот НЕ статичны
  - Создаются в runtime
  - Могут быть сотни тысяч для одной CF


Row 1
Row 2
Row 3
Col 1 -> Val 1
Col 1 -> Val 2
Col 1 -> Val 3
Col 2 -> Val 4
Column Family 1
Column Family 2
Table

HBase Timestamps

- Ячейки имеют несколько версий данных
  - Настраивается в конфигурации ColumnFamily
  - По-умолчанию равно 3
- Данные имеют timestamp
  - Задается неявно при записи
  - Явно указывается клиентом
- Версии хранятся в убывающем порядке ts
  - Последнее значение читается первым

Value = Table + RowKey + Family + Column + Timestamp

Row Key	Timestamp	CF: “Data”		CF: “Meta”		
		Url	Html	Size	Date	Log
row1	t1	Mail.Ru				Log text 1
	t2				123456	Log text 2
	t3		<html>…	1234		Log text 3
	t4		<HTML>…	2345		Log text 4
row2	t1	OK.Ru			123765	Log text 1
	t2					Log text 2

Архитектура HBase

Key	Column1	Column3	TimeStamp
Row1	value1		timestamp1
		value2	timestamp2
Row2	value4		timestamp1
Row3	value1	value6	timestamp1

HFile

Row1:ColumnFamily:column1:timestamp1:value1
Row1:ColumnFamily:column3:timestamp2:value3
Row2:ColumnFamily:column1:timestamp1:value4
Row3:ColumnFamily:column1:timestamp1:value1
Row3:ColumnFamily:column3:timestamp1:value6

KeyValues

Маштабируемость в Hbase

- **Таблица** делится на регионы
- **Регион** – это группа строк, которые хранятся вместе
  - Единица шардинга
  - Динамически делится пополам, если становится большим
- **RegionServer** – демон, который управляет одним или несколькими регионами
  - Регион принадлежит только одному RS
- **MasterServer (HMaster)** – демон, который управляет всеми RS

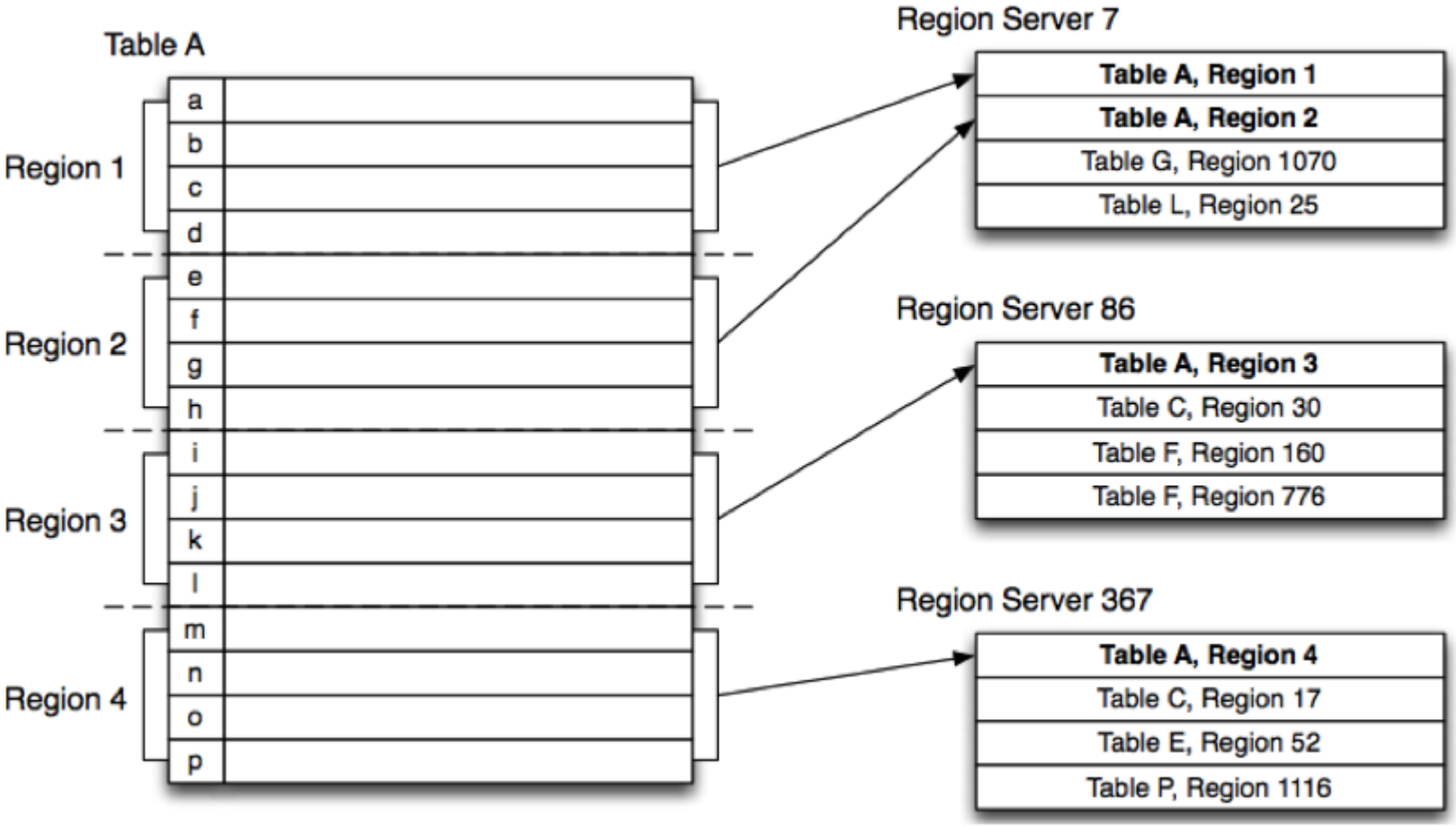


Table A
Region 1
Region 2
Region 3
Region 4
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
Region Server 7
Table A, Region 1
Table A, Region 2
Table G, Region 1070
Table L, Region 25
Region Server 86
Table A, Region 3
Table C, Region 30
Table F, Region 160
Table F, Region 776
Region Server 367
Table A, Region 4
Table C, Region 17
Table E, Region 52
Table P, Region 1116

Hbase Regions

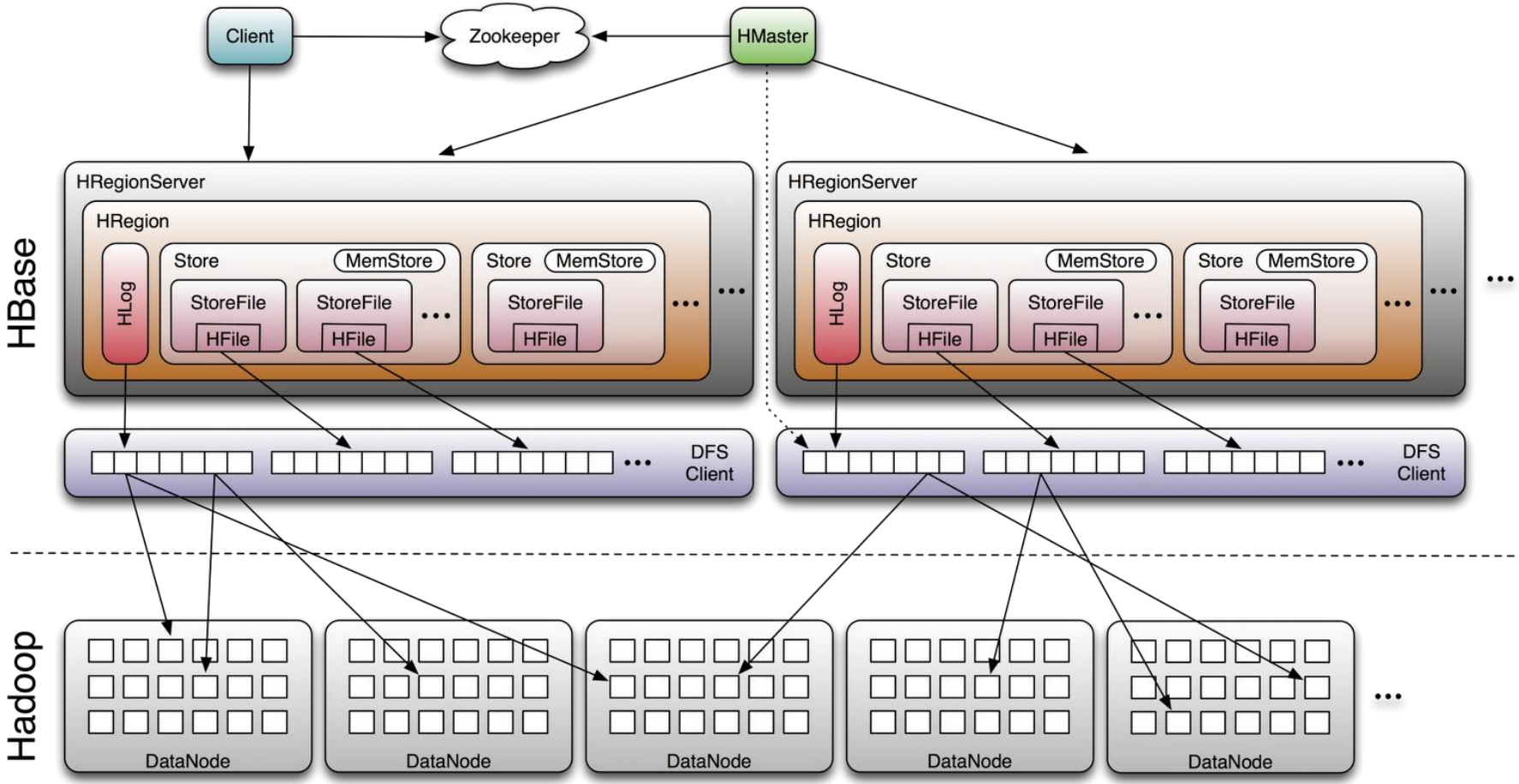
Регион – это диапазон ключей: [*Start Key; Stop Key)*

- *Start Key* включается в регион
- *Stop Key* не включается

- По-умолчанию есть только один регион
- Можно предварительно задать количество регионов
- При превышении лимита, регион разбивается на 2 части

Hbase Regions Split

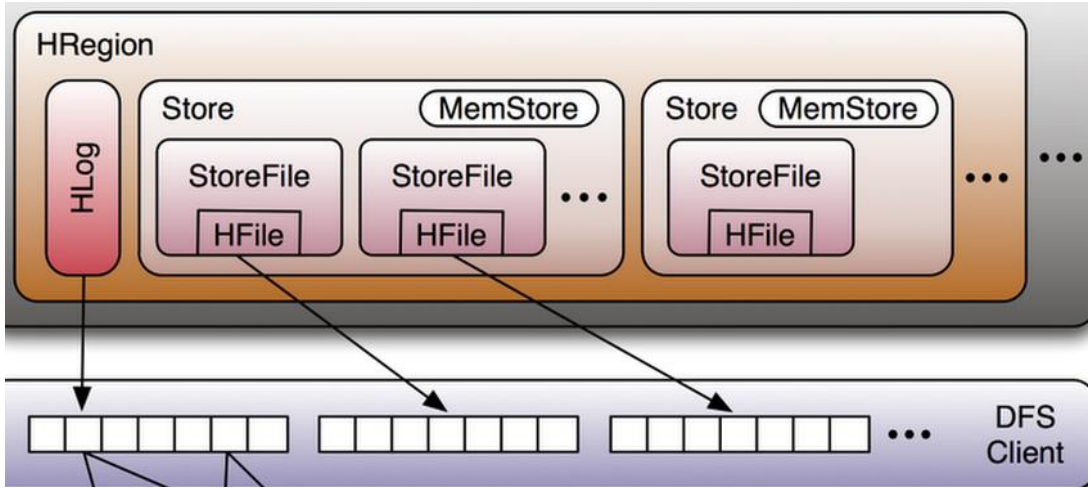
- ✚ Регионы более сбалансированы по размеру
- ✚ Быстрое восстановление, если регион повредился
- ✚ Баланс нагрузки на RegionServer
- ✚ Split – это быстрая операция
- ➖ На больших инсталляциях лучше производить в ручную


Client
Zookeeper
HMaster
HBase
HRegionServer
HRegion
HLog
Store
MemStore
StoreFile
HFile
DFS Client
Hadoop
DataNode

HBase Master

- Управляет регионами:
  - Назначает регион на RegionServer
  - Перебалансирует их для распределения нагрузки
  - Восстанавливает регион, если он становится недоступен
- Не хранит данные
  - Клиент взаимодействует напрямую с RS
  - Обычно не сильно нагружен
- Отвечает за управление схемой таблиц и ее изменений
  - Добавляет/удаляет таблицы и CF

Data Storage



Внесение изменений в данные

- В HDFS нельзя внести изменения в файл
  - Нельзя удалить key-value из HFile
  - Со временем становится много HFile’ов
- При удалении добавляется *delete marker*
  - Маркеры используются для фильтрации удаленных записей в runtime

Data Storage: compaction

- Minor Compaction
  - Маленькие HFile’ы объединяются в бОльшие файлы
  - Быстрая операция
  - Маркеры удаления не применяются
- Major Compaction
  - Для каждого региона все файлы в рамках одной CF объединяются в один файл
  - Используются маркеры удаления для того, чтобы не включать удаленные записи

Доступ к HBase

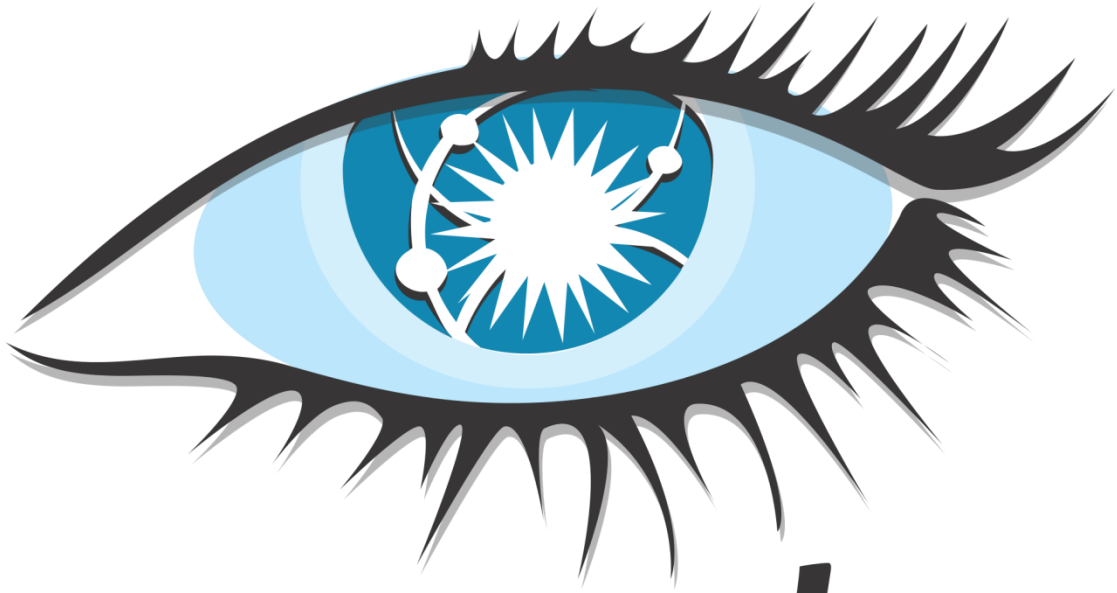
- Основные способы:
  - HBase Shell
  - Native Java API
- Другие способы:
  - Avro Server
  - PyHBase
  - REST Server
  - Thrift

Запрос данных из HBase

- По RowId или набору RowIds
  - Отдает только те строки, которые соответствуют заданным id
  - Если не заданы дополнительные критерии, то отдаются все *cells* из всех *CF* заданной таблицы
    - Это означает слияние множества файлов из HDFS
- По *ColumnFamily*
  - Уменьшает количество файлов для загрузки
- По *Timestamp/Version*
  - Может пропускать полностью Hfile’ы, которые не содержат данный диапазон timestamp’ов

Запрос данных из HBase

- По *Column Name/Qualifier*
  - Уменьшает объем передаваемых данных
- По *Value*
  - Можно пропускать ячейки используя фильтры
  - Самый медленный критерий выбора данных
  - Будет проверять каждую ячейку
- Фильтры могут применяться для каждого критерия выбора
  - *rows, families, columns, timestamps u values*
- Можно использовать множественные критерии выбора данных



cassandra

Cassandra

- Разработана в Facebook (с 2008 года)
- Следует модели Google BigTable
- Open-source проект Apache
  http://cassandra.apache.org/
- Написана на Java
- Не имеет единой точки отказа

Особенности Cassandra

- Распределенная
- Поддерживает репликацию
- Маштабируема
- Устойчива к сбоям
- Консистентность в конечном счете
- Поддержка MapReduce Hadoop
- SQL-подобный язык доступа к данным (CQL – Cassandra Query Language)

Типичный NoSQL API

- *get(key)*
- *put(key, value)*
- *delete(key)*
- *execute(key, operation, parameters)*

Data Model

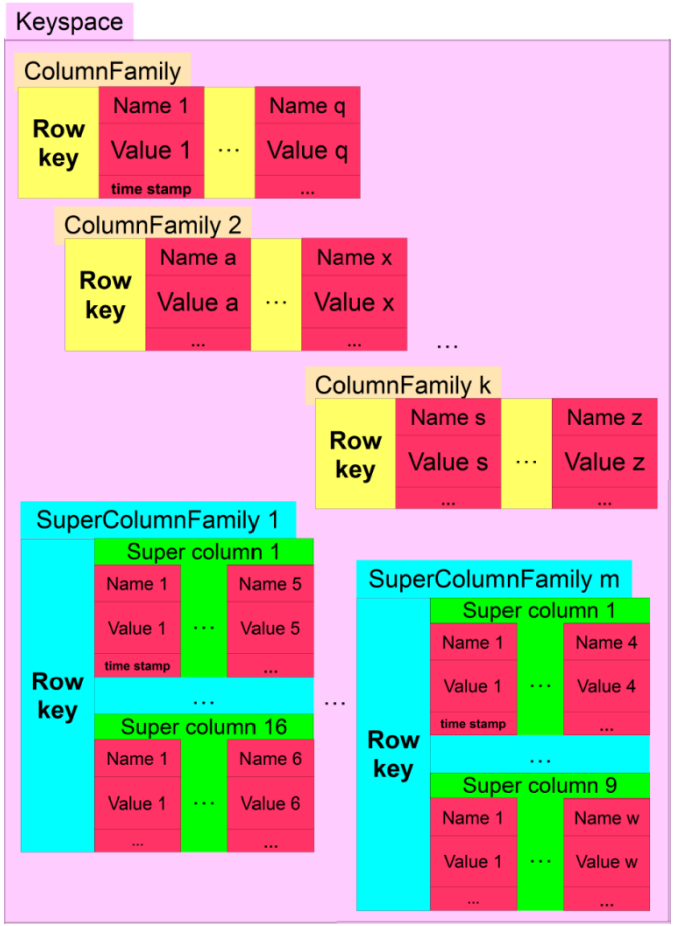
- **Column:** наименьший элемент данных, пара имя и значение
- **ColumnFamily:** структура для группировки *Columns*
- **Key:** постоянное имя записи
- **Keyspace:** самый внешний уровень организации данных (имя базы данных)

```
CREATE KEYSPACE MyKeySpace
 WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : 3 };

USE MyKeySpace;

CREATE COLUMNFAMILY MyColumns (id text, Last text, First text, PRIMARY
KEY(id));

INSERT INTO MyColumns (id, Last, First) VALUES ('1', 'Doe', 'John');

SELECT * FROM MyColumns;
```

```
id  | first | last
----+-------+------
1   | John  | Doe
(1 rows)
```

Column: **key -> string**

SuperColumn: **key -> columns**



Cassandra и Consistency

- Eventual consistency
- Cassandra имеет программируемый *read/writable consistency*

Cassandra и Consistency

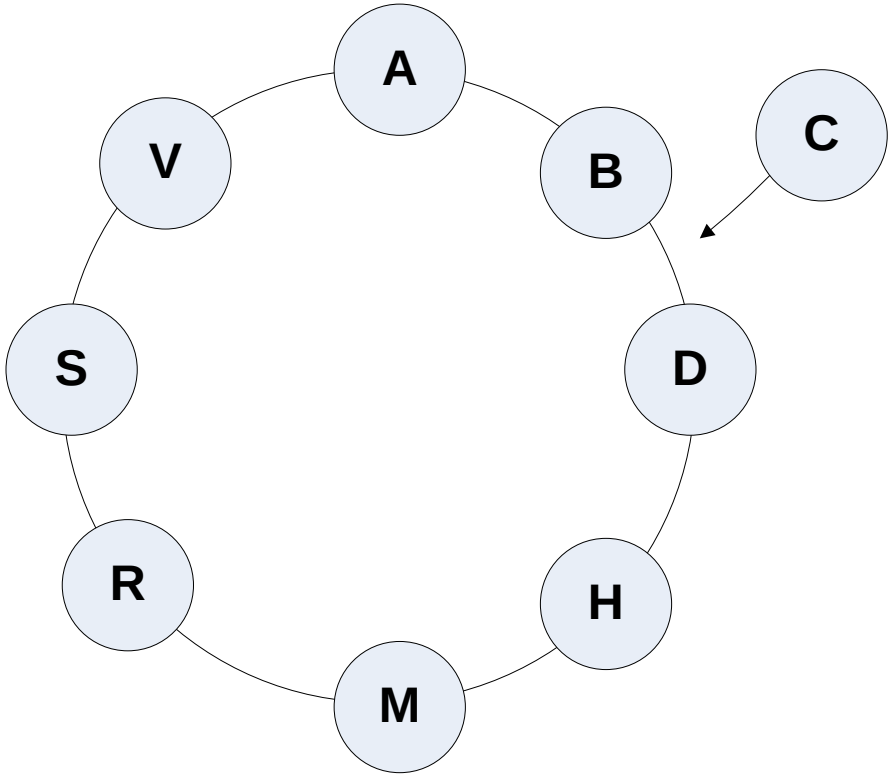
Read

- **One:** первая ответившая нода
- **Quorum:** ждем пока ответит большинство нод
- **All:** ждем ответа от всех нод

Cassandra и Consistency

Write

- **Zero:** Ничего не гарантируется
- **Any:** Гарантируется запись на одну ноду
- **One:** Гарантируется запись, в один *commit log* и в *memory table*
- **Quorum:** Гарантируется запись на N/2 + 1 ноду
- **All:** Гарантируется, что запись дойдет до всех нод


A
B
C
D
H
M
R
S
V