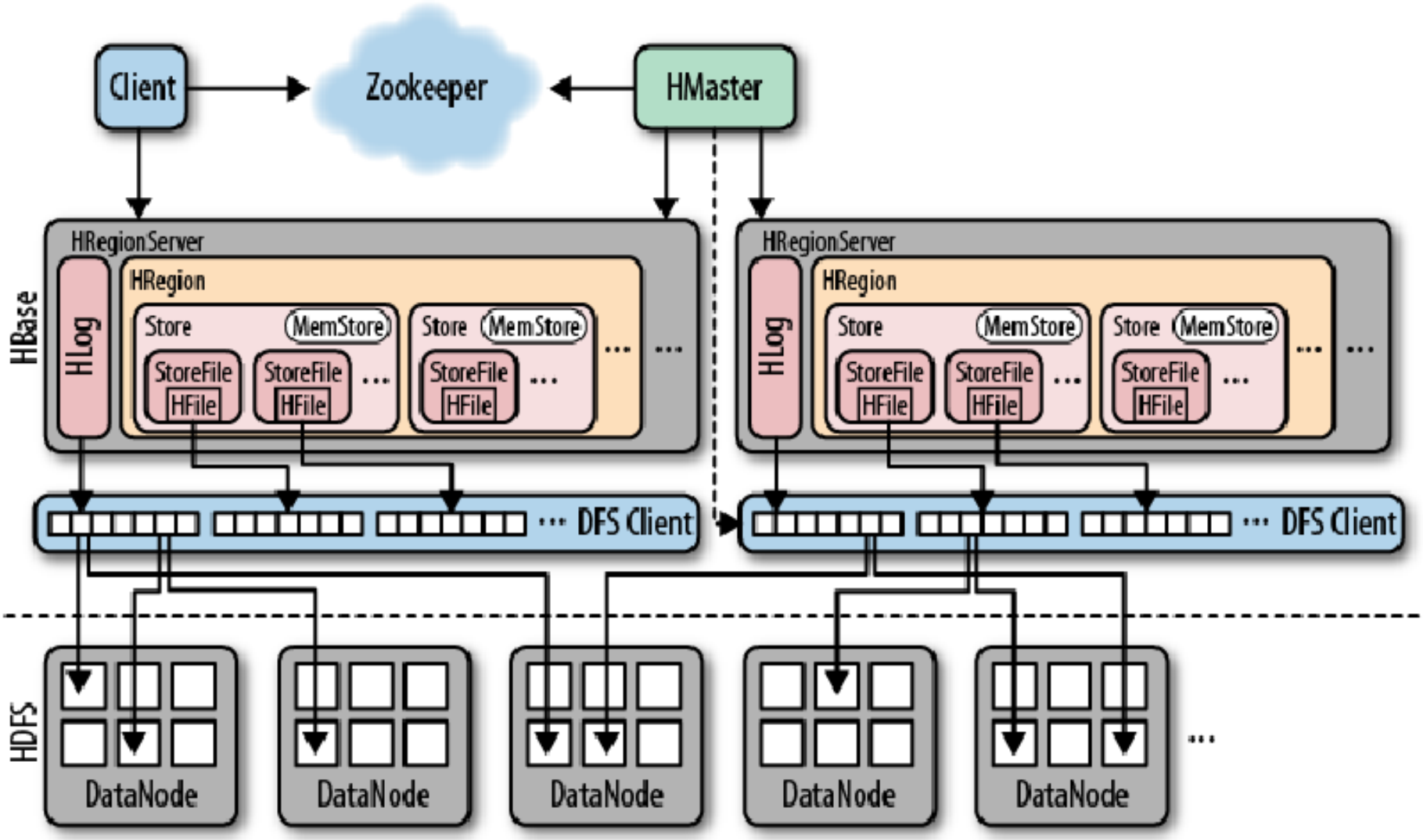
HBase. Архитектура. API. Примеры

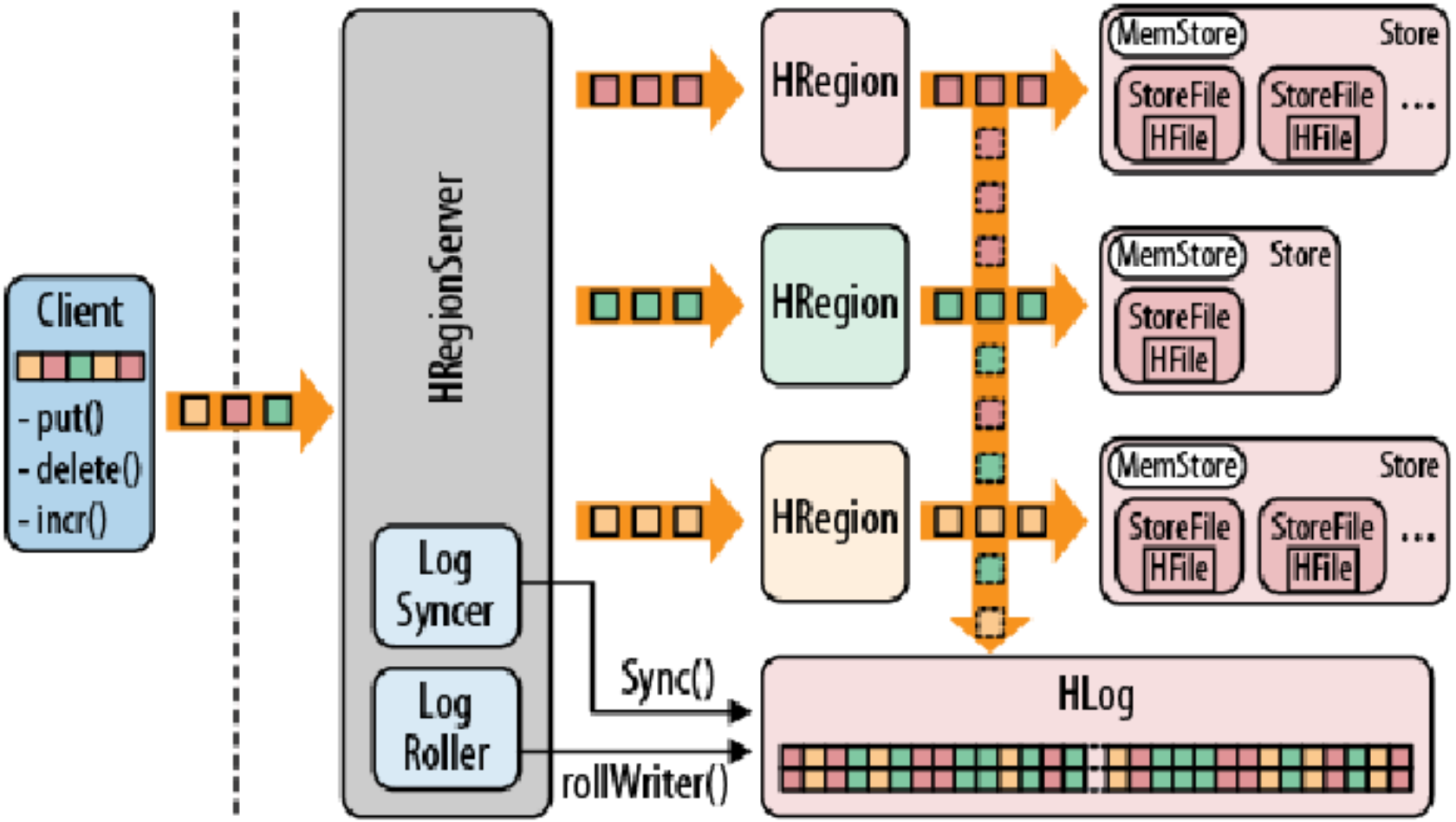
Архитектура HBase



Компоненты HBase

- HMaster – центральный компонент управляющий расположением данных в регионах
- HRegionServer – сервер, хранящий данные
- HLog – реализация Write Ahead Log
- HRegion – хранилище участка таблицы
- HFile – хранит на диске данные одного column family региона таблицы
- Zookeeper – распределенный координирующий сервис, отслеживает состояние серверов, хранит адрес региона -ROOT-

Write-Ahead Log



Запись данных

- Данные сначала записываются в HLog, далее в соответствующий memstore региона
- LogSyncer отвечает за сохранность данных – по умолчанию сразу после записи мы получаем подтверждение от файловой системы
- LogRoller через определенные интервалы времени закрывает старый файл лога и открывает новый

HFile

- HFile состоит из блоков разных типов.
- Каждый блок считывается целиком и помещается в block cache

  Виды блоков:

  DATA – данные, набор key/value

  LEAF_INDEX – Индекс блока с данными (в многоуровневом дереве индексов)

  BLOOM_CHUNK – Фильтр Блюма блока с данными

  META – Метаданные

  INTERMEDIATE_INDEX – промежуточный индекс ссылающийся на первый ключ из LEAF_index

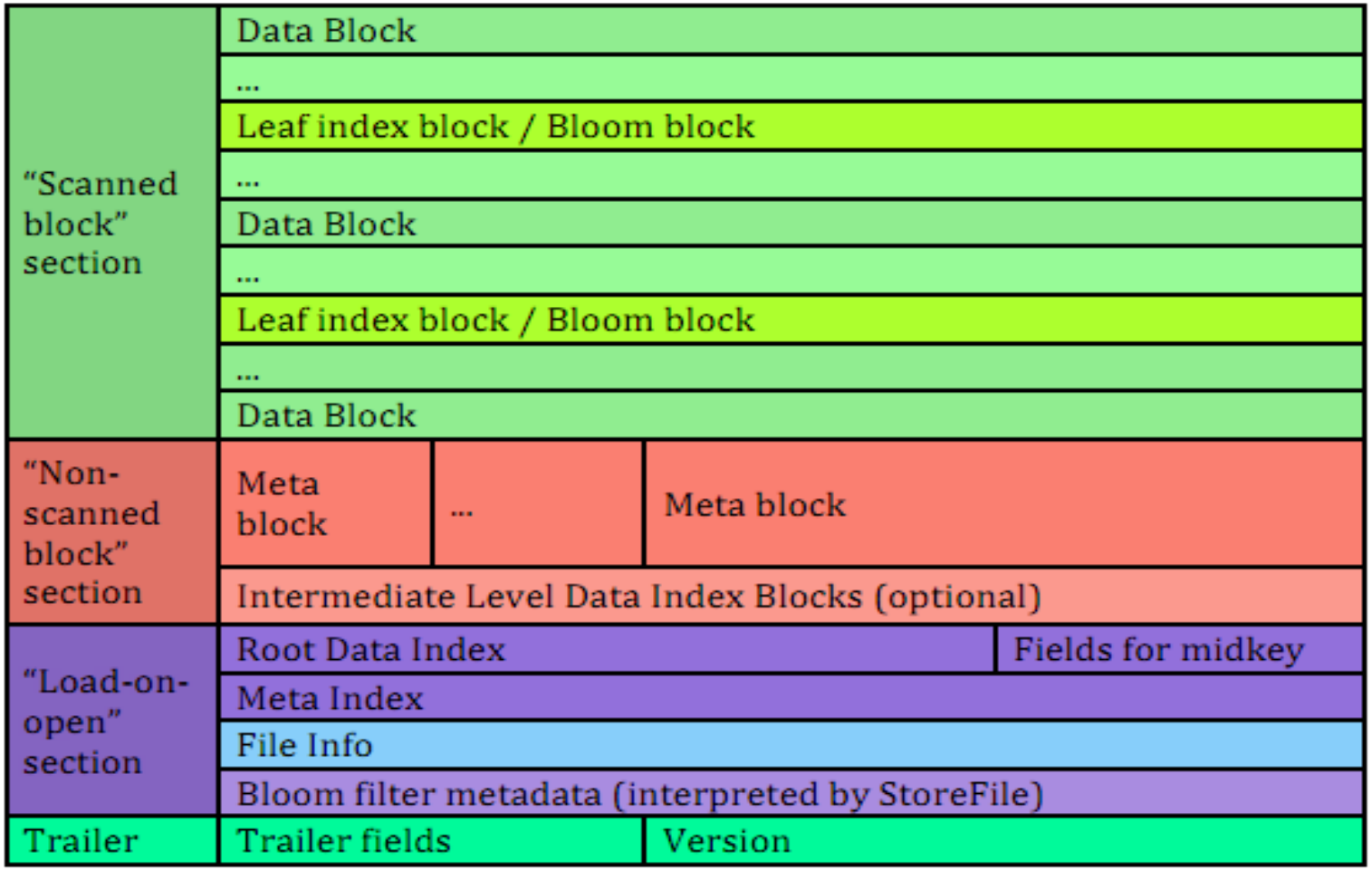
  ROOT_INDEX – Индекс верхнего уровня

  FILE_INFO – информация о файле

  BLOOM_META – Фильтр Блюма всего файла

  TRAILER – завершающий блок фиксированного размера

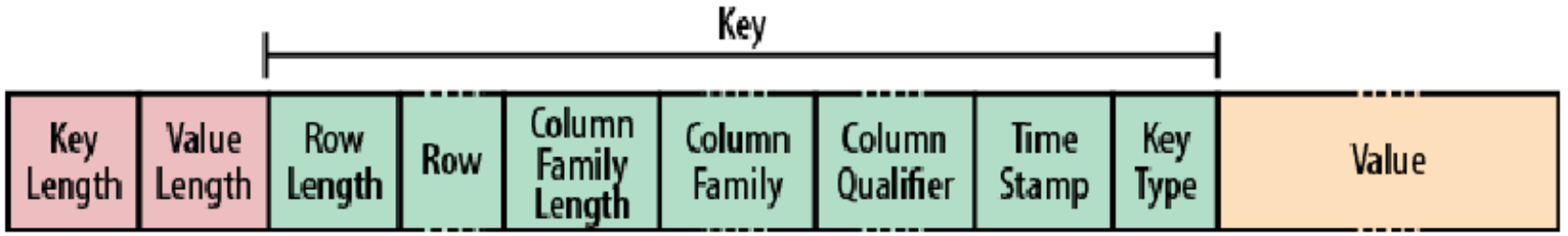
HFile


"Scanned block" section
Data Block
...
Leaf index block / Bloom block
...
Data Block
...
Leaf index block / Bloom block
...
Data Block
"Non-scanned block" section
Meta block
...
Meta block
Intermediate Level Data Index Blocks (optional)
"Load-on-open" section
Root Data Index
Fields for midkey
Meta Index
File Info
Bloom filter metadata (interpreted by StoreFile)
Trailer
Trailer fields
Version

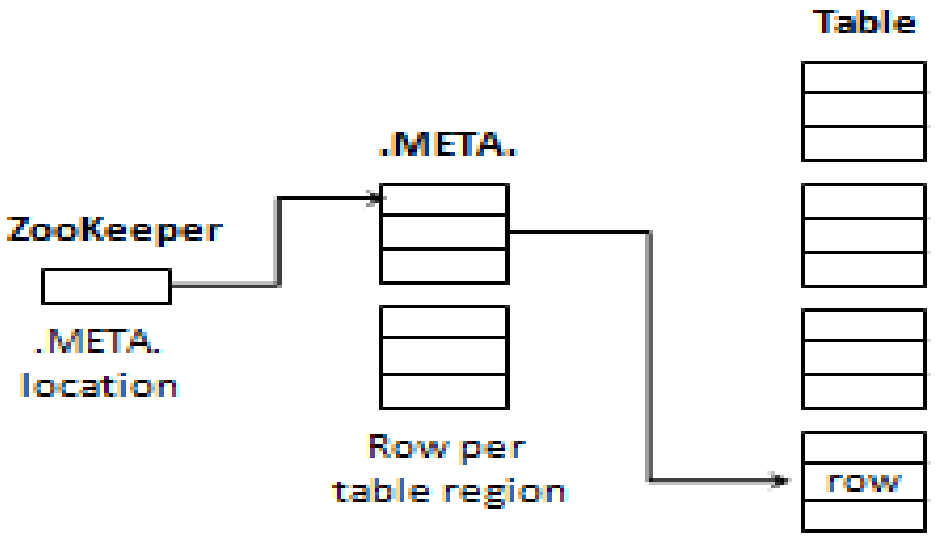

Block Type
long
Compressed size
int
Uncompressed Size
int
Offset Prev Block
long
Data

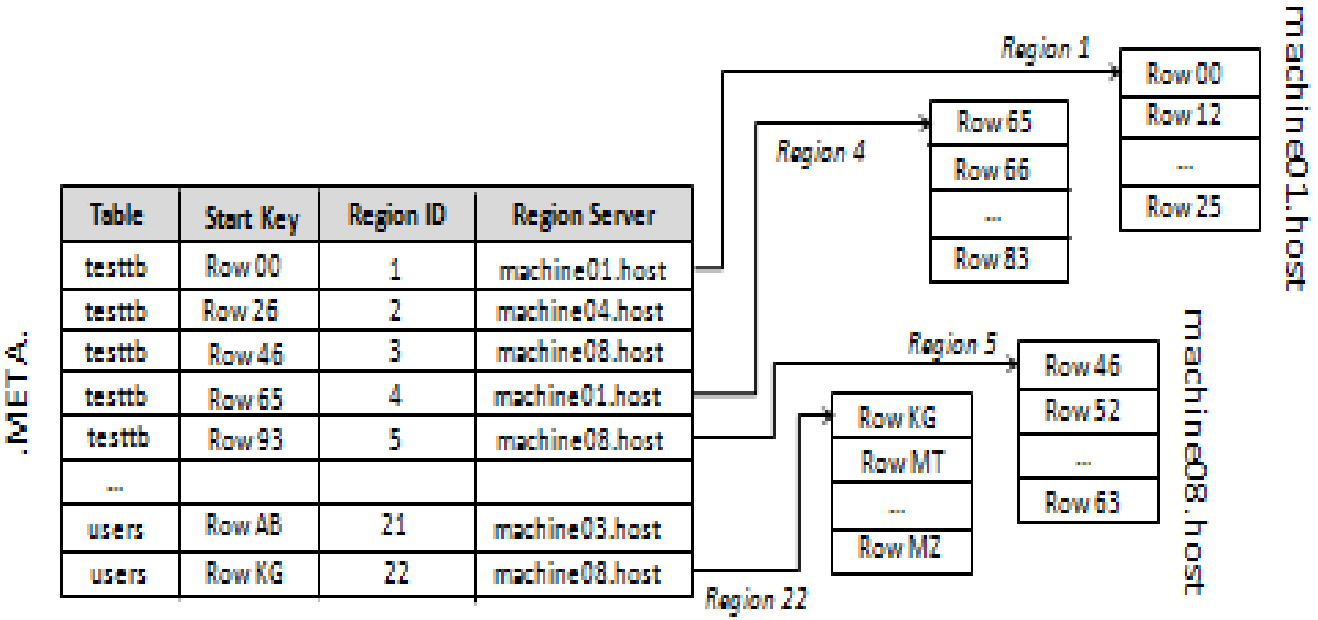
KeyValue

- Каждое значение столбца строки HBase лежит в структуре KeyValue
- Представляет собой отображение
- (row key, (column family+name), timestamp) → value



Чтение данных





Сканирование данных

- С помощью времени последнего изменения и Bloom Filter выбираются MemStore и HFile где осуществляется поиск
- Используется block index для позиционирования курсора на нужный блок в каждом из HFile и MemStore
- Осуществляется последовательное сканирование от поздних записей к старым, пока не перестанем удовлетворять условию сканирования.
- Учитываем индикатор удаленной записи

HBase API.Table

- Connection – подключение к hbase

  org.apache.hadoop.hbase.client.Connection

  Дорогая операция создания, поддерживает один поток
- Table - Базовый класс для работы с таблицами HBase

  org.apache.hadoop.hbase.client.Table
- Пример

```
Configuration config = HBaseConfiguration.create();
config.set("hbase.zookeeper.quorum", "localhost");
Connection connection = ConnectionFactory.createConnection();
Table testtable = connection.getTable(TableName.valueOf("testtable"));
```

Put

- Создаем объект
  - Put(byte[] row)
  - Put(byte[] row, long ts)
- Добавляем KeyValue
  - Put add(byte[] family, byte[] qualifier, byte[] value)
  - Put add(byte[] family, byte[] qualifier, long ts, byte[] value)
  - Put add(KeyValue kv) throws IOException
- Вызываем HBase
  - table.put(put);

Put. Пример

```
Configuration config = HBaseConfiguration.create();
config.set("hbase.zookeeper.quorum", "localhost");
Connection connection = ConnectionFactory.createConnection();
Table testtable = connection.getTable(TableName.valueOf("testtable"));
Put put1 = new Put(Bytes.toBytes( "row2"));
put1.addColumn(Bytes.toBytes("colfam1"), Bytes.toBytes("column1"), Bytes.toBytes("value"));
testtable.put(put1);
testtable.close();
 connection.close();
```

Get

- Создаем объект
  - Get(byte[] row)
- Заполняем требуемые поля и данные
  - addFamily(byte[] family)
  - addColumn(byte[] family, byte[] qualifier)
  - setTimeRange(long minStamp, long maxStamp) throws IOException
  - setTimeStamp(long timestamp)
  - setMaxVersions(int maxVersions) throws IOException
- Вызываем Hbase и обрабатываем ответ
  - Result result = table.get(get);
  - byte[] val = result.getValue(Bytes.toBytes("colfam1"), Bytes.toBytes("qual1"));

Get. Пример

```
Configuration conf = HBaseConfiguration.create();
HTable table = new HTable(conf, "testtable");
Get get = new Get(Bytes.toBytes("row1"));
get.addColumn(Bytes.toBytes("colfam1"),
                        Bytes.toBytes("qual1"));
Result result = table.get(get);
byte[] val =
result.getValue(Bytes.toBytes("colfam1"),
                        Bytes.toBytes("qual1"));
```

Delete

- Объект
  - Delete(byte[] row)
- Параметры
  - Delete deleteFamily(byte[] family)
  - Delete deleteFamily(byte[] family, long timestamp)
  - Delete deleteColumns(byte[] family, byte[] qualifier)
  - Delete deleteColumns(byte[] family, byte[] qualifier, long timestamp)
  - Delete deleteColumn(byte[] family, byte[] qualifier)
  - Delete deleteColumn(byte[] family, byte[] qualifier, long timestamp)
  - void setTimestamp(long timestamp)
- Вызов
  - table.delete(delete);

Delete.Пример

```
Delete delete = new Delete(Bytes.toBytes("row1"));
delete.setTimestamp(1);
delete.deleteColumn(Bytes.toBytes("colfam1"), Bytes.toBytes("qual1"), 1);
delete.deleteColumns(Bytes.toBytes("colfam2"), Bytes.toBytes("qual1"));
delete.deleteColumns(Bytes.toBytes("colfam2"), Bytes.toBytes("qual3"),
15);
delete.deleteFamily(Bytes.toBytes("colfam3"));
delete.deleteFamily(Bytes.toBytes("colfam3"), 3);
table.delete(delete);
table.close();
```

RowMutations

- Применяет набор команд к одной строке
- Атомарная операция
  - RowMutations(byte[] row)
  - void add(Delete d)
  - void add(Put p)
- Вызов
  - Table.mutateRow(RowMutations rm)

checkAndPut, checkAndDelete

- Аналог оптимистической блокировки.
- Сравнивает значение KeyValue с заданным, и если значения совпали, присваивает новое/удаляет cell

  boolean checkAndPut(byte[] row, byte[] family, byte[] qualifier, byte[] value, Put put)

  boolean checkAndDelete(byte[] row, byte[] family, byte[] qualifier, byte[] value, Delete delete)
- Используется в ситуациях, когда мы должны быть точно уверены что никто не изменил исходные данные
- Пример такой ситуации : изменение баланса абонента

increment, incrementColumnValue

- Данная команда предназначена для атомарного увеличения записи (например счетчика)

  increment(Increment increment)

  incrementColumnValue(byte[] row, byte[] family, byte[] qualifier, long amount)

- Increment увеличивает сразу несколько значений ячеек атомарно, но так как сканер не учитывает блокировки, то он может увидеть промежуточный результат