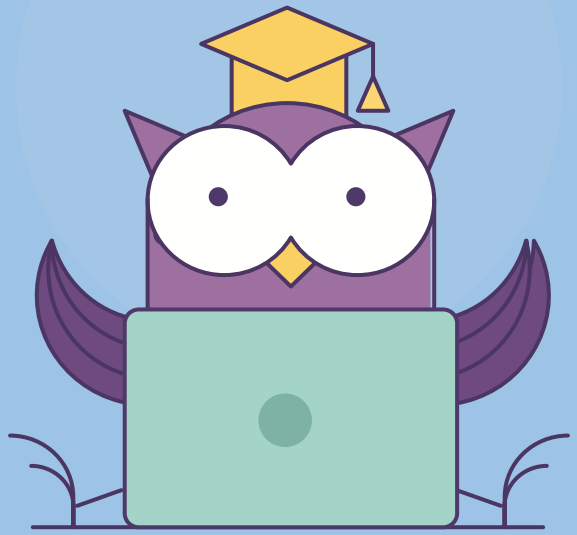
OTUS

Инженер Данных.
SQL-доступ. Apache Hive.

Меня хорошо слышно && видно?



Напишите в чат, если есть проблемы!

Ставьте + если все хорошо
Ставьте - если есть проблемы

План занятия

- SQL on Hadoop и Hive
- Архитектура Hive
- Модель данных Hive
- Стратегии Join Hive
- Оптимизация Hive
- Практика

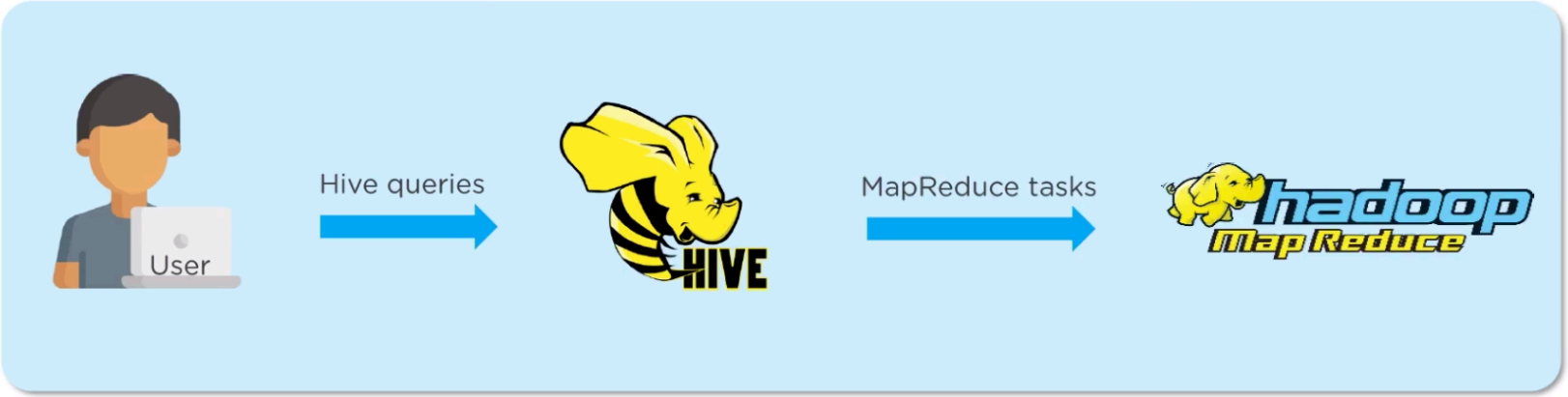
SQL on Hadoop и Hive

SQL on Hadoop

- Apache Hive
- Presto
- Cloudera Impala
- Spark SQL
- Apache Drill

Что такое Hive

- Работает поверх Hadoop
- Предоставляет SQL-подобный язык (HiveQL)
- Разработан с учетом масштабируемости и легкости использования
- Разработан в Facebook в 2007 году



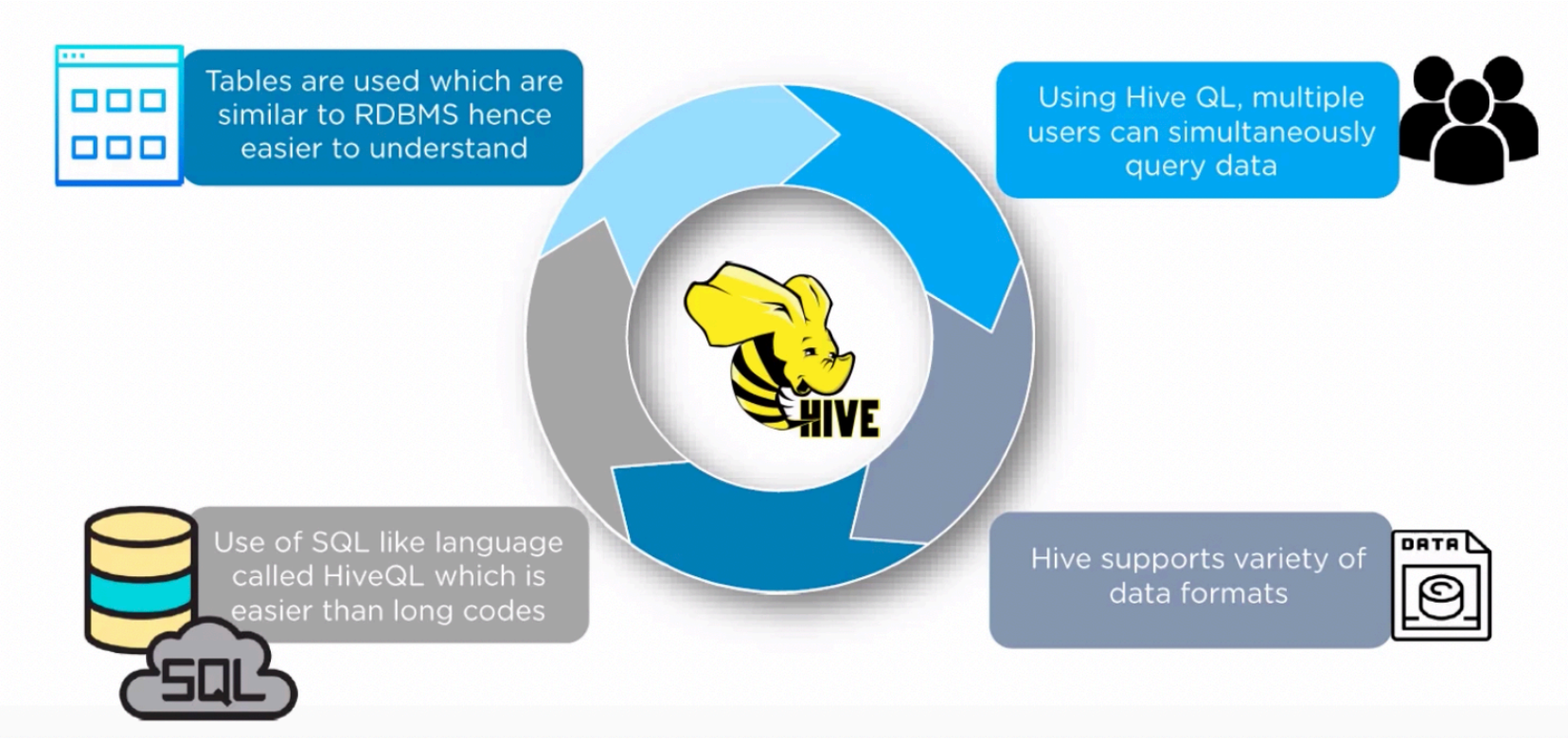
Hive предоставляет

- Возможность структурировать различные форматы данных
- Простой интерфейс для запросов, анализа и обобщения больших объемов данных
- Доступ к данных из различных источников, таких как HDFS и HBase

Чем Hive не является

- Hive:
  - Не является РСУБД
  - Не предназначен для OLTP-нагрузки
  - Не подходит для real-time & row-level UPDATES
- Запрос даже небольших объемов данных может занять минуты

Hive особенности



Hive возможности

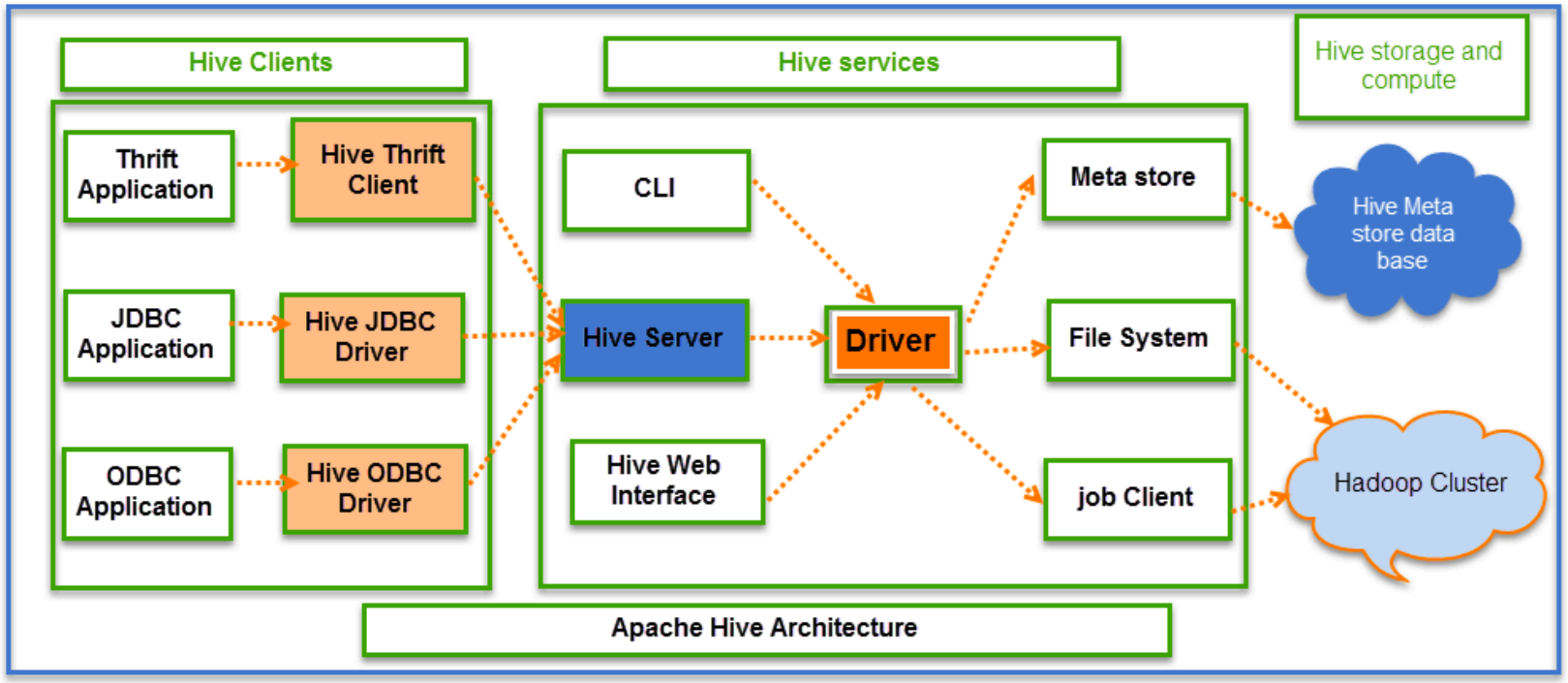
- DDL: TABLE (Internal / External), VIEW, INDEX
- SELECT, WHERE, GROUP BY, ORDER BY, JOINS, NESTED QUERIES, INSERT
- Составные типы данных
- Partitioning, Bucketing, Sampling
- UDF: UDF, UDAF, UDTF
- Различные опции загрузки данных LOAD
- Pluggable Input / Output format
- Pluggable SerDe libs

HIVE vs РСУБД

РСУБД	HIVE
Schema on Write	Schema on Read
Write / Read Many	Write once Read Many
~ 10-100 GB	~ 1TB – 1PB
Proprietary Hardware	Commodity Hardware
UPDATEs allowed	UPDATEs not recommended
Реляционная модель	Хранилище с поддержкой SQL

Архитектура Hive

Архитектура Hive



Интерфейсы Hive

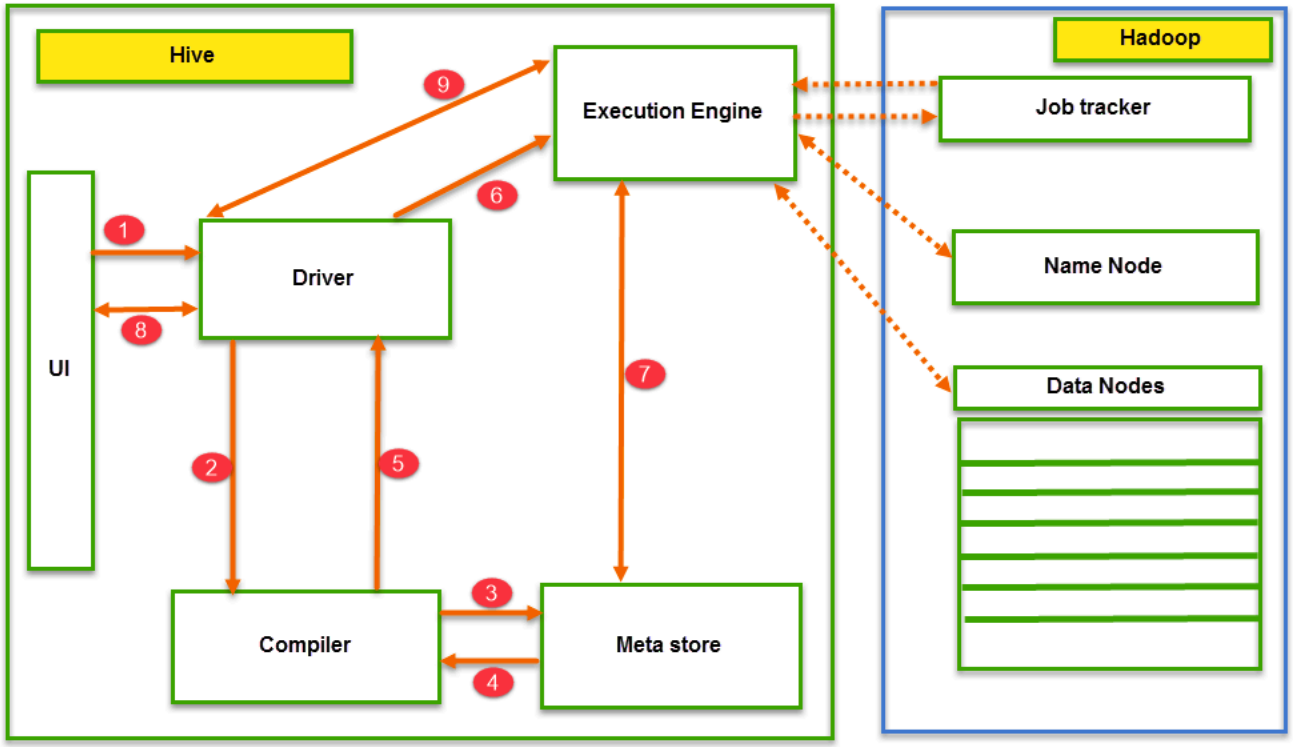
- Command Line Interface (CLI)
  - Hive
  - Hive2 (Beeline)
- Hive Web Interface
- Java Database Connectivity (JDBC)

Метаданные Hive

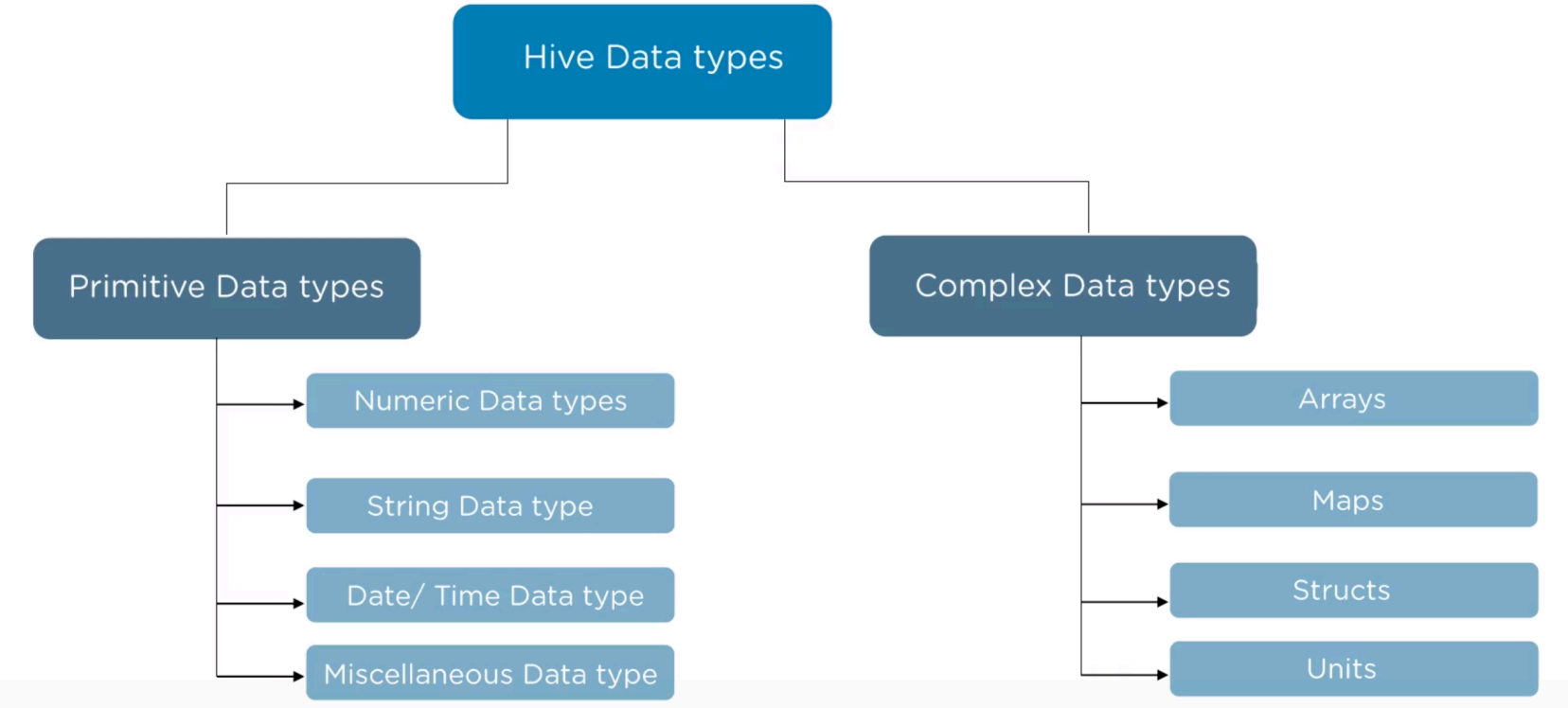
Hive хранит мета-информацию в реляционной БД

- Поставляется с Derby, “легковесной” встроенной SQL DB
- Можно относительно легко переключиться на другу БД, например, MySQL

Hive Data Flow



Типы данных в Hive



Типы данных в Hive

```
CREATE TABLE IF NOT EXISTS employee_internal (
  name string,
  work_place ARRAY<string>,
  gender_age STRUCT<gender:string,age:int>,
  skills_score MAP<string,int>,
  depart_title MAP<STRING,ARRAY<STRING>>
)
COMMENT 'This is an internal table'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
COLLECTION ITEMS TERMINATED BY ','
MAP KEYS TERMINATED BY ':'
STORED AS TEXTFILE;
```

employee_external.name	employee_external.work_place	employee_external.gender_age	employee_external.skills_score	employee_external.depart_title
Michael	["Montreal","Toronto"]	{"gender":"Male","age":30}	{"DB":80}	{"Product":["Developer^DLead"]}
Will	["Montreal"]	{"gender":"Male","age":35}	{"Perl":85}	{"Product":["Lead"],"Test":["Lead"]}
Shelley	["New York"]	{"gender":"Female","age":27}	{"Python":80}	{"Test":["Lead"],"COE":["Architect"]}
Lucy	["Vancouver"]	{"gender":"Female","age":57}	{"Sales":89,"HR":94}	{"Sales":["Lead"]}

Hive File Formats

- TEXT
- JSON
- AVRO
- ORC
- PARQUET

Hive external table

- Hive управляет метаданными
- При удалении таблицы удаляются только метаданные

```
106   --Create external table and load the data
107   CREATE EXTERNAL TABLE IF NOT EXISTS employee_external (
108       name string,
109       work_place ARRAY<string>,
110       gender_age STRUCT<gender:string,age:int>,
111       skills_score MAP<string,int>,
112       depart_title MAP<STRING,ARRAY<STRING>>
113   )
114   COMMENT 'This is an external table'
115   ROW FORMAT DELIMITED
116   FIELDS TERMINATED BY '|'
117   COLLECTION ITEMS TERMINATED BY ','
118   MAP KEYS TERMINATED BY ':'
119   STORED AS TEXTFILE
120   LOCATION '/user/cloudera/employee';
```

Hive internal / managed table

- Hive управляет схемой и данными
- По умолчанию данные хранятся в /usr/hive/warehouse/my_managed_table
- Удаляются метаданные и сами данные

```
89   --Create internal table and load the data
90   CREATE TABLE IF NOT EXISTS employee_internal (
91     name string,
92     work_place ARRAY<string>,
93     gender_age STRUCT<gender:string,age:int>,
94     skills_score MAP<string,int>,
95     depart_title MAP<STRING,ARRAY<STRING>>
96   )
97   COMMENT 'This is an internal table'
98   ROW FORMAT DELIMITED
99   FIELDS TERMINATED BY '|'
100  COLLECTION ITEMS TERMINATED BY ','
101  MAP KEYS TERMINATED BY ':'
102  STORED AS TEXTFILE;
```

Hive Views

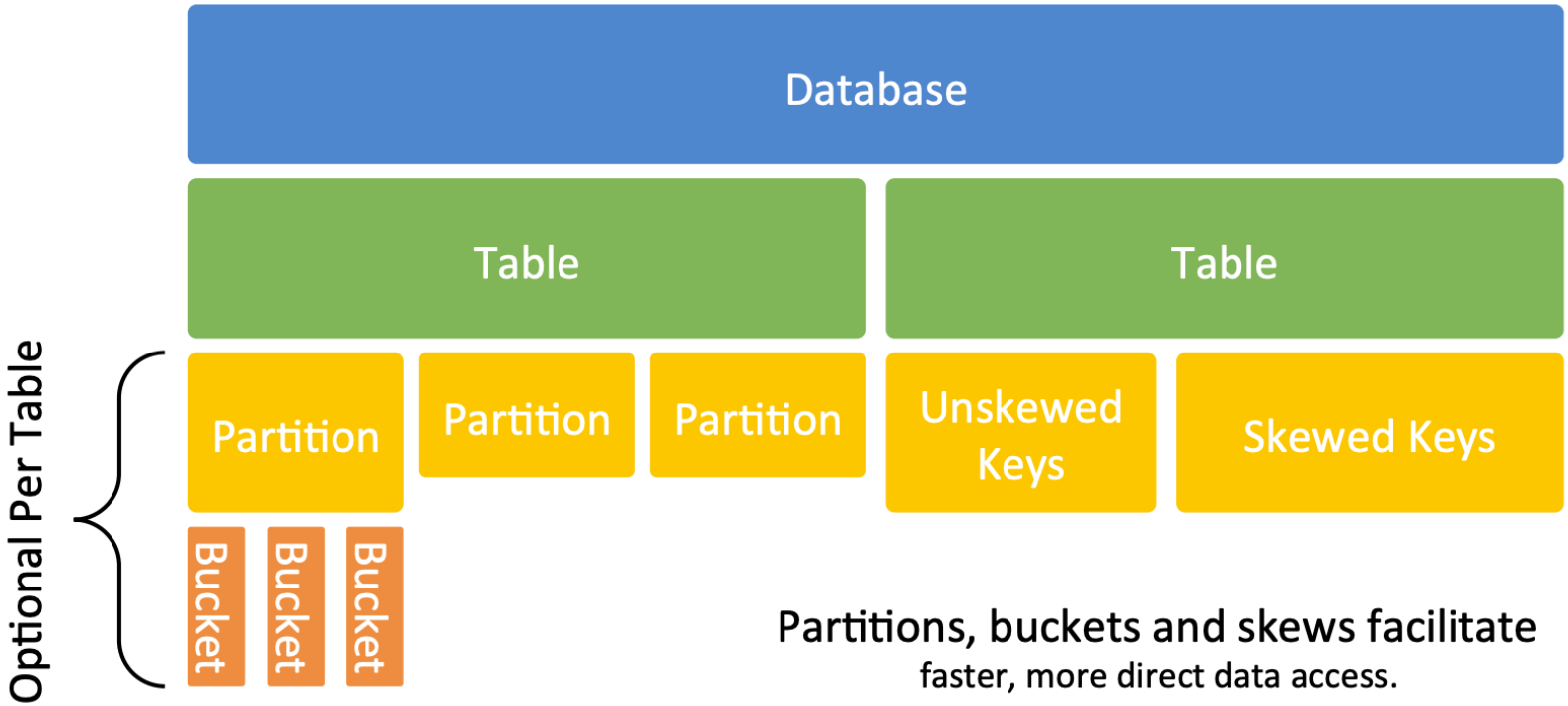
- Виртиуальная таблица
- Данные не хранятся физически в HDFS
- Удаляются только метаданные

```
CREATE VIEW my_view
            (
                        'id' int,
                        'name' string,
                        'department' string,
                        'country' string,
            )
            AS {select_statement};
```

Модель данных Hive

Partitioning, Bucketing and Skews

Модель данных Hive



Partitioning

- Горизонтальное разделение
- 1+ колонок
- Будут созданы новые директории
- Запросы использующие партиции будут быстрее
- Необходимо решить, по каким корлонкам делить

Static Partitioning

- По выбранным значениям

```
CREATE TABLE static_partioned_table
                (
                                'id' int,
                                'name' string,
                                'department' string
                )
                PARTITIONED BY ('country' string)
                ROW FORMAT DELIMETED FIELDS TERMINATED BY ','
                STORED AS ORCFile;

INSERT OVERWRITE TABLE static_partioned_table
PARTITION (country='canada')
SELECT id, name, department
FROM my_external_table
WHERE country='canada'
```

Dynamic Partitioning

- По всем значениям столбца

```
CREATE TABLE dynamic_partioned_table
                (
                                'id' int,
                                'name' string,
                                'department' string
                )
                PARTITIONED BY ('country' string)
                ROW FORMAT DELIMETED FIELDS TERMINATED BY ','
                STORED AS ORCFile;

INSERT OVERWRITE TABLE dynamic_partioned_table
PARTITION (country)
SELECT id, name, country
FROM my_external_table;
```

Partitioning

- Необходимо включить настройки

```
set hive.exec.dynamic.partition=true;
set hive.exec.dynamic.partition.mode=nonstrict;
set hive.exec.max.dynamic.partitions=1000;
set hive.exec.max.dynamic.partitions.pernode=1000;
```

Partitioning

- Мультиуровневое партиционирование возможно, но не всегда эффективно
  - Большое количество может отрицательно сказаться на Metastore
- Ограничить количество партиций
  - 1000 лучше чем 10000
- Hadoop любит большие файлы
  - Не создавайте партиции с маленькими файлами
- Используйте партиции если:
  - Данных очень много
  - В запросах часто используются определенные колонки
  - Колонки с небольшим количеством значений (кардинальность)

Partitioning

- Лучше использовать Дату, не Год-Месяц

```
SELECT * FROM TableA WHERE DateStamp IN (‘2015-01-01’, ‘2015-02-03’, ‘2016-01-01’)

vs

SELECT * FROM TableB WHERE (YEAR=2015 AND MONTH=01 AND DAY=01) OR (YEAR=2015 AND MONTH=02 AND
DAY=03) OR (YEAR=2016 AND MONTH=01 AND DAY=01)
```

Bucketing

- Делит данные вертикально с использованием хэш-функции
- Указывается количество buckets
- Особенно полезен при использовании Joins

```
CREATE TABLE bucketed_table
        (
                'id' int,
                'name' string,
                'department' string,
                'country' string
        )
        CLUSTERED BY (id) INTO 12 BUCKETS
        ROW FORMAT DELIMETED FIELDS TERMINATED BY ','
        STORED AS ORC;
```

Bucketing

- Необходимо явно указывать в настройках

```
SET hive.enforce.bucketing = true;
SET hive.exec.dynamic.partition.mode=nonstrict;

INSERT INTO TABLE sale (xdate, state)
SELECT * FROM staging_table;
```

Bucketing

- Partitioning & Bucketing можно использовать вместе
- Идеальный размер файлов 200-500 МБ
- Используйте одинаковые подходы Partition / Bucket для соединяемых таблиц

```
CREATE TABLE sale (
   id int, amount decimal, ...
) PARTITIONED BY (xdate string, state string)
CLUSTERED BY (id) SORTED BY (id) INTO 256 buckets;
```

Skewed Tables / List Bucketing

- Можно явно указать значения с перекосом в данных
- Hive разделит значения с перекосом и все остальные в разные файлы физически

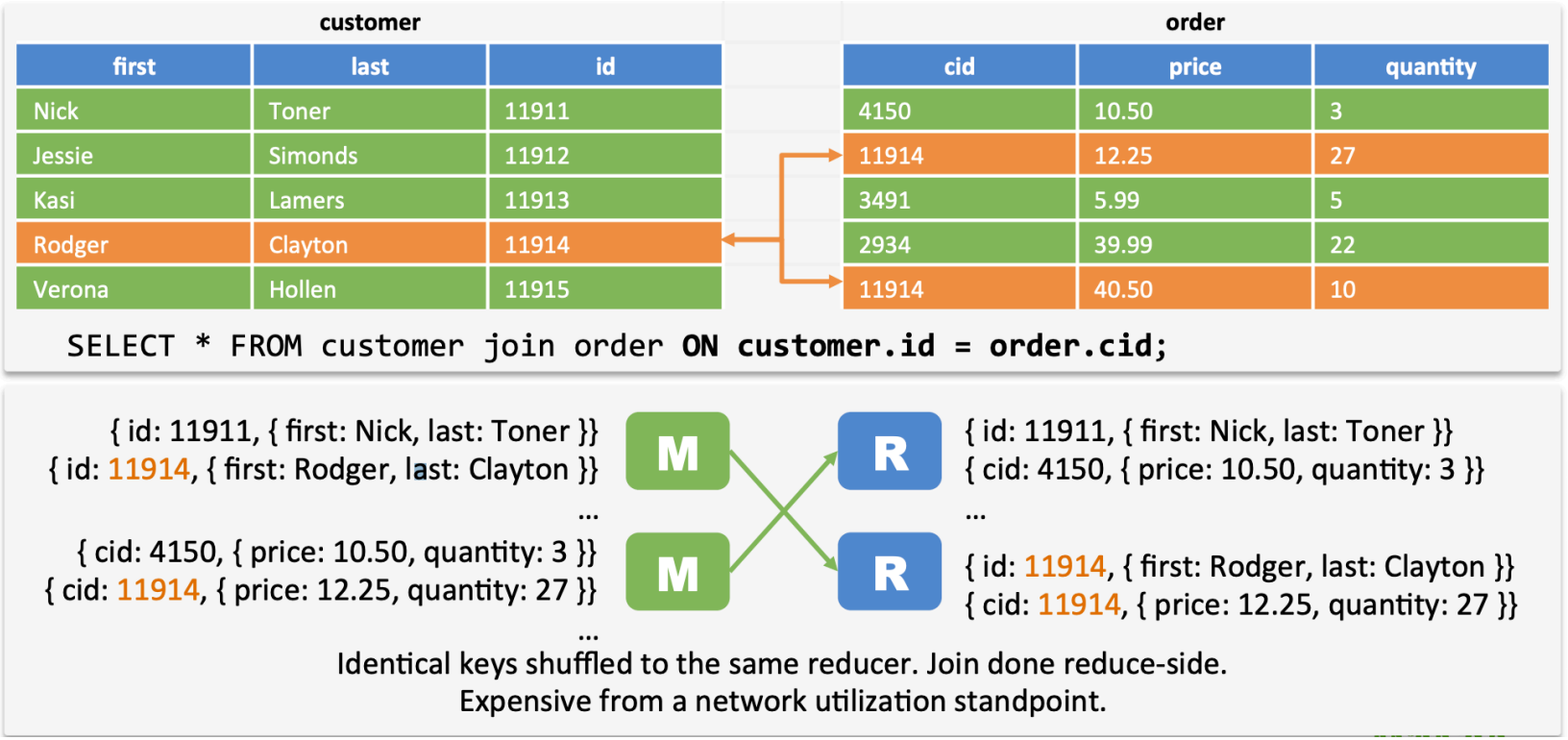
```
CREATE TABLE mytable (
    key STRING, value STRING, …
) SKEWED BY (key) ON ('key1', 'key2') STORED AS DIRECTORIES;
```

Partitioning / Bucketing / Skew best practices

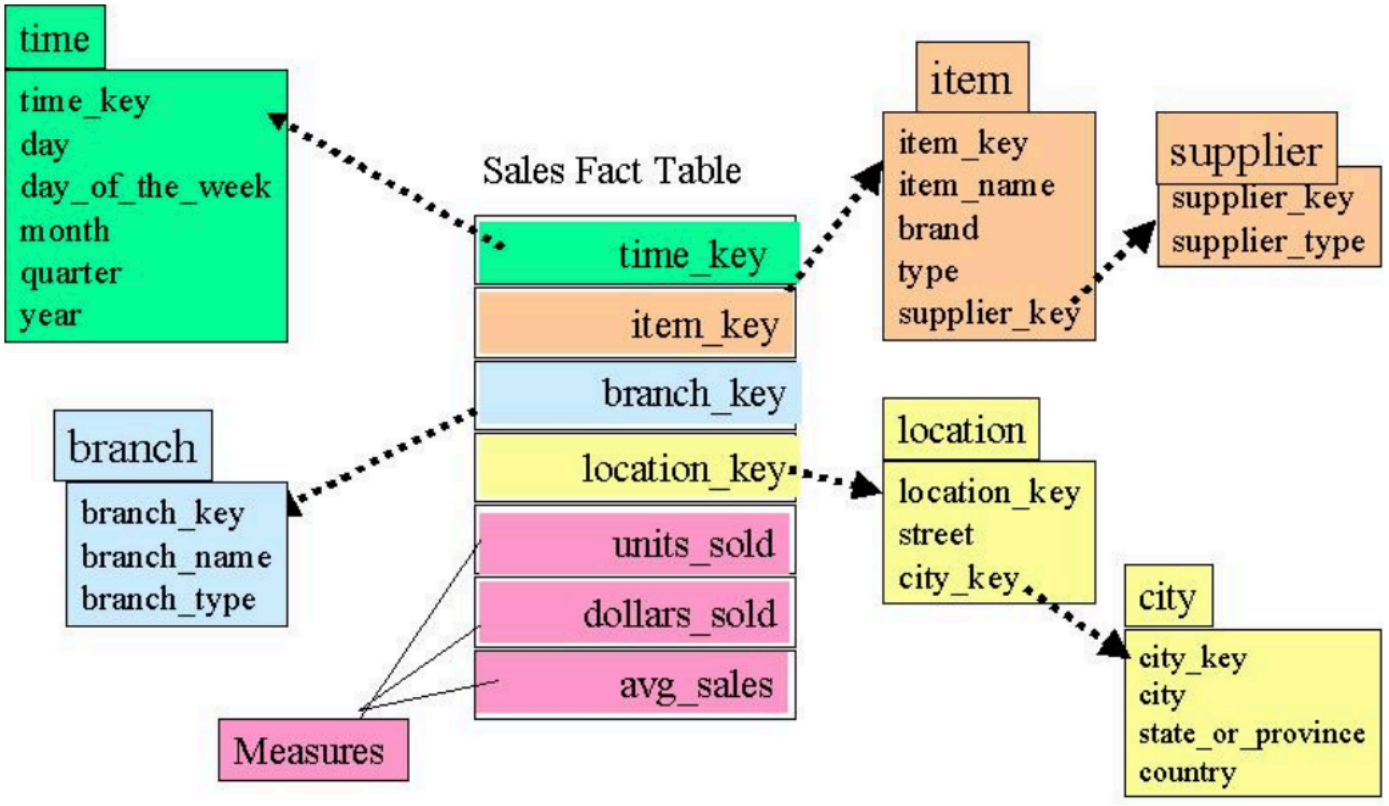
- Партиционирование полезно по хронологическим колонкам с небольшим количеством значений
- Bucketing полезен для таблиц участвующих в соединении по одному ключу
- Skew полезен для данных в которых есть явные перекосы

Стратегии Join Hive

Shuffle Joins



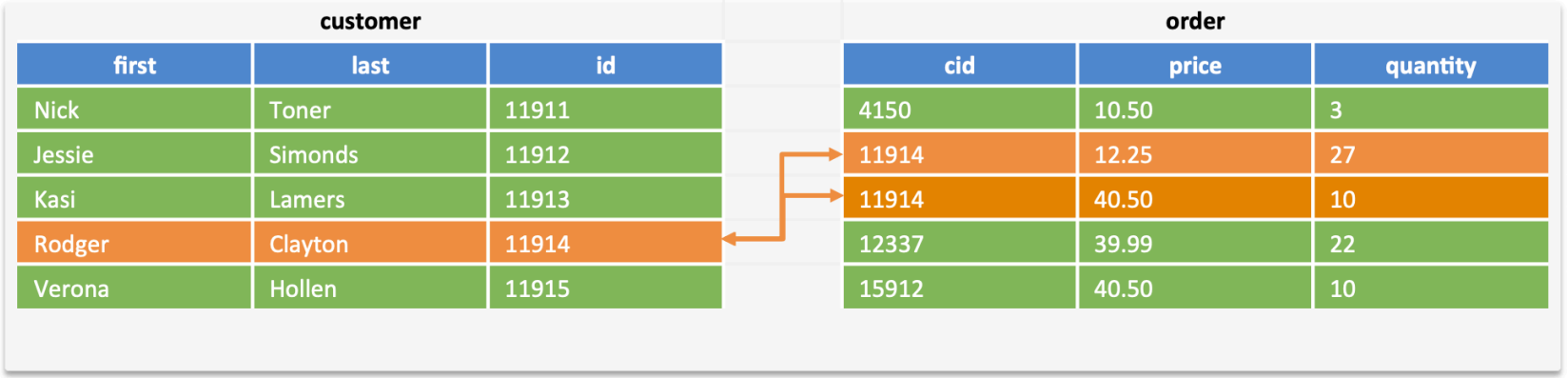
Broadcast Join (Map-side Join)



Broadcast Join (Map-side Join)

- Установить hive.auto.convert.join = true
- HIVE будет использовать broadcast join там где возможно
  - Небольшие таблицы, которые помещаются в память
- Настройка hive.auto.convert.join.noconditionaltask.size определяет размер данных для broadcast join:
  - По умолчанию 10MB
  - Рекомендуется 256MB

Sort-Merge-Bucket join



customer

first	last	id
Nick	Toner	11911
Jessie	Simonds	11912
Kasi	Lamers	11913
Rodger	Clayton	11914
Verona	Hollen	11915

order

cid	price	quantity
4150	10.50	3
11914	12.25	27
11914	40.50	10
12337	39.99	22
15912	40.50	10

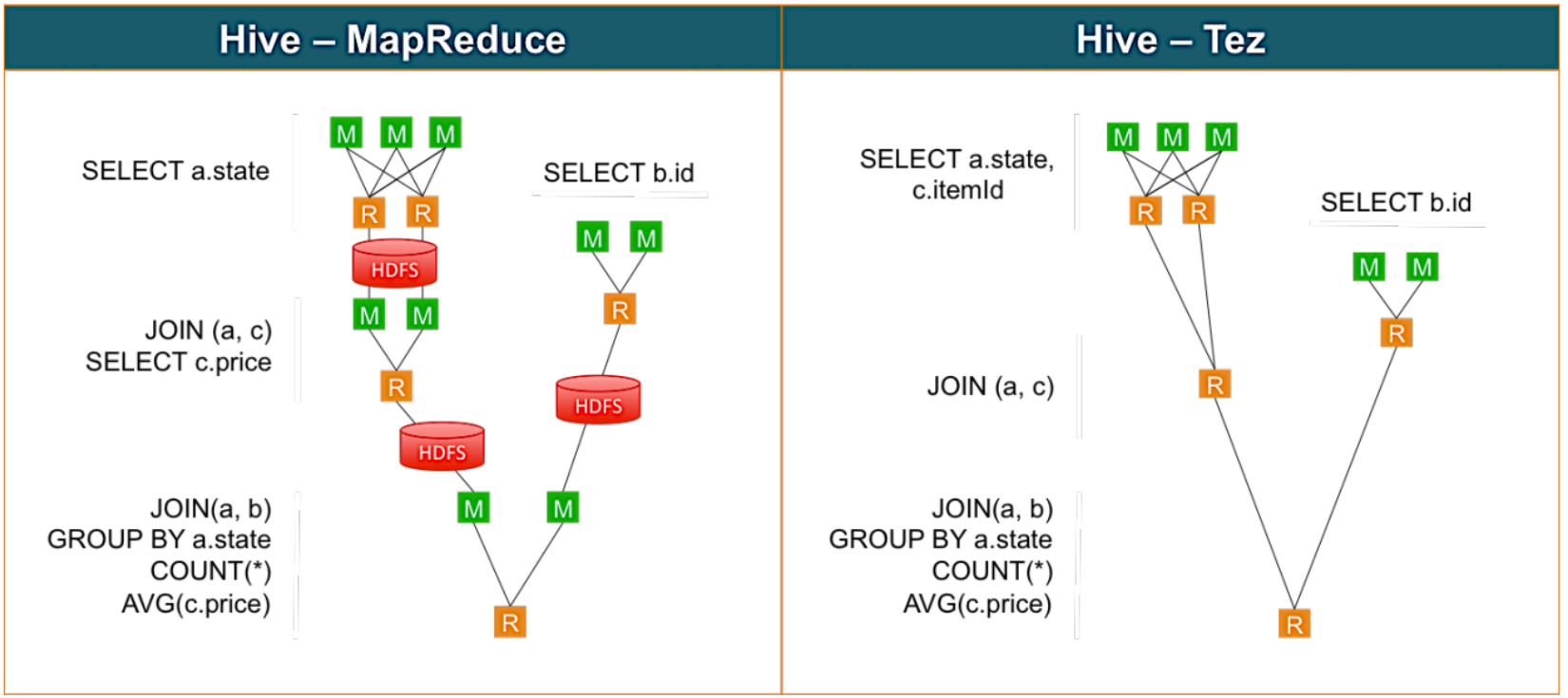
```
CREATE TABLE order (cid int, price float, quantity int)
CLUSTERED BY(cid) SORTED BY(cid) INTO 32 BUCKETS;

CREATE TABLE customer (id int, first string, last string)
CLUSTERED BY(id) SORTED BY(id) INTO 32 BUCKETS;

SELECT * FROM customer join order ON customer.id = order.cid;
```

Оптимизация Hive

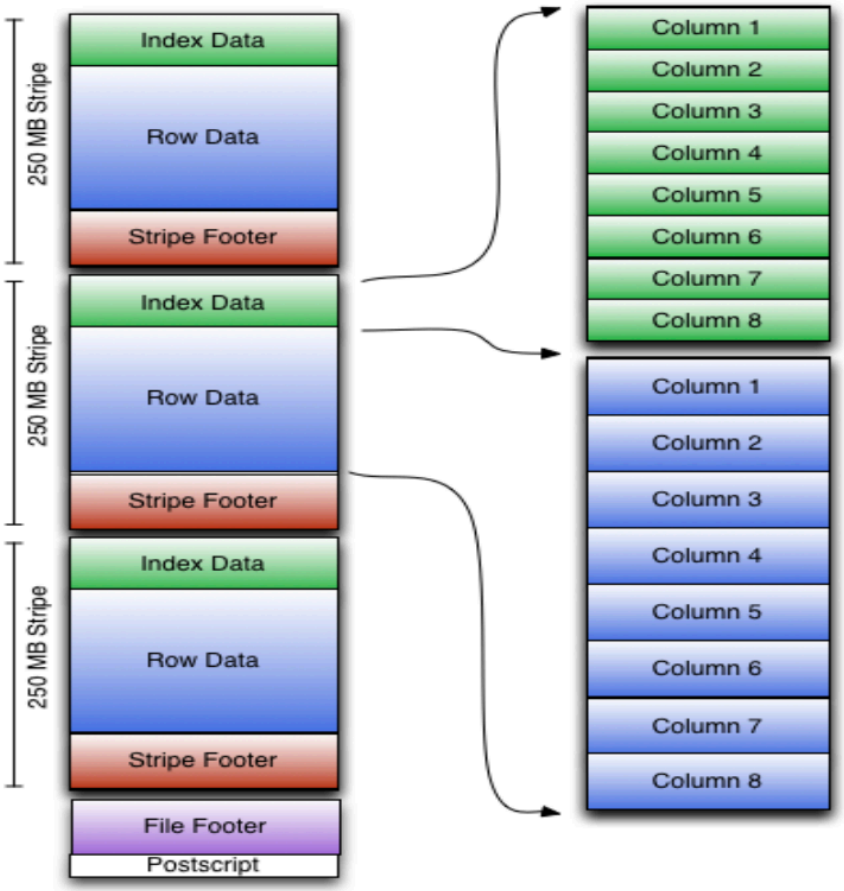
Оптимизация HIVE

- Tez
- ORCFile
- Vectorization
- Use Cost Based Optimization (CBO)
- Хорошие SQL-запросы
- Читать Hive Explain
- Использовать Hive LLAP

Tez


Hive – MapReduce
SELECT a.state
JOIN (a, c)
SELECT c.price
JOIN(a, b)
GROUP BY a.state
COUNT(*)
AVG(c.price)
SELECT b.id
M
R
HDFS
Hive – Tez
SELECT a.state,
c.itemId
JOIN (a, c)
JOIN(a, b)
GROUP BY a.state
COUNT(*)
AVG(c.price)
SELECT b.id

ORC


250 MB Stripe
Index Data
Row Data
Stripe Footer
250 MB Stripe
Index Data
Row Data
Stripe Footer
250 MB Stripe
Index Data
Row Data
Stripe Footer
File Footer
Postscript
Column 1
Column 2
Column 3
Column 4
Column 5
Column 6
Column 7
Column 8
Column 1
Column 2
Column 3
Column 4
Column 5
Column 6
Column 7
Column 8

Vectorization

- Векторизованные вычисления для основных операций: scans, filters, aggregates, and joins
- Процессинг блока данных в единицу времени (1024 строки вместо 1)
- Для ORC-файлов

```
SET hive.vectorized.execution.enabled = true;
SET hive.vectorized.execution.reduce.enabled=true;
```

Cost-Based Optimization (CBO)

- Наличие статистик для данных
  - На уровне таблицы
  - На уровне колонки

```
SET hive.cbo.enable=true;
SET hive.compute.query.using.stats = true;
```

```
Hive.stats.autogather=true;
```

```
ANALYZE TABLE table-name COMPUTE STATISTICS;
```

```
ANALYZE TABLE table-name COMPUTE STATISTICS for COLUMNS col1, col2;
```

```
ANALYZE TABLE table-name partition (col1='x') COMPUTE STATISTICS;
```

HIVE EXPLAIN

- Инструмент для получения деталей действий Hive
- Последовательность выполнения операций
- Убедиться в использовании правильных стратегий соединения таблиц

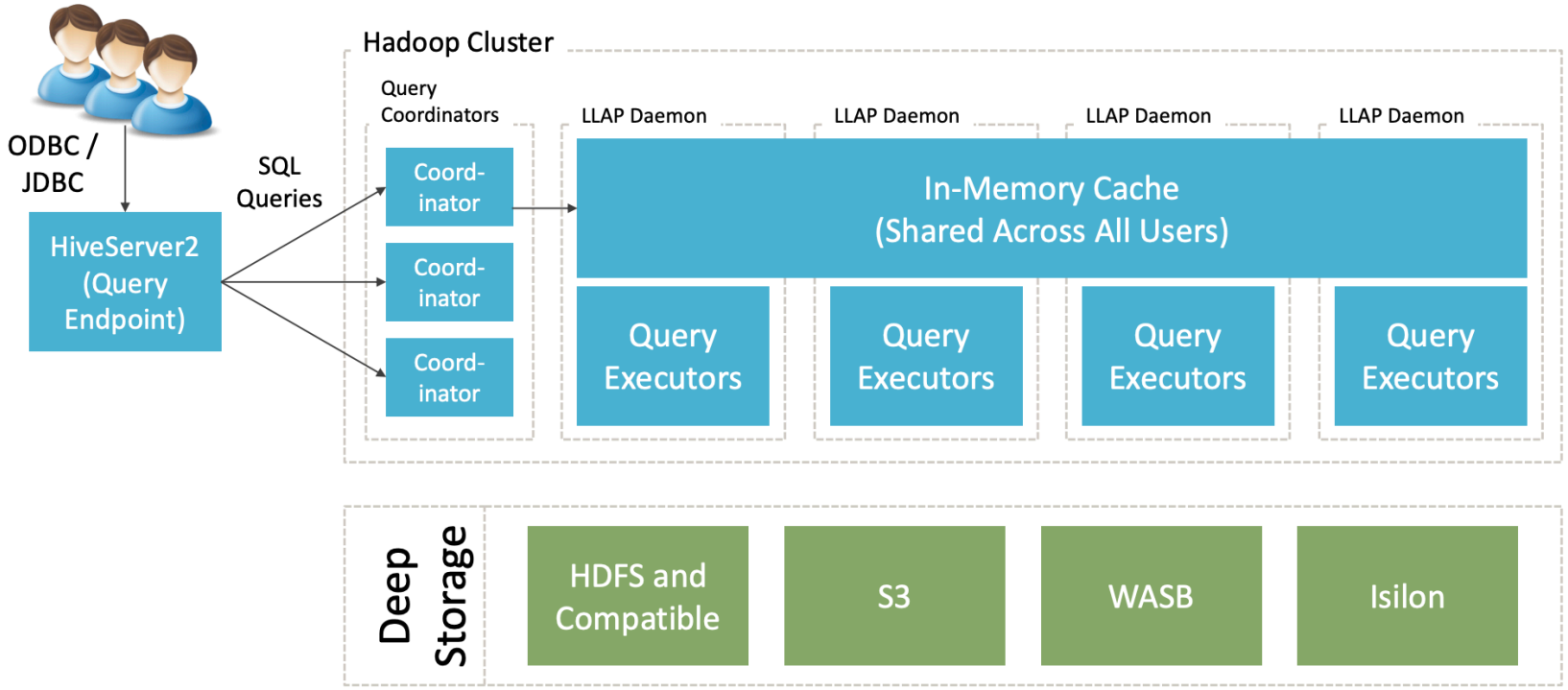
```
EXPLAIN {Hive Query}
```

Materialized views

```
SELECT distinct dest,origin
FROM flights;

SELECT origin, count(*)
FROM flights
GROUP BY origin
HAVING origin = 'OAK';
```

```
CREATE MATERIALIZED VIEW flight_agg
AS
   SELECT dest,origin,count(*)
   FROM flights
   GROUP BY dest,origin;
```

Hive LLAP


ODBC / JDBC
SQL Queries
HiveServer2 (Query Endpoint)
Hadoop Cluster
Query Coordinators
Coord-inator
Coord-inator
Coord-inator
LLAP Daemon
LLAP Daemon
LLAP Daemon
LLAP Daemon
In-Memory Cache (Shared Across All Users)
Query Executors
Query Executors
Query Executors
Query Executors
Deep Storage
HDFS and Compatible
S3
WASB
Isilon

Практика

Практика

- DDL
- DML
- Export / Import
- Data Aggregation / Window Functions

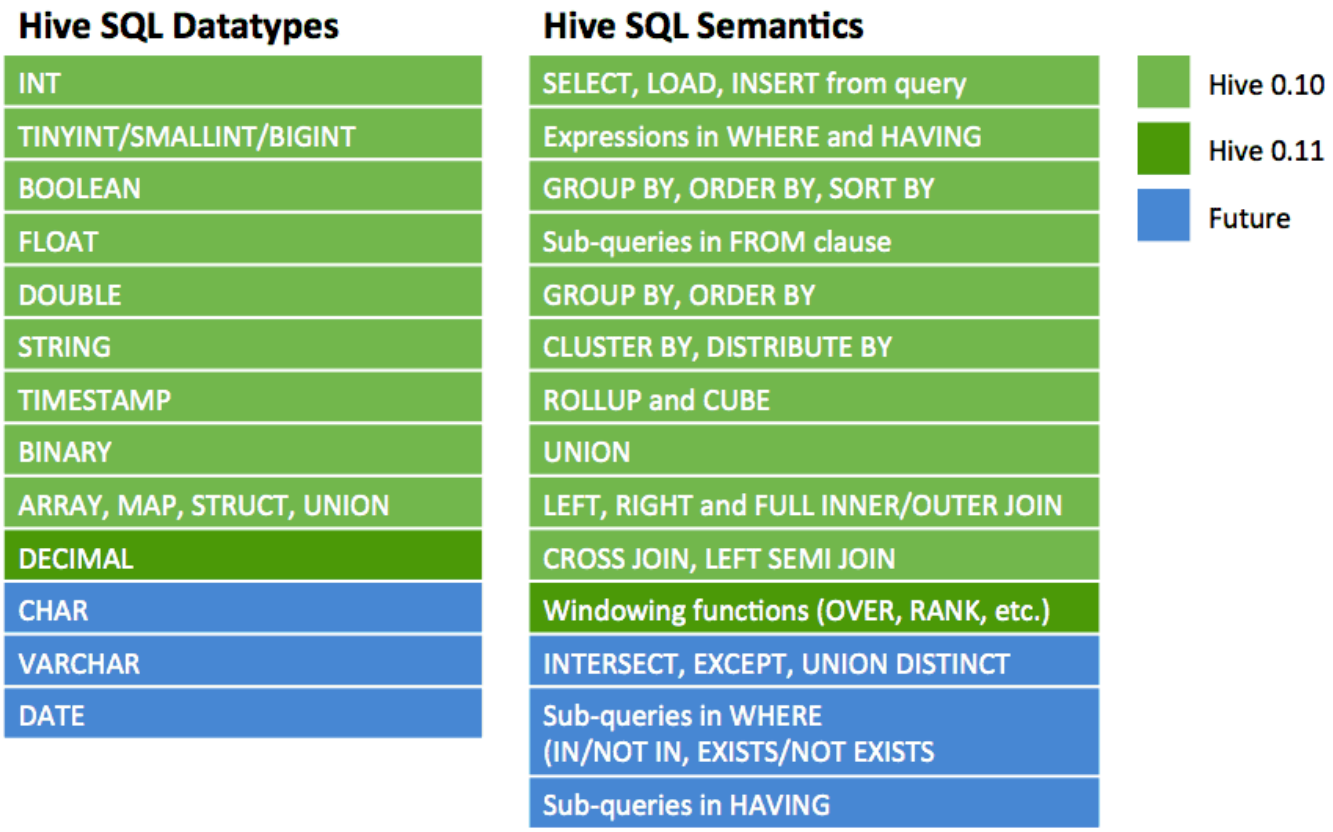
Результаты занятия

- SQL on Hadoop и Hive
- Архитектура Hive
- Модель данных Hive
- Стратегии Join Hive
- Оптимизация Hive
- Практика

Ссылки на материалы

- Hive Data Modeling and Query Optimization
- Hive Join Strategies
- How to understand and analyze Hive query plan
- Apache Hive 3: A New Horizon Meetup + Slides
- Hive LLAP Technical Preview
- AN APACHE HIVE BASED DATA WAREHOUSE

Hive cheatsheet



Домашнее задание

- Развернуть дистрибутив CDH
- Самостоятельно проделать манипуляции с Hive

Рефлексия

- Что вам запомнилось больше всего
- Пройти опрос

Ваши вопросы?