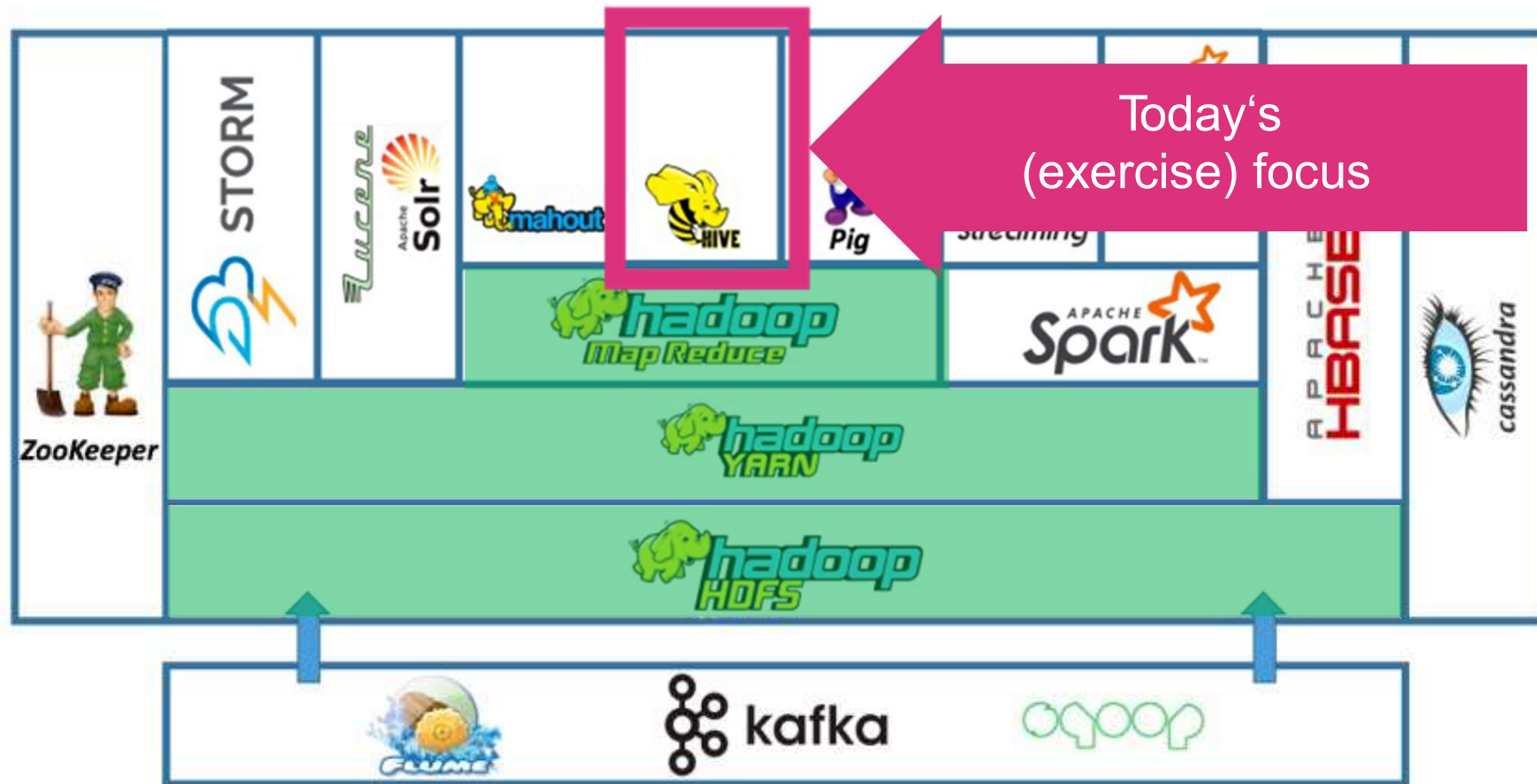


# Hive и HiveQL

## Краткое введение в Hive и HiveQL

# The Hadoop Ecosystem



# Apache Hive



## Apache Hive

- Initial Release in **October 2010**
- written in **Java**
- current Version: **3.1.2**

<http://hive.apache.org>

### - Hive YES:

- прост в использовании (HiveQL)
- на основе Hadoop (HDFS, YARN)
- хорошие возможности горизонтального масштабирования (например, путем разделения HDFS и YARN)

лучше всего использовать для:

- (BigData) задачи хранения данных
- озеро данных (a DataLake)
- интерфейс для аналитиков, специалистов по данным, разработчиков
- специальные (пакетные) запросы, агрегация и анализ больших объемов данных (PB!) и сотен узлов

### - Hive NOT:

- транзакционная база данных
- highly responsive

# HiveQL

- SQL like **query language**
- Поддерживается: **Hive CLI** (устарело), **Beeline CLI** и большинством клиентов **JDBC**.
- Поддерживаемые Hive форматы файлов HDFS :
  - **Text File** (даже сжатый gzip или bzip2)
  - **Sequence File**
  - **RC File**
  - **ORC**
  - **Parquet**
  - **Avro**

# HiveQL

## Hive Text File Format

**Hive Text** является форматом хранения по умолчанию. Используется текстовый формат для обмена данными с другим клиентским приложением. Данные хранятся в строках, каждая строка является записью. Каждая строка заканчивается символом новой строки (**\n**).

Текстовый формат представляет собой простой формат плоского файла. Вы можете использовать сжатие ( **BZIP2** ) в текстовом файле, чтобы уменьшить пространство для хранения.

команда **Hive CREATE TABLE:**

```
Create table textfile_table  
(column_specs)  
stored as textfile;
```

# HiveQL

## Hive Sequence File Format

**Файлы последовательности** — это плоские файлы **Hadoop**, в которых значения хранятся в виде двоичных пар «ключ значение». Файлы последовательности имеют двоичный формат, и эти файлы можно разбивать.

Основным преимуществом использования файла последовательности является **объединение двух или более файлов в один файл**.

команда **Hive CREATE TABLE:**

```
Create table sequencefile_table  
(column_specs)  
stored as sequencefile;
```

# HiveQL

## Hive RC File Format

**RCFile** — это формат файла со столбцами строк. Это еще одна форма формата файла **Hive**, которая обеспечивает высокую степень сжатия на уровне строк. Если вам нужно выполнять несколько строк одновременно, вы можете использовать формат **RCFile**.

**RCFile** очень похож на формат файла последовательности. Этот формат файла также хранит данные в виде пар ключ-значение.

команда **Hive CREATE TABLE:**

```
Create table RCfile_table  
(column_specs)  
stored as rcfile;
```

# HiveQL

## Hive AVRO File Format

**AVRO** — это проект с открытым исходным кодом, который предоставляет услуги сериализации данных и обмена данными для **Hadoop**. Можно обмениваться данными между экосистемой **Hadoop** и программой, написанной на любом языке программирования.

**Avro** — один из популярных форматов файлов в приложениях на основе **Big Data Hadoop**.

команда **Hive CREATE TABLE:**

```
Create table avro_table  
(column_specs)  
stored as avro;
```



# HiveQL

## Hive ORC File Format

**ORC file** — формат файла **Optimized Row Columnar**.

Формат файла **ORC** обеспечивает высокоэффективный способ хранения данных в таблице **Hive**. Эта файловая система фактически была разработана для преодоления ограничений других форматов файлов **Hive**.

Использование файлов **ORC** повышает производительность, когда **Hive** читает, записывает и обрабатывает данные из больших таблиц.

команда **Hive CREATE TABLE**:

```
Create table orc_table  
(column_specs)  
stored as orc;
```

# HiveQL

## Hive Parquet File Format

**Parquet** — это формат двоичных файлов, ориентированный на столбцы.

Паркет высокоэффективен для типов объемных запросов.

**Parquet** особенно хорош для запросов, сканирующих определенные столбцы в определенной таблице.

Таблица **Parquet** использует сжатие **Snappy, gzip**.

команда **Hive CREATE TABLE**:

```
Create table parquet_table  
(column_specs)  
stored as parquet;
```

# HDFS/Hive - Wordcount


- пример **WordCount**, полученный с использованием **Hive** и **HiveQL**:

```
CREATE TABLE faust (line STRING);

LOAD DATA INPATH '/user/hadoop/faust' OVERWRITE INTO TABLE faust;

CREATE TABLE word_counts AS
SELECT word, count(1) AS count FROM
(SELECT explode(split(line, '\\s')) AS word FROM faust) temp
GROUP BY word ORDER BY word;
```

```
SELECT * from word_counts ORDER BY count DESC LIMIT 10;
```



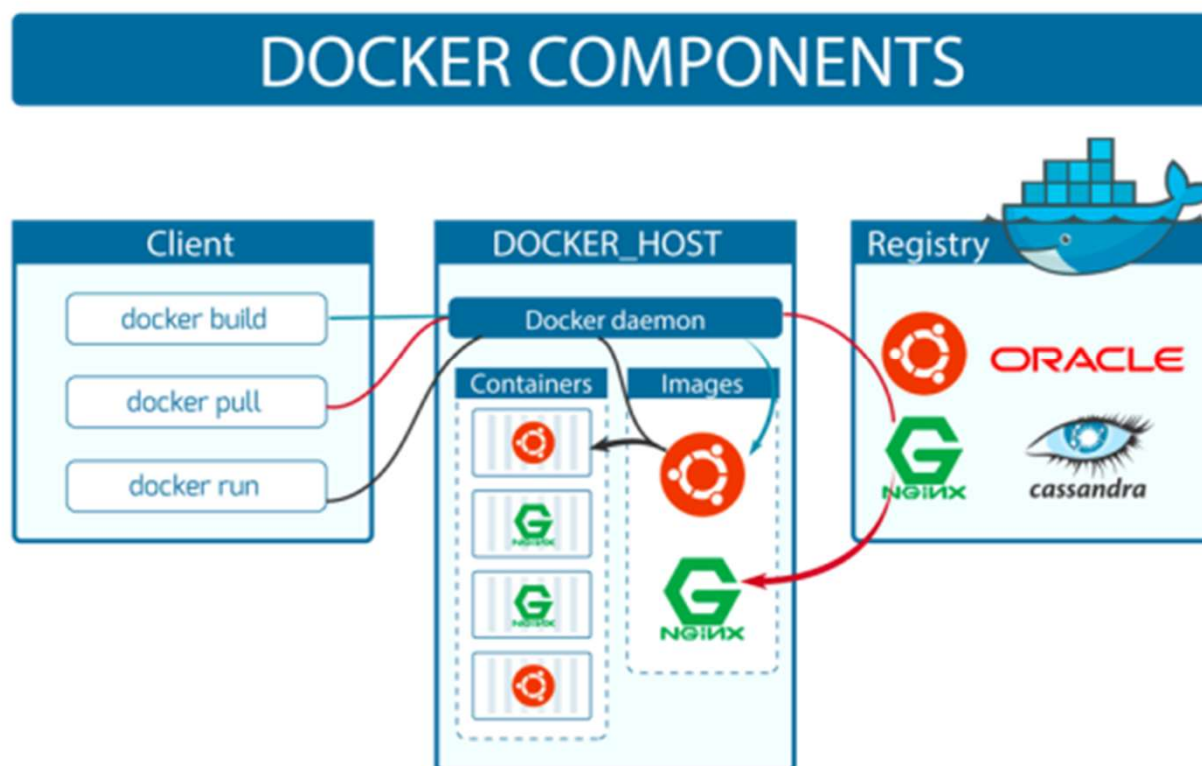
word	count
und	509
die	463
der	440
ich	435
Und	400
nicht	346
zu	319
[...]	

# Подготовка к упражнениям

## Установка и настройка Hive

# Docker

Чтобы ускорить процесс и не тратить время на установку и настройку **Hive** и других инструментов, воспользуемся уже подготовленным **докер-контейнером**.



# Docker Images/Dockerfiles

main ▾ BigDataAnalytic\_Practice / docker22 /

Go to file Add file ▾ ⋮

BosenkoTM Add files via upload ec798fc · 19 days ago History

..		
airflow	Add files via upload	19 days ago
hadoop_base	Add files via upload	19 days ago
hive_base	Add files via upload	19 days ago
hiveserver_base	Add files via upload	19 days ago
pentaho	Add files via upload	19 days ago
spark_base	Add files via upload	19 days ago
README.md	Add files via upload	19 days ago

⋮ README.md

## Docker Images

Эта папка содержит все изображения Docker, используемые в этой лекции. Более подробную информацию, например, о том, как их запускать, останавливать и использовать, см. в файлах readme образов Docker (во вложенных папках). Вы можете создавать контейнеры самостоятельно или извлекать их из репозитория Docker:

```
https://hub.docker.com/u/marcelmittelstaedt
```

### Images:

- [Hadoop Base Image](#) Hadoop 3.1.2 Base Image (Ubuntu 18.04)
- [Hadoop and Hive Base Image](#) Hadoop 3.1.2 and Hive 3.1.2 Base Image (Ubuntu 18.04)
- [Hadoop, Hive and HiveServer2 Base Image](#) Hadoop 3.1.2, Hive 3.1.2 and HiveServer2 Base Image (Ubuntu 18.04)
- [Spark Base Image](#) Spark 2.3.4 on Hadoop 3.1.2 as well as Hive 3.1.2 and HiveServer2 Base Image (Ubuntu 18.04)
- [Airflow Base Image](#) Airflow 1.10.5 with PostgreSQL 10.10 as Metadata Store Base Image (Ubuntu 18.04)
- [Pentaho Data Integration Base Image](#) Pentaho Data Integration 8.0 Base Image (Ubuntu 18.04)

dockerhub

Explore Repositories Organizations Get Help ▾ marcelmittelstaedt ▾

Repositories Using 1 of 1 private repositories. [Get more](#)

marcelmittelsta... Filter by repository name...

Create Repository +

REPOSITORY	DESCRIPTION	LAST MODIFIED
marcelmittelstaedt / airflow	Airflow 10.1.5 Image using PostgreSQL 10.10 for Metadata (Ubuntu...	2 days ago
marcelmittelstaedt / spark_base	Hadoop 3.1.2 and Spark 2.3.4 Base Image (Ubuntu 18.04)	5 days ago
marcelmittelstaedt / hiveserver_base	Hive 3.1.2, Hadoop 3.1.2 and HiveServer2 Base Image (Ubuntu 18...	5 days ago
marcelmittelstaedt / hive_base	Hive 3.1.2 and Hadoop 3.1.2 Base Image (Ubuntu 18.04)	6 days ago
marcelmittelstaedt / hadoop_base	Hadoop 3.1.2 Base Image (Ubuntu 18.04)	6 days ago
marcelmittelstaedt / ubuntu_18_04_base	Ubuntu 18.04 Base Image created from scratch using debootstrap.	8 days ago

<https://hub.docker.com/u/marcelmittelstaedt>

[https://github.com/BosenkoTM/BigDataAnalytic\\_Practice/tree/main/docker22](https://github.com/BosenkoTM/BigDataAnalytic_Practice/tree/main/docker22)

# Setup Docker Container

3. Install and setup docker **На кластере МГПУ** шаг выполнен.

```
sudo apt-get update
sudo apt-get install docker.io
sudo usermod -aG docker $USER
# exit and login again
```

4. Pull Hadoop with Hive Image

```
sudo docker pull marcelmittelstaedt/hive_base:latest
```

5. Start Container from pulled image:

```
sudo docker run -dit --name hive_base_container -p 8088:8088 -p 9870:9870 -p 9864 marcelmittelstaedt/hive_base:latest
```

# Setup Docker Container

## 6. Show Running Container:

```
sudo docker ps -a
```

```
lenipuz@161:~$ sudo docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
7b5ba57bd5ae   marcelmittelstaedt/hive_base:latest "/startup.sh"           About a minute Up About a minute 0.0.0.0:8088->8088/tcp, :::8088->8088/tcp, 0.0.0.0:9864->9864/tcp, :::9864->9864/tcp, 0.0.0.0:9870->9870/tcp, :::9870->9870/tcp
hive_base_container
```

## 7. Show Logs of container (wait till finished):

```
sudo docker logs hive_base_container
```

```
[...]
Stopping namenodes on [localhost]
Stopping datanodes
Stopping secondary namenodes [c821a0e1bdcf]
Stopping nodemanagers
Stopping resourcemanager
Container Startup finished.
```



# Setup Docker Container

## 8. Get a shell inside the container:

```
lemp@u20-16:~$ sudo docker exec -it hive_base_container bash  
root@7b5ba57bd5ae:/#
```

## 9. Switch to hadoop user:

```
root@7b5ba57bd5ae:/# sudo su hadoop  
hadoop@7b5ba57bd5ae:/$ cd  
hadoop@7b5ba57bd5ae:~$
```

## 10. Start DFS and YARN:

```
start-all.sh
```

# Test Hive

## 11. Test if Hive Installation and Configuration is successful. Start Hive:

```
hive
```



```
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hadoop/hive/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Hive Session ID = c120d0b1-9025-43db-96e4-48ccfb875f1a

Logging initialized using configuration in jar:file:/home/hadoop/hive/lib/hive-common-3.1.0.jar!/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Hive Session ID = fdd6f06a-d4e4-48e6-8971-997e8a0a8e2c
hive>
```

# Install and Setup Hive

## 12. Execute First SQL Query:

```
hive> show databases;  
OK  
default  
Time taken: 0.083 seconds, Fetched: 1 row(s)  
hive>
```

# Hive: Создание и работа с внешними таблицами

Использование общедоступного набора данных [IMDb.com](https://www.imdb.com)

# Получение данных IMDb и перемещение их в HDFS

## 1. Get IMDb Data (<https://www.imdb.com/interfaces/>):

```
wget https://datasets.imdbws.com/title.basics.tsv.gz  
wget https://datasets.imdbws.com/title.ratings.tsv.gz
```

## 2. Uncompress IMDb Data:

```
gunzip title.basics.tsv.gz  
gunzip title.ratings.tsv.gz
```

## 3. Create HDFS Directories for IMDb Data:

```
hadoop fs -mkdir /user/hadoop/imdb  
hadoop fs -mkdir /user/hadoop/imdb/title_basics  
hadoop fs -mkdir /user/hadoop/imdb/title_ratings
```

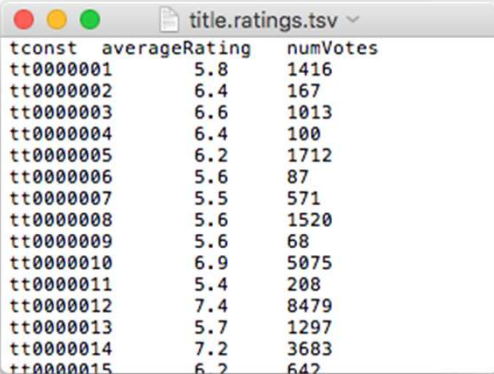
# Создание Внешних Таблиц В Hive

## 4. Transfer IMDb data files to HDFS:

```
hadoop fs -put title.basics.tsv /user/hadoop/imdb/title_basics/title.basics.tsv
hadoop fs -put title.ratings.tsv /user/hadoop/imdb/title_ratings/title.ratings.tsv
```

## 5. Create External Table `title_ratings` (file `title.ratings.tsv`) in Hive:

```
hive > CREATE EXTERNAL TABLE IF NOT EXISTS title_ratings(
    tconst STRING,
    average_rating DECIMAL(2,1),
    num_votes BIGINT
) COMMENT 'IMDb Ratings'
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' STORED AS
TEXTFILE LOCATION '/user/hadoop/imdb/title_ratings'
TBLPROPERTIES ('skip.header.line.count'='1');
```



tconst	averageRating	numVotes
tt0000001	5.8	1416
tt0000002	6.4	167
tt0000003	6.6	1013
tt0000004	6.4	100
tt0000005	6.2	1712
tt0000006	5.6	87
tt0000007	5.5	571
tt0000008	5.6	1520
tt0000009	5.6	68
tt0000010	6.9	5075
tt0000011	5.4	208
tt0000012	7.4	8479
tt0000013	5.7	1297
tt0000014	7.2	3683
tt0000015	6.2	642

# Создание Внешних Таблиц В Hive

6. Create External Table `title_basics` for file `title.basics.tsv` in Hive:

tconst	titleType	primaryTitle	originalTitle	isAdult	startYear	endYear	runtimeMinutes	genres
tt0000001	short	Carmencita	Carmencita	0	1894	\N	1	Documentary,Short
tt0000002	short	Le clown et ses chiens	Le clown et ses chiens	0	1892	\N	5	Animation,Short
tt0000003	short	Pauvre Pierrot	Pauvre Pierrot	0	1892	\N	4	Animation,Comedy,Romance
tt0000004	short	Un bon bock	Un bon bock	0	1892	\N	\N	Animation,Short
tt0000005	short	Blacksmith Scene	Blacksmith Scene	0	1893	\N	1	Comedy,Short
tt0000006	short	Chinese Opium Den	Chinese Opium Den	0	1894	\N	1	Short
tt0000007	short	Corbett and Courtney Before the Kinetograph	Corbett and Courtney Before the Kinetograph	0	1894	\N	1	Short,Sport
tt0000008	short	Edison Kinetoscopic Record of a Sneeze	Edison Kinetoscopic Record of a Sneeze	0	1894	\N	1	Documentary,Short
tt0000009	movie	Miss Jerry	Miss Jerry	0	1894	\N	45	Romance
tt0000010	short	Employees Leaving the Lumière Factory	La sortie de l'usine Lumière à Lyon	0	1895	\N	1	Documentary,Short
tt0000011	short	Akrobatisches Potpourri	Akrobatisches Potpourri	0	1895	\N	1	Documentary,Short
tt0000012	short	The Arrival of a Train	L'arrivée d'un train à La Ciotat	0	1896	\N	1	Documentary,Short
tt0000013	short	The Photographical Congress Arrives in Lyon	Neuville-sur-Saône: Débarquement du congrès des photographes à Lyon	0	1895	\N	1	Documentary,Short
tt0000014	short	Tables Turned on the Gardener	L'arroseur arrosé	0	1895	\N	1	Comedy,Short
tt0000015	short	Autour d'une cabine	Autour d'une cabine	0	1894	\N	2	Animation,Short
tt0000016	short	Barque sortant du port	Barque sortant du port	0	1895	\N	1	Documentary,Short
tt0000017	short	Italienischer Bauerntanz	Italienischer Bauerntanz	0	1895	\N	1	Documentary,Short
tt0000018	short	Das boxende Känguruh	Das boxende Känguruh	0	1895	\N	1	Short
tt0000019	short	The Clown Barber	The Clown Barber	0	1898	\N	\N	Comedy,Short
tt0000020	short	The Derby 1895	The Derby 1895	0	1895	\N	1	Documentary,Short,Sport
tt0000022	short	Blacksmith Scene	Les forgerons	0	1895	\N	1	Documentary,Short
tt0000023	short	The Sea Baignade en mer		0	1895	\N	1	Documentary,Short
tt0000024	short	Opening of the Kiel Canal	Opening of the Kiel Canal	0	1895	\N	\N	News,Short
tt0000025	short	The Oxford and Cambridge University Boat Race	The Oxford and Cambridge University Boat Race	0	1895	\N	\N	News,Short,Sport
tt0000026	short	The Messers. Lumière at Cards	Partie d'écarté	0	1896	\N	1	Documentary,Short
tt0000027	short	Cordeliers' Square in Lyon	Place des Cordeliers à Lyon	0	1895	\N	1	Documentary,Short
tt0000028	short	Fishing for Goldfish	La pêche aux poissons rouges	0	1895	\N	1	Documentary,Short
tt0000029	short	Baby's Dinner	Repas de bébé	0	1895	\N	1	Documentary,Short
tt0000030	short	Rough Sea at Dover	Rough Sea at Dover	0	1895	\N	1	Documentary,Short

# Создание Внешних Таблиц В Hive

6. Create External Table `title_basics` for file `title.basics.tsv` in Hive:

```
hive > CREATE EXTERNAL TABLE IF NOT EXISTS title_basics (  
    tconst STRING,  
    title_type STRING,  
    primary_title STRING,  
    original_title STRING,  
    is_adult DECIMAL(1,0),  
    start_year DECIMAL(4,0),  
    end_year STRING,  
    runtime_minutes INT,  
    genres STRING  
) COMMENT 'IMDb Movies' ROW FORMAT DELIMITED FIELDS TERMINATED BY  
'\t' STORED AS TEXTFILE LOCATION '/user/hadoop/imdb/title_basics'  
TBLPROPERTIES ('skip.header.line.count'='1');
```



# Создание Внешних Таблиц В Hive

## 7. Query Table `title_basics` in Hive using SQL (HiveQL):

```
hive> select * from title_basics limit 3;
OK
tt0000001 short Carmencita Carmencita 0 1894 NULL 1 Documentary,Short
tt0000002 short Le clown et ses chiens Le clown et ses chiens 0 1892 NULL 5 Animation,Short
tt0000003 short Pauvre Pierrot Pauvre Pierrot 0 1892 NULL 4 Animation,Comedy,Romance
Time taken: 0.139 seconds, Fetched: 5 row(s)
hive>
```

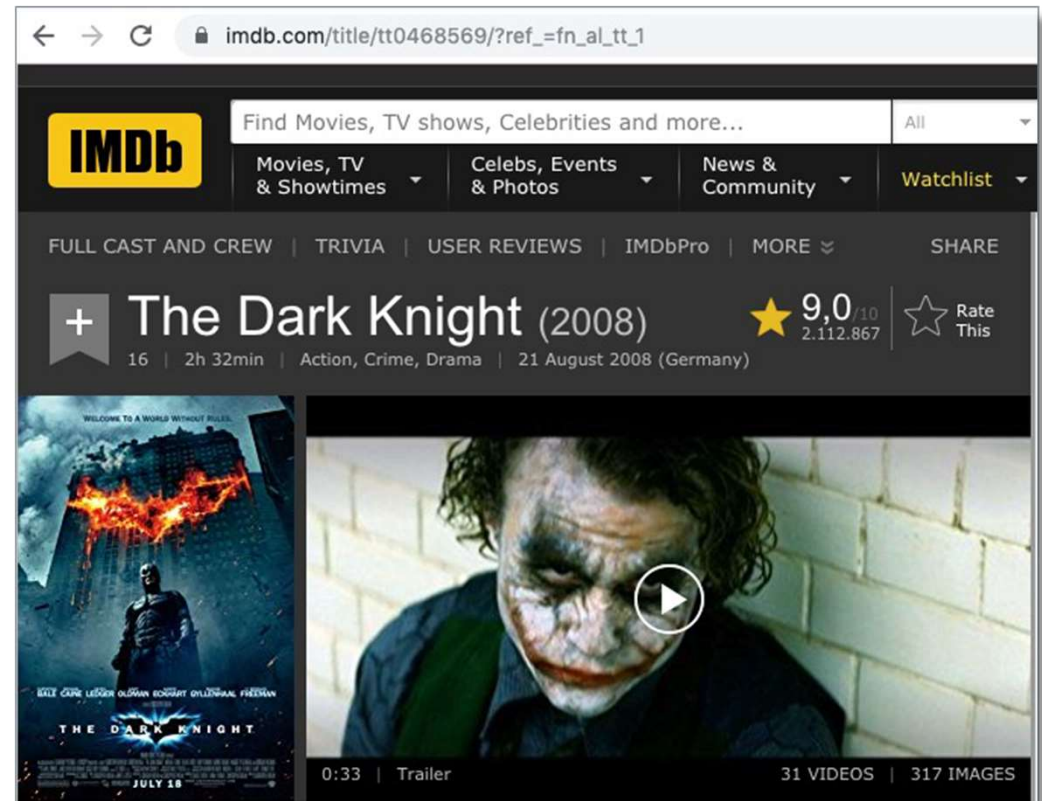
## 8. Query Table `title_ratings` in Hive using SQL (HiveQL):

```
hive> select * from title_ratings limit 3;
OK
tt0000001 5.6 1540
tt0000002 6.1 186
tt0000003 6.5 1199
Time taken: 0.119 seconds, Fetched: 3 row(s)
hive>
```

# Создание Внешних Таблиц В Hive

9. Run a complex query which starts a MapReduce Job on Yarn, e.g. get Rating of movie „*The Dark Knight*“:

```
SELECT
  *
FROM
  title_basics b
  JOIN title_ratings r ON (b.tconst=r.tconst)
WHERE
  original_title = 'The Dark Knight'
  AND title_type='movie';
```




# Создание Внешних Таблиц В Hive

## 9. Execute Query

```
hive> SELECT * FROM title_basics b JOIN title_ratings r ON (b.tconst=r.tconst) WHERE original_title =  
'The Dark Knight' and title_type='movie';  
[...]  
Starting Job = job_1613323804972_0003, Tracking URL = http://154172c92bc7:8088/proxy/application_1613323804972_0003/  
Kill Command = /home/hadoop/hadoop/bin/mapred job -kill job_1613323804972_0003  
Hadoop job information for Stage-3: number of mappers: 3; number of reducers: 0  
2021-02-14 17:37:59,326 Stage-3 map = 0%, reduce = 0%  
2021-02-14 17:38:20,617 Stage-3 map = 50%, reduce = 0%, Cumulative CPU 35.59 sec  
2021-02-14 17:38:21,656 Stage-3 map = 67%, reduce = 0%, Cumulative CPU 52.65 sec  
2021-02-14 17:38:22,718 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 56.84 sec  
MapReduce Total cumulative CPU time: 56 seconds 840 msec  
Ended Job = job_1613323804972_0003  
MapReduce Jobs Launched:  
Stage-Stage-3: Map: 3 Cumulative CPU: 56.84 sec HDFS Read: 650217476 HDFS Write: 376 SUCCESS  
Total MapReduce CPU Time Spent: 56 seconds 840 msec  
OK  
tt0468569 movie The Dark Knight The Dark Knight 0 2008 NULL 152 Action,Crime,Drama tt0468569 9.0 2111245  
Time taken: 53.094 seconds, Fetched: 1 row(s)  
hive>
```

# Создание Внешних Таблиц В Hive

9. Take a look at YARN (<http://XXX.XXX.XXX.XXX:8088/cluster/>):



## RUNNING Applications

Logged in as: dr.who

Cluster

[About](#)  
[Nodes](#)  
[Node Labels](#)  
[Applications](#)  
[NEW](#)  
[NEW SAVING](#)  
[SUBMITTED](#)  
[ACCEPTED](#)  
[RUNNING](#)  
[FINISHED](#)  
[FAILED](#)  
[KILLED](#)  
[Scheduler](#)

Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved
5	0	1	4	4	11 GB	16 GB	0 B	4	8	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes
1	0	0	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0

Show 20 entries

Search:

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU Vcores	Allocated Memory MB	Reserved CPU Vcores	Reserved Memory MB	% of Queue	% of Cluster	Progress	Tracking UI	Blacklisted Nodes
<a href="#">application_1613323804972_0005</a>	hadoop	SELECT * FROM title_bas...title_type='movie' (Stage-3)	MAPREDUCE	default	0	Sun Feb 14 18:41:12 +0100 2021	N/A	RUNNING	UNDEFINED	4	4	11264	0	0	68.8	68.8	<div></div>	<a href="#">ApplicationMaster</a>	0

Showing 1 to 1 of 1 entries

First Previous 1 Next Last

# Задание

**Hive:** Создание и работа с внешними таблицами

Использование общедоступного набора данных [IMDb.com](https://www.imdb.com)

# HDFS и Hive QL упражнения - IMDB

1. Выполните все действия на слайдах выше
2. Скачать <https://datasets.imdbws.com/name.basics.tsv.gz>
3. Создать каталог HDFS `/user/hadoop/imdb/name_basics/` для файла **name.basics.tsv**
4. Создайте внешнюю Hive таблицу `name_basics` для **name.basics.tsv**
5. Используйте HiveQL, чтобы ответить на следующие вопросы::
  - a) Сколько фильмов и сериалов находится в наборе данных IMDB?
  - b) Кто самый молодой актер/сценарист/... в наборе данных?
  - c) Создайте список (`tconst`, `original_title`, `start_year`, `average_rating`, `num_votes`), который состоит из:
    - фильм вышел в 2010 году или позднее;
    - фильм имеет средний рейтинг, равный или превышающий 8,1
    - проголосовали более 100 000 раз
  - d) Сколько фильмов находится в списке c)?

# HDFS и Hive QL упражнения - IMDB

5. Используйте **HiveQL**, чтобы ответить на следующие вопросы::

е) Мы хотим знать, какие годы были великими для кинематографа.

Создайте список с одной строкой в год и соответствующим количеством фильмов, которые:

- имеют средний рейтинг выше 8;
- были проголосованы более 100 000 раз в порядке убывания количества фильмов.

СПАСИБО ЗА ВНИМАНИЕ