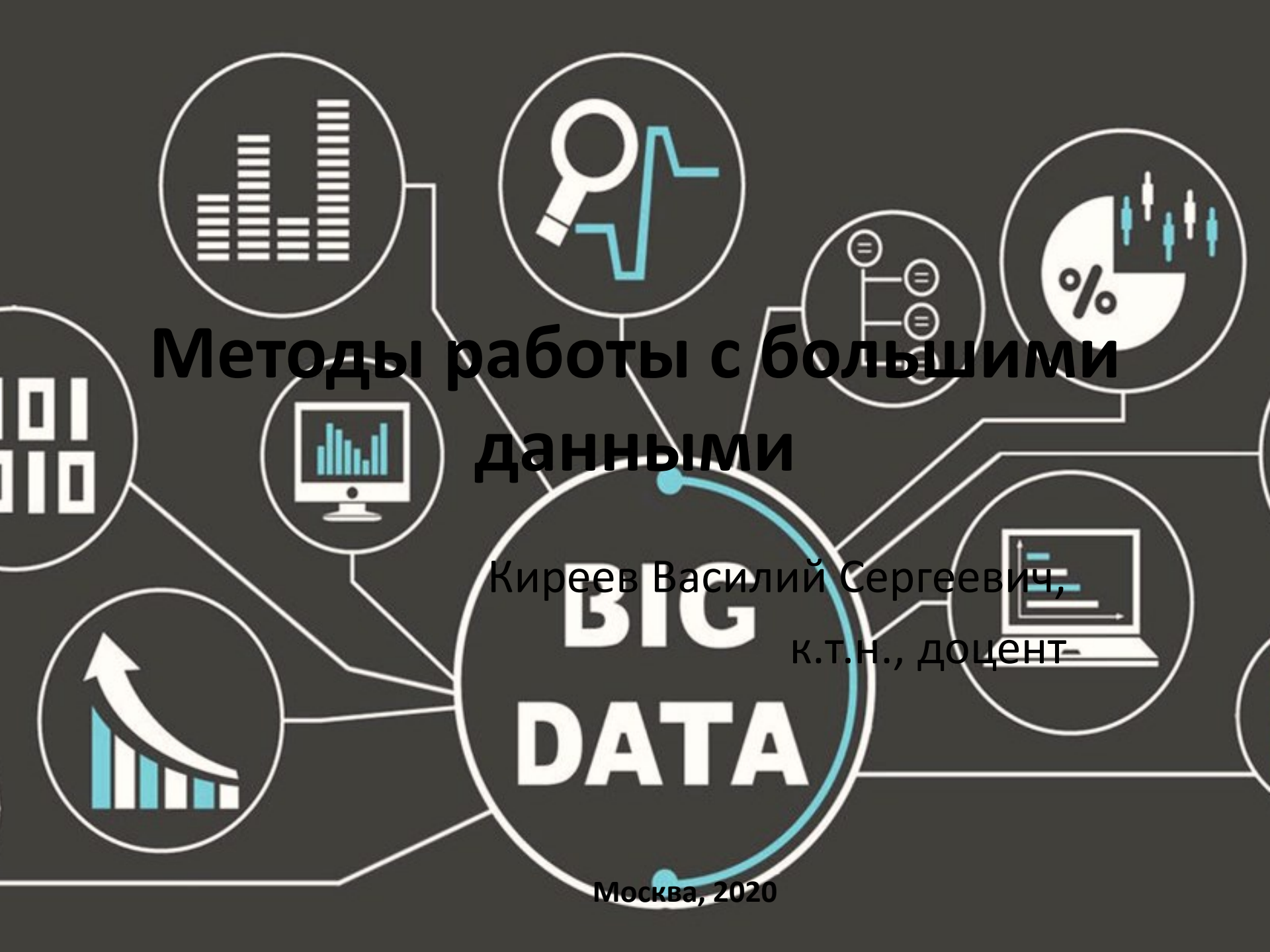


Методы работы с большими данными

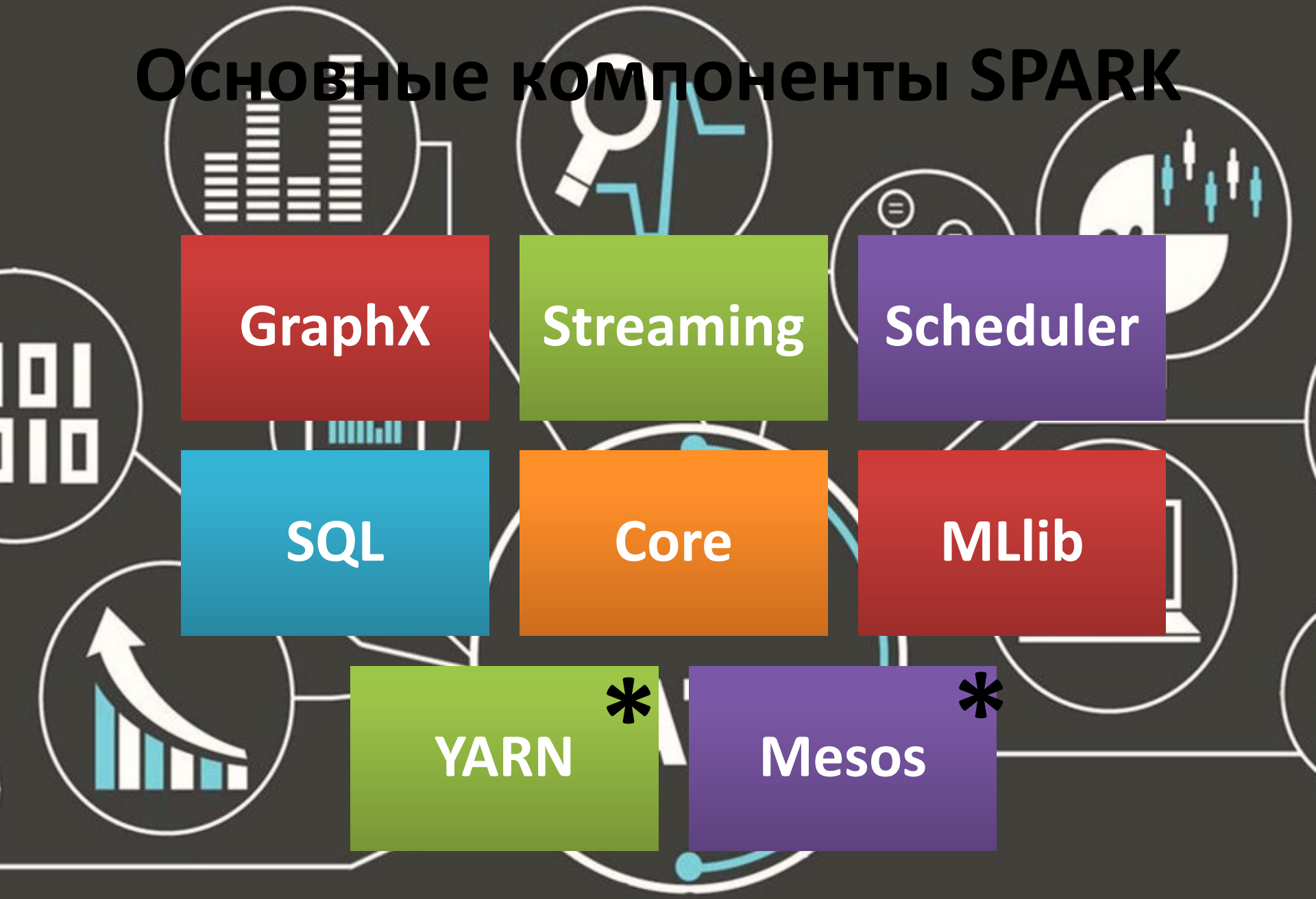
Киреев Василий Сергеевич,
к.т.н., доцент

Москва, 2020

APACHE SPARK

APACHE SPARK представляет собой кластерную вычислительную платформу с открытым исходным кодом, аналогичную Hadoop, но с некоторыми полезными особенностями, которые делают ее превосходным инструментом для решения некоторых видов задач. А именно, помимо интерактивных запросов Spark поддерживает распределенные наборы данных в оперативной памяти, оптимизируя решение итеративных задач. Сегодня Spark применяется во многих крупнейших компаниях, таких, как Amazon, eBay и Yahoo! Многие организации эксплуатируют Spark в кластерах, включающих тысячи узлов.

Основные компоненты SPARK



SPARK YARN

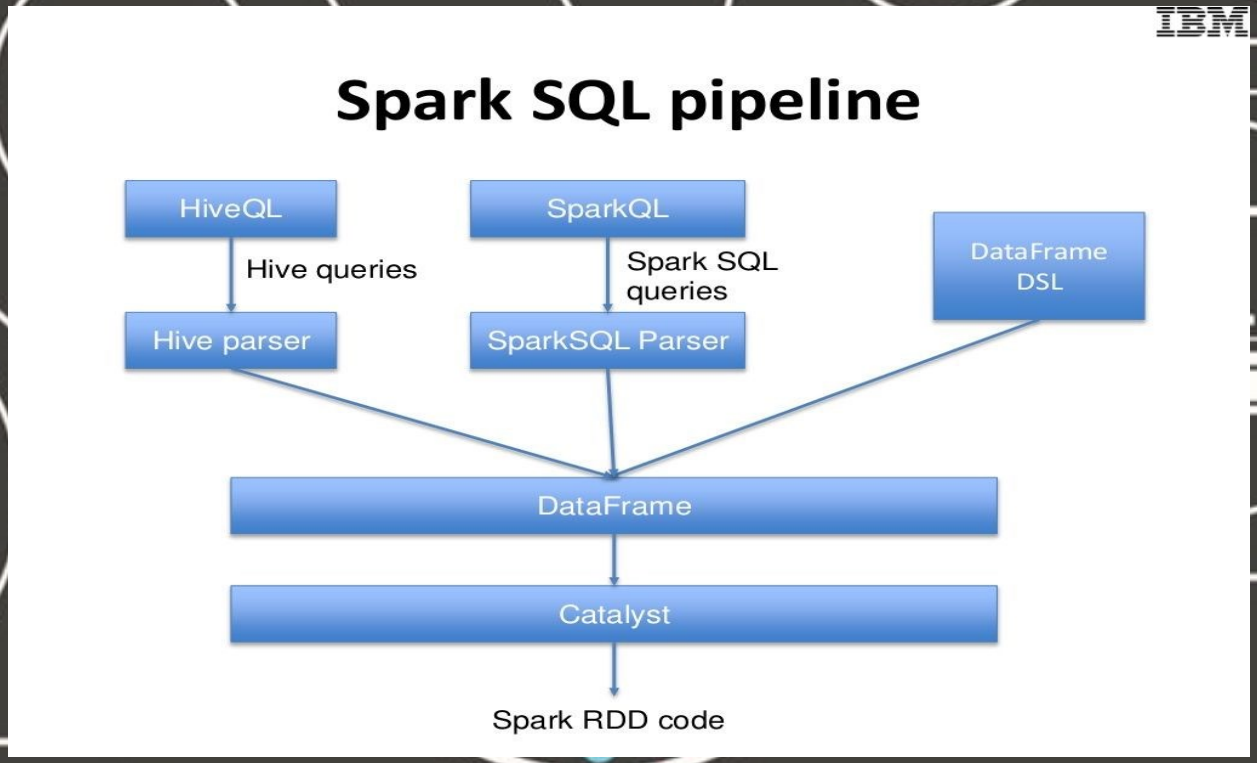
SPARK YARN - это часть системы Hadoop, являющаяся структурой общего управления ресурсами для распределенных рабочих нагрузок. YARN поддерживает множество различных вычислительных фреймворков, таких как Spark и Tez, а также функции Map-reduce.

SPARK Core

Spark Core – это базовый инструмент для крупномасштабной параллельной и распределенной обработки данных. Кроме того, есть дополнительные библиотеки, встроенные поверх ядра. Они позволяют разделить рабочие нагрузки для стриминга, SQL и машинного обучения. Отвечают за управление памятью и восстановление после ошибок, планирование, распределение и мониторинг задач в кластере, а также взаимодействие с системами хранения.

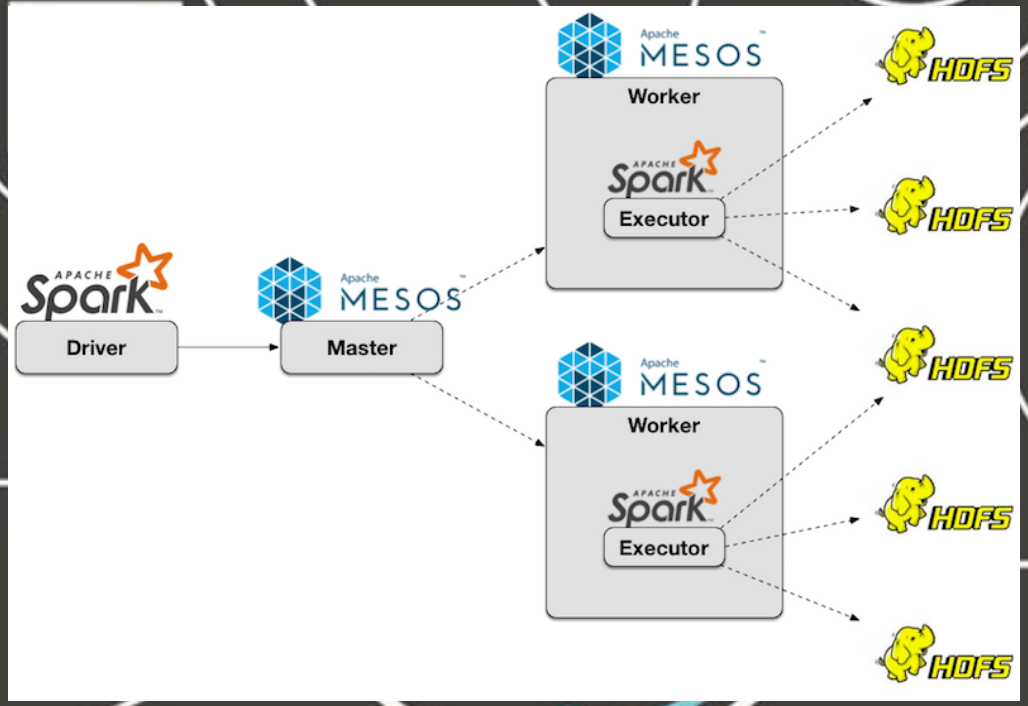
SPARK SQL

Spark SQL – интегрирует реляционную обработку с API функционального программирования в Spark. Поддерживает извлечение данных, как через SQL, так и через Hive Query Language. API DataFrame и Dataset в Spark SQL обеспечивают самый высокий уровень абстракции для структурированных данных.



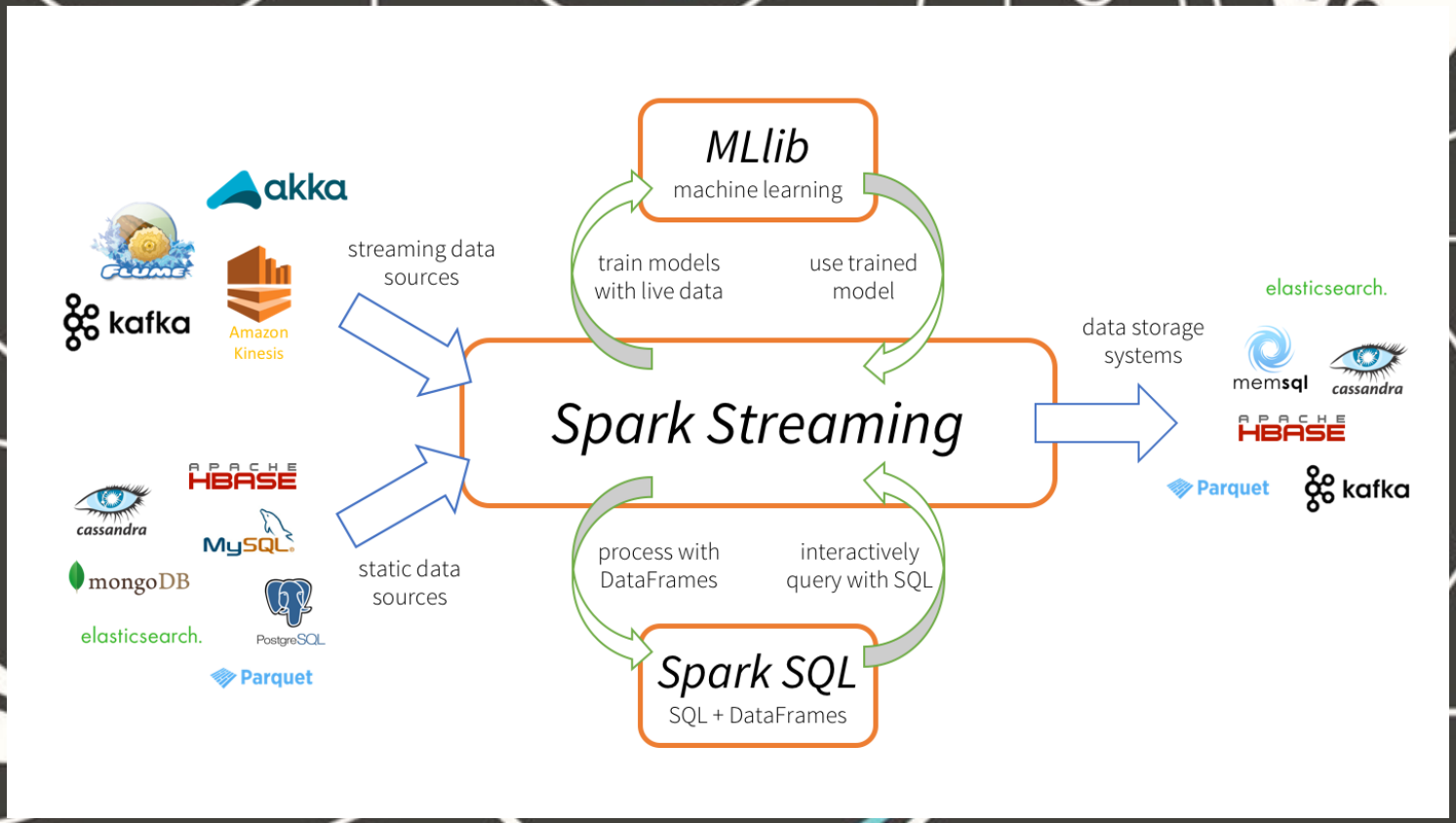
SPARK Mesos

Apache Mesos — это централизованная отказоустойчивая система управления кластером, разработанная для распределенных компьютерных сред с целью обеспечения изоляции ресурсов и удобного управления кластерами подчиненных узлов.



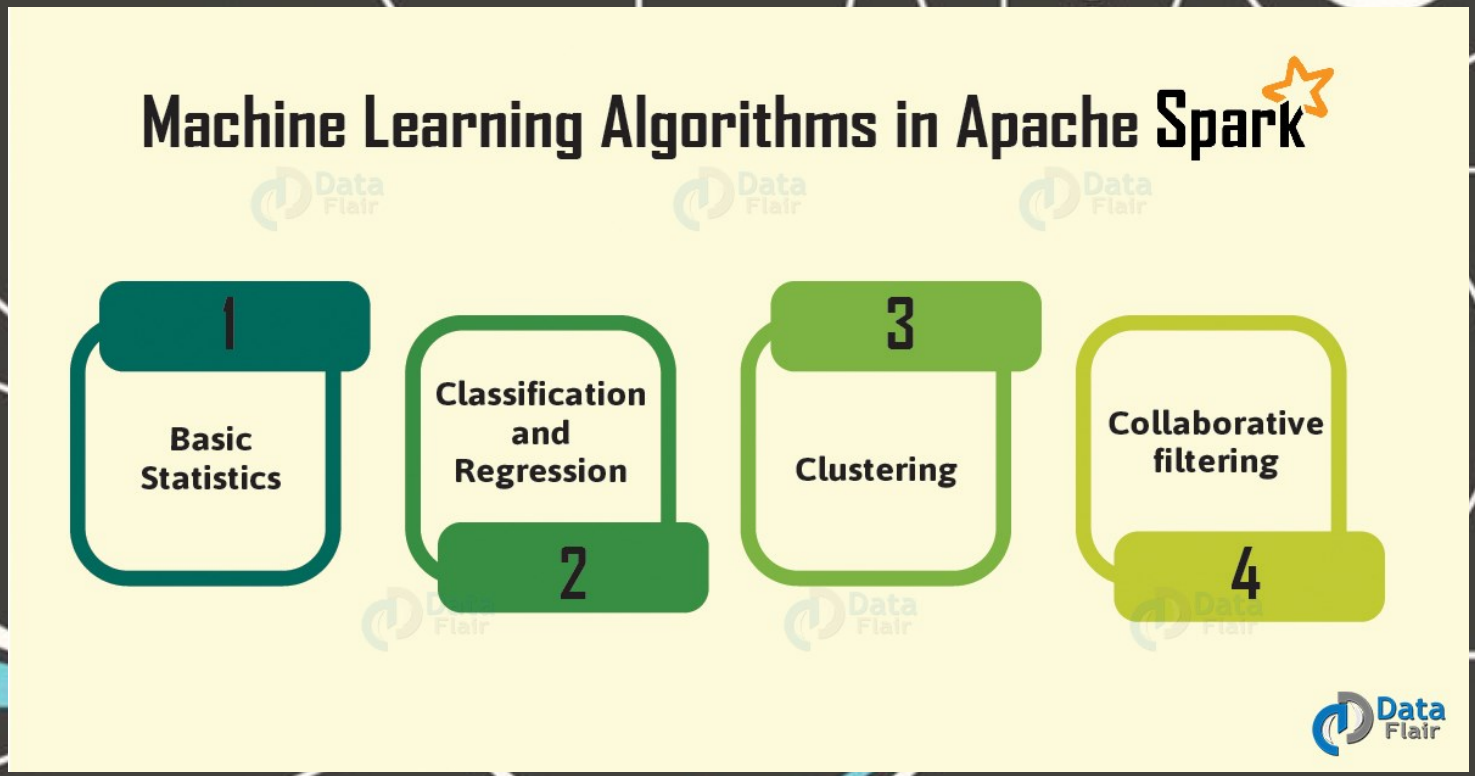
SPARK Streaming

Spark Streaming — это компонент Spark, который нужен для обработки потоковых данных в реальном времени.



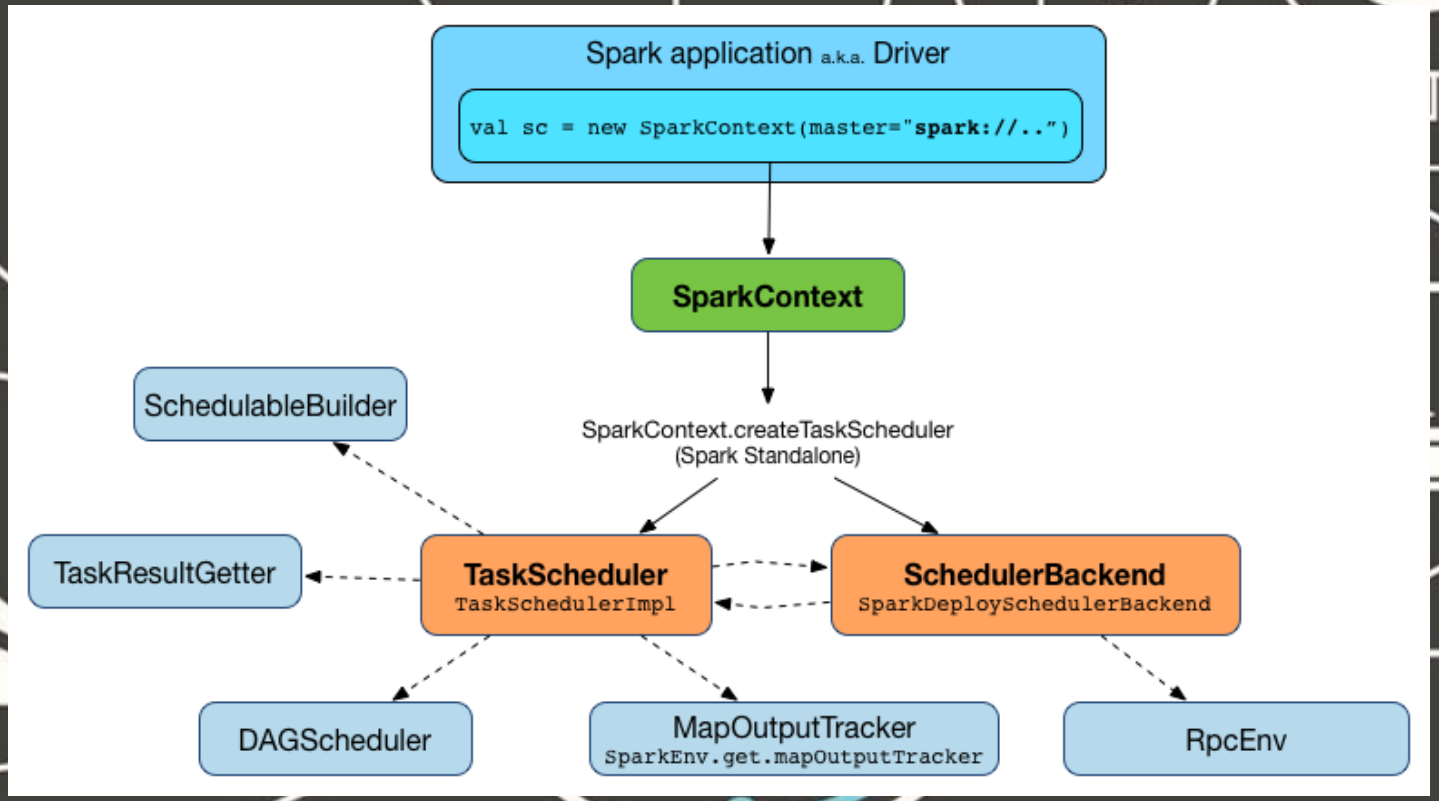
SPARK MLlib

MLlib (Машинное обучение): MLlib расшифровывается как библиотека машинного обучения. Нужна для реализации машинного обучения в Apache Spark



SPARK Scheduler

TaskScheduler — получает наборы задач (как наборы задач), представленные ему от DAGScheduler для каждого этапа, и отвечает за отправку задач в кластер, их запуск, повторную попытку, если есть сбои.

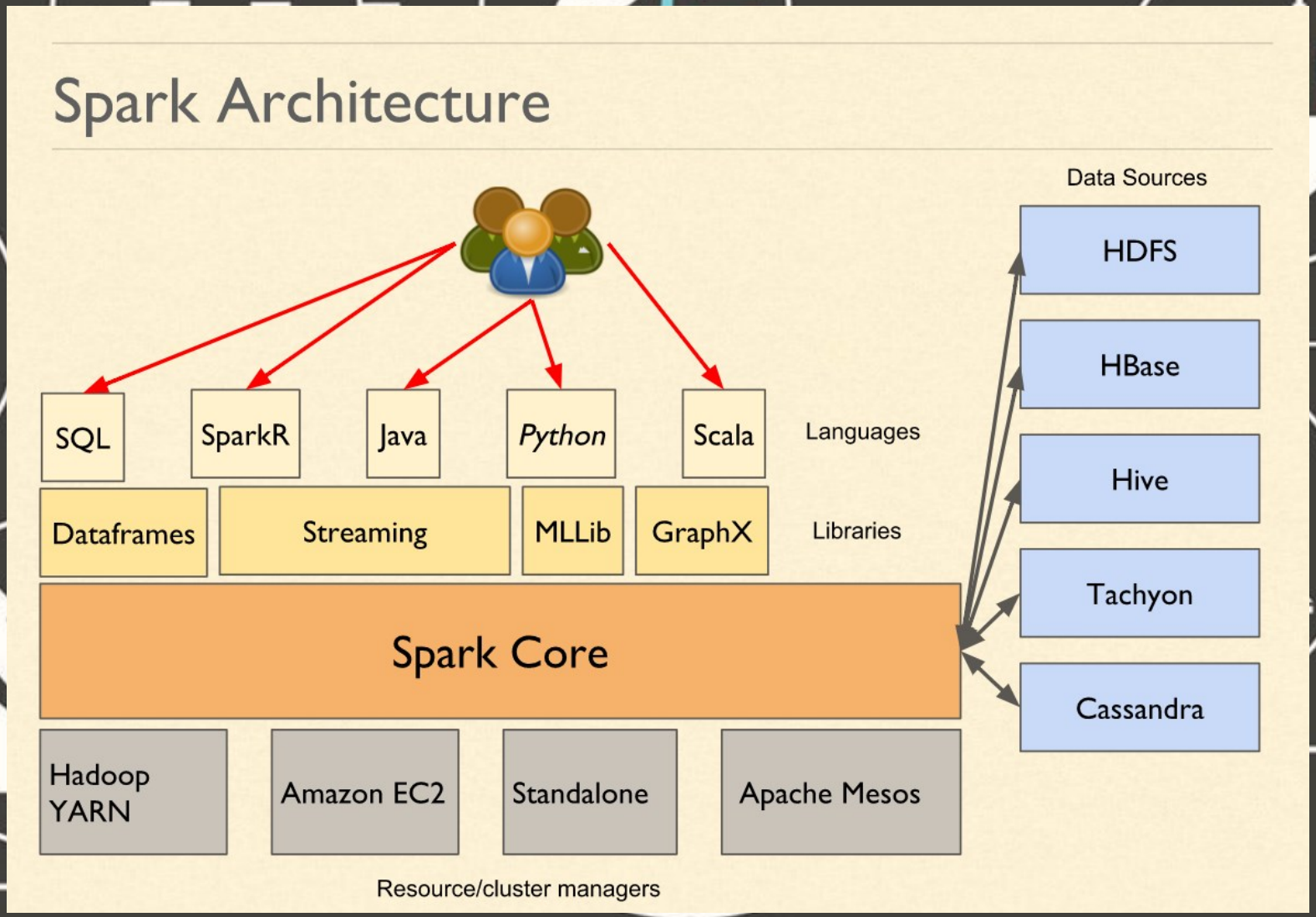


SPARK GraphX

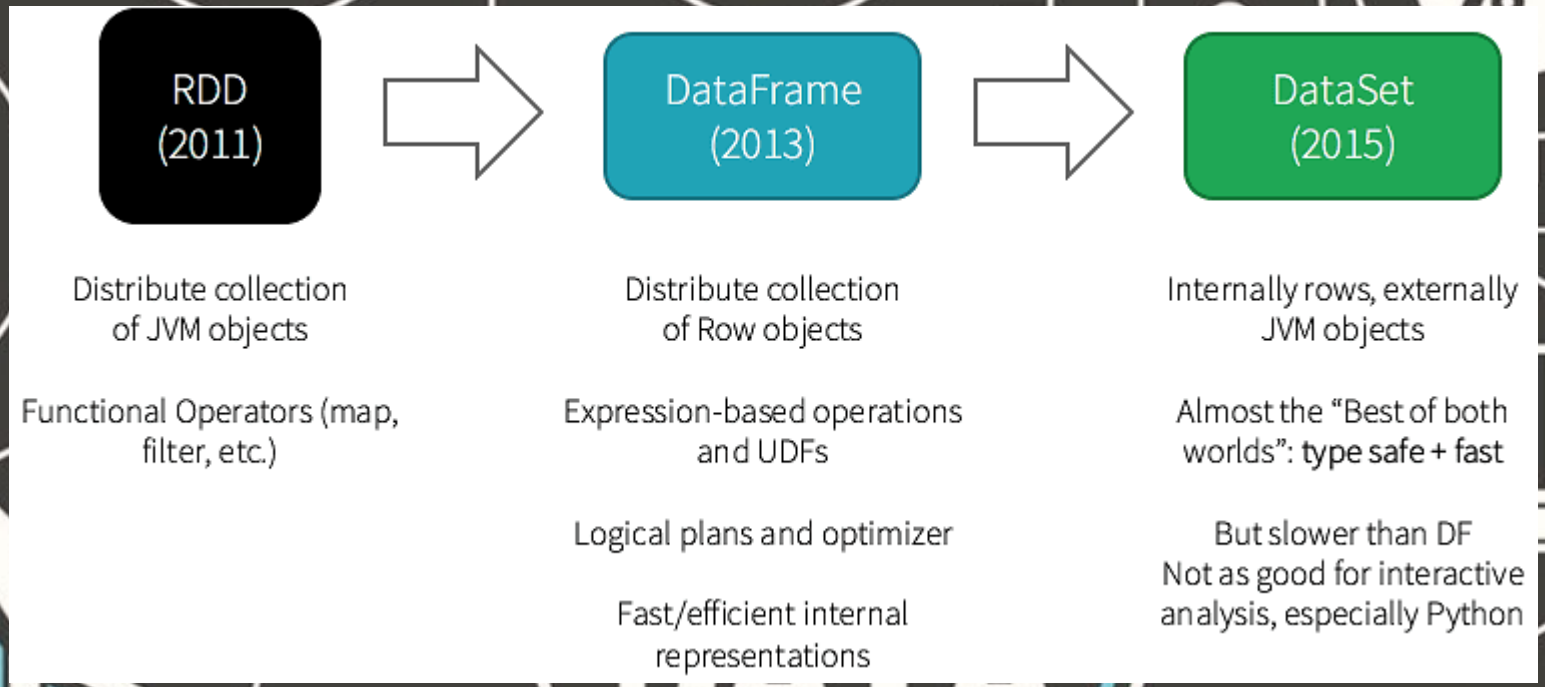
GraphX — API Spark для графов и параллельных вычислений с графами. Так что он является расширением Spark RDD с графом устойчивого распределения свойств (Resilient Distributed Property Graph).

DD (resilient distributed dataset, отказоустойчивый распределенный набор данных) – это распределенная структура данных, размещаемая в оперативной памяти. Каждый RDD представляет собой фрагмент данных, распределенных по узлам кластера. RDD являются неизменяемыми структурами, поэтому после преобразований создаются новые RDD. RDD обрабатываются параллельно с помощью таких преобразований/действий, как отображение, фильтрация и др. Эти операции выполняются одновременно во всех разделах (partition). RDD являются отказоустойчивыми: если раздел теряется в результате сбоя узла, он может быть восстановлен из исходных источников.

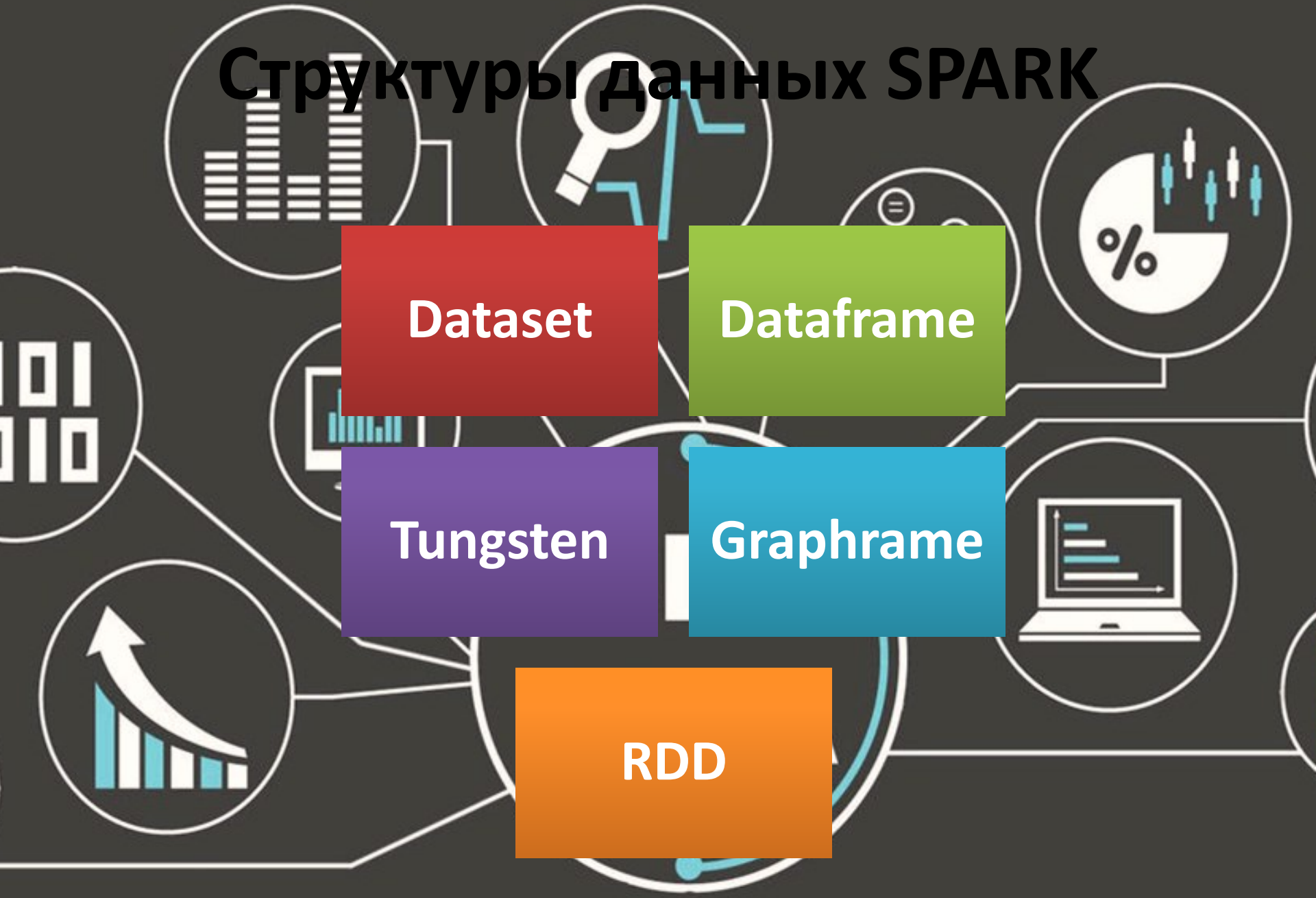
Архитектура SPARK



Эволюция структур данных SPARK



Структуры данных SPARK



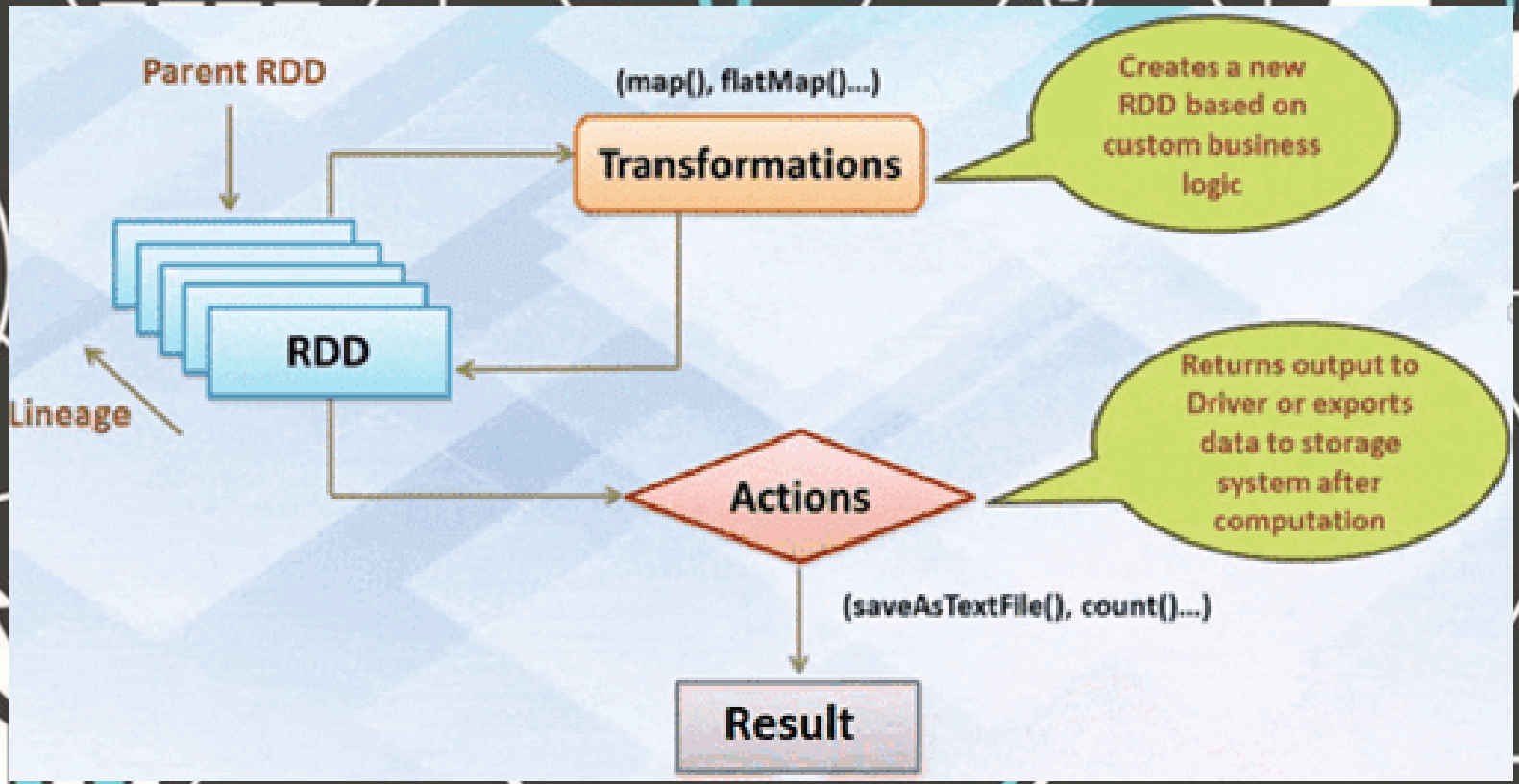
SPARK RDD

RDD (resilient distributed dataset) – это распределенная структура данных, размещаемая в оперативной памяти. Каждый RDD представляет собой фрагмент данных, распределенных по узлам кластера. RDD являются неизменяемыми структурами, поэтому после преобразований создаются новые RDD. RDD обрабатываются параллельно с помощью таких преобразований/действий, как отображение, фильтрация и др. Эти операции выполняются одновременно во всех разделах (partition). RDD являются отказоустойчивыми: если раздел теряется в результате сбоя узла, он может быть восстановлен из исходных источников.

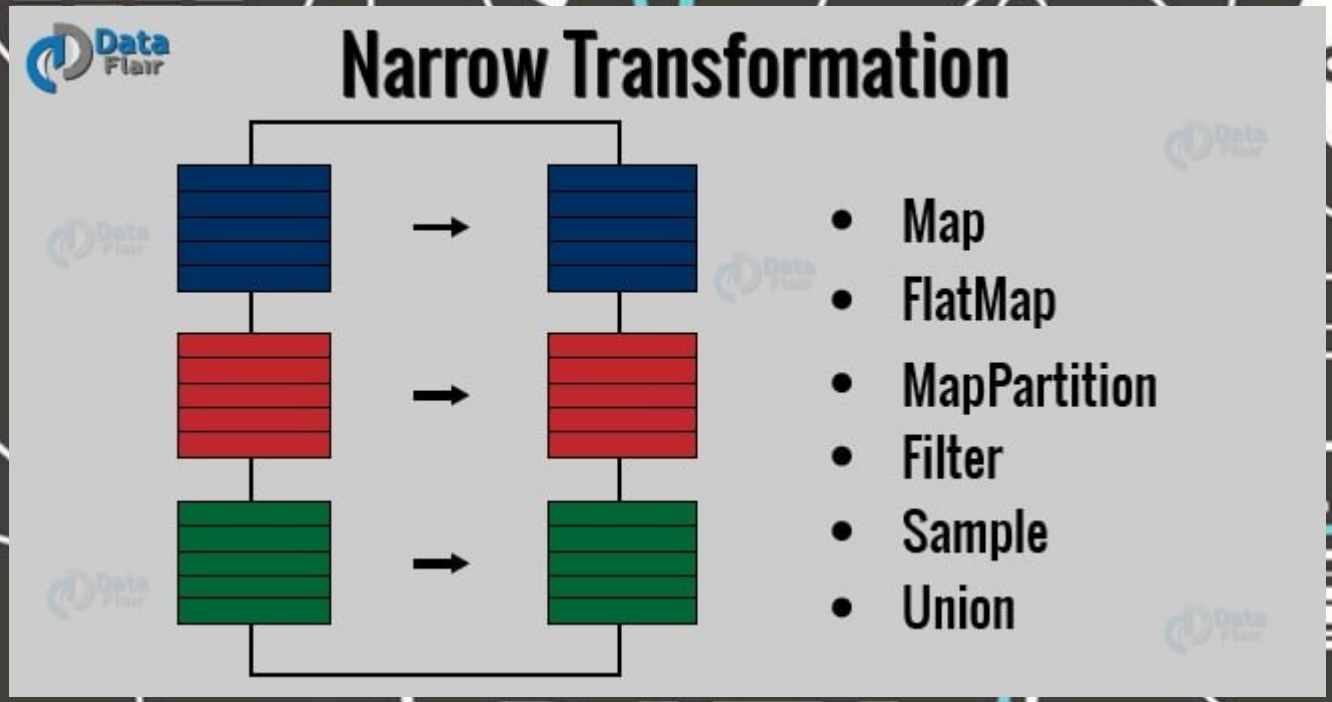
Некоторые методы работы с RDD:

.parallelize(data)	.filter(функция)	.take(n)
.textFile(путь)	.union(другой RDD)	.intersection(другой RDD)
.map(функция)	.collect()	.count()

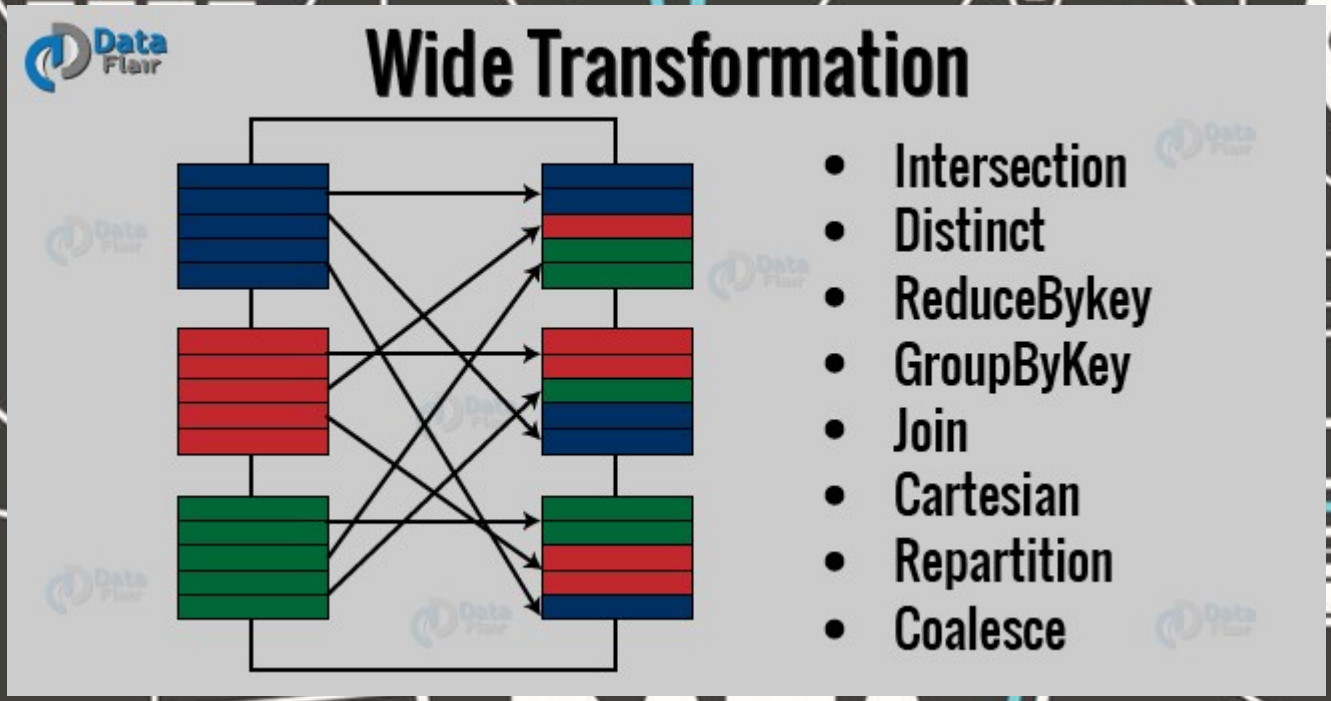
SPARK RDD


Parent RDD
(map(), flatMap()...)
Transformations
Creates a new RDD based on custom business logic
RDD
Lineage
Actions
Returns output to Driver or exports data to storage system after computation
(saveAsTextFile(), count()...)
Result

Узкие трансформации



Широкие транформации



SPARK Dataframe

DataFrame – это распределенная коллекция данных, организованных посредством именованных столбцов. Данная абстракция предназначена для выборки, фильтрации, агрегации и визуализации структурированных данных. DataFrame формирует схему посредством отражения (reflection). DataFrame является «ленивой» структурой данных, то есть содержит логический план для вычисления набора данных, при этом вычисления не выполняются до тех пор, пока пользователь не запросит специальную «операцию вывода», например, сохранение. Такой подход обеспечивает эффективную оптимизацию всех операций.

Некоторые методы работы с Dataframe

.show()	.select()	.orderBy()
.printSchema()	.count()	.filter()
.describe()	. groupBy()	

Дополнительно

Действия - это любая операция RDD, которая не выдает RDD в качестве вывода. Некоторые примеры общих действий - это подсчет данных, или нахождение максимума или минимума, или возвращение первого элемента и т. Д.

Трансформации- это ленивые операции, которые создают один или несколько новых RDD, не изменяя их

SPARK Dataset

Dataset – -это одна из базовых структур данных SparkSQL. Он помогает хранить промежуточные данные для обработки данных SPARK. Dataset SPARK с типом строки очень похож на фреймы данных, которые работают как табличная форма в устойчивом распределенном наборе данных(RDD). Dataset'ы в SPARK известны своими специфическими функциями, такими как безопасность типов, неизменяемость, схемы, оптимизация производительности, ленивая оценка, сериализация и сборка мусора. Dataset в SPARK поддерживает как безопасность во время компиляции, так и оптимизацию, что делает его предпочтительным выбором для реализации в среде SPARK Framework.

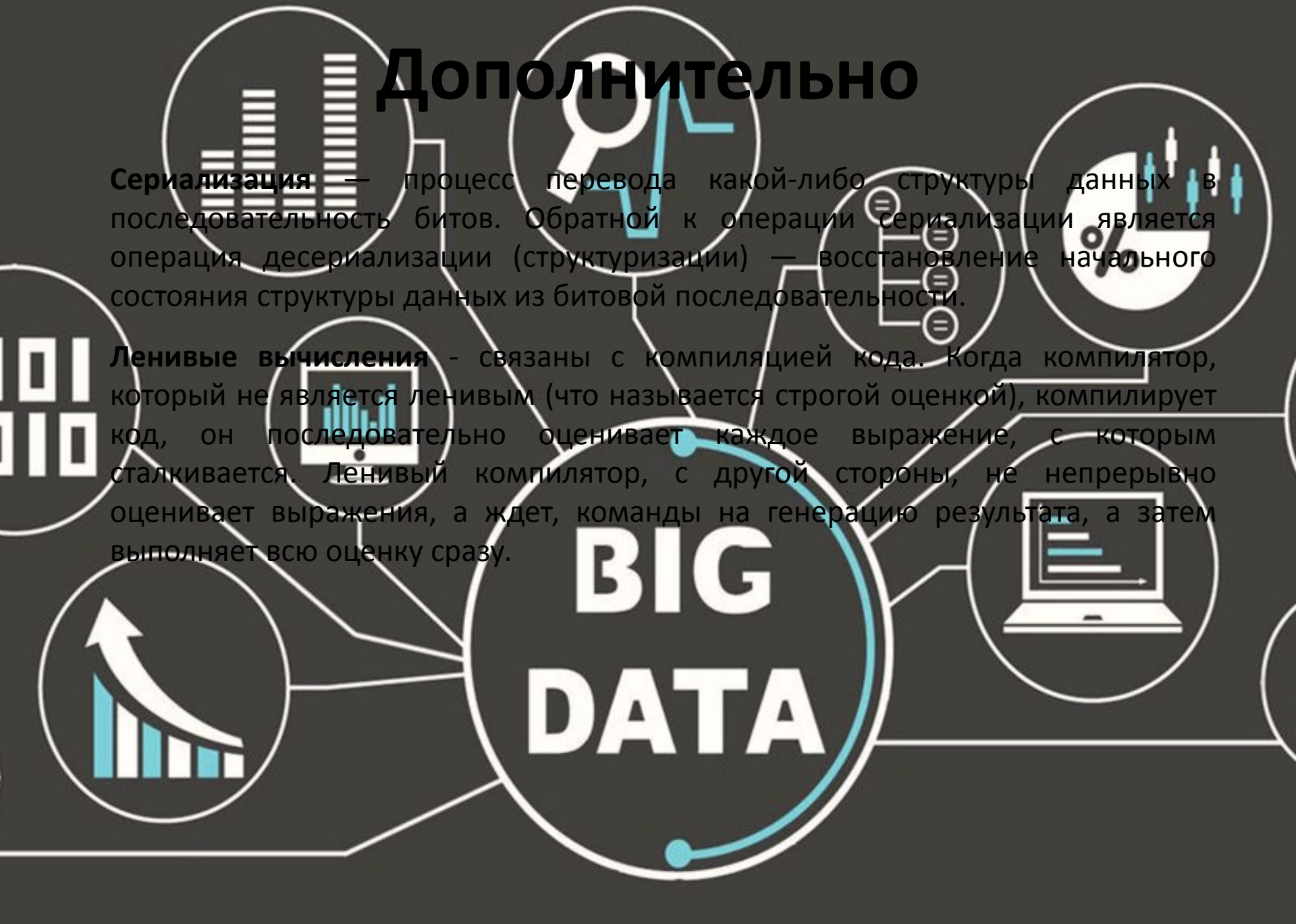
Некоторые методы работы с Dataset

.show()	.orderBy()	
.toDF()	.count()	
. groupBy()		

Дополнительно

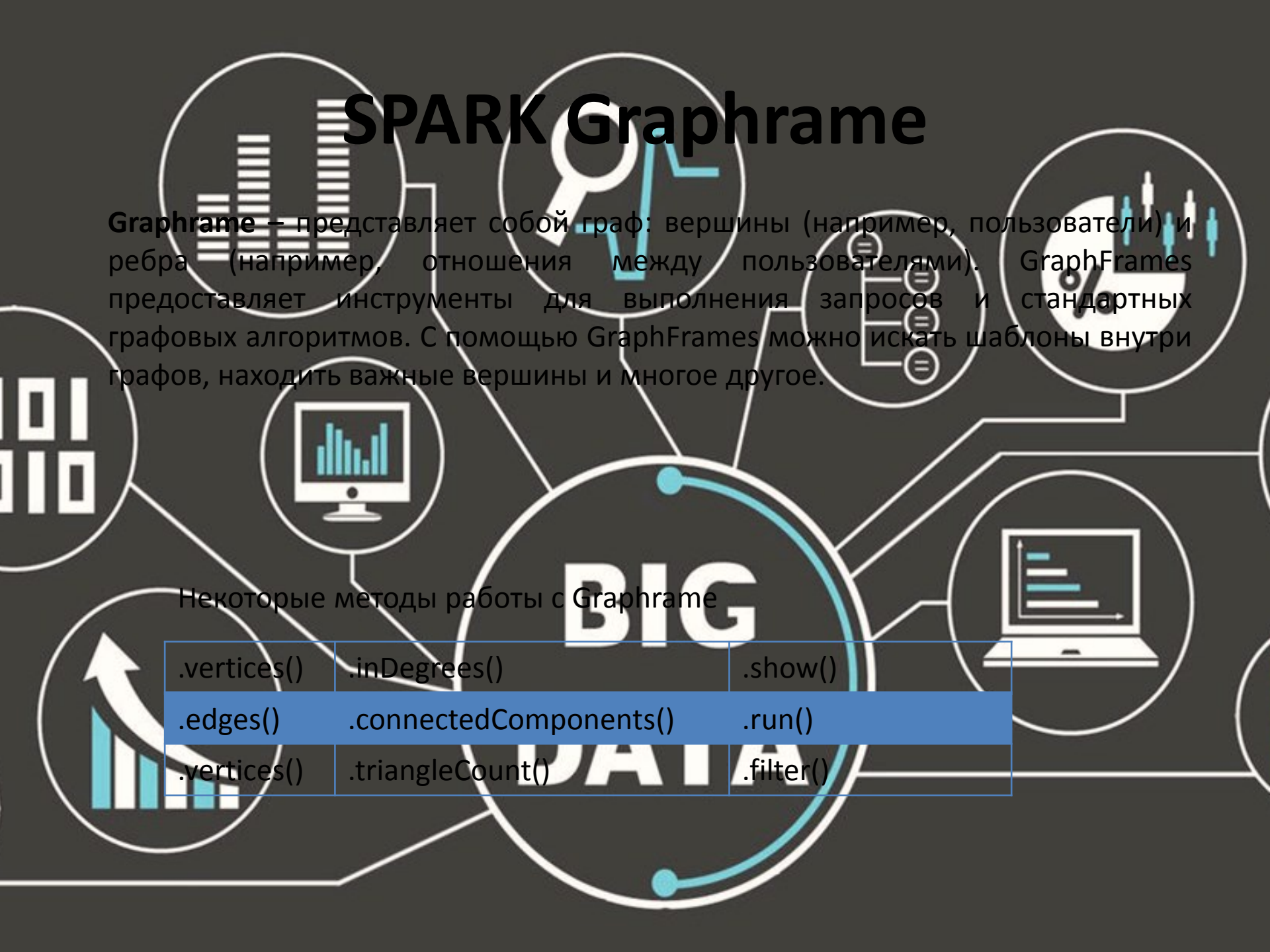
Сериализация — процесс перевода какой-либо структуры данных в последовательность битов. Обратной к операции сериализации является операция десериализации (структуризации) — восстановление начального состояния структуры данных из битовой последовательности.

Ленивые вычисления - связаны с компиляцией кода. Когда компилятор, который не является ленивым (что называется строгой оценкой), компилирует код, он последовательно оценивает каждое выражение, с которым сталкивается. Ленивый компилятор, с другой стороны, не непрерывно оценивает выражения, а ждет, команды на генерацию результата, а затем выполняет всю оценку сразу.



SPARK Graphrame

Graphrame – представляет собой граф: вершины (например, пользователи) и ребра (например, отношения между пользователями). GraphFrames предоставляет инструменты для выполнения запросов и стандартных графовых алгоритмов. С помощью GraphFrames можно искать шаблоны внутри графов, находить важные вершины и многое другое.



Некоторые методы работы с Graphrame

.vertices()	.inDegrees()	.show()
.edges()	.connectedComponents()	.run()
.vertices()	.triangleCount()	.filter()

SPARK Tungsten

Tungsten - **это** новый компонент **Spark** SQL, который обеспечивает более эффективные операции **Spark**, работая непосредственно на уровне байтов. Так как **Tungsten** больше не зависит от работы с объектами Java, мы можем использовать хранилище как в куче (в JVM), так и вне памяти. В режиме без кучи оба уменьшают накладные расходы на сборку мусора, поскольку данные не хранятся как объекты Java.

Примеры

```
from pyspark.sql import *

department1 = Row(id='123456', name='Computer Science')
Employee = Row("firstName", "lastName", "email", "salary")
employee1 = Employee('michael', 'armbrust', 'no-reply@berkeley.edu', 100000)
employee2 = Employee('xiangrui', 'meng', 'no-reply@stanford.edu', 120000)
departmentWithEmployees1 = Row(department=department1, employees=[employee1, employee2])
print(departmentWithEmployees1.employees[0].email)
```

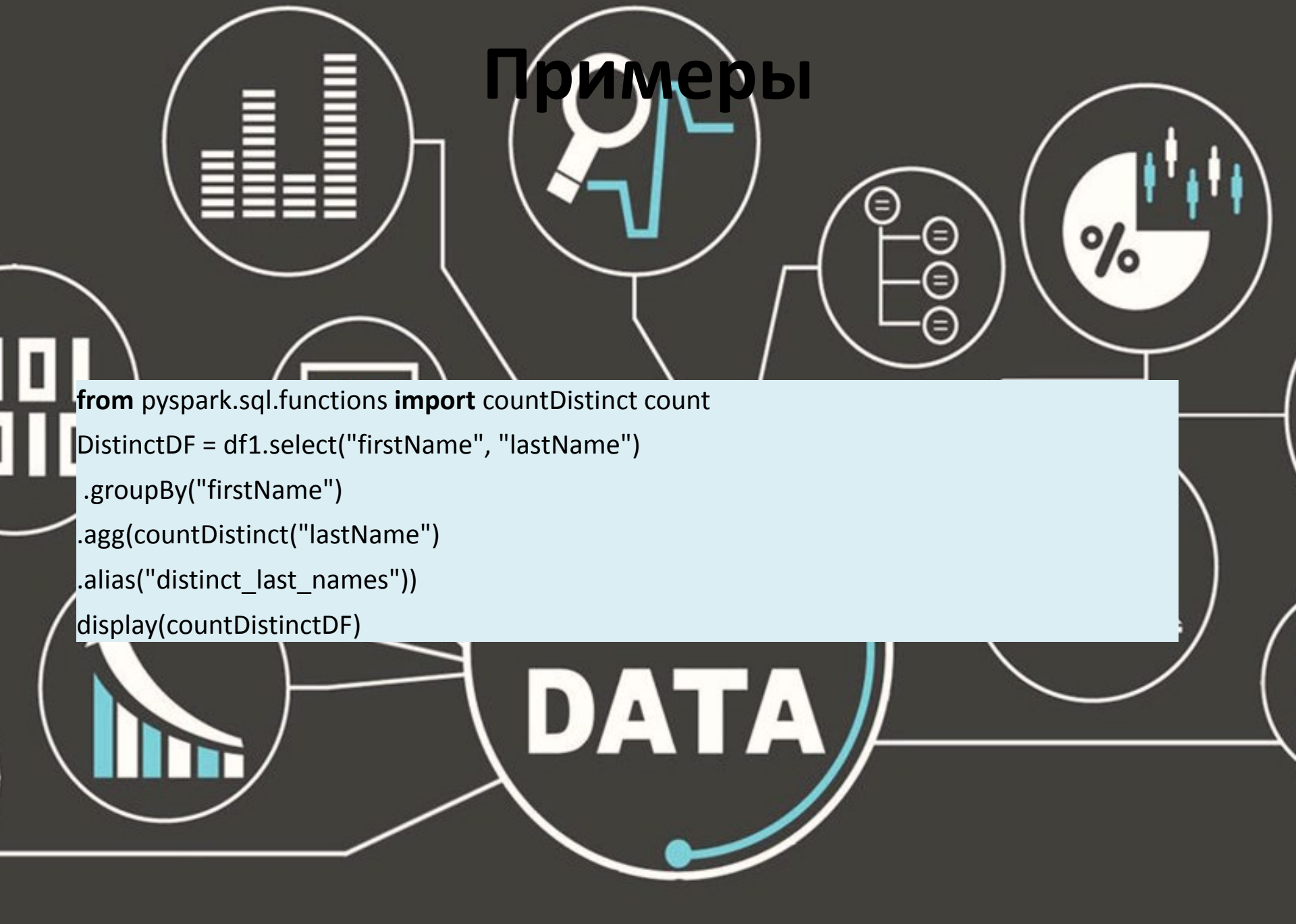
Примеры


```
departmentsWithEmployeesSeq1 = [departmentWithEmployees1,
departmentWithEmployees2]
df1 = spark.createDataFrame(departmentsWithEmployeesSeq1)
display(df1)
```

Примеры


```
filterDF = df1.filter(flattenDF.firstName == "xiangrui").sort(flattenDF.lastName)
display(df1)
```

Примеры

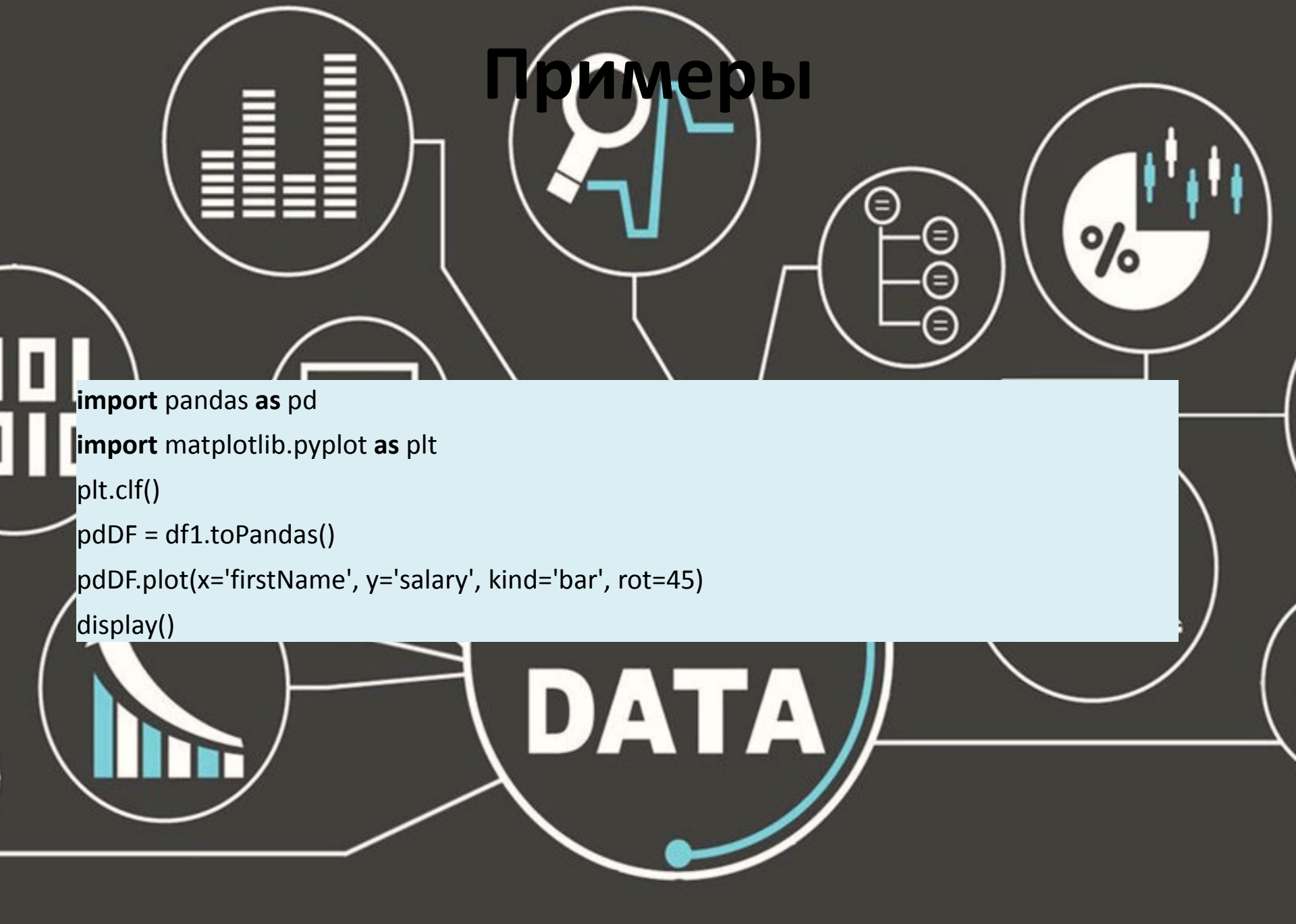



```
from pyspark.sql.functions import countDistinct count
DistinctDF = df1.select("firstName", "lastName")
 .groupBy("firstName")
.agg(countDistinct("lastName")
.alias("distinct_last_names"))
display(countDistinctDF)
```

Примеры


```
salarySumDF = df1.agg({"salary" : "sum"})
display(salarySumDF)
```

Примеры




```
import pandas as pd
import matplotlib.pyplot as plt
plt.clf()
pdDF = df1.toPandas()
pdDF.plot(x='firstName', y='salary', kind='bar', rot=45)
display()
```