Analyzing Flight Delays with Apache Spark GraphFrames and MapR-DB

Agenda

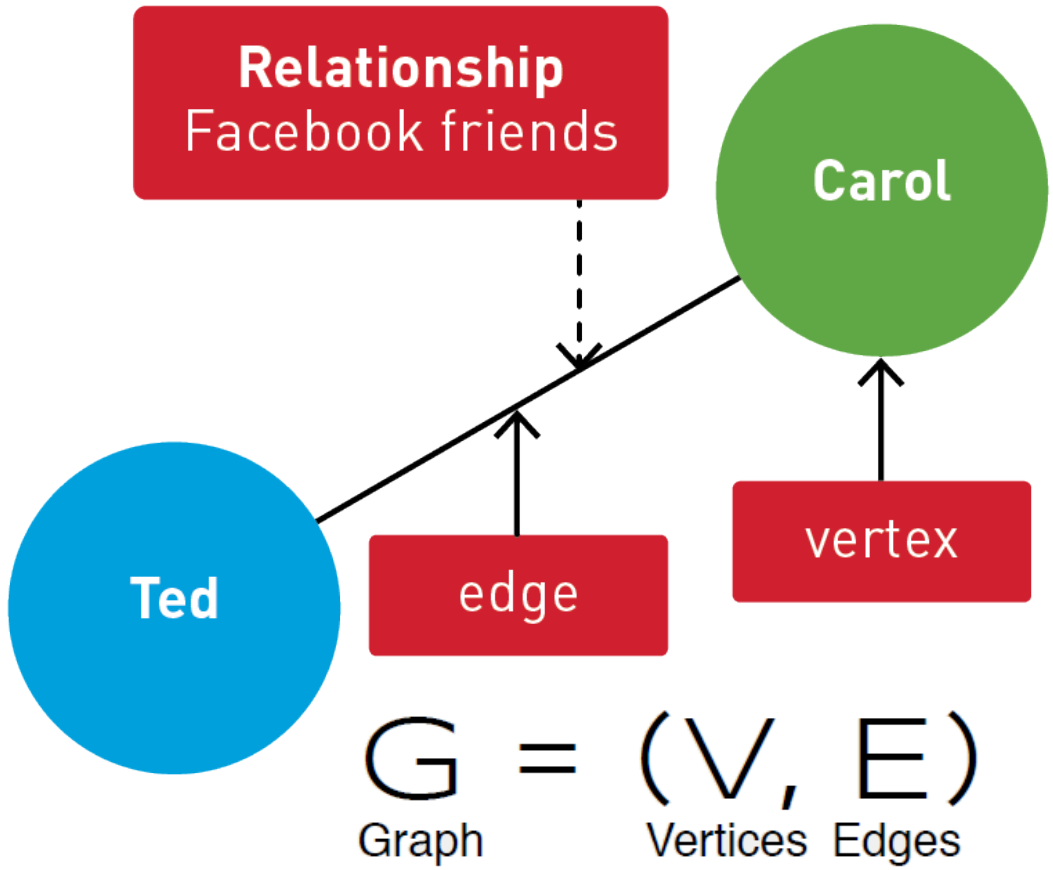
- Introduction to Graphs
- Introduction to GraphFrames with a simple Flight Dataset
- Use GraphFrames with Flight Dataset for 2018

2

Intro to Graphs

What is a Graph?

- Graph: Models Relations between Objects
- Graph: Vertices connected by Edges
- Vertices: the objects
- Edges: the relationships between Vertices

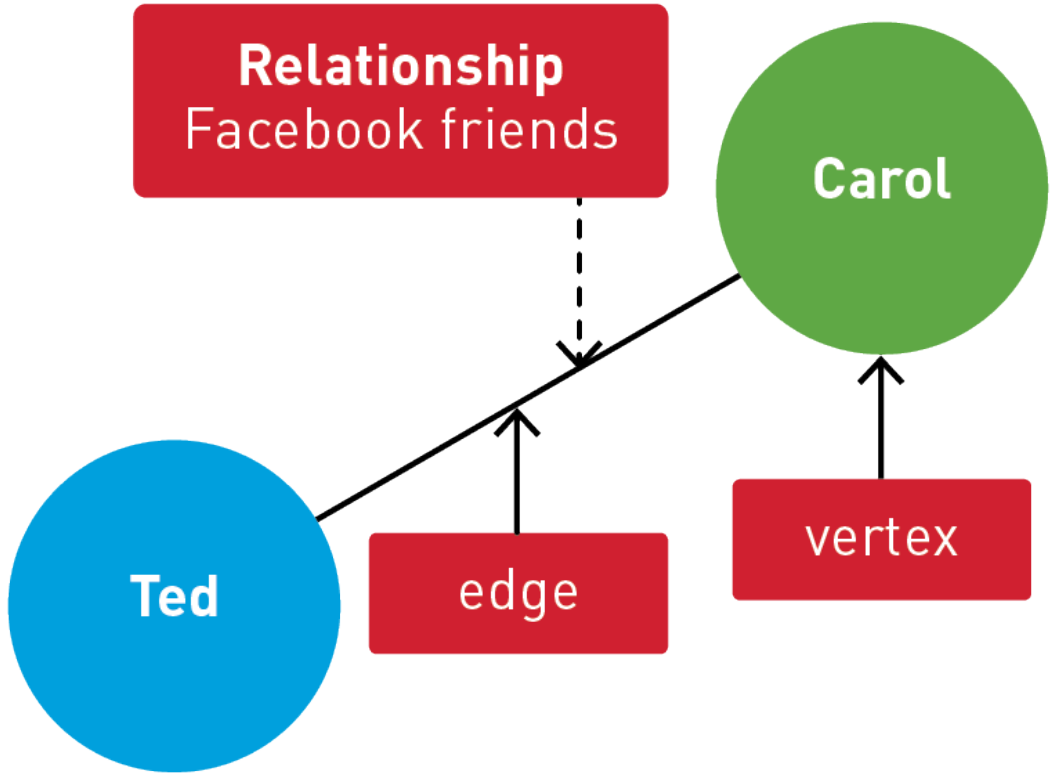


Regular Graphs vs Directed Graphs

Regular graph: each vertex has the same number of edges

Example: Facebook friends

- Ted is a friend of Carol
- Carol is a friend of Ted

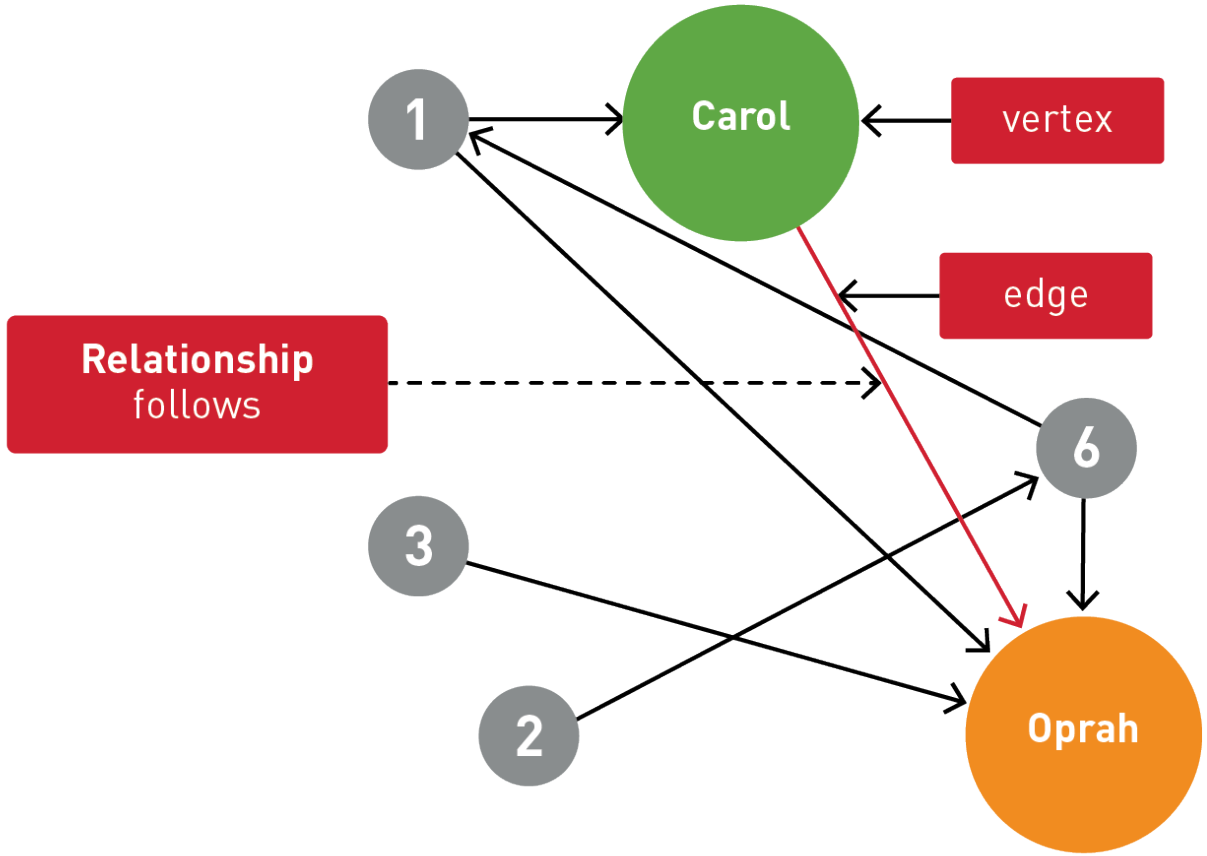


Regular Graphs vs Directed Graphs

Directed graph: edges have a direction

Example: Twitter followers

- Carol follows Oprah
- Oprah does not follow Carol

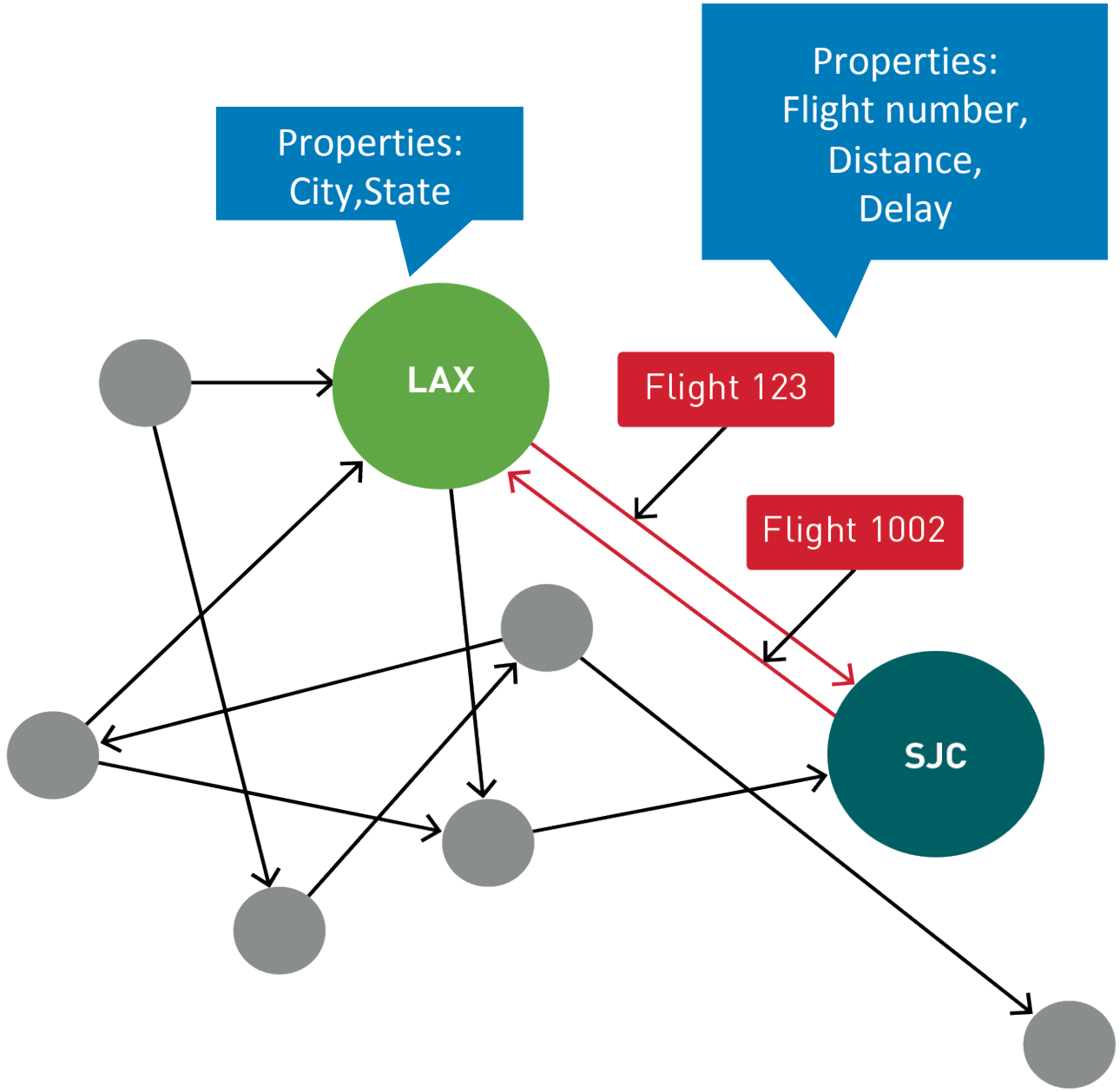


Property Graph

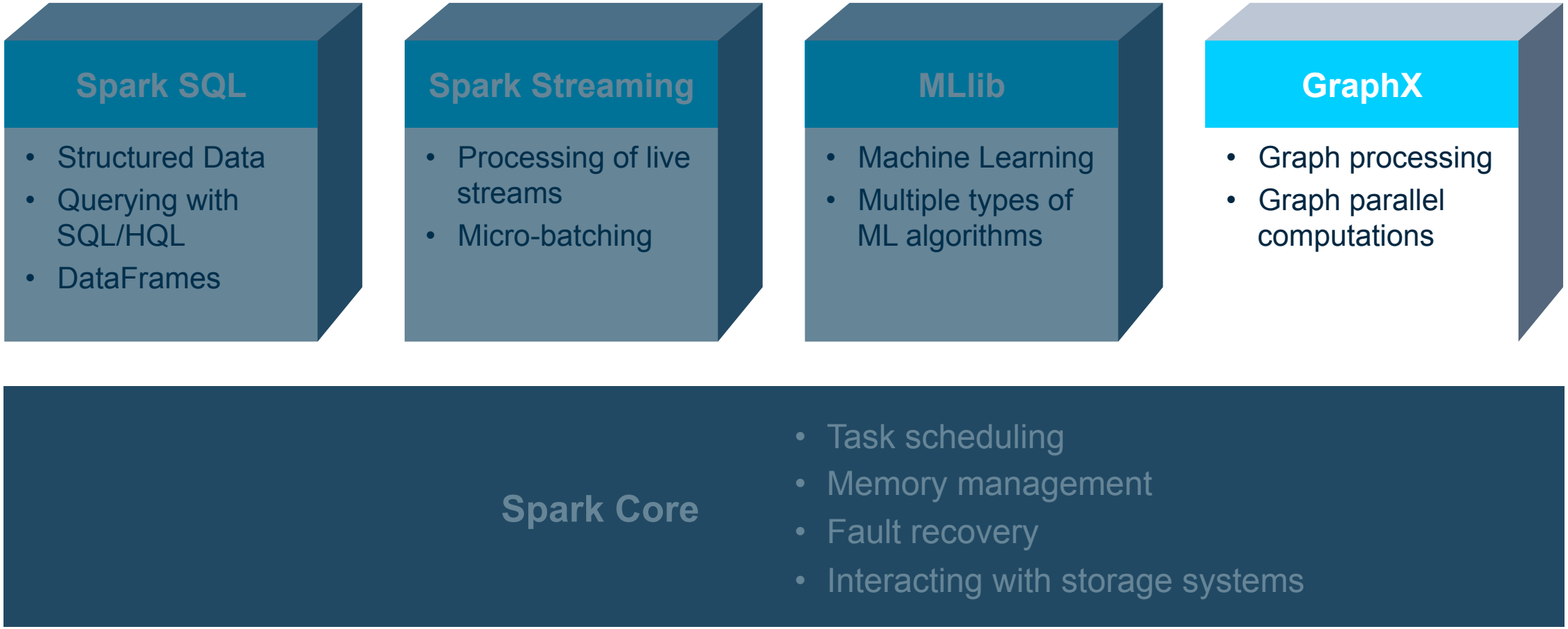
Property Graph:

- Edges and Vertexes have properties
- Vertex can have multiple directed edges in parallel
    - Allows **multiple relationships**

Spark GraphX supports a distributed property graph.



What is GraphX?

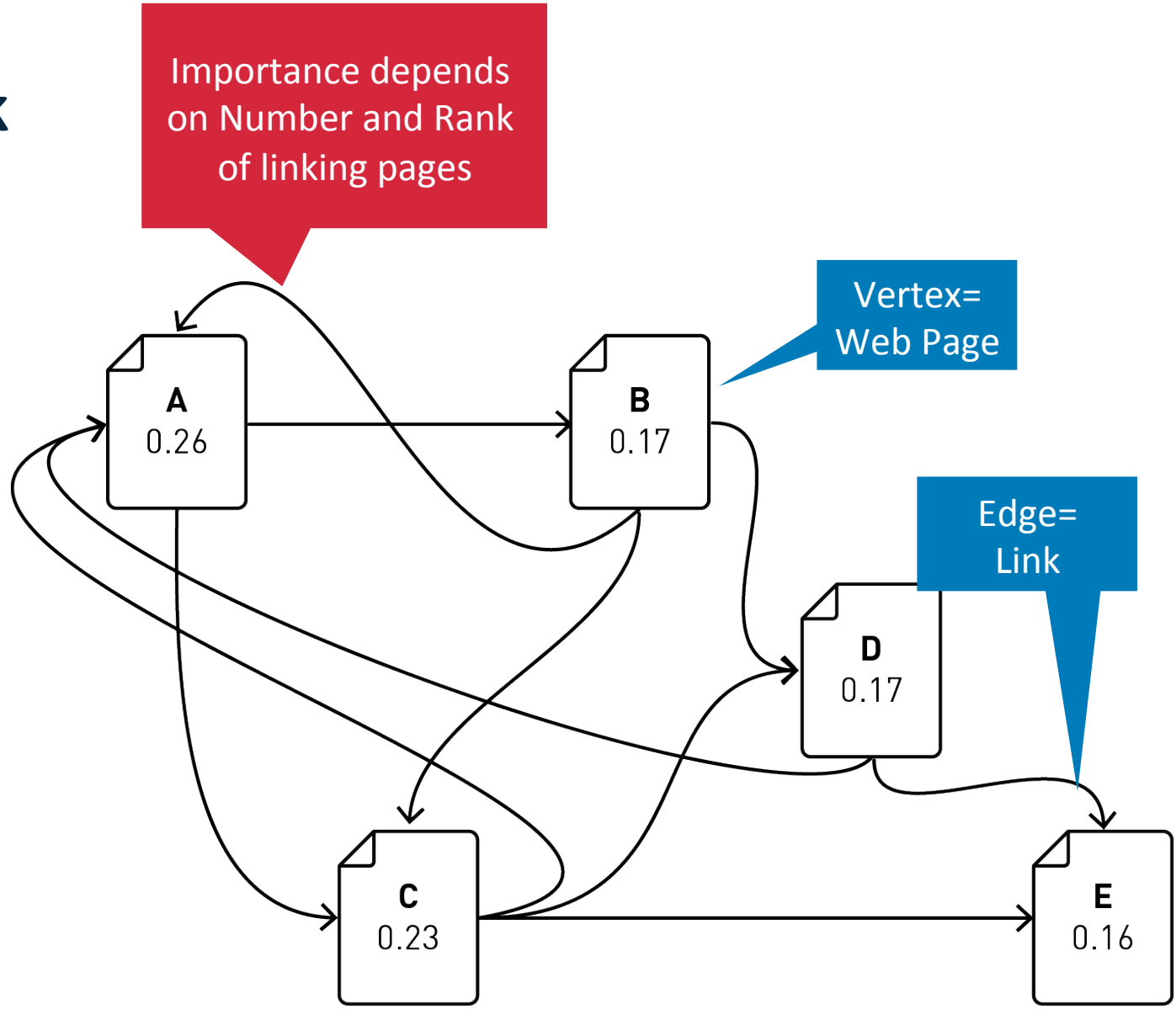


Graph Algorithms and Graph Queries with GraphFrames

Graph Algorithms: PageRank

Web Sites

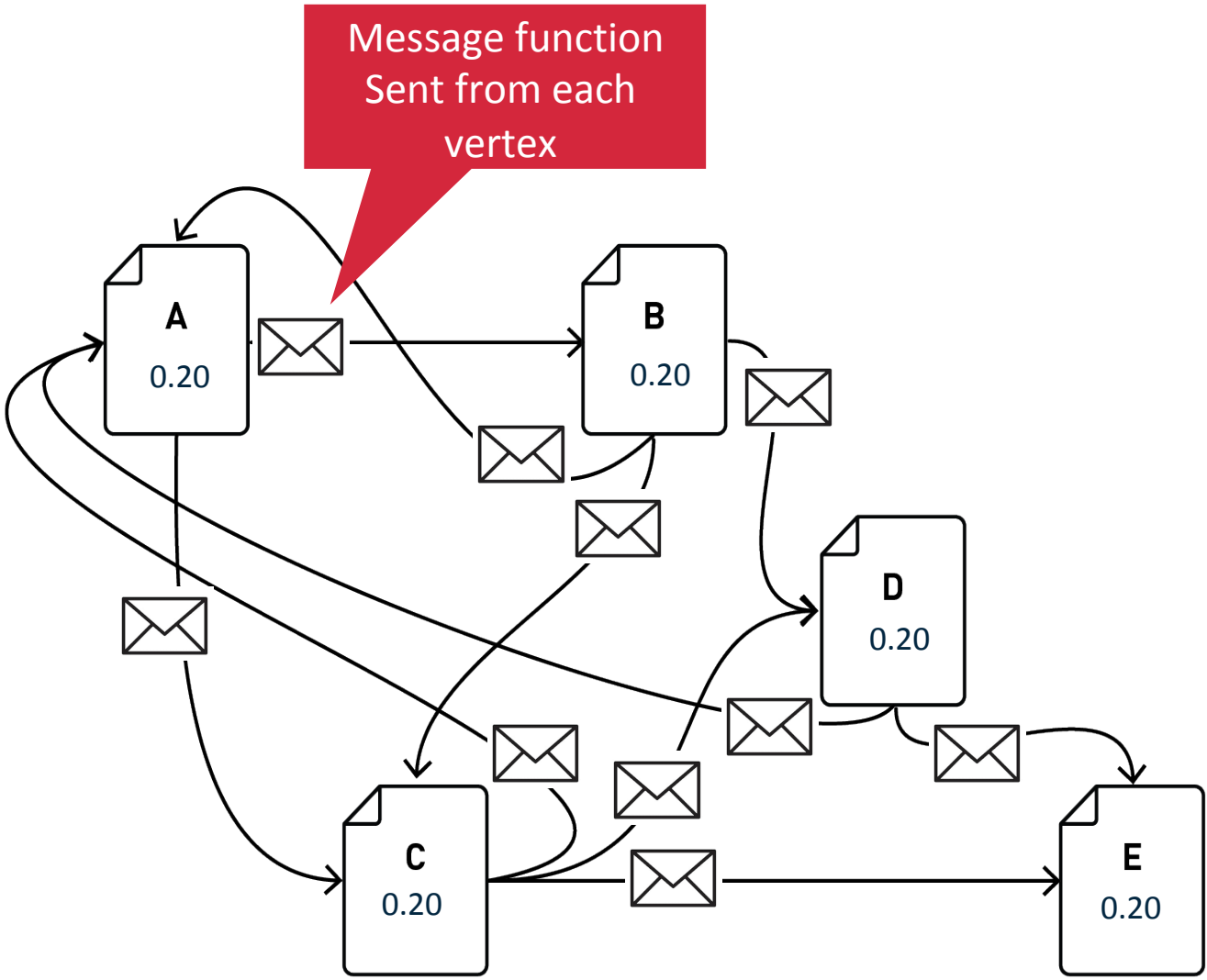
- Vertices = Web Pages
- Edges = Links between Pages
- PageRank Importance =
  - Iterative Number of Links to a page and it's linking pages
- Twitter Example: who has the most twitter followers



Graph Algorithms: PageRank

Visualize PageRank

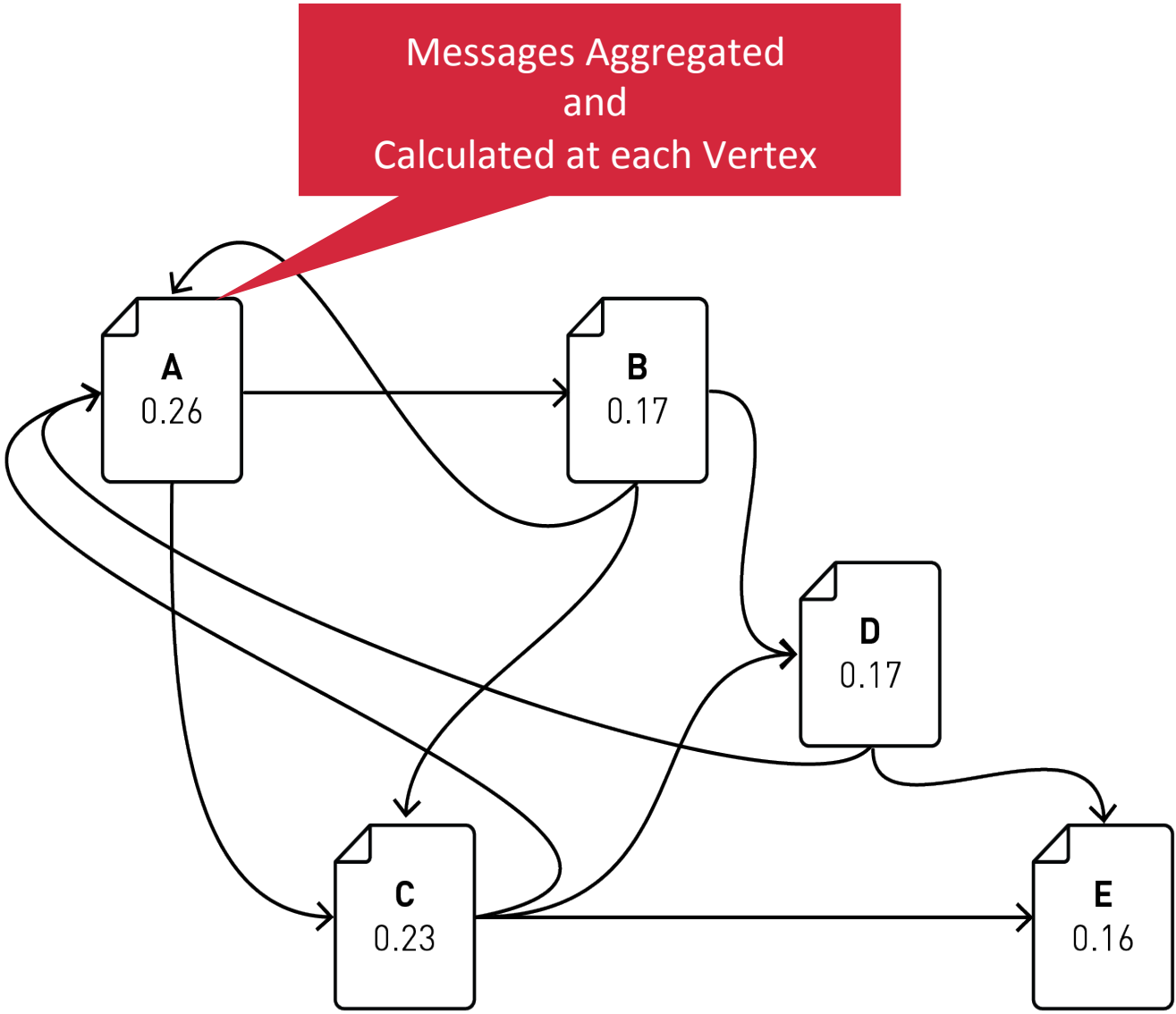
1. Each page sends message function with it's "rank" to neighbors



Graph Algorithms: PageRank

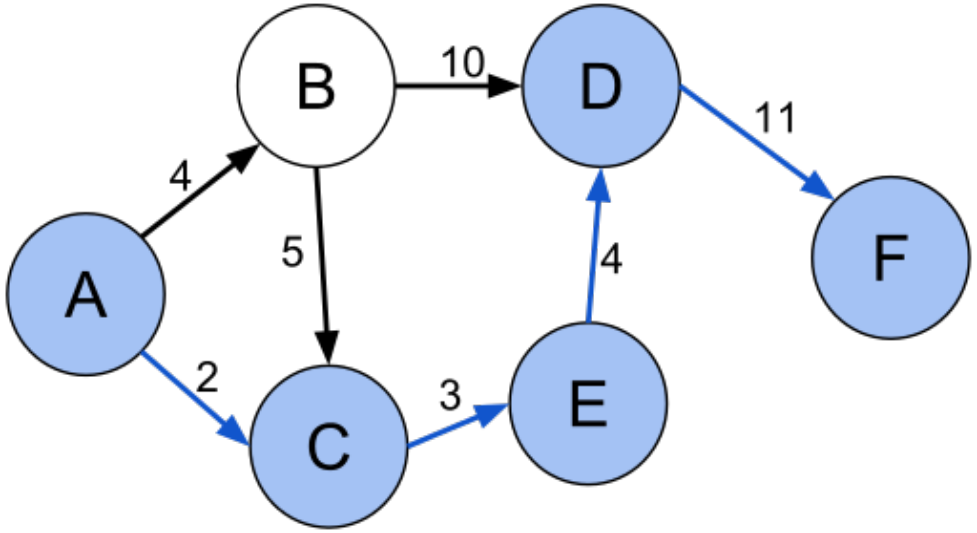
Visualize PageRank

1. Each page sends message function with it's "rank" to neighbors
2. Messages are Aggregated and Calculated at each destination vertex
3. Sum of messages becomes new vertex Page rank
4. Repeat

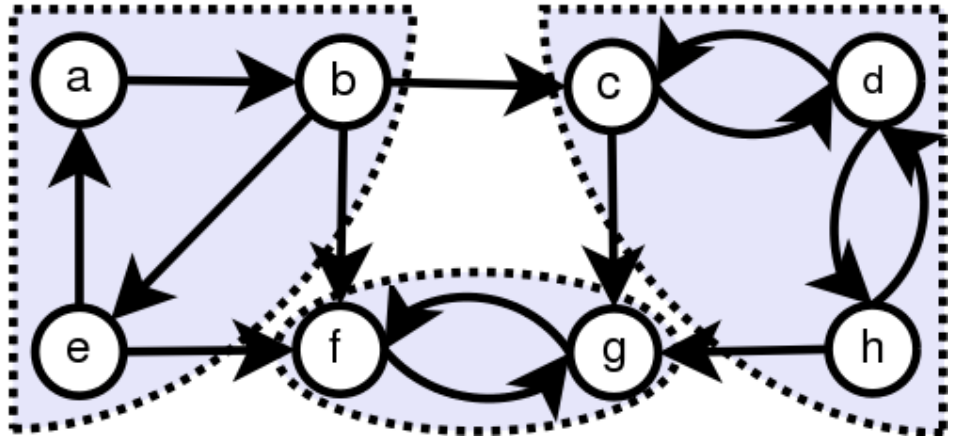


Graph Algorithms

- Many Graph Algorithms Aggregate properties of neighbors:
  - PageRank
  - Connected Components
  - Shortest Path



Shortest Path A to F



Connected Components

Reference https://en.wikipedia.org/

Graph Motif Queries

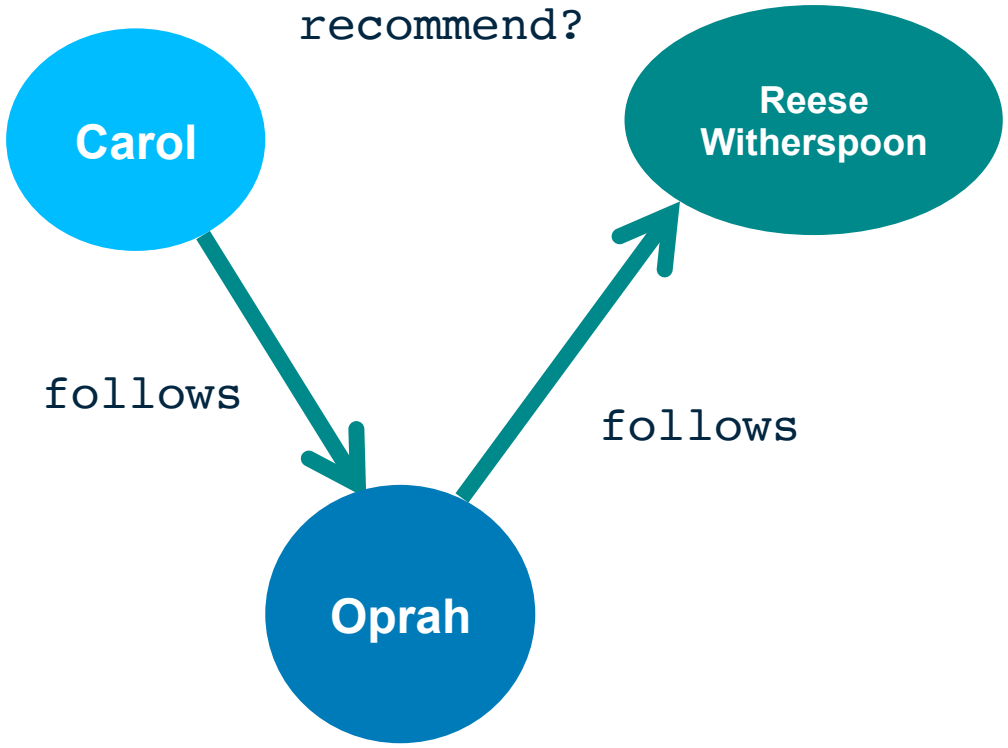
Graph Motif: recurrent patterns in a graph

Graph Motif Query: Search a graph for occurrences of a given a pattern

Twitter Example:

Who should we recommend for Carol to Follow?

- Carol follows Oprah
- Oprah follows Reese Witherspoon
- Recommend Carol to follow Reese



Graph Motif Queries

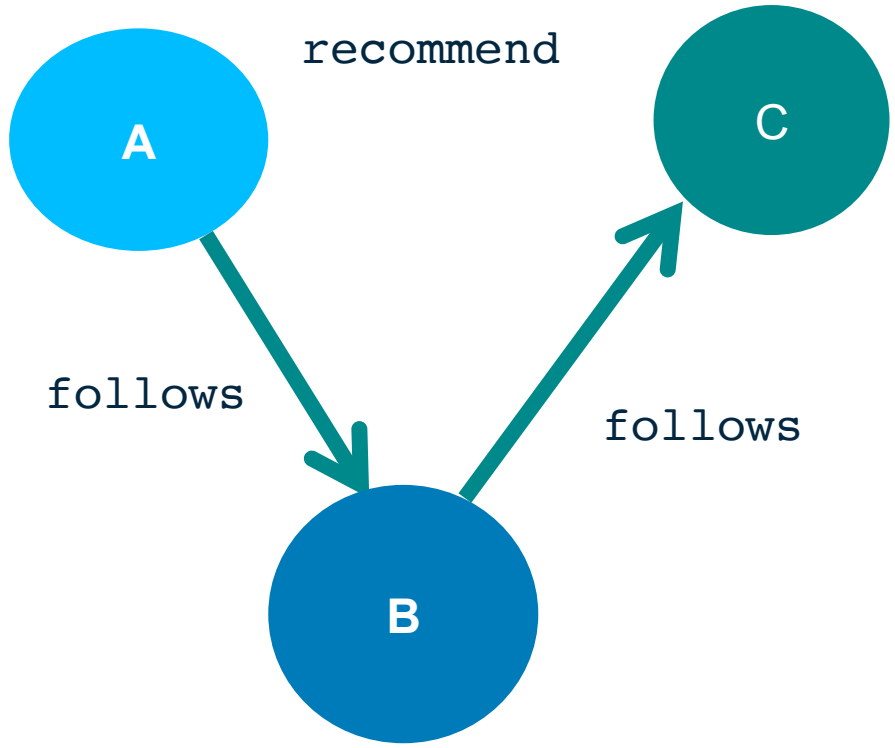
Graph Motif: recurrent patterns in a graph

Graph Motif Query: Search a graph for occurrences of a given a pattern

Twitter Example:

Recommend who to Follow? Search for patterns

- A follows B
- B follows C
- A does not follow C

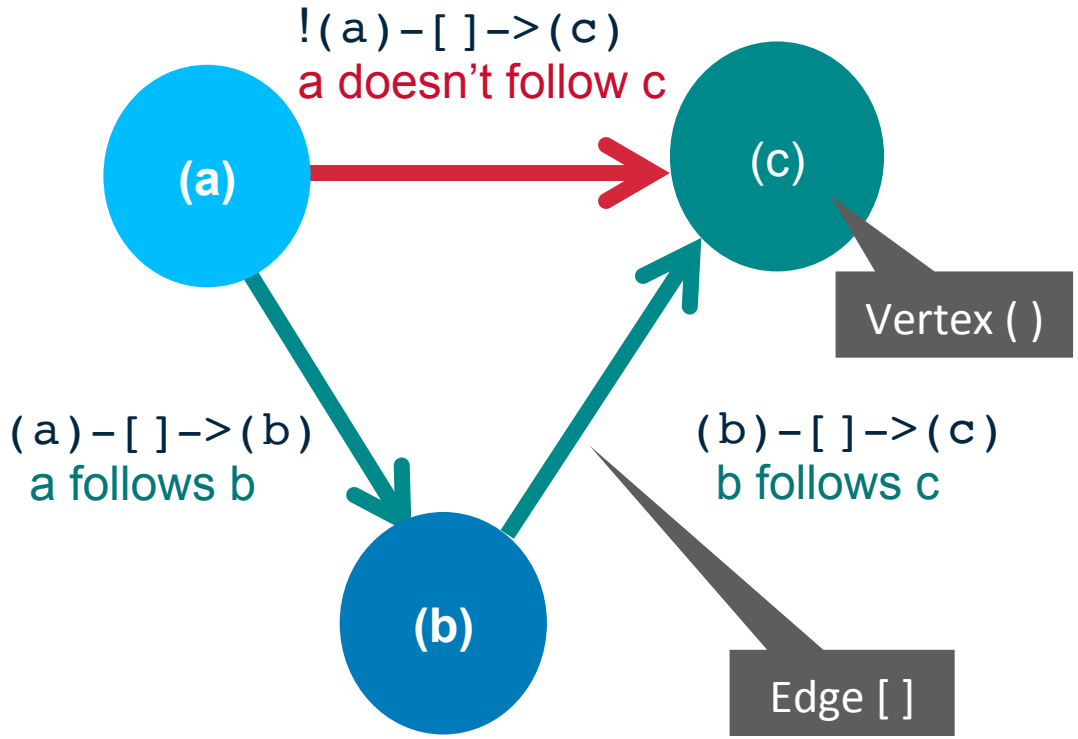


Graph Query: Motif Find Structural Pattern

Twitter: A follows B; B follows C; A doesn't follow C

```
graph.find("(a)-[]->(b); (b)-[]->(c); !(a)-[]->(c)")
```

Search for a pattern



Separate Systems

Graph Algorithms

Graph Queries

Image reference Spark Summit

GraphFrames: Graph Algorithms + Graph Queries

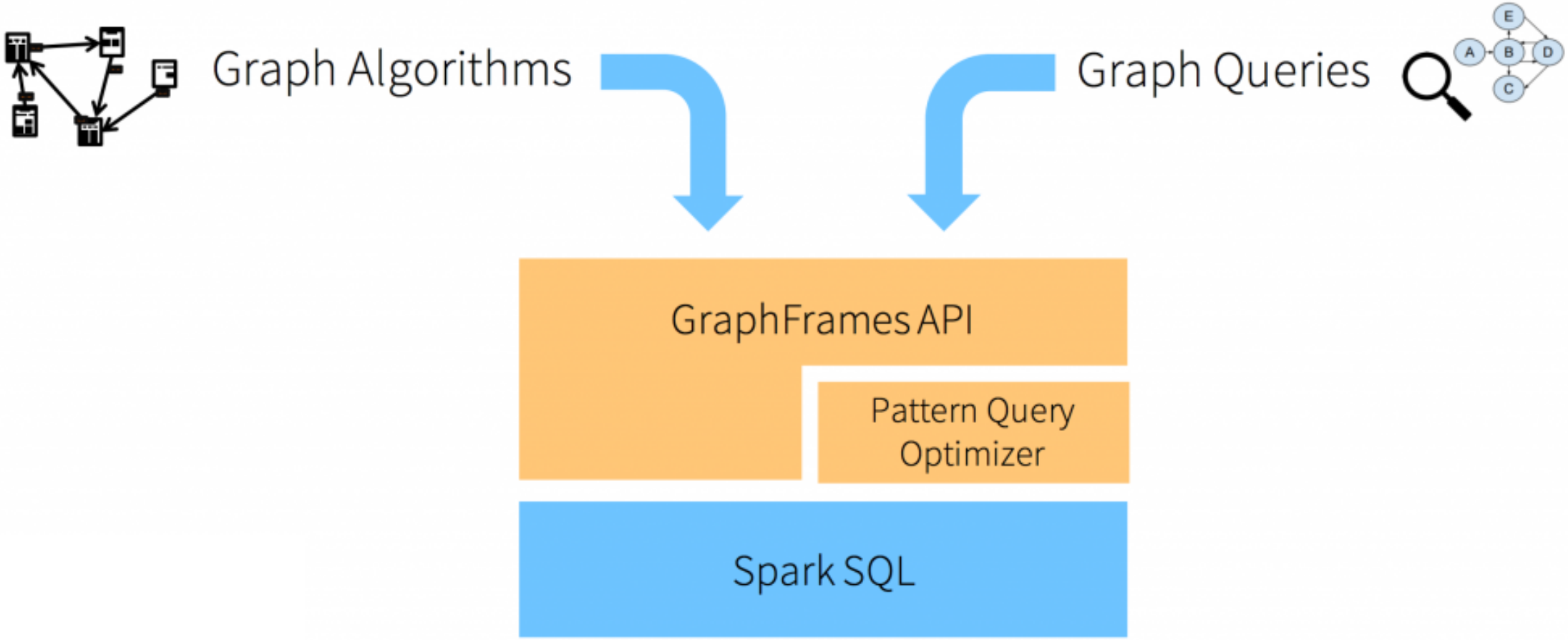


Image reference Spark Summit

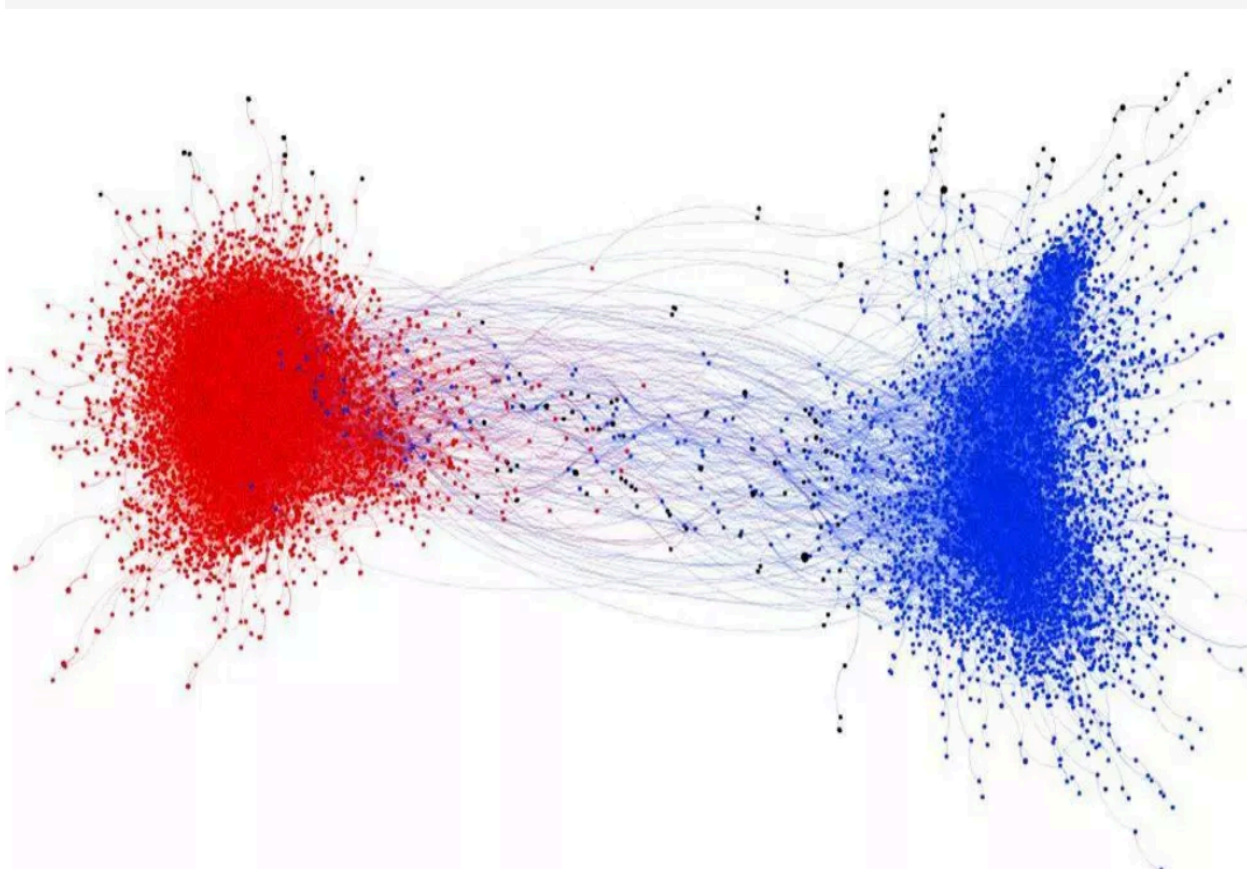
Graph Examples

Real World Graphs: Twitter

Twitter Tweets:

morally outraged tweets retweeted within political sphere

But rarely outside sphere

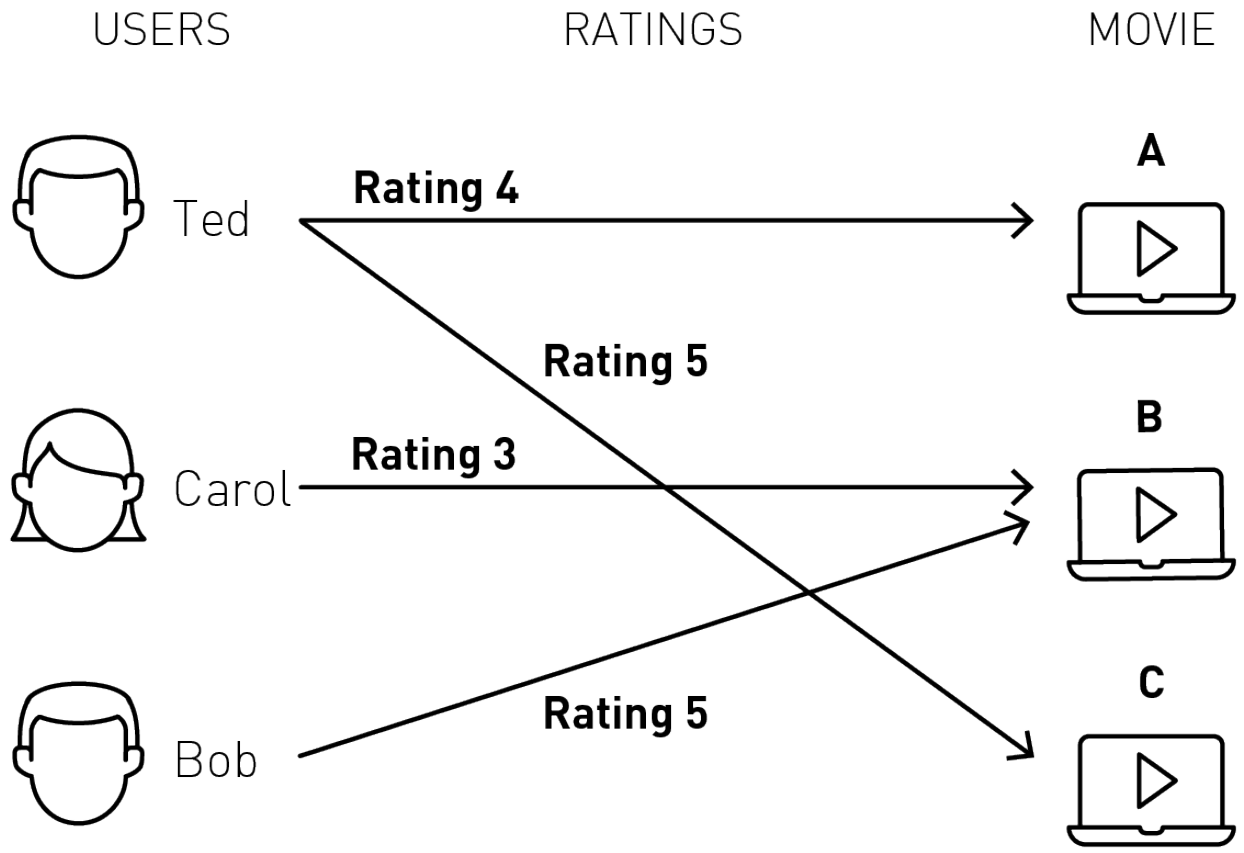


Reference National Academy of Sciences

Graph: Recommendation Engines

Recommendation Engine:

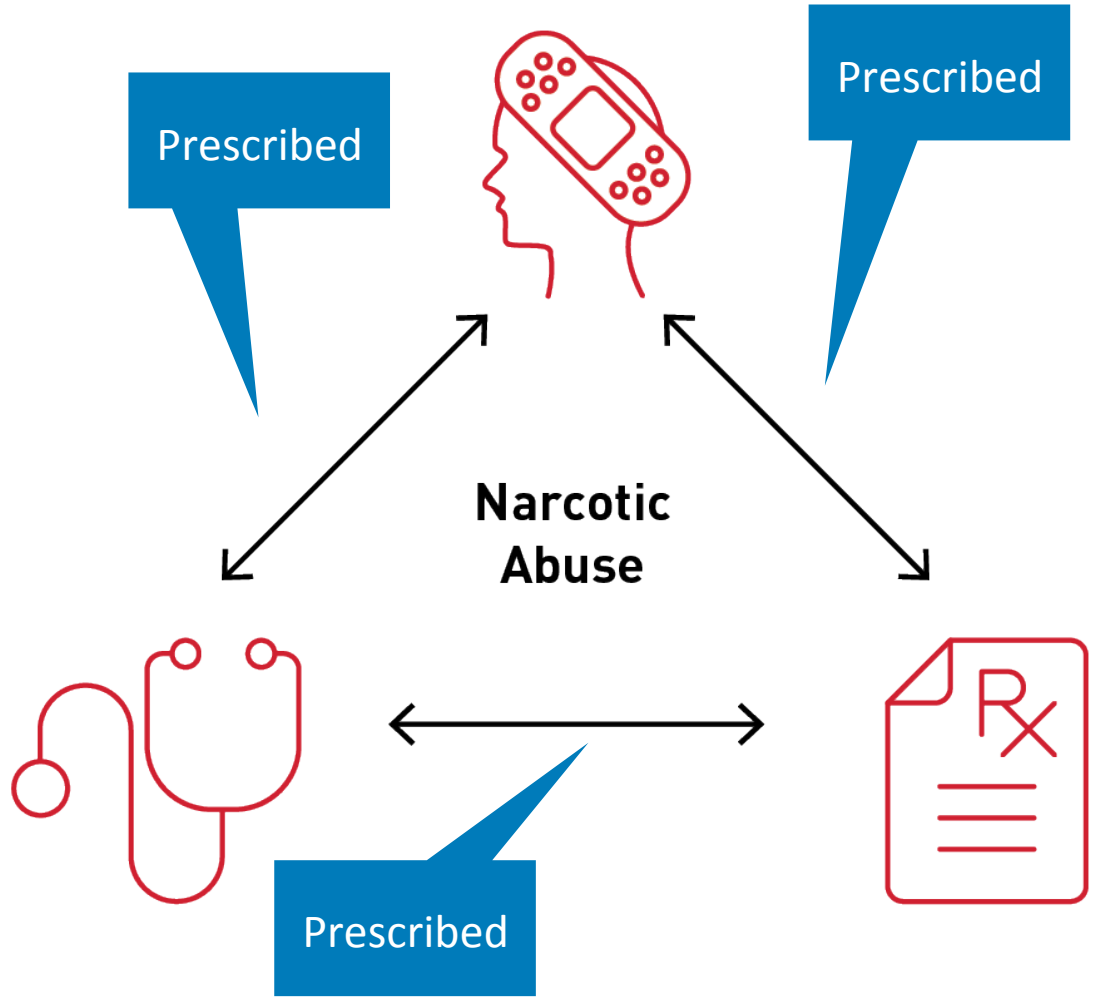
- Vertices = Users, Products
- Edges = Ratings or Purchases
- Calculate how similar users rated similar products



Real World Graphs: Fraud

Healthcare Fraud:

- Vertices = Doctors, Patients, Prescriptions
- Edges = prescribed
- Calculate Narcotic Abuse, Patient Similarity, Over prescribing



Real World Graphs: Fraud

Credit Card Aplication Fraud:

- Vertices = Credit Card Applicant, Phone, email, address, ssn
- Edges = Identifier
- Detect People sharing identifiers such as telephone number

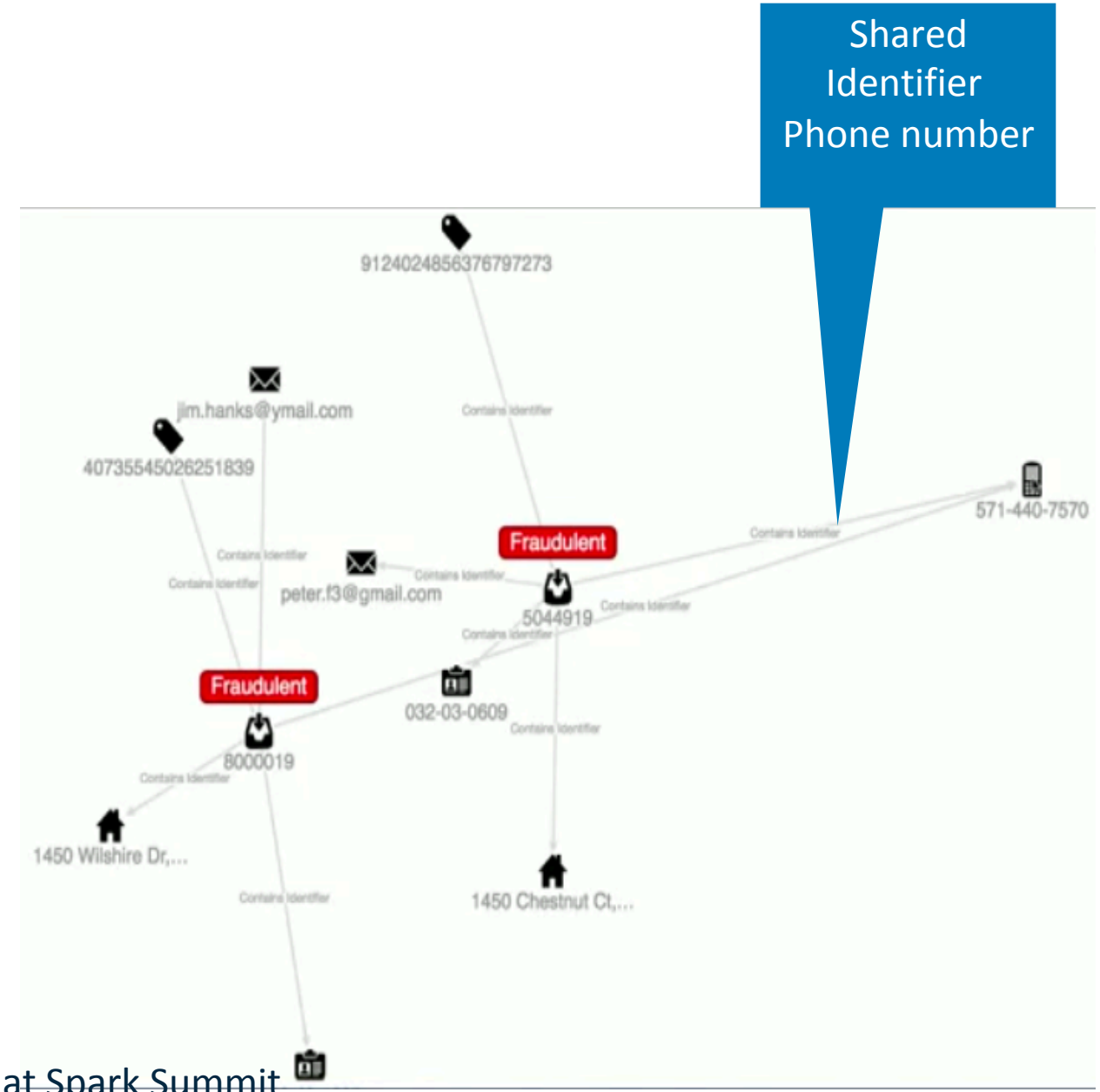
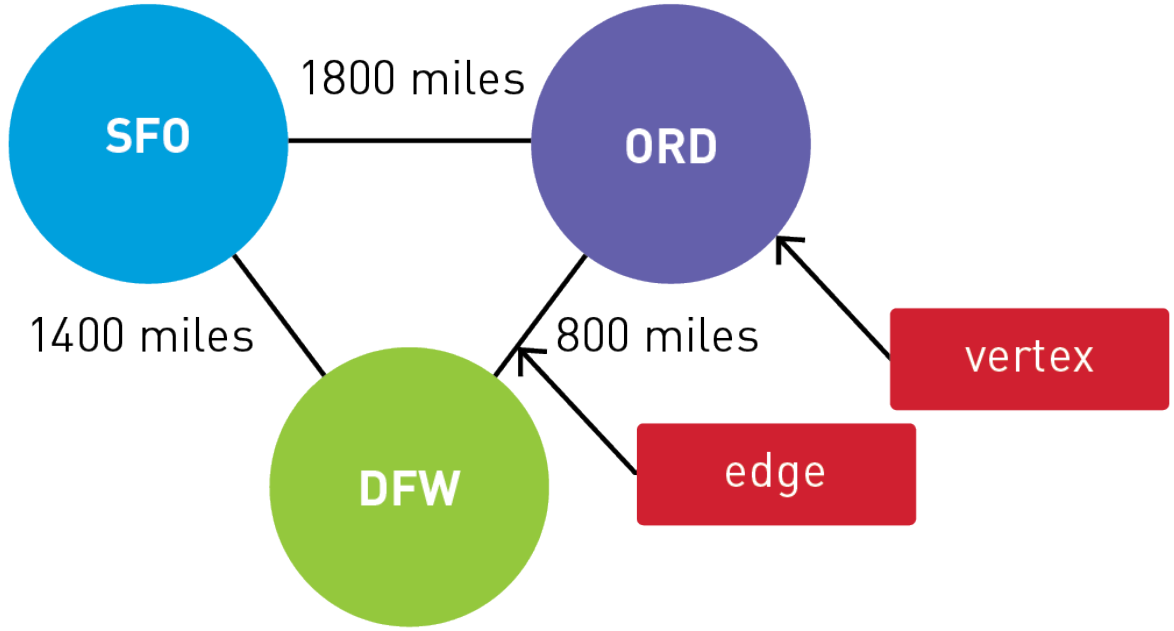


Image reference Capitol One at Spark Summit

A Simple Flight Example with GraphFrames

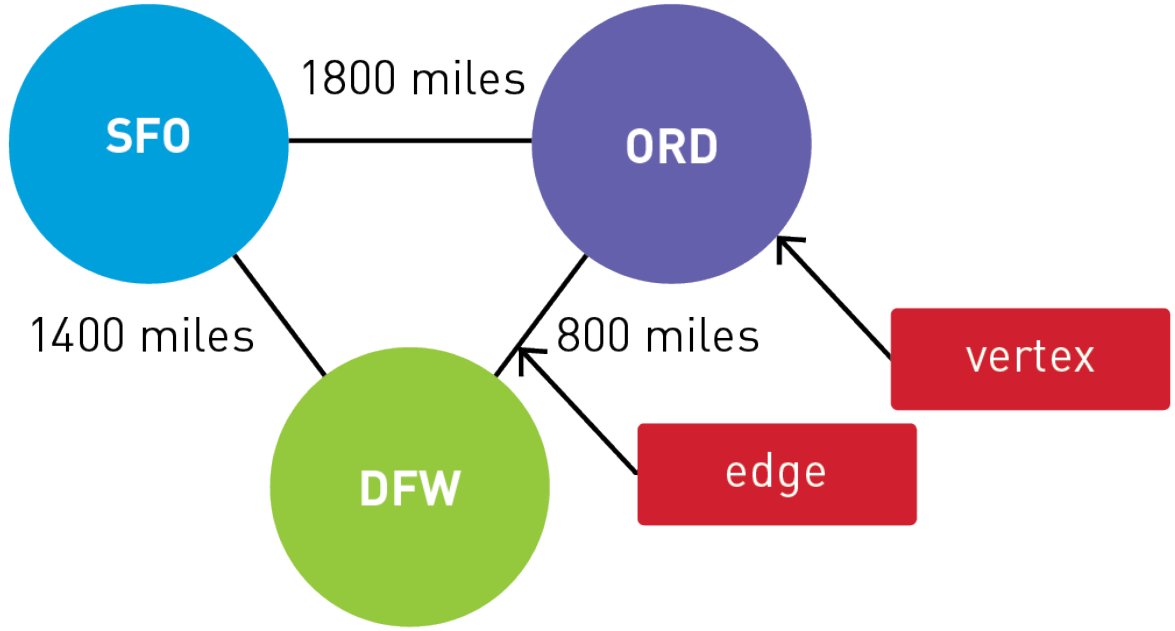
Simple Flight Example with GraphFrames

Originating Airport	Destination Airport	Distance	Delay
SFO	ORD	1800 miles	40
ORD	DFW	800 miles	0
DFW	SFO	1400 miles	10



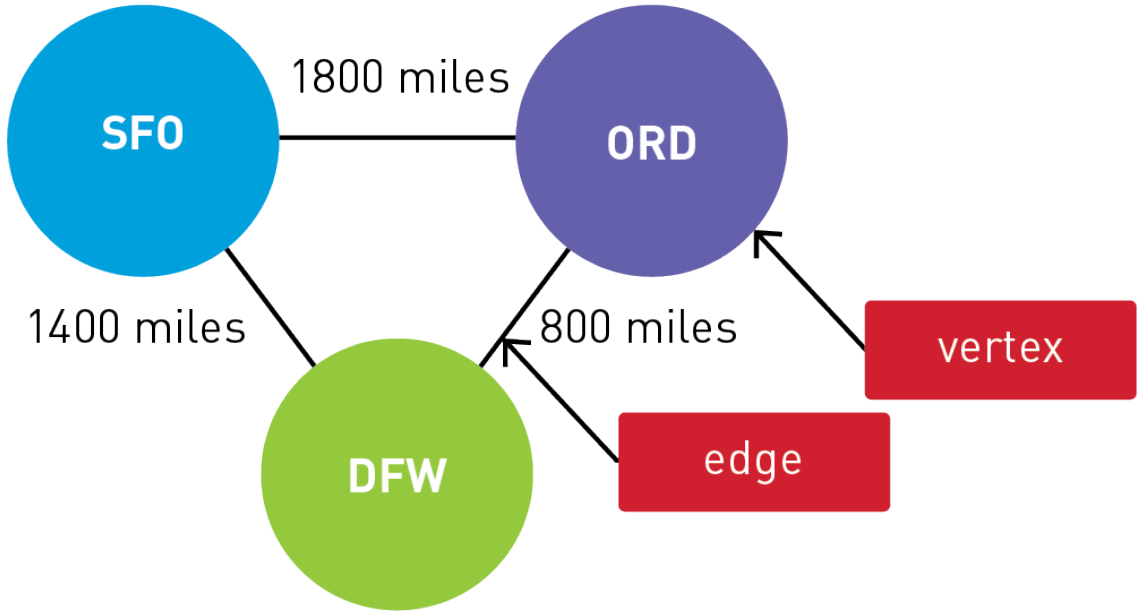
Vertex Table

id	city
SFO	San Francisco
ORD	Chicago
DFW	Texas



Edges Table

Src	Dst	Distance	Delay
SFO	ORD	1800	40
ORD	DFW	800	0
DFW	SFO	1400	10



Create a Vertices DataFrame

Id	City
SFO	San Francisco
ORD	Chicago
DFW	Dallas

```
case class Airport(id: String, city: String)

val airports=Array(Airport("SFO","San Francisco"),
 Airport("ORD","Chicago"), Airport("DFW","Dallas Fort Worth"))

val vertices = spark.createDataset(airports).toDF
vertices.show
+---+-----------------+
| id|             city|
+---+-----------------+
|SFO|    San Francisco|
|ORD|          Chicago|
|DFW|Dallas Fort Worth|
+---+-----------------+
```

Create an Edges DataFrame

Src	Dst	Distance	Delay
SFO	ORD	1800	40
ORD	DFW	800	0
DFW	SFO	1400	10

```
case class Flight(id: String, src: String, dst: String,
      dist: Double, delay: Double)
val flights=Array(
      Flight("SFO_ORD_2017-01-01_AA”,"SFO”,"ORD”,1800, 40),
      Flight("ORD_DFW_2017-01-01_UA","ORD","DFW",800, 0),
      Flight("DFW_SFO_2017-01-01_DL","DFW","SFO",1400, 10))
val edges = spark.createDataset(flights).toDF
edges.show
+--------------------+---+---+------+-----+
|                  id|src|dst|  dist|delay|
+--------------------+---+---+------+-----+
|SFO_ORD_2017-01-0...|SFO|ORD|1800.0| 40.0|
|ORD_DFW_2017-01-0...|ORD|DFW| 800.0|  0.0|
|DFW_SFO_2017-01-0...|DFW|SFO|1400.0| 10.0|
+--------------------+---+---+------+-----+
```

Create the GraphFrame

```
val graph = GraphFrame(vertices, edges)

graph.vertices.show

+---+-----------------+
| id|             name|
+---+-----------------+
|SFO|    San Francisco|
|ORD|          Chicago|
|DFW|Dallas Fort Worth|
+---+-----------------+
```

GraphFrame Edges

Src	Dst	Distance	Delay
SFO	ORD	1800	40
ORD	DFW	800	0
DFW	SFO	1400	10

```
graph.edges.show
```

result:

```
+--------------------+---+---+------+-----+
|                  id|src|dst|  dist|delay|
+--------------------+---+---+------+-----+
|SFO_ORD_2017-01-0...|SFO|ORD|1800.0| 40.0|
|ORD_DFW_2017-01-0...|ORD|DFW| 800.0|  0.0|
|DFW_SFO_2017-01-0...|DFW|SFO|1400.0| 10.0|
+--------------------+---+---+------+-----+
```

Graph Operators

To answer questions such as:

How many airports are there?

How many flight routes are there?

What are the longest distance routes?

Which airport has the most incoming flights?

What are the top 10 flights?

Query the GraphFrame

```
// How many airports?
graph.vertices.count

result:  = 3

// How many flights?
graph.edges.count

result: = 3
```

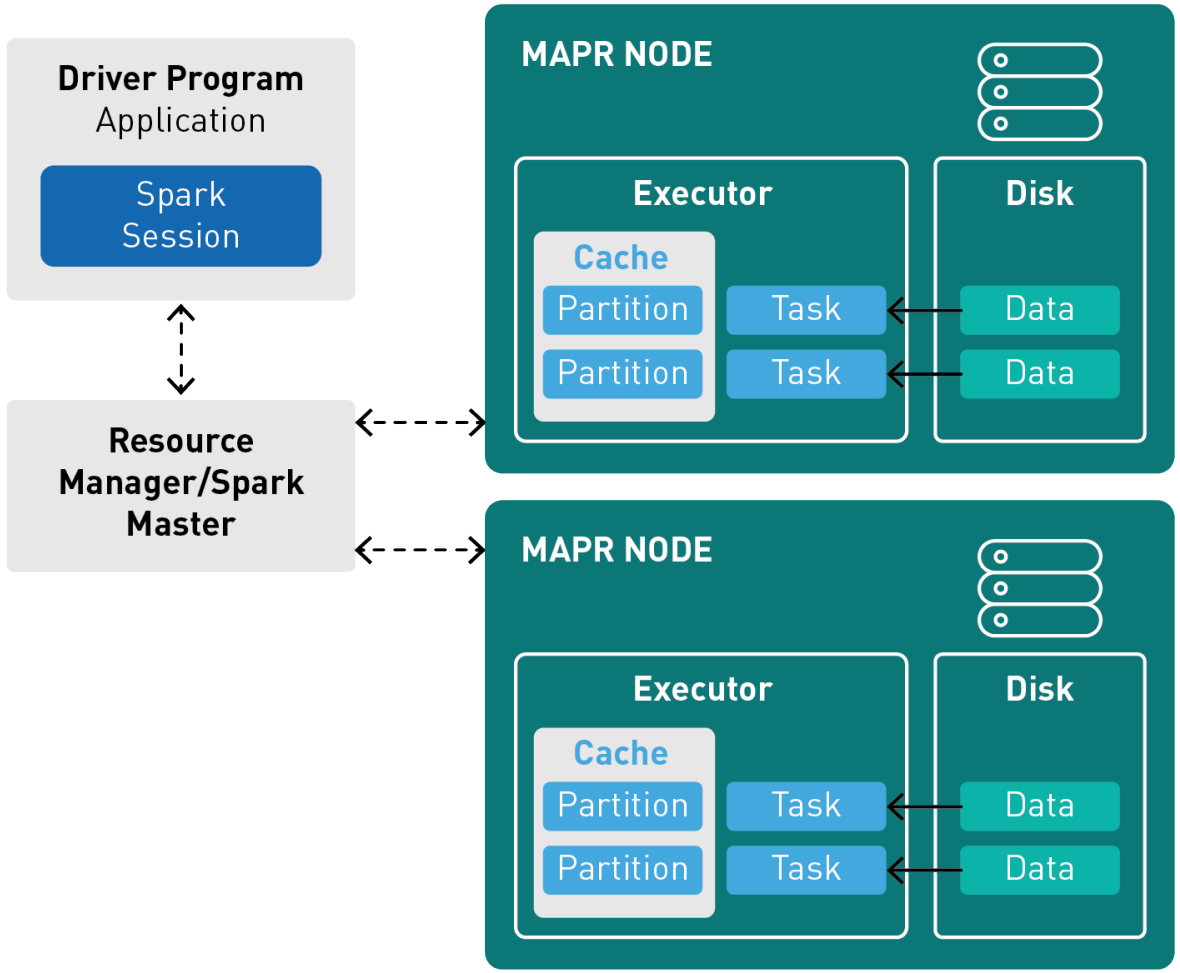
Query the GraphFrame

```
// flight routes >  800 miles distance?
graph.edges.filter("dist > 800").show
```

```
+--------------------+---+---+------+-----+
|                  id|src|dst|  dist|delay|
+--------------------+---+---+------+-----+
|SFO_ORD_2017-01-0...|SFO|ORD|1800.0| 40.0|
|DFW_SFO_2017-01-0...|DFW|SFO|1400.0| 10.0|
+--------------------+---+---+------+-----+
```

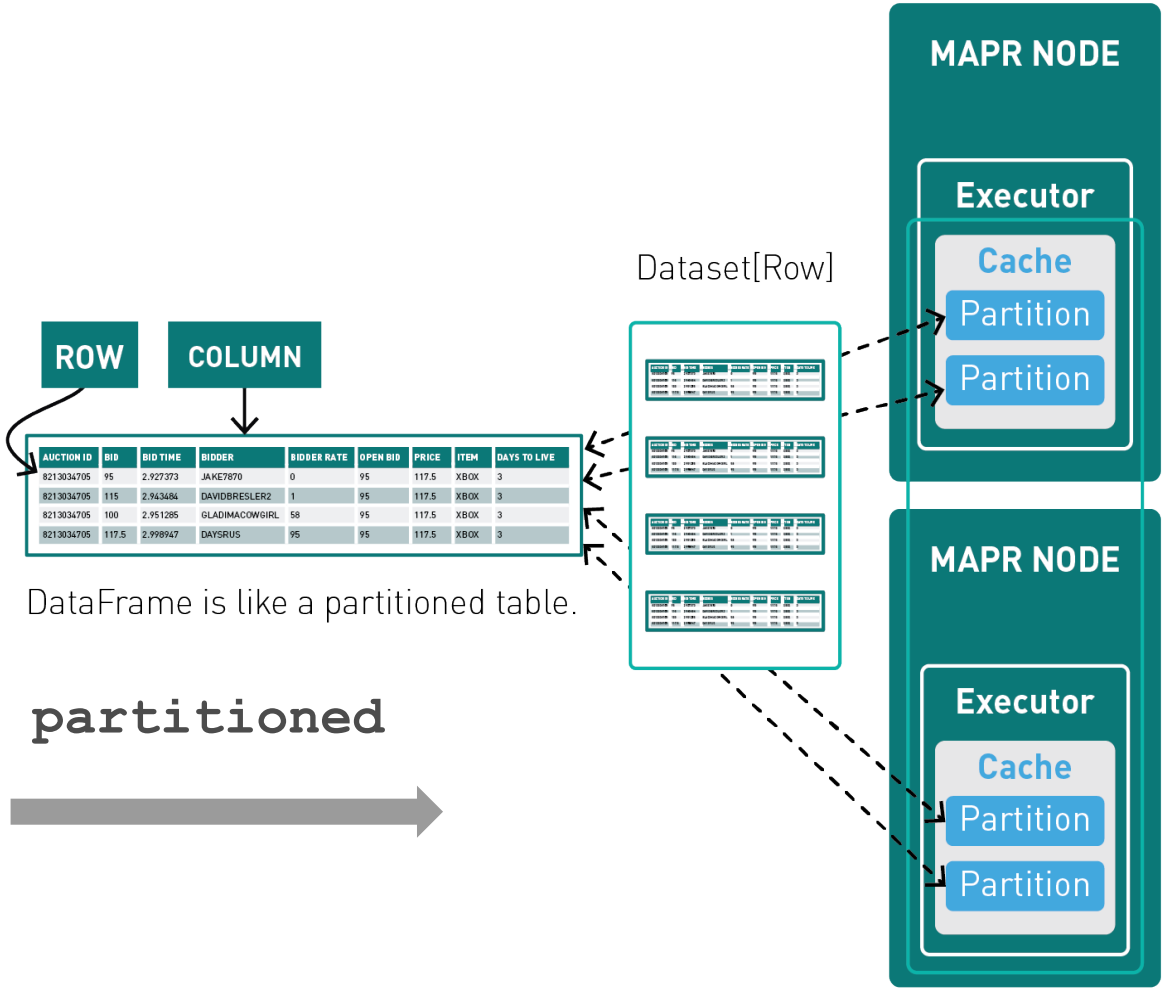
Loading and Exploring the MapR-DB Flight Table with DataFrames

How a Spark Application Runs on a Cluster



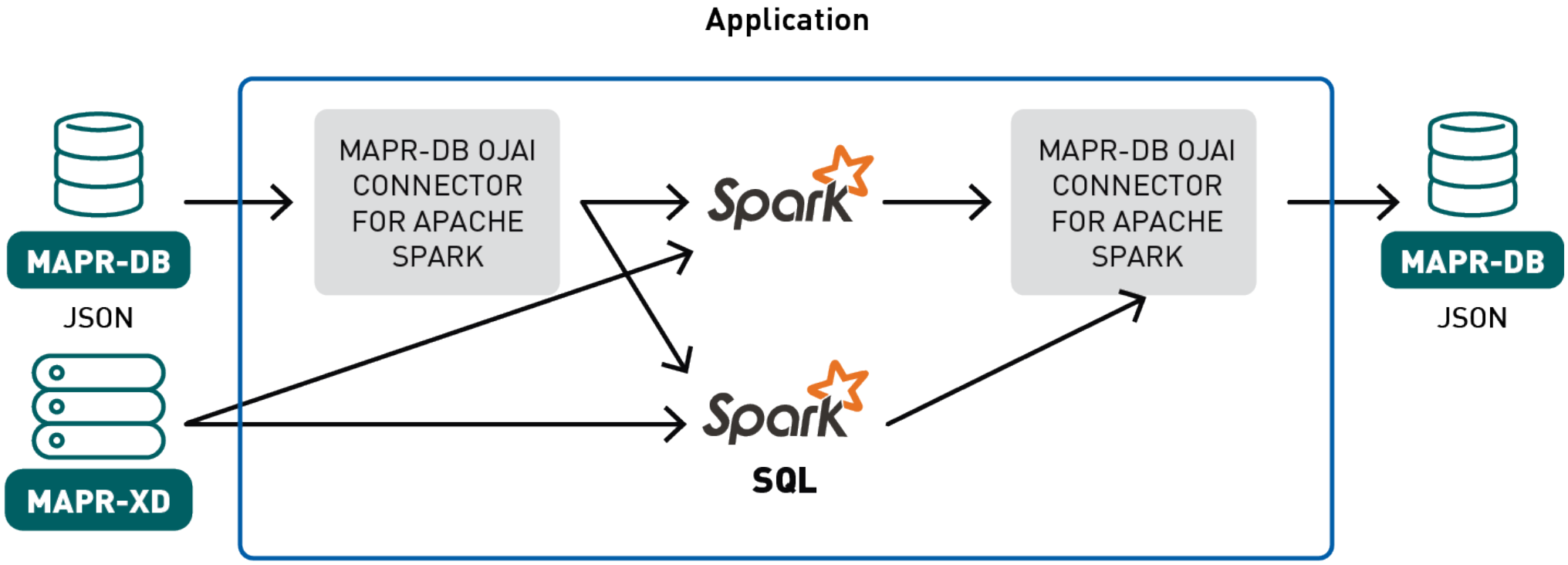
Spark Distributed Datasets

- A Dataset is a collection of Typed Objects
  - Dataset[T]
  - (can use SQL and functions)
- A DataFrame is a Dataset of Row objects
  - Dataset[Row]
  - (can use SQL)
- **Partitioned** across a cluster
- Operated on in **parallel**
- can be **Cached**

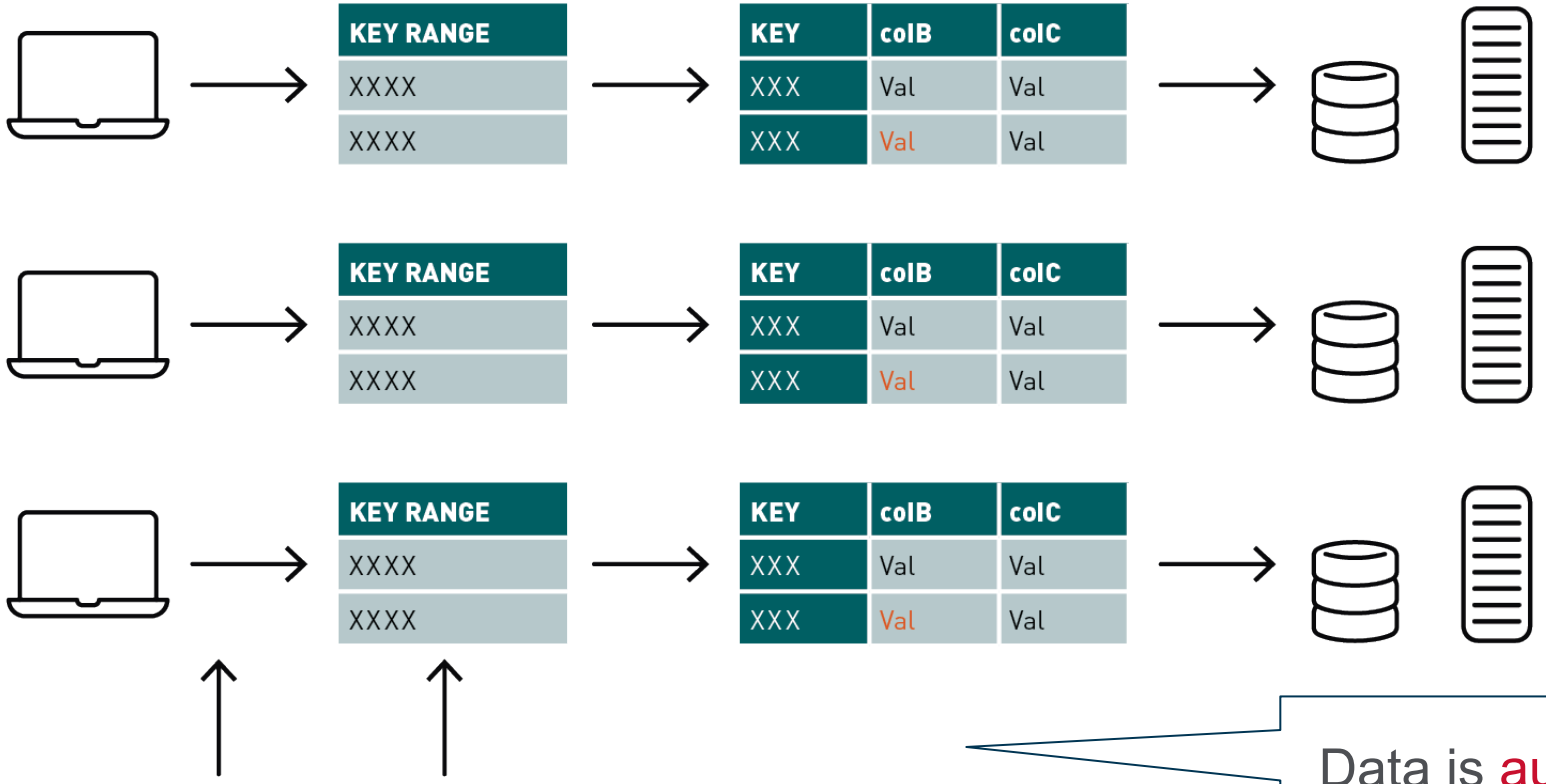


Spark SQL Querying MapR-DB JSON

- Spark SQL queries and updates to MapR-DB
- With projection and filter pushdown, custom partitioning, and data locality



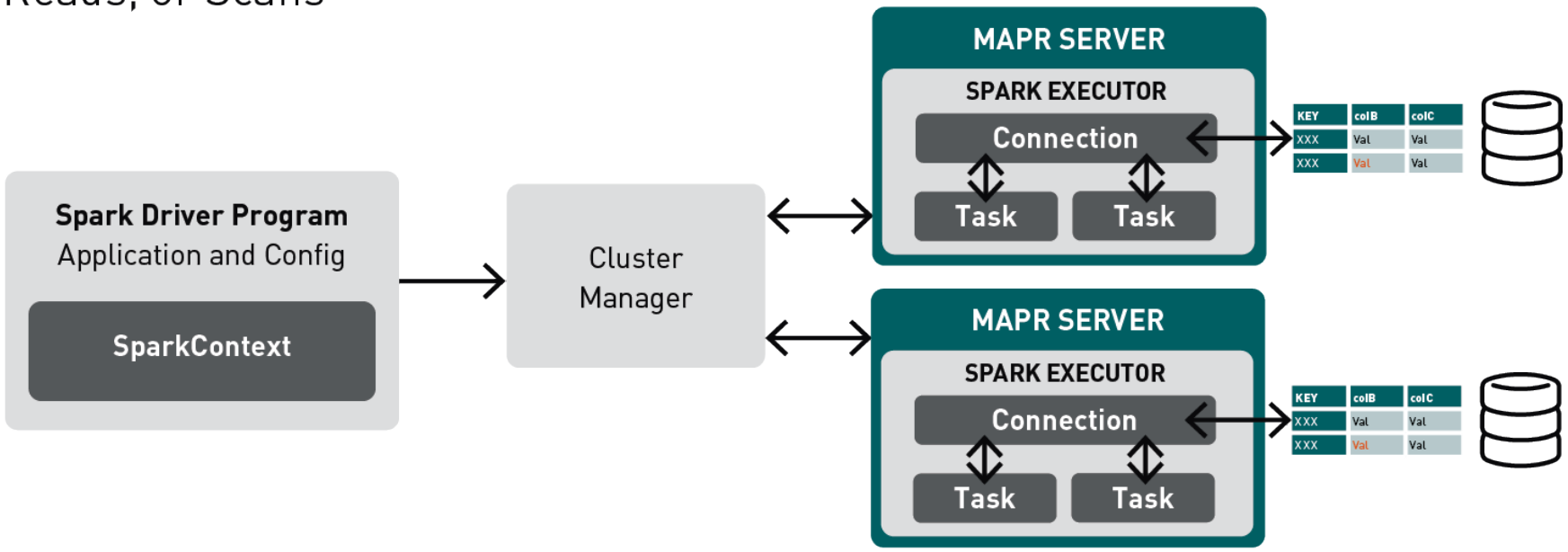
Designed for Partitioning and Scaling



Data is automatically partitioned and sorted by id row key

Spark MapR-DB Connector

Connection in Every Spark Executor
Allowing for Distributed Parallel Writes, Reads, or Scans



Flight Dataset

Field	Description	Example Value
id	Unique identifier: composed of carrier code, date, origin, destination, flight number	AA_2017-02-22_SFO_ORD_150
dofW (Integer)	Day of week (1=Monday 7=Sunday)	1
carrier (String)	Carrier code	AA
origin(String)	Origin Airport Code	JFK
dest(String	Destination airport code	LAX
crsdephour(Integer)	Scheduled departure hour	9
crsdeptime(Double)	Scheduled departure time	900.0
depdelay (Double)	Departure delay in minutes	40.0
crsarrtime (Double)	Scheduled arrival time	1230.0
arrdelay (Double)	Arrival delay minutes	40.0
crselapsedtime (Double)	Elapsed time	390.0
dist (Double)	Distance	2475.0


```
{
   “id": ”ATL_LGA_2017-01-01_AA_1678",
   "dofW": 7,
   "carrier": "AA",
   ”src": "ATL",
   ”dst": "LGA",
   "crsdephour": 17,
   "crsdeptime": 1700,
   "depdelay": 0.0,
   "crsarrtime": 1912,
   "arrdelay": 0.0,
   "crselapsedtime": 132.0,
   "dist": 762.0
}
```

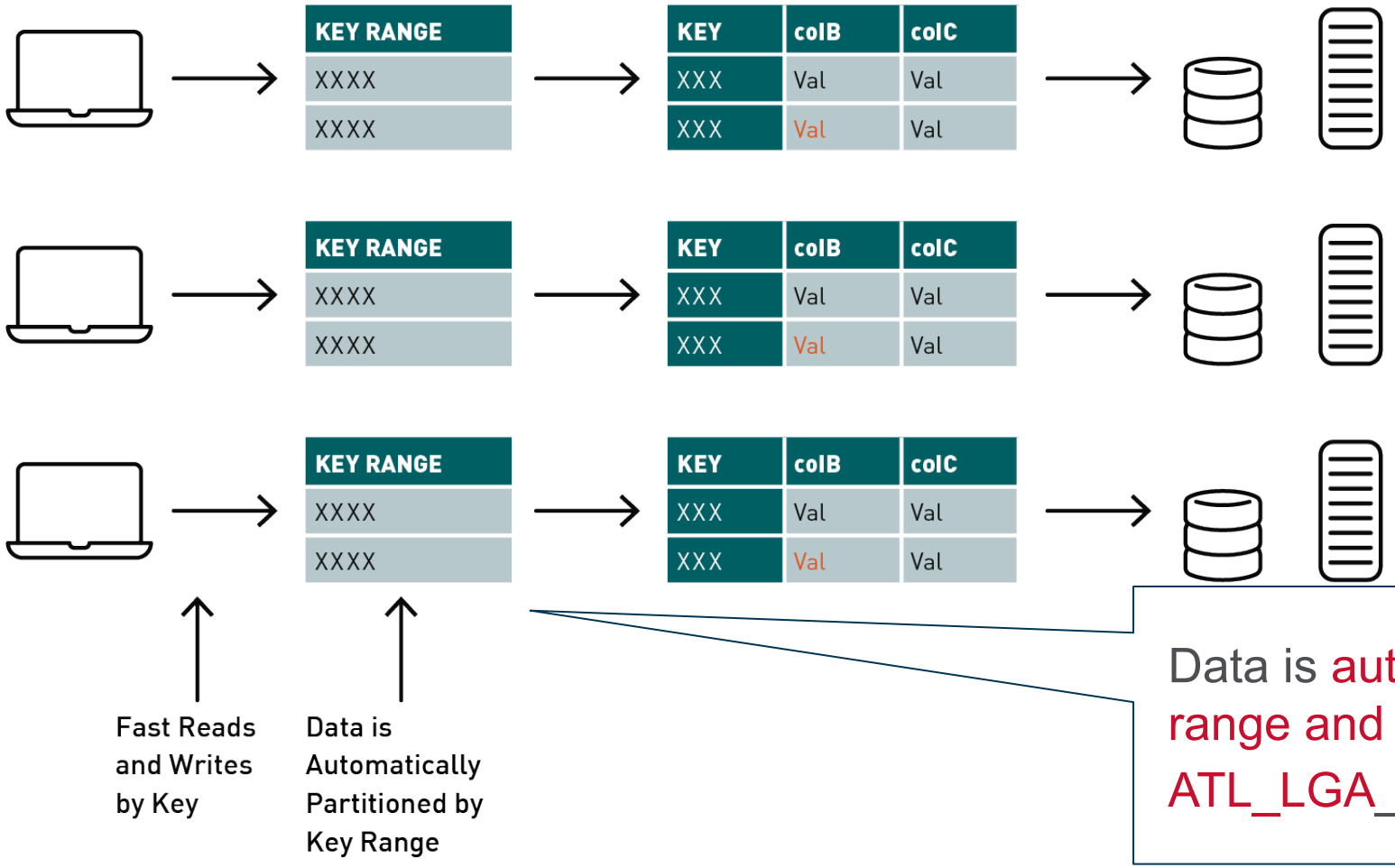

Table is automatically partitioned and sorted by id row key

MapR-DB JSON Document Store

```
{
   “id": ”ATL_LGA_2017-01-01_AA_1678",
   "dofW": 7,
   "carrier": "AA",
   ”src": "ATL",
   ”dst": "LGA",
   "crsdephour": 17,
   "crsdeptime": 1700,
   "depdelay": 0.0,
   "crsarrtime": 1912,
   "arrdelay": 0.0,
   "crselapsedtime": 132.0,
   "dist": 762.0
}
```

Data is automatically partitioned and sorted by id row key

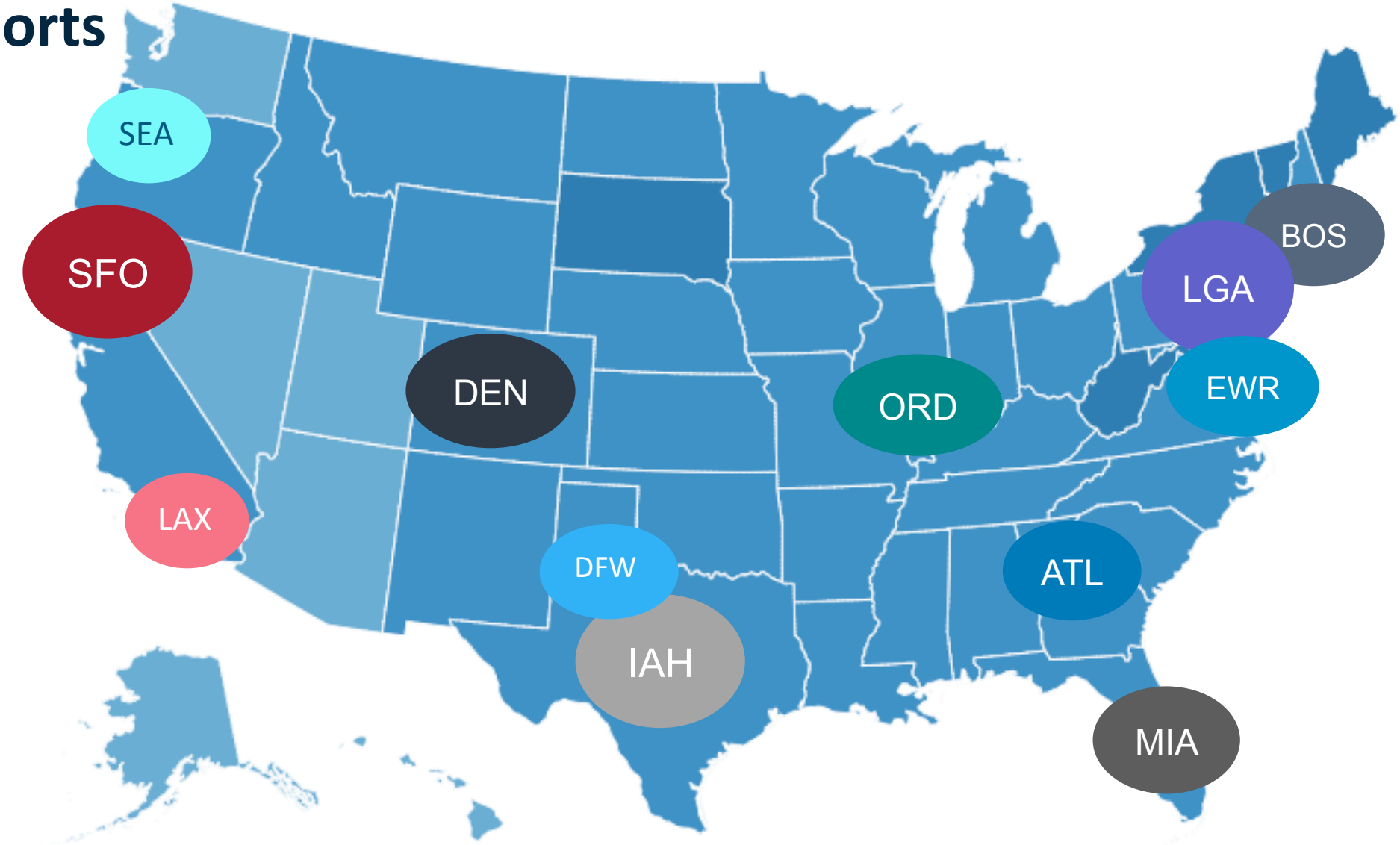
Row key = Table is Partitioned by src,dst vertexes



Data is automatically partitioned by key range and sorted = src_dst

ATL_LGA_2017-01-01_AA_1678

Airports



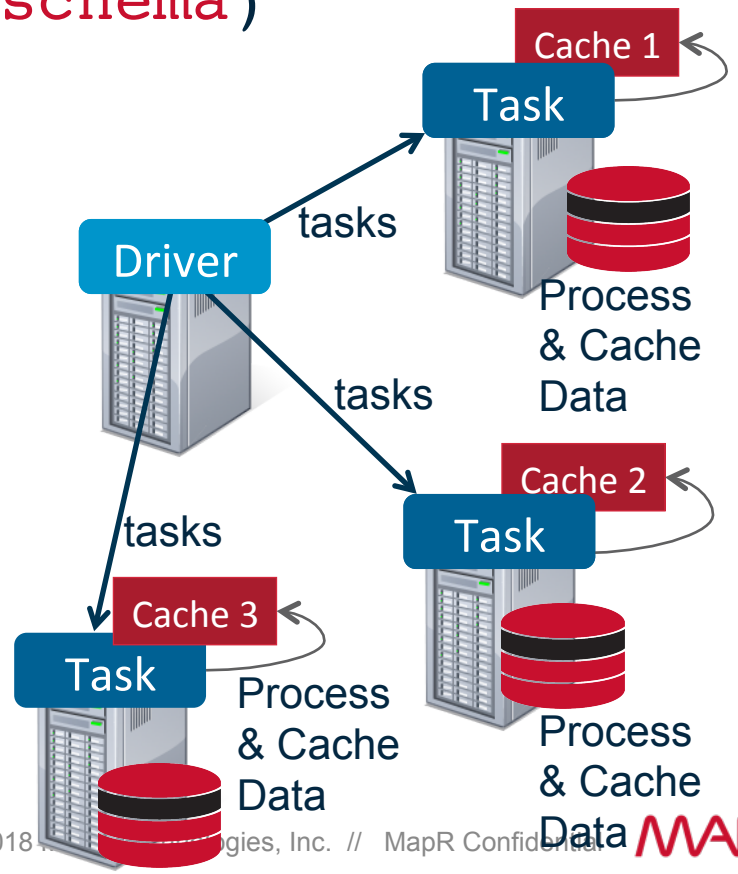
Load the data into a Dataset: Define the Schema

```
case class Flight(id: String,fldate: String,month:Integer, dofW: Integer,
carrier: String, src: String,dst: String, crsdephour: Integer, crsdeptime: Integer,
depdelay: Double, crsarrtime: Integer, arrdelay: Double, crselapsedtime: Double, dist: Double)

val schema = StructType(Array(
  StructField("id", StringType, true),
  StructField("fldate", StringType, true),
  StructField("month", IntegerType, true),
  StructField("dofW", IntegerType, true),
  StructField("carrier", StringType, true),
  StructField("src", StringType, true),
  StructField("dst", StringType, true),
  StructField("crsdephour", IntegerType, true),
  StructField("crsdeptime", IntegerType, true),
  StructField("depdelay", DoubleType, true),
  StructField("crsarrtime", IntegerType, true),
  StructField("arrdelay", DoubleType, true),
  StructField("crselapsedtime", DoubleType, true),
  StructField("dist", DoubleType, true)
))
```

Read Dataset from MapR-DB

```
var tableName = "/user/mapr/flighttable"
val df = spark.sparkSession
     .loadFromMapRDB[Flight](tableName, schema)
```



Show the first rows of the DataFrame

```
df.show(5)
```

columns

```
+--------------------+----------+-----+----+-------+---+---+----------+----------+--------+----------+--------+--------------+-----+
|                  id|    fldate|month|dofW|carrier|src|dst|crsdephour|crsdeptime|depdelay|crsarrtime|arrdelay|crselapsedtime| dist|
+--------------------+----------+-----+----+-------+---+---+----------+----------+--------+----------+--------+--------------+-----+
|ATL_BOS_2018-01-0...|2018-01-01|    1|   1|     DL|ATL|BOS|         9|       850|     0.0|      1116|     0.0|         146.0|946.0|
|ATL_BOS_2018-01-0...|2018-01-01|    1|   1|     DL|ATL|BOS|        11|      1122|     8.0|      1349|     0.0|         147.0|946.0|
|ATL_BOS_2018-01-0...|2018-01-01|    1|   1|     DL|ATL|BOS|        14|      1356|     9.0|      1623|     0.0|         147.0|946.0|
|ATL_BOS_2018-01-0...|2018-01-01|    1|   1|     DL|ATL|BOS|        16|      1620|     0.0|      1851|     3.0|         151.0|946.0|
|ATL_BOS_2018-01-0...|2018-01-01|    1|   1|     DL|ATL|BOS|        19|      1940|     6.0|      2210|     0.0|         150.0|946.0|
+--------------------+----------+-----+----+-------+---+---+----------+----------+--------+----------+--------+--------------+-----+
```

row

Data is automatically partitioned and sorted by row key = src dst

ATL_BOS_2018-01-01_AA_1678

Originating airports with highest number of Departure Delays

```
df.filter($"depdelay" > 40).groupBy(”src”)
      .count().orderBy(desc(“count”)).show(5)
```

```
+---+-----+
|src|count|
+---+-----+
|ORD| 4033|
|ATL| 3106|
|DFW| 2782|
|EWR| 2328|
|DEN| 2304|
+---+-----+
```

MapR-DB Projection and Filter push down

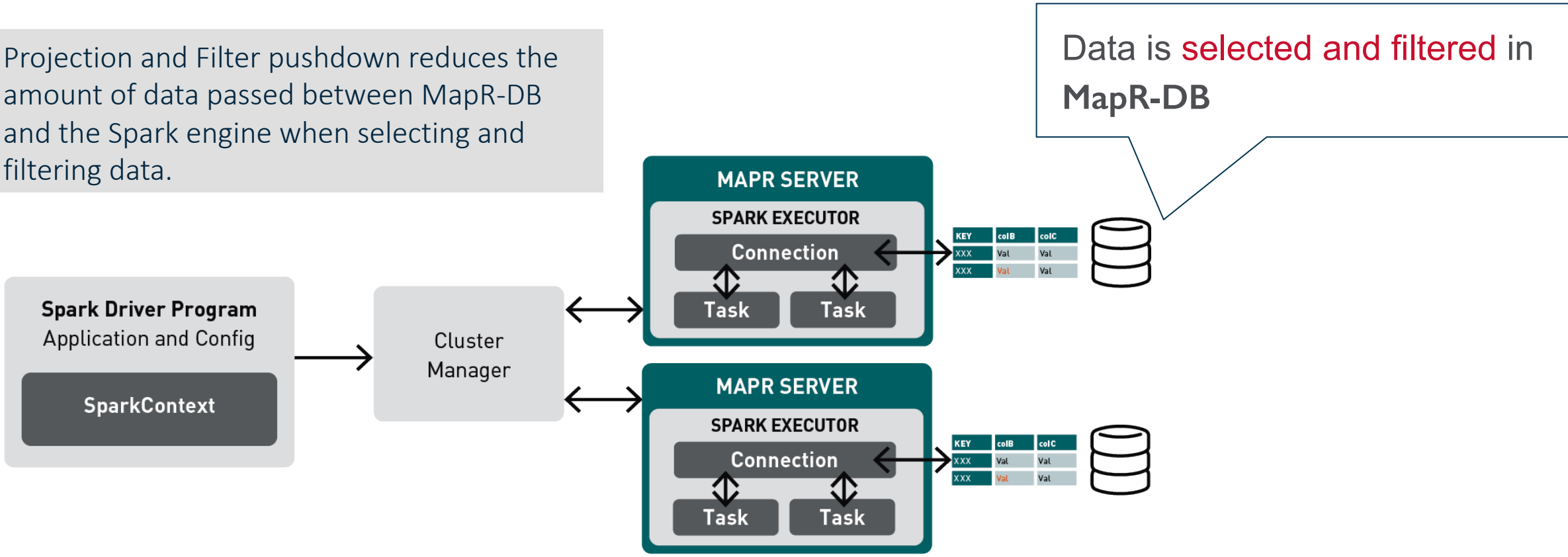
```
df.filter($"depdelay" > 40).groupBy("src")
       .count.orderBy(desc("count" )).explain

== Physical Plan ==
*(3) Sort [count#549L DESC NULLS LAST], true, 0
+- Exchange rangepartitioning(count#549L DESC NULLS LAS    200)
   +- *(2) HashAggregate(keys=[src#5], functions=[c    t(1)])
      +- Exchange hashpartitioning(src#5, 200)
         +- *(1) HashAggregate(keys=[src#5], functions=[partial_count(1)])
            +- *(1) Project [src#5]
               +- *(1) Filter (isnotnull(depdelay#9) && (depdelay#9 > 40.0))
                  +- *(1) Scan MapRDBRelation(/user/mapr/flighttable
[src#5,depdelay#9] PushedFilters: [IsNotNull(depdelay), GreaterThan(depdelay,
40.0)], ReadSchema: struct<src:string,depdelay:double>
```

Project and Filter pushed into MapR-DB

Spark MapR-DB Projection Filter push down

Projection and Filter pushdown reduces the amount of data passed between MapR-DB and the Spark engine when selecting and filtering data.



Register Dataframe as a Temporary View

df.cache

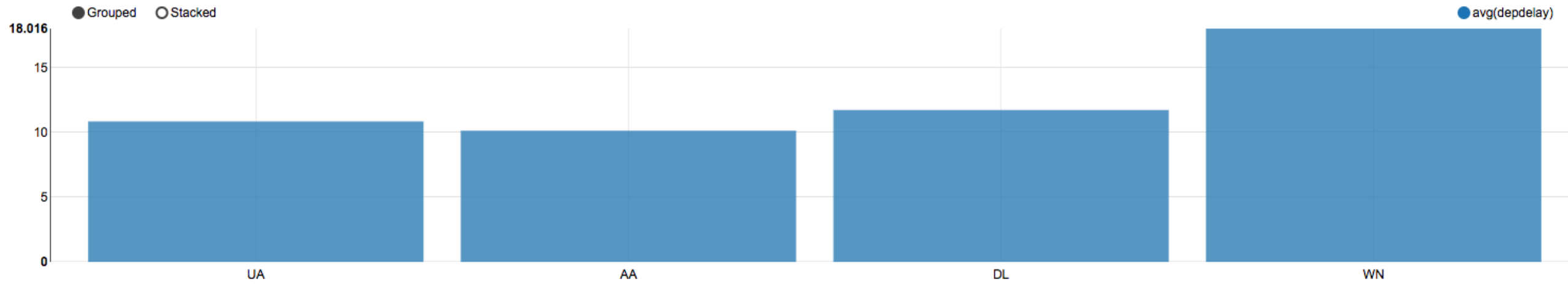
df.count()

df.createOrReplaceTempView("flights")

Long = 282628

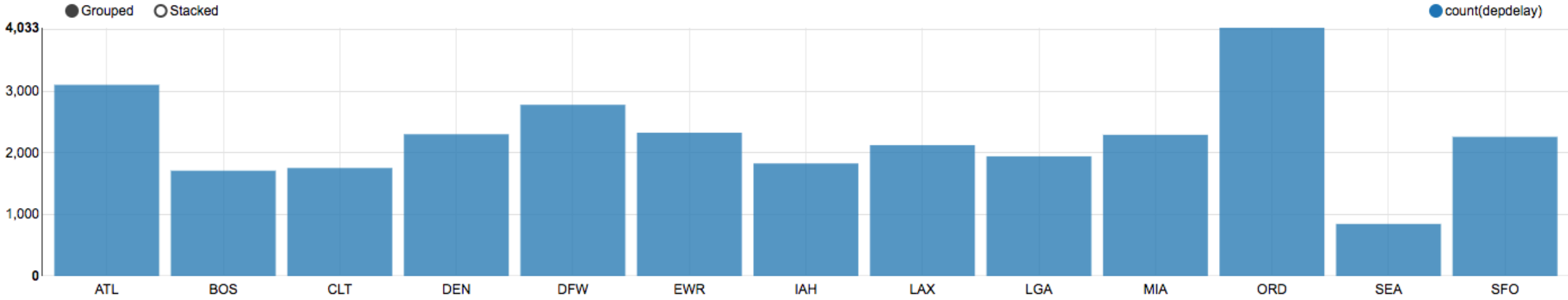
Average Departure Delay by Carrier

```
%sql select carrier, avg(depdelay) from flights
 group by carrier
```



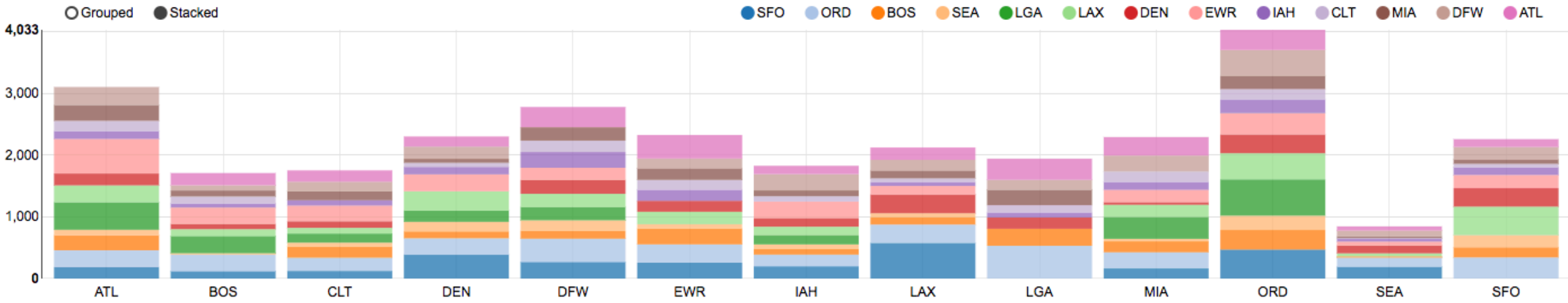
Count of Departure Delays by Origin

```
%sql select src, count(depdelay) from flights
where depdelay > 40 group by src
```



Count of Departure Delays by Origin, Destination

```
%sql select src,dst count(depdelay) from flights
where depdelay > 40 group by src,dst
```



Explore MapR-DB Flight Table with GraphFrames

GraphFrame and DataFrame

To answer questions such as:

How many flight routes are there?

What are the longest distance routes?

Which airport has the most incoming flights?

What are the top 10 flight routes?

Read Vertices DataFrame from a JSON File

```
val airports = spark.read.json(file)
airports.show
```

```
+-------------+-------+-----+---+
|         City|Country|State| id|
+-------------+-------+-----+---+
|      Chicago|    USA|   IL|ORD|
|     New York|    USA|   NY|JFK|
|     New York|    USA|   NY|LGA|
|       Boston|    USA|   MA|BOS|
|      Houston|    USA|   TX|IAH|
|       Newark|    USA|   NJ|EWR|
|       Denver|    USA|   CO|DEN|
|        Miami|    USA|   FL|MIA|
|San Francisco|    USA|   CA|SFO|
|      Atlanta|    USA|   GA|ATL|
|       Dallas|    USA|   TX|DFW|
|    Charlotte|    USA|   NC|CLT|
|  Los Angeles|    USA|   CA|LAX|
|      Seattle|    USA|   WA|SEA|
+-------------+-------+-----+---+
```

Create the GraphFrame

```
val graph = GraphFrame(airports, df)

// graph.edges is a DataFrame
graph.edges.show
```

```
+--------------------+----------+-----+---+-------+---+---+----------+----------+-------+----------+-------+--------------+-----+
|                  id|    fldate|month|dofW|carrier|src|dst|crsdephour|crsdeptime|depdelay|crsarrtime|arrdelay|crselapsedtime| dist|
+--------------------+----------+-----+---+-------+---+---+----------+----------+-------+----------+-------+--------------+-----+
|ATL_BOS_2018-01-0...|2018-01-01|    1|  1|     DL|ATL|BOS|         9|       850|    0.0|      1116|    0.0|         146.0|946.0|
|ATL_BOS_2018-01-0...|2018-01-01|    1|  1|     DL|ATL|BOS|        11|      1122|    8.0|      1349|    0.0|         147.0|946.0|
|ATL_BOS_2018-01-0...|2018-01-01|    1|  1|     DL|ATL|BOS|        14|      1356|    9.0|      1623|    0.0|         147.0|946.0|
|ATL_BOS_2018-01-0...|2018-01-01|    1|  1|     DL|ATL|BOS|        16|      1620|    0.0|      1851|    3.0|         151.0|946.0|
|ATL_BOS_2018-01-0...|2018-01-01|    1|  1|     DL|ATL|BOS|        19|      1940|    6.0|      2210|    0.0|         150.0|946.0|
+--------------------+----------+-----+---+-------+---+---+----------+----------+-------+----------+-------+--------------+-----+
```

GraphFrame API

Category	Methods
Graph Topology	vertices, edges, triplets
Graph Structure	inDegrees, outDegrees, degrees
Graph Algorithms	pageRank, bfs, aggregatedMessages, shortestPaths, connectedComponents, triangleCount
Graph Queries	Motif find

DataFrame Queries

Operation	Description
`select(col)`	Selects set of columns
`sort(sortcol)`	Returns new DataFrame sorted by specified column
`filter(expr); where(condition)`	Filter based on the SQL expression or condition
`groupBy(cols: Columns)`	Groups DataFrame using specified columns
`join (DataFrame, joinExpr)`	Joins with another DataFrame using given join expression
`count`	Count of rows
`avg, count, min, max, sum (col)`	Average , count , min , max on values in a group

Graph Vertices and Edges are DataFrames

```
graph.vertices.filter("State='TX'").show
```

```
+-------+-------+-----+---+
|   City|Country|State| id|
+-------+-------+-----+---+
|Houston|    USA|   TX|IAH|
| Dallas|    USA|   TX|DFW|
+-------+-------+-----+---+
```

GraphFrame DataFrame Queries

```
// How many airports?
graph.vertices.count

result:  = 13

// How many flights?
graph.edges.count

result: = 282628
```

What are the 4 Longest Distance Flights?

```
// Show the longest distance flight routes
graph.edges.groupBy("src", "dst")
       .max("dist").sort(desc("max(dist)")).show(4)
```

```
+---+---+---------+
|src|dst|max(dist)|
+---+---+---------+
|MIA|SEA|   2724.0|
|SEA|MIA|   2724.0|
|BOS|SFO|   2704.0|
|SFO|BOS|   2704.0|
+---+---+---------+
```

What is the average delay for delayed flights from Atlanta?

```
graph.edges.filter("src = 'ATL' and depdelay > 1")
.groupBy("src", "dst").avg("depdelay").sort(desc("avg(depdelay)")).show
```

```
+---+---+------------------+
|src|dst|     avg(depdelay)|
+---+---+------------------+
|ATL|EWR|  58.1085801063022|
|ATL|ORD| 46.42393736017897|
|ATL|DFW|39.454460966542754|
|ATL|LGA| 39.25498489425982|
|ATL|CLT| 37.56777108433735|
|ATL|SFO| 36.83008356545961|
+---+---+------------------+
```

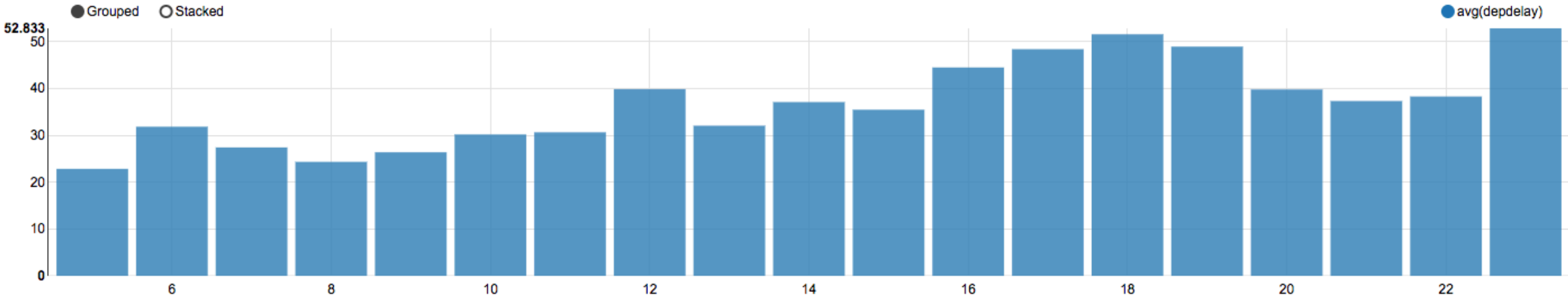
MapR-DB Projection and Filter push down

```
graph.edges.filter("src = 'ATL' and depdelay > 1")
.groupBy("src", "dst").avg("depdelay").sort(desc("avg(depdelay)")).explain

== Physical Plan ==
*(3) Sort [avg(depdelay)#273 DESC NULLS LAST], true, 0
+- Exchange rangepartitioning(avg(depdelay)#273 DESC NULLS LAST, 200)
   +- *(2) HashAggregate(keys=[src#5, dst#6], functions=[avg(depdelay#9)])
      +- Exchange hashpartitioning(src#5, dst#6, 200)
         +- *(1) HashAggregate(keys=[src#5, dst#6], functions=[partial_avg(depdelay#9)])
            +- *(1) Filter (((isnotnull(src#5) && isnotnull(depdelay#9)) &&
                     (src#5 = ATL)) && (depdelay#9 > 1.0))
               +- *(1) Scan MapRDBRelation(/user/mapr/flighttable
[src#5,dst#6,depdelay#9] PushedFilters: [IsNotNull(src), IsNotNull(depdelay),
EqualTo(src,ATL), GreaterThan(depdelay,1.0)], ReadSchema:
struct<src:string,dst:string,depdelay:double>
```

What is the Average Delay for delayed flights from Atlanta by Hour?

```
z.show( graph.edges
        .filter("src = 'ATL' and depdelay > 1”)
        .groupBy("crsdephour")
        .avg("depdelay”) )
```



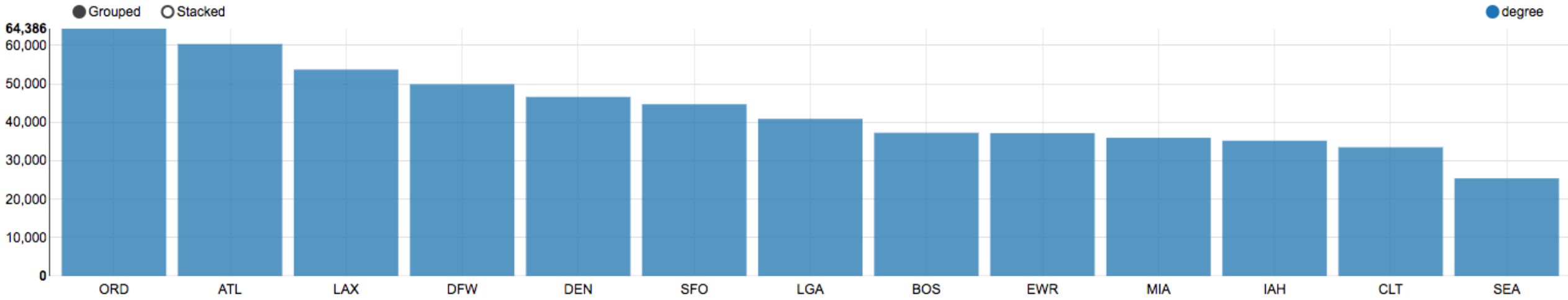
GraphFrame API

Category	Methods
Graph Topology	vertices, edges, triplets
Graph Structure	inDegrees, outDegrees, degrees
Graph Algorithms	pageRank, bfs, aggregatedMessages, shortestPaths, connectedComponents
Graph Queries	Motif find

Which Airports have the most incoming and outgoing flights?

WHAT ARE THE HIGHEST DEGREE VERTEXES?

```
z.show( graph.degrees.orderBy(desc("degree")) )
```



GraphFrame API

Category	Methods
Graph Topology	vertices, edges, triplets
Graph Structure	inDegrees, outDegrees, degrees
Graph Algorithms	pageRank, bfs, aggregatedMessages, shortestPaths, connectedComponents
Graph Queries	Motif find

Use Pagerank to find most important airports

```
val ranks = graph.pageRank.resetProbability(0.15).maxIter(10).run()
ranks.vertices.orderBy($"pagerank".desc).show(5)
```

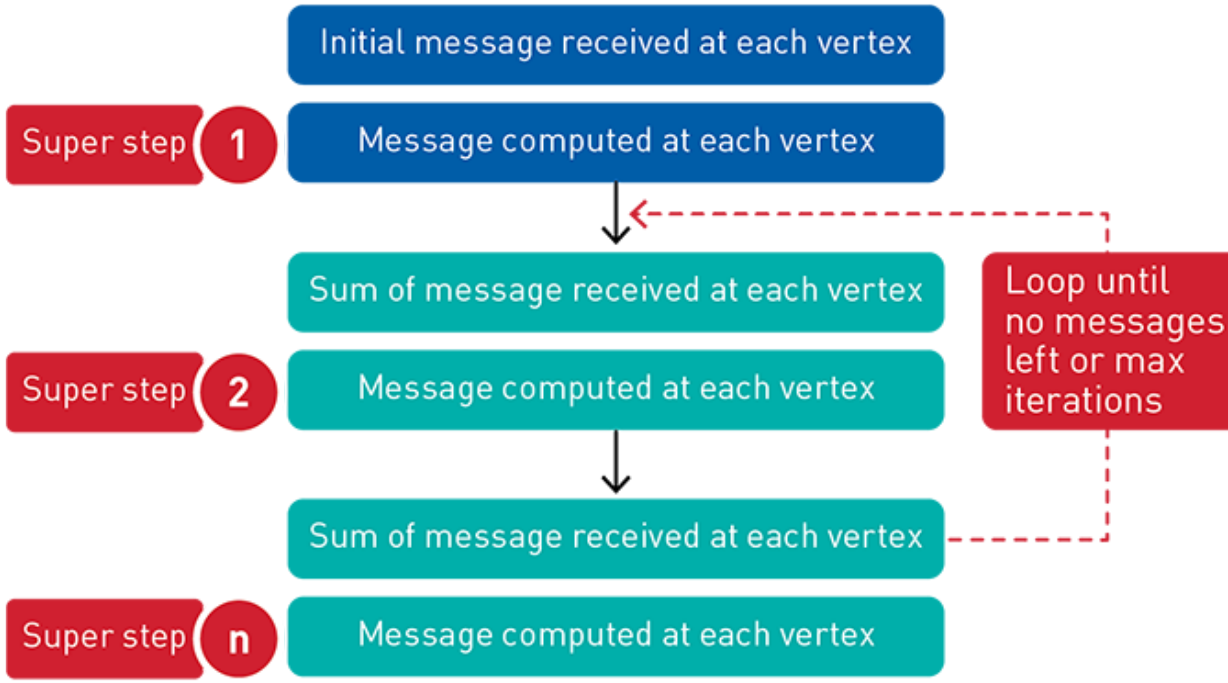
```
+-------------+-------+-----+---+------------------+
|         City|Country|State| id|          pagerank|
+-------------+-------+-----+---+------------------+
|      Chicago|    USA|   IL|ORD| 1.512992983935868|
|      Atlanta|    USA|   GA|ATL| 1.425548154421664|
|  Los Angeles|    USA|   CA|LAX| 1.278700100175873|
|       Dallas|    USA|   TX|DFW| 1.199925217168806|
|       Denver|    USA|   CO|DEN| 1.127519432436076|
+-------------+-------+-----+---+------------------+
```

GraphFrame API

Category	Methods
Graph Topology	vertices, edges, triplets
Graph Structure	inDegrees, outDegrees, degrees
Graph Algorithms	pageRank, bfs, aggregatedMessages, shortestPaths, connectedComponents
Graph Queries	Motif find

Aggregate Messages to calculate avg delay

```
val AM = AggregateMessages
val msgToSrc = AM.edge("depdelay")
val agg = { graph
   .aggregateMessages
   .sendToSrc(msgToSrc)
   .agg(avg(AM.msg).as("avgdelay"))}
agg.show()
+---+------------------+
| id|          avgdelay|
+---+------------------+
|EWR|17.818079459546404|
|MIA|17.768691978431264|
|ORD|  16.5199551010227|
+---+------------------+
```



What are the most Frequent Flight Routes?

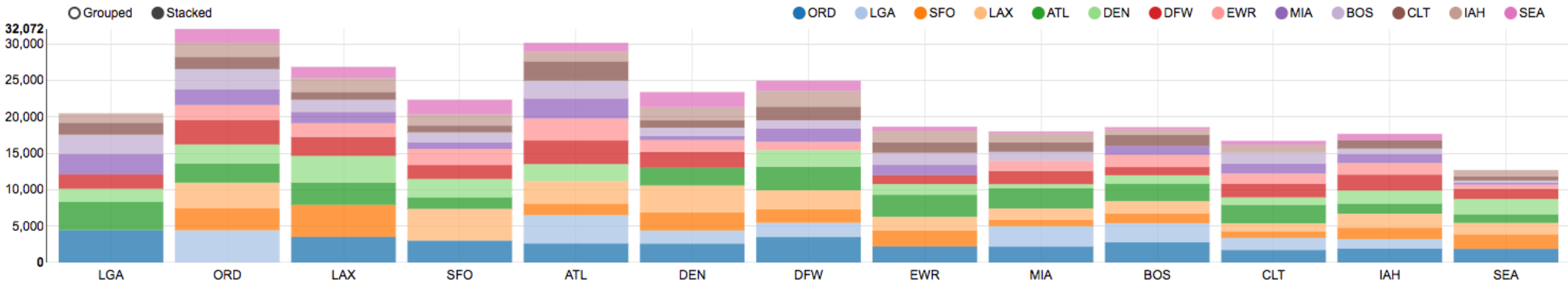
```
// count of flight routes
val flightroutecount=graph.edges
       .groupBy("src", "dst”)
       .count().orderBy(desc("count"))
flightroutecount.show(5)
+---+---+-----+
|src|dst|count|
+---+---+-----+
|LGA|ORD| 4442|
|ORD|LGA| 4426|
|LAX|SFO| 4406|
|SFO|LAX| 4354|
|ATL|LGA| 3884|
+---+---+-----+
```

```
// how many routes?
flightroutecount.count
Long = 148
```

What are the most Frequent Flight Routes?

(HIGHEST COUNT OF FLIGHTS)

```
z.show (flightroutecount )
```

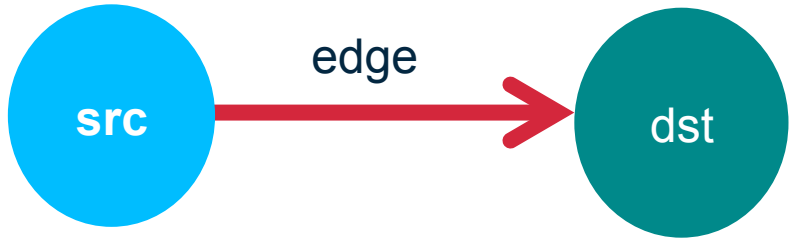


GraphFrame API

Category	Methods
Graph Topology	vertices, edges, triplets
Graph Structure	inDegrees, outDegrees, degrees
Graph Algorithms	pageRank, bfs, aggregatedMessages, shortestPaths, connectedComponents
Graph Queries	Motif find

Triplets = 2 Vertices and 1 Connecting Edge DataFrames

```
graph.triplets
 .show(3)
```

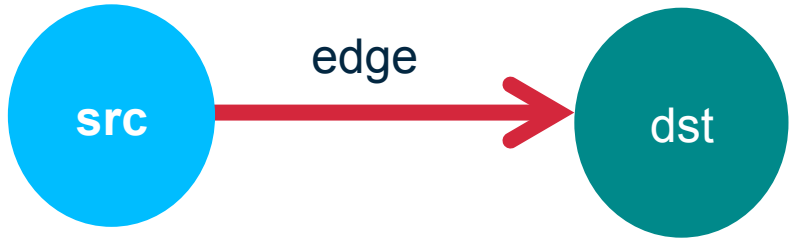



```
+--------------------+--------------------+--------------------+
|                 src|                edge|                 dst|
+--------------------+--------------------+--------------------+
|[Atlanta, USA, GA...|[ATL_BOS_2018-01-...|[Boston, USA, MA,...|
|[Atlanta, USA, GA...|[ATL_BOS_2018-01-...|[Boston, USA, MA,...|
|[Atlanta, USA, GA...|[ATL_BOS_2018-01-...|[Boston, USA, MA,...|
+--------------------+--------------------+--------------------+
```

Triplets = 2 Vertices and 1 Connecting Edge DataFrames

```
graph.triplets
  .filter("src.State='TX'")
  .show
```

DataFrames
Refine the result




```
+--------------------+--------------------------------------------------------------------------------------------------------------+--------------------+
|src                 |edge                                                                                                          |dst                 |
+--------------------+--------------------------------------------------------------------------------------------------------------+--------------------+
|[Dallas, USA, TX, DFW]|[DFW_ATL_2018-01-01_AA_1473, 2018-01-01, 1, 1, AA, DFW, ATL, 10, 1026, 26.0, 1327, 21.0, 121.0, 731.0]|[Atlanta, USA, GA, ATL]|
|[Dallas, USA, TX, DFW]|[DFW_ATL_2018-01-01_AA_1675, 2018-01-01, 1, 1, AA, DFW, ATL, 13, 1255, 32.0, 1557, 16.0, 122.0, 731.0]|[Atlanta, USA, GA, ATL]|
|[Dallas, USA, TX, DFW]|[DFW_ATL_2018-01-01_AA_2408, 2018-01-01, 1, 1, AA, DFW, ATL, 18, 1835, 4.0, 2141, 0.0, 126.0, 731.0]  |[Atlanta, USA, GA, ATL]|
|[Dallas, USA, TX, DFW]|[DFW_ATL_2018-01-01_AA_2479, 2018-01-01, 1, 1, AA, DFW, ATL, 9, 855, 0.0, 1200, 0.0, 125.0, 731.0]    |[Atlanta, USA, GA, ATL]|
|[Dallas, USA, TX, DFW]|[DFW_ATL_2018-01-01_AA_2497, 2018-01-01, 1, 1, AA, DFW, ATL, 21, 2055, 0.0, 2359, 0.0, 124.0, 731.0]  |[Atlanta, USA, GA, ATL]|
+--------------------+--------------------------------------------------------------------------------------------------------------+--------------------+
```

Motif find

Search for a pattern

```
graph.find("(src)-[edge]->(dst)")
.show(3)
```


```
+--------------------+--------------------+--------------------+
|                 src|                edge|                 dst|
+--------------------+--------------------+--------------------+
|[Atlanta, USA, GA...|[ATL_BOS_2018-01-...|[Boston, USA, MA,...|
|[Atlanta, USA, GA...|[ATL_BOS_2018-01-...|[Boston, USA, MA,...|
|[Atlanta, USA, GA...|[ATL_BOS_2018-01-...|[Boston, USA, MA,...|
+--------------------+--------------------+--------------------+
```

Next: use flightroutecount with Motif find

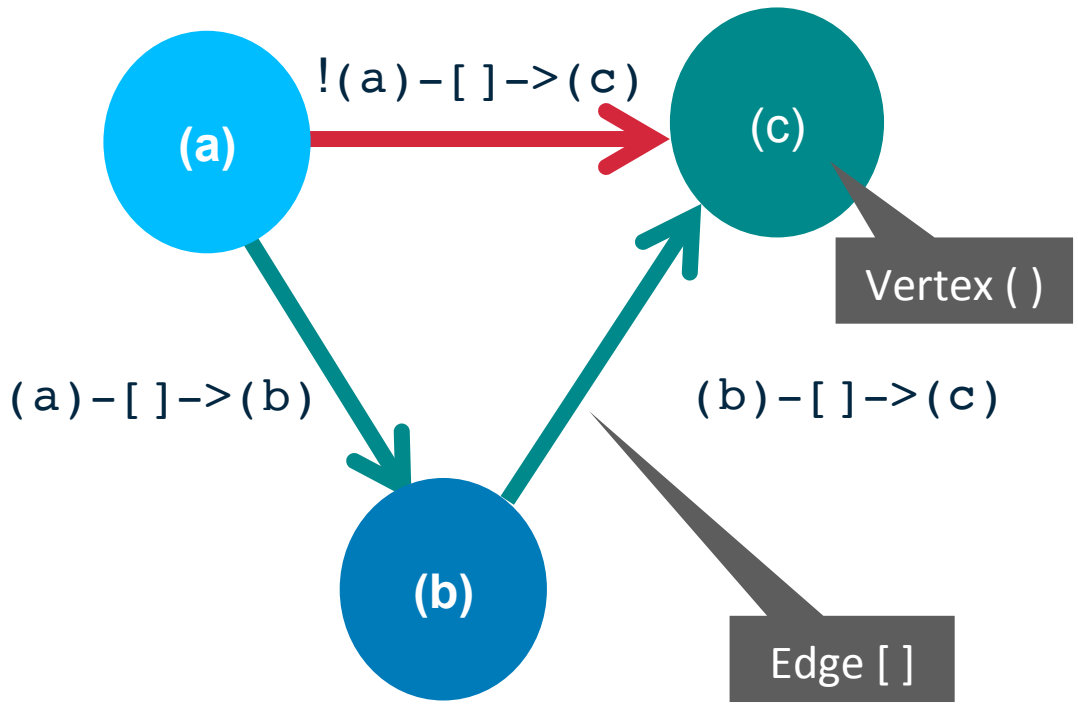
```
// count of flight routes
val flightroutecount=graph.edges
       .groupBy("src", "dst”)
       .count().orderBy(desc("count"))
flightroutecount.show(5)
+---+---+-----+
|src|dst|count|
+---+---+-----+
|LGA|ORD| 4442|
|ORD|LGA| 4426|
|LAX|SFO| 4406|
|SFO|LAX| 4354|
|ATL|LGA| 3884|
+---+---+-----+
```

Motif Find Flights with No Direct Connection

```
val subGraph = GraphFrame(graph.vertices, flightroutecount)
val res = subGraph
 .find("(a)-[]->(b); (b)-[]->(c); !(a)-[]->(c)")
 .filter("c.id !=a.id”)
```

Search for a pattern

DataFrames
Refine the result:
Remove duplicates



Motif Find Flights with No Direct Connection

```
val subGraph = GraphFrame(graph.vertices, flightroutecount)
val res = subGraph
  .find("(a)-[]->(b); (b)-[]->(c); !(a)-[]->(c)")
  .filter("c.id !=a.id”)
```

a	c
[Newark,USA,NJ,EWR]	[New York,USA,NY,LGA]
[New York,USA,NY,LGA]	[Newark,USA,NJ,EWR]
[Los Angeles,USA,CA,LAX]	[New York,USA,NY,LGA]
[New York,USA,NY,LGA]	[Los Angeles,USA,CA,LAX]
[New York,USA,NY,LGA]	[San Francisco,USA,CA,SFO]
[San Francisco,USA,CA,SFO]	[New York,USA,NY,LGA]
[New York,USA,NY,LGA]	[Seattle,USA,WA,SEA]
[Seattle,USA,WA,SEA]	[New York,USA,NY,LGA]

GraphFrame API

Category	Methods
Graph Topology	vertices, edges, triplets
Graph Structure	inDegrees, outDegrees, degrees
Graph Algorithms	pageRank, bfs, aggregatedMessages, shortestPaths, connectedComponents
Graph Queries	Motif find

Compute shortest paths from each Airport to LGA

```
val results = graph.shortestPaths.landmarks(Seq("LGA")).run()
```

```
+---+----------+
| id| distances|
+---+----------+
|IAH|[LGA -> 1]|
|CLT|[LGA -> 1]|
|LAX|[LGA -> 2]|
|DEN|[LGA -> 1]|
|DFW|[LGA -> 1]|
|SFO|[LGA -> 2]|
|LGA|[LGA -> 0]|
|ORD|[LGA -> 1]|
|MIA|[LGA -> 1]|
|SEA|[LGA -> 2]|
|ATL|[LGA -> 1]|
|BOS|[LGA -> 1]|
|EWR|[LGA -> 2]|
+---+----------+
```

GraphFrame API

Category	Methods
Graph Topology	vertices, edges, triplets
Graph Structure	inDegrees, outDegrees, degrees
Graph Algorithms	pageRank, bfs, aggregatedMessages, shortestPaths, connectedComponents
Graph Queries	Motif find

Breadth First Search for Direct Flights between LAX and LGA

```
graph.bfs.fromExpr("id = 'LAX'")
      .toExpr("id = 'LGA'").maxPathLength(1).run().show()
```

```
+----+-------+-----+---+
|City|Country|State| id|
+----+-------+-----+---+
+----+-------+-----+---+
```

Breadth First Search for Flights between LAX and LGA

```
graph.bfs.fromExpr("id = 'LAX'")
        .toExpr("id = 'LGA'").maxPathLength(2).run().show(5)
```

```
+--------------------+--------------------+--------------------+--------------------+--------------------+
|                from|                  e0|                  v1|                  e1|                  to|
+--------------------+--------------------+--------------------+--------------------+--------------------+
|[Los Angeles, USA...|[LAX_IAH_2018-01-...|[Houston, USA, TX...|[IAH_LGA_2018-01-...|[New York, USA, N...|
|[Los Angeles, USA...|[LAX_IAH_2018-01-...|[Houston, USA, TX...|[IAH_LGA_2018-01-...|[New York, USA, N...|
|[Los Angeles, USA...|[LAX_IAH_2018-01-...|[Houston, USA, TX...|[IAH_LGA_2018-01-...|[New York, USA, N...|
|[Los Angeles, USA...|[LAX_IAH_2018-01-...|[Houston, USA, TX...|[IAH_LGA_2018-01-...|[New York, USA, N...|
|[Los Angeles, USA...|[LAX_IAH_2018-01-...|[Houston, USA, TX...|[IAH_LGA_2018-01-...|[New York, USA, N...|
+--------------------+--------------------+--------------------+--------------------+--------------------+
```

Motif Search for Flights between LAX and LGA

```
graph.find("(a)-[ab]->(b); (b)-[bc]->(c)")
      .filter("a.id = 'LAX'")
      .filter("c.id = 'LGA'").show(4)
```

Search for a pattern

DataFrames Refine the result

a	ab	b	bc	c
[Los Angeles,USA,CA,LAX]	[LAX_IAH_2018-01-01_UA_1125,2018-01-01,1,1,UA,LAX,IAH,8,817,0.0,1333,0.0,196.0,1379.0]	[Houston,USA,TX,IAH]	[IAH_LGA_2018-01-01_UA_1254,2018-01-01,1,1,UA,IAH,LGA,12,1230,0.0,1655,0.0,205.0,1416.0]	[New York,USA,NY,LGA]
[Los Angeles,USA,CA,LAX]	[LAX_IAH_2018-01-01_UA_1125,2018-01-01,1,1,UA,LAX,IAH,8,817,0.0,1333,0.0,196.0,1379.0]	[Houston,USA,TX,IAH]	[IAH_LGA_2018-01-01_UA_1284,2018-01-01,1,1,UA,IAH,LGA,18,1805,52.0,2233,22.0,208.0,1416.0]	[New York,USA,NY,LGA]
[Los Angeles,USA,CA,LAX]	[LAX_IAH_2018-01-01_UA_1125,2018-01-01,1,1,UA,LAX,IAH,8,817,0.0,1333,0.0,196.0,1379.0]	[Houston,USA,TX,IAH]	[IAH_LGA_2018-01-01_UA_1410,2018-01-01,1,1,UA,IAH,LGA,14,1441,8.0,1912,6.0,211.0,1416.0]	[New York,USA,NY,LGA]
[Los Angeles,USA,CA,LAX]	[LAX_IAH_2018-01-01_UA_1125,2018-01-	[Houston,USA,TX,IAH]	[IAH_LGA_2018-01-01_UA_1501,2018-01-	[New York,USA,NY,LGA]

Breadth First Search for Flights between LAX and LGA with AA

```
val paths = graph.bfs.fromExpr("id = 'LAX'").toExpr("id = 'LGA'")
   .maxPathLength(3).edgeFilter("carrier = 'AA'").run()
```

BFS

```
paths.filter("e0.crsarrtime<e1.crsdeptime-60 and e0.fldate=e1.fldate")
.select("e0.id","e1.id").show(5)
```

DataFrames
Refine the result

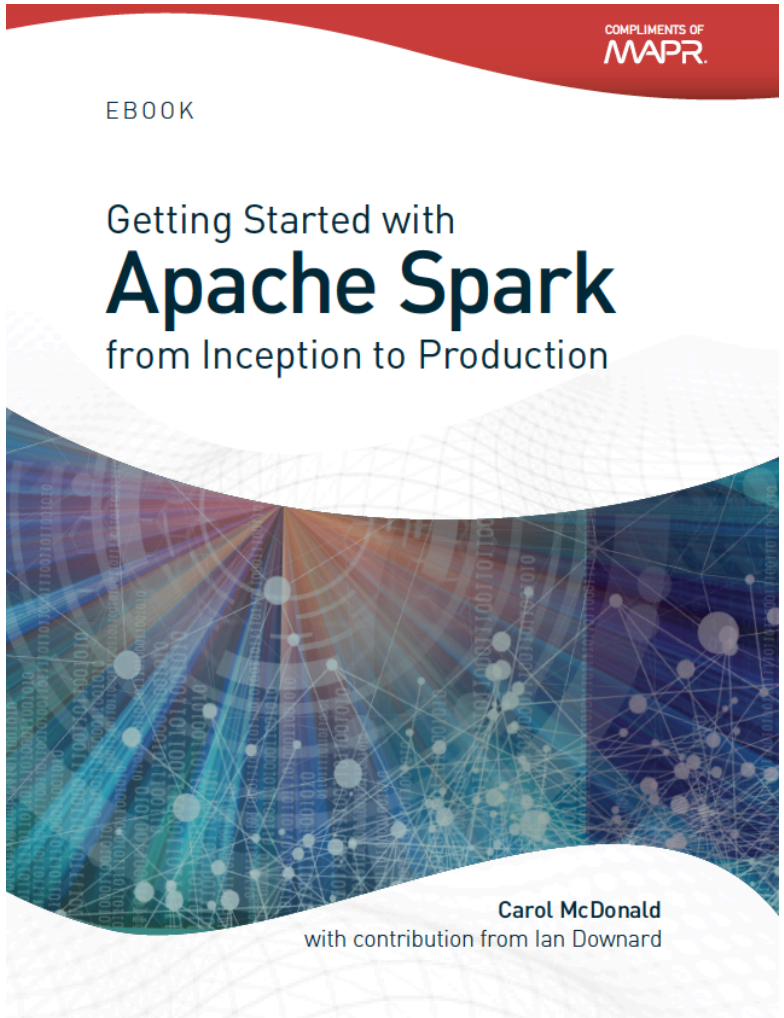
```
+-------------------------+-------------------------+
|id                       |id                       |
+-------------------------+-------------------------+
|LAX_BOS_2018-02-03_AA_1098|BOS_LGA_2018-02-03_AA_2126|
|LAX_BOS_2018-02-03_AA_1379|BOS_LGA_2018-02-03_AA_2126|
|LAX_CLT_2018-02-14_AA_1905|CLT_LGA_2018-02-14_AA_1740|
|LAX_CLT_2018-02-14_AA_1905|CLT_LGA_2018-02-14_AA_1910|
|LAX_CLT_2018-02-14_AA_1905|CLT_LGA_2018-02-14_AA_1954|
+-------------------------+-------------------------+
```

Resources

New Spark Ebook

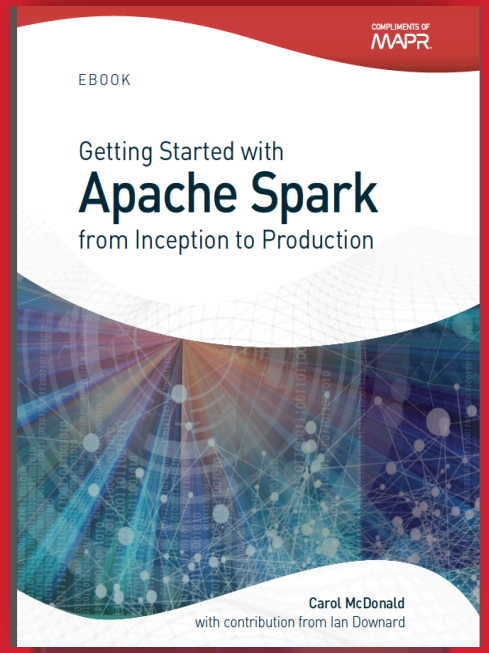
Link to Code for this webinar is in appendix of this book.

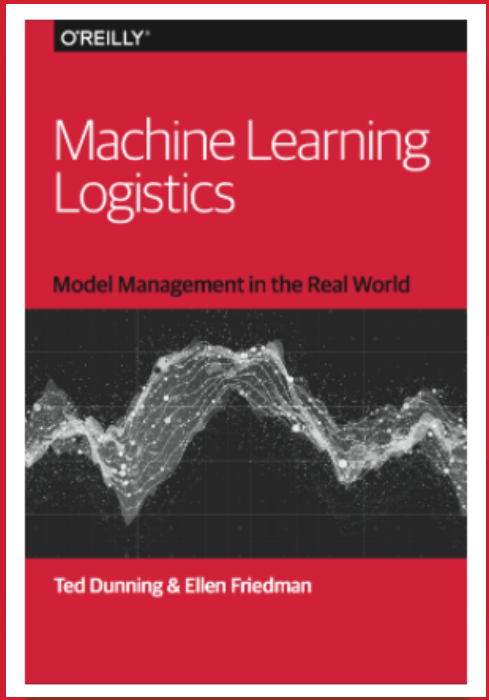
https://mapr.com/ebook/getting-started-with-apache-spark-v2/



MAPR

Complimentary books



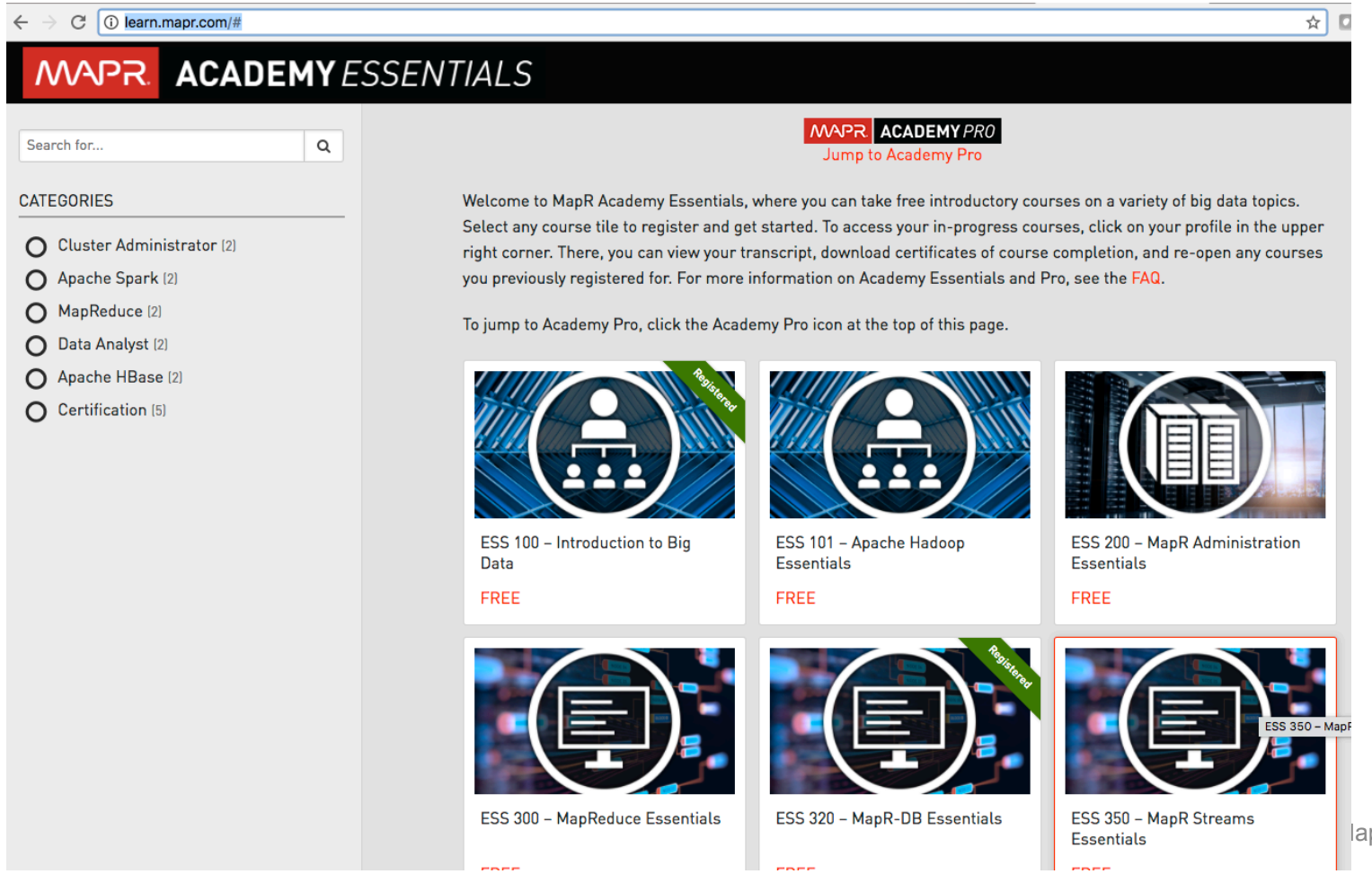


Get your copies

download at mapr.com/ebooks

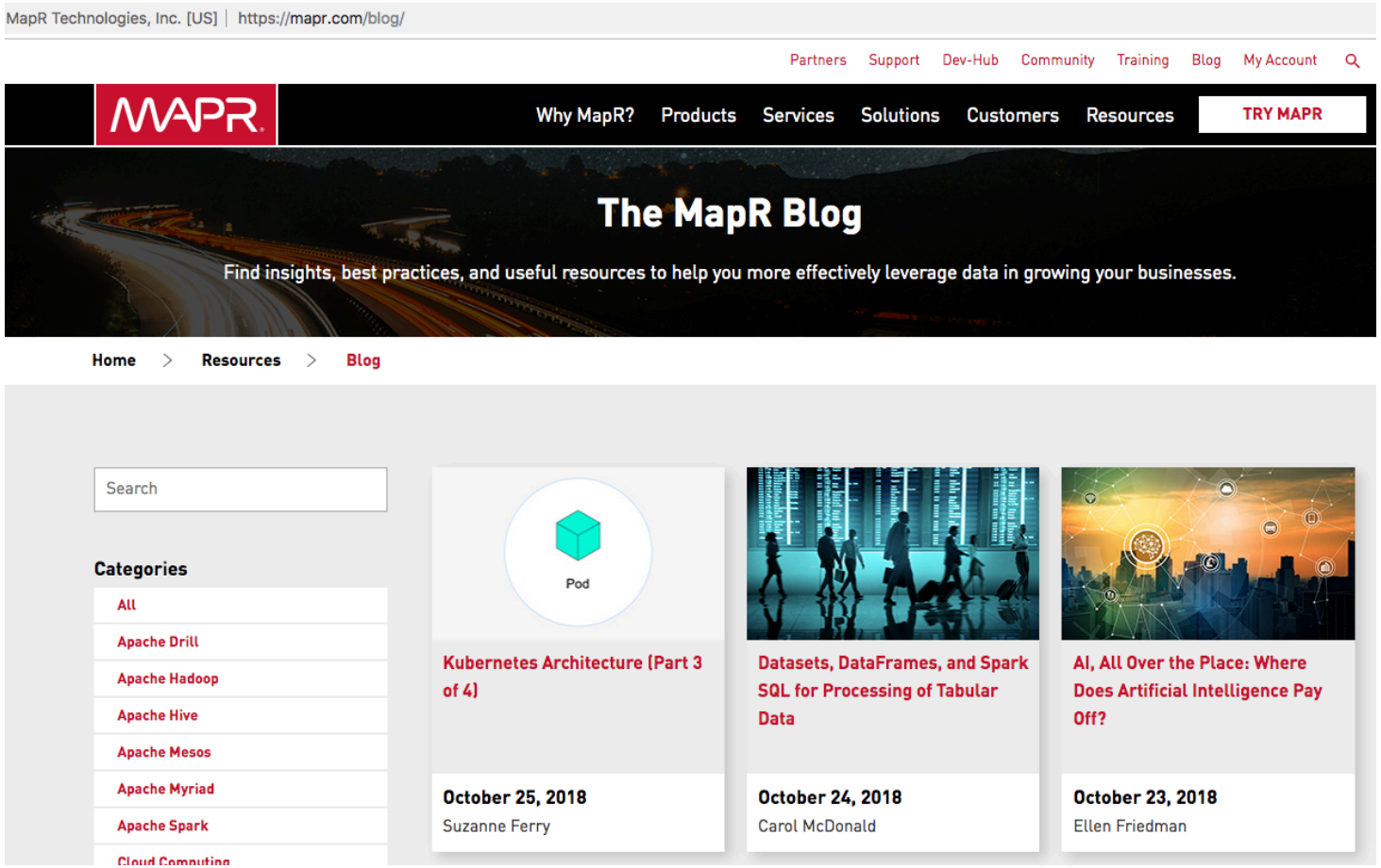
To Learn More: New Spark 2.0 training

- **MapR Free ODT http://learn.mapr.com/**



MapR Blog

https://mapr.com/blog/



MapR Data Platform

Link to Code for this webinar is in appendix of the book.
https://mapr.com/ebook/getting-started-with-apache-spark-v2/