|  |  |  |
| --- | --- | --- |
| RDD | Dataframe | Dataset |
| Resilent distributed dataset | Resilent distributed dataset | Resilent distributed dataset |
| Array of Object | Table form | Combination of RDD and Data frame therefore any form |
| Process structured as well unstructured data | Process structured data | Process structured as well as unstructured data |
| If you want to give schema to data, you need to provide externally | Automatic schema detection is possible using inferschema | Automatic schema detection is possible using inferschema |
| We can perform transformation and actions | We can perform SQL and HQL queries | We can perform transformations , actions as well SQL and HQL queries |
| Came with Spark 1.0 | Came with Spark 1.3 | Came with Spark 1.6 |
| Compile time error checking | Runtime error checking | Compile time error checking possible |
| Need to optimize RDD externally using caching persisting  Partitioning  Broadcasting  accumulating | Catalyst Optimizers are available  It is library of trees and rules.  Tree:- collection of nodes  Rules :-function which can transform the trees.  It uses pattern matching to find and replace the tree .  So catalyst optimizer find the tree which need to be transform using pattern matching and skip for which pattern does not match.  Rules executes multiple times on trees  Optimizer has phases  Analysis:- analysis of logical plan  Eg. If we have sql query select name from employee  So initially we don’t know it is valid column name or not and its type.  Catalog has all the information about fields and their types  So catalyst uses rules and catalog object to track particular column exist or not and what is its type.  Then mapping input correct name  Determines which rules are creating same output and giving them unique ids  Mapping correct return type  logical optimization:- converting queries to simple forms.  physical planning:-catalyst generates multiple plan and choose best plan depending on cost.  code generation:-  it creates syntax tree the it passes to compiler which convert queries to java byte code | Catalyst Optimizers |
| Java serialization for distributing data and its schema to nodes across cluster.  And its expensive | Dataframe stores data into binary format in of-heap memory and perform its operations in off-heap memory so java serialization not needed and its cheaper | Dataframe stores data into binary format in of-heap memory and perform its operations in off-heap memory so java serialization not needed and its cheaper |
| Garbage collection handling is hard | Produce less garbage | Produce less garbage |
| Data can come by any datasources eg. Media streams | Data generally come from,  Csv,tsv,mysql,hql,avro | Data can come from any datasources |
| RDD aggregation operations are hard to use | dataframe aggregation operation is easy to use | Dataset aggregation operation is easy to use |
| Speed is slow | Speed is very high | Speed is very high. |
| Not space efficient | Space efficient | Space efficient |
| Ways to create RDD:-  Sc.parallelize()  Sc.textfile()  Spark.read.csv().rdd  Data.toDF().rdd  Data.toDS().rdd  Rdd2= Rdd1.map(x => x/2) | Ways to create dataframe:-  Data.toDF()  Spark.read.format(“csv format”).csv(//:/)  Spark.read.format(“tsv”).tsv(“//:/)  Spark.read.format(“xml format”).xml(//:/)  Spark.read.format(“hive”).option(  url= “jdbc url”,  driver= “mysql driver”,  username= “”  password=””  tablename=””  ).load()  Spark.read.format(“mysql”).option(  url=”jdc url”  driver= “”  username=””  password=””  tablenam=””  ).load()  Sc.parallelize(“//://).toDF() | Ways to create dataset:-  Data.toDS()  Spark.read.format(“csv”).csv(//:/).as[schema]  Spark.read.format(“tsv”).tsv(//:/).as[schema]  Spark.read.format(“xml”).xml(//:/).as[schema]  Spark.reas.format(“hive”).option  (url=”jdbc url”,  Driver=”hivedriver”,  Username=””,  Password=””,  Tablename=””)  .load().as[schema]  Spark.read.format(“mysql”).option  (url=”jdbc url”,  Driver=”mysql driver”,  Username=””,  Password=””,  Tablename=””)  .load().as[schema] |
| Supports java,scala,python | Supports scala ,java ,python | Supports scala and java only |

**What is case classes?**

Useful for immutable object and u don’t need to implement any logic

Useful for pattern matching

Use to give schema to objects

It provides, equal ,hashcode,toString,apply,unapply method , no need to use getter and setter methods for it.

Apply method is use to create instance of class

Unapply method is used to destruct instance to simple forms

we extend trait using case classes

Syntax case class classname(parameters)

* We don’t need to provide new keyword to create case classes because it has apply method which creates object
* What is catalyst optimizer?
* getNumPartition?
* Hive Optimization techniques?
* reduceByKey and groupByKey?
* Coalesce –
* How to deploy Spark application
* Which are deploy modes in spark?
* When we use cluster mode and client mode
* Incremental load-
* Sqoop reducer chalta he kya

Split by will be use when primary key is evenly distributed