1. **WHAT IS PARAMETER SUBSTITUTION IN PIG? GIVE AN EXAMPLE**

If we want to run the script with different parameters. Pig supports parameter substitution, where parameters in the script are substituted with values supplied at runtime. Parameters are denoted by identifiers prefixed with a $ character.

For Example, Script that runs daily, may use the date to determine which input files it runs over.

1. **DIFF BETWEEN PIG GROUP BY & CO GROUP BY**

**Group**: The group statement collects together records with the same key.

**Cogroup**: Cogroup is a generalization of group. Instead of collecting records of one input based on a key, it collects of n inputs based on a key. The result is a record with a key and one bag for each input. Each bag contains all records from that input that have the given value for the key.

1. **WHEN DO U CHOOSE MAP REDUCE , PIG & HIVE**

**?**

1. **FLATTEN COMMAND PURPOSE ...WHEN TO USE THE SAME ..ARE THERE ANY CONSTRAINTS TO USE THE SAME?**

Sometimes there is data in a tuple or bag and if we want to remove the level of nesting from that data then Flatten modifier in Pig can be used. Flatten un-nests bags and tuples. For tuples, the Flatten operator will substitute the fields of a tuple in place of a tuple whereas un-nesting bags is a little complex because it requires creating new tuples.

**CONSTRAINTS?**

1. **STEPS FOR DEVELOPING PIG UDFS?**

* We have to develop a class by extending a base class of “Eval Function (EvalFunc)”
* To write our own customizable business logic, we must have to override a method called “public string exec(Tuple input)”
* For compilation & execution purpose we must have to add the dependent JAR files to the build path & create our own JAR file with the same business logic which need to be exported to the hadoop environment where it is running.
* In order to refer the same exported JAR file with in our PIG script, we must have to use register keyword (as first line of pig script we have to use).

1. **WHAT ARE PERFORMANCE OPTIMIZATIONS IN PIG**

* **Use Types:** If types are not specified in the load statement, Pig assumes the type of =double= for numeric computations. A lot of the time, your data would be much smaller, maybe, integer or long. Specifying the real type will help with speed of arithmetic computation.
* **Project early and often:** Pig does not (yet) determine when a field is no longer needed and drop the field from the row.
* **Filter early and often:** As with early projection, in most cases it is beneficial to apply filters as early as possible to reduce the amount of data flowing through the pipeline.
* **Use the parallel:** You can set the number of reduce tasks for the MapReduce jobs generated by Pig using parallel feature.
* **Drop Nulls before join:** With the introduction of nulls, join and cogroup semantics were altered to work with nulls. The semantic for cogrouping with nulls is that nulls from a given input are grouped together, but nulls across inputs are not grouped together. This preserves the semantics of grouping and the semantics of join (nulls are not joined across inputs). Since flattening an empty bag results in an empty row (and no output), in a standard join the rows with a null key will always be dropped.
* **Compress the results of the intermediate files:** If your Pig script generates a sequence of MapReduce jobs, you can compress the output of the intermediate jobs using LZO compression.

1. **WHAT ARE DIFF TYPES OF JOINS IN PIG....WHERE CAN WE USE THEM?**

There are 3 types of Joins in PIG 1. Replicated Joins, 2. Skewed Joins and 3. Merge Joins

1. **Relicated Joins**

Replicate join is a special type of join that works well if one or more relations are small enough to fit into main memory. In such cases, Pig can perform a very efficient join because all of the hadoop work is done on the map side. In this type of join the large relation is followed by one or more small relations. The small relations must be small enough to fit into main memory; if they don't, the process fails and an error is generated.

Eg:

big = LOAD 'big\_data' AS (b1,b2,b3);

tiny = LOAD 'tiny\_data' AS (t1,t2,t3);

mini = LOAD 'mini\_data' AS (m1,m2,m3);

C = JOIN big BY b1, tiny BY t1, mini BY m1 USING 'replicated';

1. **Skewed Joins**

Parallel joins are vulnerable to the presence of skew in the underlying data. If the underlying data is sufficiently skewed, load imbalances will swamp any of the parallelism gains. In order to counteract this problem, skewed join computes a histogram of the key space and uses this data to allocate reducers for a given key. Skewed join does not place a restriction on the size of the input keys. It accomplishes this by splitting the left input on the join predicate and streaming the right input. The left input is sampled to create the histogram.

Eg:

big = LOAD 'big\_data' AS (b1,b2,b3);

massive = LOAD 'massive\_data' AS (m1,m2,m3);

C = JOIN big BY b1, massive BY m1 USING 'skewed';

1. **Merge Joins**

Pig has implemented a merge join algorithm, or sort-merge join, although in this case the sort is already assumed to have been done (see the Conditions, below). Pig implements the merge join algorithm by selecting the left input of the join to be the input file for the map phase, and the right input of the join to be the side file. It then samples records from the right input to build an index that contains, for each sampled record, the key(s) the filename and the offset into the file the record begins at. This sampling is done in an initial map only job. A second MapReduce job is then initiated, with the left input as its input. Each map uses the index to seek to the appropriate record in the right input and begin doing the join.

Eg:

C = JOIN A BY a1, B BY b1 USING 'merge';

1. **HOW DO YOU HANDLE THE NULL RECORDS USING PIG?**

**“**is not null” operator is used to filter data with null values. Check below example

**Sample data: “nulldata.log”**

1000 ABC 12000

1001 15000

1002 XYZ 16000

Pig script to filter null values

Data = LOAD ‘nulldata.log” using PigStorage(“\t”) as (id:int,name:chararray,sal:int);

fildata = filter Data by name is no null;

STORE fildata INTO ‘nulldata’;

1. **EXPLAIN THE INTERNAL EXECUTION FLOW OF PIG...HOW BAGS AND TUPLES WILL BE EVALUATED?**

Pig is a high level and abstract language over Mapreduce to process the data. As part of the processing Pig will make use of different kinds of transformations or build in operators i.e. during the processing the data will flow through these transformations only to get the desired output. And hence we can fall Pig as “Data flow language” or “Transformation Language”.

**How bags and tuples will be evaluated**

**?**