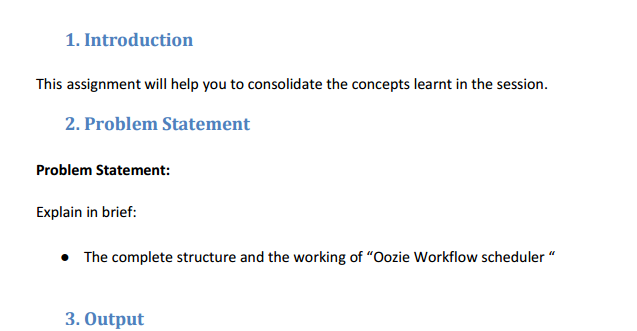
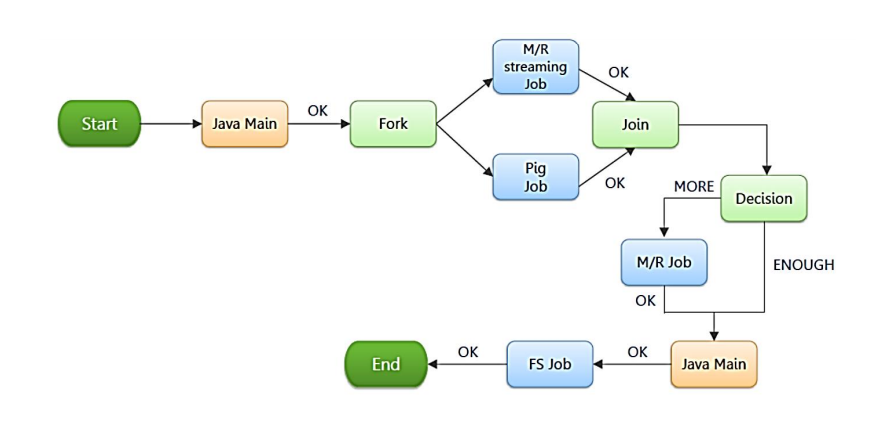
**Assignment 34.1**

****

**The complete structure and the working of Oozie Workflow scheduler**

* Workflow in case of Oozie means sequence of actions which will be arranged in a control dependency DAG (Direct Acyclic Graph).
* The actions will be in such a manner like the previous output will be the input of the next action.
* Every actions will dependent upon its previous action.
* A workflow action could be in the form of Hive action, Pig action, Java action, Shell action, etc.
* There is a advantage in Oozie framework –it will be fully integrated with Apache Hadoop stack
* As well it will be supported by Hadoop jobs for Apache MapReduce, Pig, Hive, and Sqoop.
* It can also be used for schedule jobs specific a system, like Java programs.

**Oozie workflow**



* Oozie workflow - collection of actions which is arranged in directed acyclic graph (DAG).
* Directed acyclic graph (DAG) contains 2 types of nodes and they are :

1. **Control nodes**

2. **Action nodes.**

1. ***Control Nodes -*** are the once which are used to define job chronology.

* They areprovided with the rules for starting and ending a workflow
* And also to control the workflow execution path with the help of possible decision points known *as fork and join nodes.*

Each Control node is achieved this by using all these 5 types of nodes-

1. **Start Control Node**

2. **End Control Node**

3. **Kill control Node**

4. **Decision Control Node**

5. **Fork and Join Control Nodes**

1. ***Start Control Node***

* start node is entry point of each workflow job
* It will indicate first workflow node and the workflow job must be a transition.
* Once a workflow gets started, it will automatically gets translated to node which is specified in the start .
* There must be a start node at the workflow definition.

**Syntax:**

<workflow-app name="[WF-DEF-NAME]" xmlns="uri:oozie:workflow:0.1">

...

<start to="[NODE-NAME]"/>

...

</workflow-app>

The to attribute is the name of first workflow node to execute.

Example:

<workflow-app name="foo-wf" xmlns="uri:oozie:workflow:0.1">

...

<start to="firstHadoopJob"/>

...

</workflow-app>

1. ***Kill control Node***

* kill node will allow workflow job to kill by itself.
* workflow job will kill and finishes in error (KILLED).
* When one or more actions were started then the workflow job will be executed.
* Once the kill node is reached then the actions will also be killed.
* workflow definition might have no kill mode or more kill nodes.

**Syntax:**

<workflow-app name="[WF-DEF-NAME]" xmlns="uri:oozie:workflow:0.1">

...

<kill name="[NODE-NAME]">

<message>[MESSAGE-TO-LOG]</message>

</kill>

...

</workflow-app>

The name attribute in the kill node is the name of the Kill action node.

The content of the message element will be logged as the kill reason for the workflow job.

A kill node does not have transition elements because it ends the workflow job, as KILLED .

Example:

<workflow-app name="foo-wf" xmlns="uri:oozie:workflow:0.1">

...

<kill name="killBecauseNoInput">

<message>Input unavailable</message>

</kill>

...

</workflow-app>

1. ***End Control Node***

* end node – will be at the end for a workflow job.
* Which indicates when the workflow job will get complete.
* Once workflow job reach the end it will get finished successfully (SUCCEEDED).
* When more actions gets started then workflow job will get executed.
* When the end node gets reached, the actions will be killed.
* If such is the scenario then workflow job will be considered successfully run.
* A workflow definition must have one end node.

**Syntax:**

<workflow-app name="[WF-DEF-NAME]" xmlns="uri:oozie:workflow:0.1">

...

<end name="[NODE-NAME]"/>

...

</workflow-app>

The name attribute is the name of the transition to do to end the workflow job.

Example:

<workflow-app name="foo-wf" xmlns="uri:oozie:workflow:0.1">

...

<end name="end"/>

</workflow-app>

1. *Decision Control Node*

* Decision node – will enable workflow to select the execution path.
* Decision node behavior will be considered as switch-case statement.
* Decision node will consist of list of predicates-transition pairs plus as well default transition.
* Predicates will get evaluated in such a order until one gets evaluated as true .
* And transition will take place.
* If no predicate is evaluated as true then default transition will take place.
* Predicates will be as JSP Expression Language (EL) expressions (will be same as 4.2 section in this document) and will be resolved into a Boolean, true or false .

For example:

${fs:fileSize('/usr/foo/myinputdir') gt 10 \* GB}

**Syntax:**

<workflow-app name="[WF-DEF-NAME]" xmlns="uri:oozie:workflow:0.1">

...

<decision name="[NODE-NAME]">

<switch>

<case to="[NODE\_NAME]">[PREDICATE]</case>

...

<case to="[NODE\_NAME]">[PREDICATE]</case>

<default to="[NODE\_NAME]"/>

</switch>

</decision>

...

</workflow-app>

* Name attribute will act as decision node.
* Each case elements will contain a predicate as transition name.
* The predicate ELs will be evaluated in such a manner until one gets returned true.
* And transition will be taken.
* By default each element will indicate transition to take place if no predicate is evaluated as true .
* All decision nodes will have a default element in order to avoid the workflow getting into error state .
* if no predicate was evaluated to true.

**Example:**

<workflow-app name="foo-wf" xmlns="uri:oozie:workflow:0.1">

...

<decision name="mydecision">

<switch>

<case to="reconsolidatejob">

${fs:fileSize(secondjobOutputDir) gt 10 \* GB}

</case>

<case to="rexpandjob">

${fs:filSize(secondjobOutputDir) lt 100 \* MB}

</case>

<case to="recomputejob">

${ hadoop:counters('secondjob')[RECORDS][REDUCE\_OUT] lt 1000000 }

</case>

<default to="end"/>

</switch>

</decision>

...

</workflow-app>

**5. Fork and Join Control Nodes**

* Fork node will splits the path of execution into multiple number of concurrent paths of execution.
* Join node will wait until every concurrent execution path for previous fork node which arrive to it.
* Fork and join nodes will be used in pairs.
* The join node will assume concurrent execution paths as children which is of same fork node.

**Syntax:**

<workflow-app name="[WF-DEF-NAME]" xmlns="uri:oozie:workflow:0.1">

...

<fork name="[FORK-NODE-NAME]">

<path start="[NODE-NAME]" />

...

<path start="[NODE-NAME]" />

</fork>

...

<join name="[JOIN-NODE-NAME]" to="[NODE-NAME]" />

...

</workflow-app>

* Name attribute in fork node will be the name of the workflow fork node.
* Start attribute in path elements of fork node will indicate the name of workflow node which acts as a part of the concurrent execution paths.
* Name attribute in join node is name of the workflow join node.
* Attribute in join node will indicate name of the workflow node
* It will get executed after all concurrent execution paths of corresponding fork which arrive to the join node.

Example:

<workflow-app name="sample-wf" xmlns="uri:oozie:workflow:0.1">

...

<fork name="forking">

<path start="firstparalleljob"/>

<path start="secondparalleljob"/>

</fork>

<action name="firstparallejob">

<map-reduce>

<job-tracker>foo:9001</job-tracker>

<name-node>bar:9000</name-node>

<job-xml>job1.xml</job-xml>

</map-reduce>

<ok to="joining"/>

<error to="kill"/>

</action>

<action name="secondparalleljob">

<map-reduce>

<job-tracker>foo:9001</job-tracker>

<name-node>bar:9000</name-node>

<job-xml>job2.xml</job-xml>

</map-reduce>

<ok to="joining"/>

<error to="kill"/>

</action>

<join name="joining" to="nextaction"/>

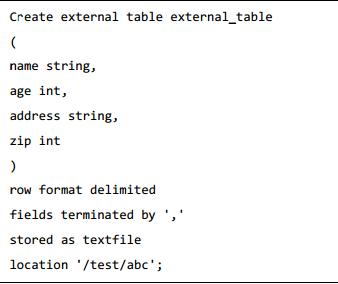
...

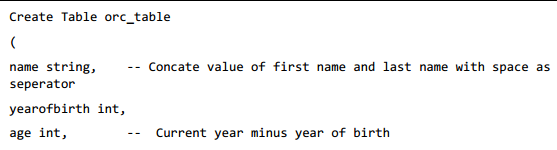
</workflow-app>

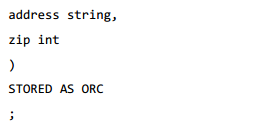
* **Action Node**
* Action node will represents workflow task
* For example- moving files into the HDFS, running the MapReduce, Pig or[Hive](http://www.guru99.com/hive-tutorials.html)jobs, importing data using Sqoop or running a shell script of a program written in Java.

**Example** - Suppose I want to load a data from external hive table to an ORC Hive table.

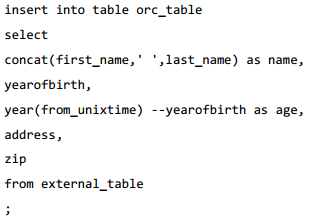
Step 1: **creating Hive external table (say external.hive) in text format**



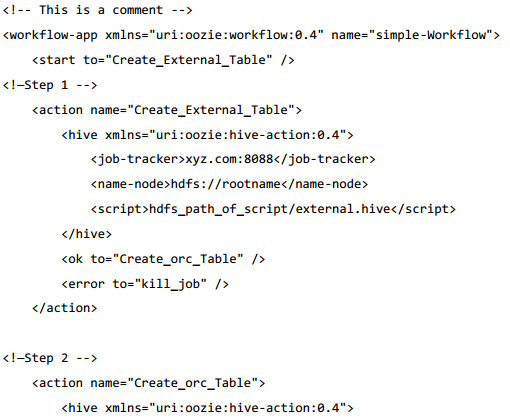
Step 2: **creating Hive external table in ORC format** **(say external.hive)**



Step3: **Inserting into Table orc from Text Table**



**Instead manual process one can use Oozie scheduler**





**Explanation**

* Action Nodes – will define the type of job in which the node runs.
* Hive node inside action node will define hive action.
* It could be a pig, java, shell action, etc.
* There can be a user defined tag for each type of action .
* job tracker is defined for the above job with the name node details, script to be used and the param entity.
* Script tag will define the script to run the hive action.
* Param tag will define value of the pass into hive script.
* (Passing database name is done in step 3)