# 1 Benchmark Function (main function)

**run\_benchmark**()

* **READ** the configuration parameters into the memory (config params `file.configuration.name ...`)
* save the current time as the start time of the 'benchmark' action
* **READ** the bulk file data into the partitioned collection bulk\_data\_partitions (config param 'file.bulk.name')

partition key = modulo (ASCII value of 1st byte of key

\* 251

+ ASCII value of 2nd byte of key),

number partitions (config param 'benchmark.number.partitions')

* create a separate database connection (without auto commit behaviour) for each partition
* trial\_max = 0

trial\_min = 0

trial\_no = 0

trial\_sum = 0

**WHILE** trial\_no < config\_param 'benchmark.trials'

**duration\_trial = DO** **run\_trial**(database connections,

trial\_no,

bulk\_data\_partitions)

IF trial\_max == 0 OR duration\_trial > trial\_max

trial\_max = duration\_trial

END IF

IF trial\_min == 0 OR duration\_trial < trial\_min

trial\_min = duration\_trial

END IF

trial\_sum + duration\_trial

**ENDWHILE**

* partition\_key = 0

**WHILE** partition\_key < config\_param 'benchmark.number.partitions'

close the database connection

**ENDWHILE**

* **WRITE** an entry for the action 'benchmark' in the result file (config param 'file.result.name')

* INFO Duration (ms) trial min. : trial\_min

INFO Duration (ms) trial max. : trial\_max

INFO Duration (ms) trial average : trial\_sum / config\_param 'benchmark.trials'

* INFO Duration (ms) benchmark run : duration\_benchmark

# 2 Trial Function

**run\_trial**(database connections,

trial\_no,

bulk\_data\_partitions)

* save the current time as the start time of the 'trial' action
* INFO Start trial no. Trial\_no
* create the database table (config param 'sql.create')

**IF** error

drop the database table (config param 'sql.drop')

create the database table (config param 'sql.create')

**ENDIF**

* **DO** **run\_insert**(database connections,

trial\_no,

bulk\_data\_partitions)

* **DO** **run\_select**(database connections,

trial\_no,

bulk\_data\_partitions)

* drop the database table (config param 'sql.drop')
* **WRITE** an entry for the action 'trial' in the result file (config param 'file.result.name')
* INFO Duration (ms) trial : duration\_trial
* **RETURN** duration = end time - start time

# 3 Insert Control Function

**run\_insert**(database connections,

trial\_no,

bulk\_data\_partitions)

* save the current time as the start time of the 'query' action
* partition\_key = 0

**WHILE** partition\_key < config\_param 'benchmark.number.partitions'

**IF** config\_param 'benchmark.core.multiplier' = 0

**DO** **run\_insert\_helper**(database connections(partition\_key),

bulk\_data\_partitions(partition\_key),

partition\_key)

**ELSE**

**DO** **run\_insert\_helper** (database connections(partition\_key),

bulk\_data\_partitions(partition\_key,

partition\_key)) as a thread

**ENDIF**

**ENDWHILE**

* **WRITE** an entry for the action 'query' in the result file (config param 'file.result.name')

# 4 Insert Helper Function

**run\_insert\_helper** (database connection,

bulk\_data\_partition,

partition\_key)

- INFO Start insert partition\_key=partition\_key

* count = 0

collection batch\_collection = empty

**WHILE** iterating through the collection bulk\_data\_partition

count + 1

add the SQL statement in config param 'sql.insert' with the current bulk\_data entry to the collection batch\_collection

**IF** config\_param 'benchmark.batch.size' > 0

**IF** count modulo config param 'benchmark.batch.size' = 0

execute the SQL statements in the collection batch\_collection

batch\_collection = empty

**ENDIF**

**ENDIF**

**IF** config param 'benchmark.transaction.size' > 0

AND count modulo config param 'benchmark.transaction.size' = 0

commit

**ENDIF**

**ENDWHILE**

* **IF** collection batch\_collection is not empty

execute the SQL statements in the collection batch\_collection

**ENDIF**

* commit
* INFO End insert partition\_key=partition\_key

# 5 Select Control Function

**run\_select**(database connections,

trial\_no,

bulk\_data\_partitions)

* save the current time as the start time of the 'query' action
* partition\_key = 0

**WHILE** partition\_key < config\_param 'benchmark.number.partitions'

**IF** config\_param 'benchmark.core.multiplier' = 0

DO **run\_select\_helper**(database connections(partition\_key),

bulk\_data\_partitions(partition\_key,

partition\_key)

**ELSE**

**DO** **run\_select\_helper** (database connections(partition\_key),

bulk\_data\_partitions(partition\_key,

partition\_key) as a thread

**ENDIF**

**ENDWHILE**

* **WRITE** an entry for the action 'query' in the result file (config param 'file.result.name')

# 6 Select Helper Function

**run\_select\_helper** (database connection,

bulk\_data\_partition,

partition\_key)

* INFO Start select partition\_key=partition\_key
* execute the SQL statement in config param 'sql.select'
* count = 0

**WHILE** iterating through the result set

count + 1

**ENDWHILE**

* **IF NOT** count = size(bulk\_data\_partition)

display an error message

**ENDIF**

* INFO End select partition\_key=partition\_key