**Report for Bug:** [**84613**](https://github.com/elastic/elasticsearch/issues/84613) **- SQL: GROUP BY constant with ORDER BY aggregate crashes the server**

File Changes are in GitHub branch [- davidkelly-84613](https://github.com/denisanzora/ElasticSearch581/tree/davidkelly-84613)

Steps for the fix:

1. Pre/post-factoring activities
   1. Setup up a local environment to run the application
   2. Make sure all tests run.
   3. Make corrective changes
      1. Extract method into a new method.
   4. Added unit tests (see appendix)
   5. Ran integrations tests via Postman
2. Bug localization
   1. Study to the bug and look for comments by other Elasticsearch developers. I found other bugs that were similar and that were fixed to help find a location in the code.
   2. I added logging in some areas that were possible locations and eventually found where the bug was happening.
3. Change impact analysis and assessment criteria
   1. The strategy for a fix was to handle the SQL query error by not crashing the system. This was done by changing the code to interrupt the thread versus halting the program. Several tests were completed after to make sure there were no side effects. The logs were clean from error.
4. Clarity of the report
   1. The repro steps were located in the bug report. However, the exact steps to reproduce the bug were not clear and needed to be refined and tested.
5. The choice of techniques and design decisions
   1. My decision for a fix was to prevent the application from crashing. I did this by logging the error and allowing the user to correct their query.
6. Build and run Elasticsearch locally to create a localhost port.
7. Ran the units successfully.
8. Test that the port for the API is working properly.
   1. Request: curl -u elastic:password -s -k http://localhost:9200/\_cluster/health
   2. Response: {"cluster\_name":"runTask","status":"yellow","timed\_out":false,"number\_of\_nodes":1,"number\_of\_data\_nodes":1,"active\_primary\_shards":2,"active\_shards":2,"relocating\_shards":0,"initializing\_shards":0,"unassigned\_shards":1,"delayed\_unassigned\_shards":0,"number\_of\_pending\_tasks":0,"number\_of\_in\_flight\_fetch":0,"task\_max\_waiting\_in\_queue\_millis":0,"active\_shards\_percent\_as\_number":66.66666666666666}
9. Tools:
   1. Intellij was the IDE.
   2. Java and JUnit
   3. Curl was used to test simple api calls.
   4. Postman was used to test more complex calls.
10. Reproduce the problem:
    1. Using the sample request in the bug description, setup the local environment to run those sql queries in Postman to run the requests.
    2. Push data into Elasticsearch:
       1. Request:
       * URI: localhost:9200/\_bulk
       * Request Type: POST
       * Auth: elastic:password
       * ContentType: application/json
       * Body:

{"index": {"\_index":"person"}}

{"name" : "dave", "surname":"kelly", "salary": 1000}

{"index": {"\_index":"person"}}

{"name" : "john", "surname":"doe", "salary": 2000}

* + 1. Response:

{

    "took": 104,

    "errors": **false**,

    "items": [

        {

            "index": {

                "\_index": "person",

                "\_id": "luGvNoQBtT-rbOfOhOQh",

                "\_version": 1,

                "result": "created",

                "\_shards": {

                    "total": 2,

                    "successful": 1,

                    "failed": 0

                },

                "\_seq\_no": 2,

                "\_primary\_term": 1,

                "status": 201

            }

        },

        {

            "index": {

                "\_index": "person",

                "\_id": "l-GvNoQBtT-rbOfOhOQh",

                "\_version": 1,

                "result": "created",

                "\_shards": {

                    "total": 2,

                    "successful": 1,

                    "failed": 0

                },

                "\_seq\_no": 3,

                "\_primary\_term": 1,

                "status": 201

            }

        }

    ]

}

* 1. Pull data from Elasticsearch:
     + URI: localhost:9200/\_sql
     + Request Type: GET
     + Auth: elastic:password
     + ContentType: application/json
     + Body:

{

 "query":"SELECT 1 as c, MAX(salary) FROM person GROUP BY c ORDER BY MAX(salary) desc"

}

Result: Crashes server

1. Fix the problem:
   1. Change code - [ElasticsearchUncaughtExceptionHandler.java](https://github.com/denisanzora/ElasticSearch581/compare/main...davidKelly#diff-3663fdf752682bad01dbae901878fa715cb45bbda66c9868183a6a577d9361f6)
   2. Unit tests - [ElasticsearchUncaughtExceptionHandlerTests.java](https://github.com/denisanzora/ElasticSearch581/compare/main...davidKelly#diff-00b743146c8f61b0ad3cfa713af86ec90e03748a4b0cfbbf2b98909066d47ec4)
      1. assertAssertionError()
      2. assertNonAssertionError()
   3. Integration Tests
      1. Use Postman to load data and then query the data. Check for errors.
   4. Change documentation - [stopping.asciidoc](https://github.com/denisanzora/ElasticSearch581/compare/main...davidKelly#diff-6a785baf28264c00fdcd12c3cf10fd9b1f1d2a32f2496df3a9e67cf06d1de057)
      1. A new section was add to stopping.asciidoc.

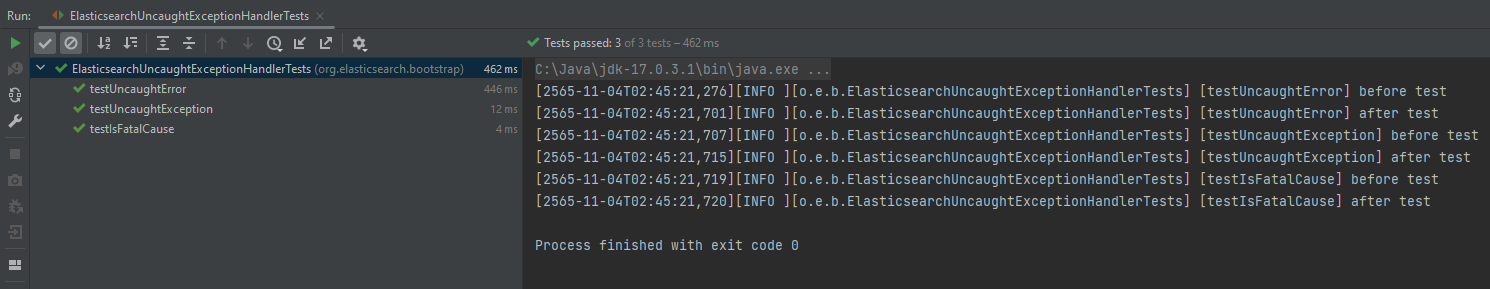
=== Handling non-Fatal Errors

If Elasticsearch encounters an AssertionError, the thread will be interrupted and the Elasticsearch virtual machine will continue. Thread interruption is a gentle way to nudge a thread. It is used to give threads a chance to exit cleanly.

1. Commits pushed to branch [davidkelly-84613](https://github.com/denisanzora/ElasticSearch581/tree/davidkelly-84613). note: commits were squashed to 1 commit. This is a requirement to do a PR for Elasticsearch.
   1. [Added thread.interupt to avoid a crash](https://github.com/denisanzora/ElasticSearch581/commit/28d83e844b51ce7c3a7453dd288b36a5ce337d9e)
   2. [Added tests for ElasticsearchUncaughtExceptionHandler AssertionError](https://github.com/denisanzora/ElasticSearch581/commit/21870f254d2ec715f01bff0f719eecb617ca222b)
   3. [Fix formatting](https://github.com/denisanzora/ElasticSearch581/commit/a0b77aff82285911801ff4538cf6e4a78f2000b7)
   4. [Update stopping.asciidoc documentation](https://github.com/denisanzora/ElasticSearch581/commit/2c0062a5ac996bc4133efde8d3009b9e4bca7a89)
2. Code Reviewed conducted by Garhgaj and Denis

Appendix

1. Test results.



1. Tests added (highlighted):

public void testIsFatalCause() {

assertFatal(new OutOfMemoryError());

assertFatal(new StackOverflowError());

assertFatal(new InternalError());

assertFatal(new UnknownError());

assertFatal(new IOError(new IOException()));

assertNonFatal(new RuntimeException());

assertNonFatal(new UncheckedIOException(new IOException()));

assertAssertionError(new AssertionError());

assertNonAssertionError(new IOException())

}

private void assertAssertionError(Throwable cause) {

assertTrue(ElasticsearchUncaughtExceptionHandler.isAssertionError(cause));

}

private void assertNonAssertionError(Throwable cause) {

assertFalse(ElasticsearchUncaughtExceptionHandler.isAssertionError(cause));

}