拉勾教育
—互联网人实战大学—

《31 讲带你搞懂 SkyWalking》

徐郡明 资深技术专家

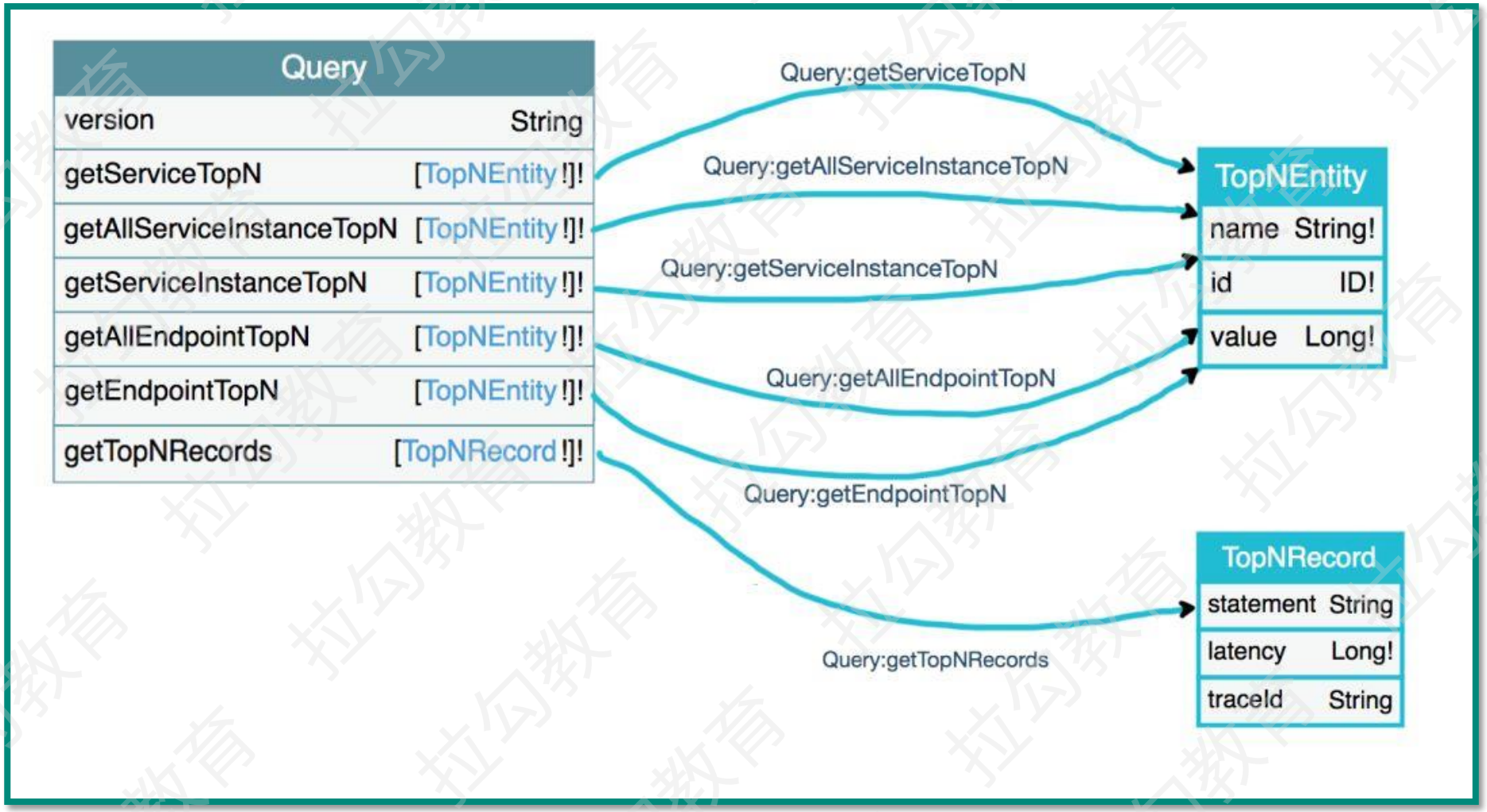
— 拉勾教育出品 —

拉勾教育
—互联网人实战大学—

第29讲：深入 query-graphql 插件 SW Rocketbot 背后的英雄（下）

L / A / G / O / U

TopN 查询


Query
version String
getServiceTopN [TopNEntity !]!
getAllServiceInstanceTopN [TopNEntity !]!
getServiceInstanceTopN [TopNEntity !]!
getAllEndpointTopN [TopNEntity !]!
getEndpointTopN [TopNEntity !]!
getTopNRecords [TopNRecord !]!
Query:getServiceTopN
Query:getAllServiceInstanceTopN
Query:getServiceInstanceTopN
Query:getAllEndpointTopN
Query:getEndpointTopN
Query:getTopNRecords
TopNEntity
name String!
id ID!
value Long!
TopNRecord
statement String
latency Long!
traceId String

TopN 查询

拉勾教育
— 互联网人实战大学 —

```
private int serviceId; //查询哪个 DB的慢查询
private String metricName; //查询的 Index别名，即
top_n_database_statement
private int topN; //返回 N个耗时最大的慢查询，默认20
private Order order; //排序方式，查询 DB慢查询自然是 DES
private Duration duration; //查询的时间范围
```

TopN 查询

拉勾教育
— 互联网人实战大学 —

```
SearchSourceBuilder sourceBuilder = SearchSourceBuilder.searchSource();
BoolQueryBuilder boolQueryBuilder = QueryBuilders.boolQuery();
//指定查询的时间范围
boolQueryBuilder.must().add(QueryBuilders.rangeQuery(TopN.TIME_BUCKET).gte(
startSecondTB).lte(endSecondTB));
//指定查询的 DB对应的 serviceId
boolQueryBuilder.must().add(QueryBuilders.termQuery(TopN.SERVICE_ID,
serviceId));
sourceBuilder.query(boolQueryBuilder);
//按照 latency进行排序，指定返回 topN条记录
sourceBuilder.size(topN).sort(TopN.LATENCY, order.equals(Order.DES) ?
SortOrder.DESC : SortOrder.ASC);
SearchResponse response = getClient().search(metricName, sourceBuilder);
```

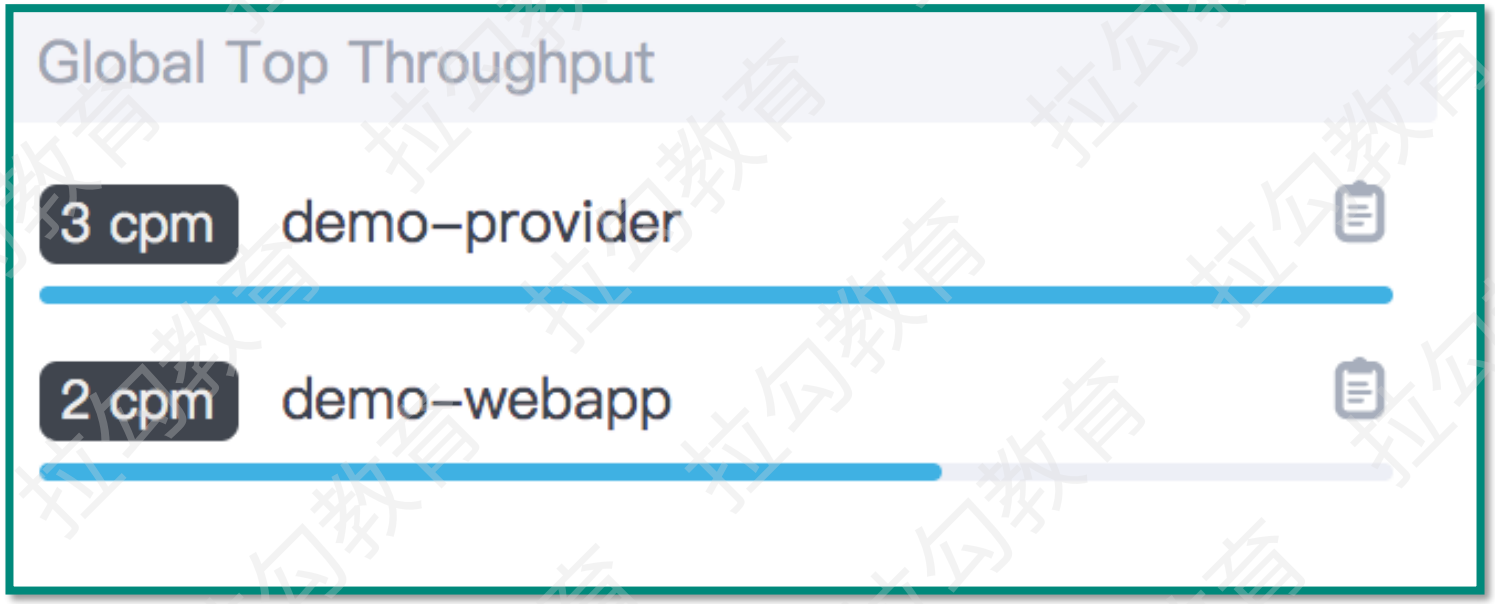
TopN 查询

拉勾教育
— 互联网人实战大学 —

```
AggregationQuery
  getAllEndpointTopN(String, int, Duration, Order): List<TopNEntity>
  getAllServiceInstanceTopN(String, int, Duration, Order): List<TopNEntity>
  getEndpointTopN(int, String, int, Duration, Order): List<TopNEntity>
  getServiceInstanceTopN(int, String, int, Duration, Order): List<TopNEntity>
  getServiceTopN(String, int, Duration, Order): List<TopNEntity>
```

TopN 查询

拉勾教育
— 互联网人实战大学 —



TopN 查询

拉勾教育
— 互联网人实战大学 —

```
SearchSourceBuilder sourceBuilder = SearchSourceBuilder.searchSource();
//指定查询的起止时间，示例中起止时间分别是201901072044~201901072059
sourceBuilder.query(QueryBuilders.rangeQuery(Metrics.TIME_BUCKET).lte(e
ndTB).gte(startTB));
boolean asc = false; //确定排序方式，示例中查询服务的吞吐量是从高到低排序的
if (order.equals(Order.ASC)) { asc = true; }

TermsAggregationBuilder aggregationBuilder = AggregationBuilders
    .terms(Metrics.ENTITY_ID) //按照entity_id进行聚合，在以 service_cpm 为别
名的 Index中 entity_id字段记录的是 serviceId
    .field(Metrics.ENTITY_ID)
    .order(BucketOrder.aggregation(valueCName, asc)) //按照指定字段排序，示
例中以 service_cpm 为别名的 Index会按照 value字段进行排序
    .size(topN) //返回记录的数量，Skywalking Rocketbot传递的topN参数为10
    .subAggregation( //根据 entity_id分组后会计算 valueCName字段的平均值，
生成的新字段名称也为valueCName
        AggregationBuilders.avg(valueCName).field(valueCName)
    );
sourceBuilder.aggregation(aggregationBuilder);
//发送SearchRequest请求
SearchResponse response = getClient().search(indexName, sourceBuilder);
```

TopN 查询

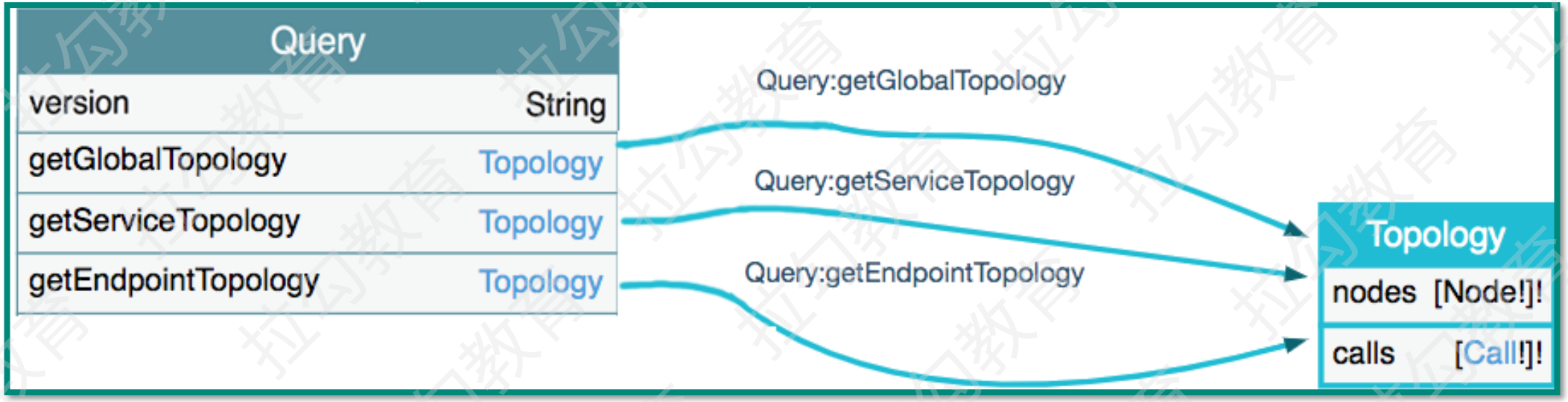
拉勾教育
— 互联网人实战大学 —

```
List<TopNEntity> topNEntities = new ArrayList<>();
Terms idTerms = response.getAggregations().get(Metrics.ENTITY_ID);
for (Terms.Bucket termsBucket : idTerms.getBuckets()) {
    TopNEntity topNEntity = new TopNEntity();
    topNEntity.setId(termsBucket.getKeyAsString()); //获取 ServiceId
    Avg value = termsBucket.getAggregations().get(valueCName); // 获取 cpm平均值
    topNEntity.setValue((long)value.getValue());
    topNEntities.add(topNEntity);
}
return topNEntities;
```

TopologyQuery

拉勾教育
— 互联网人实战大学 —

TopologyQuery



TopologyQuery


```
▼ serviceRelationServerCalls = {ArrayList@7224}  size = 2
  ▼ 0 = {Call$CallDetail@7239}
    ▶ f id = "1_2"
    ▶ f source = {Integer@7242} 1
    ▶ f target = {Integer@7243} 2
    ▶ f detectPoint = {DetectPoint@7222} "SERVER"
    ▶ f componentId = {Integer@7242} 1
  ▼ 1 = {Call$CallDetail@7240}
    ▶ f id = "2_3"
    ▶ f source = {Integer@7243} 2
    ▶ f target = {Integer@7246} 3
    ▶ f detectPoint = {DetectPoint@7222} "SERVER"
    ▶ f componentId = {Integer@7246} 3
```

serviceId=1表示的用户

serviceId=2的服务调用了serviceId=3的服务

TopologyQuery


```
▼ serviceRelationClientCalls = {ArrayList@7257} size = 1
  ▼ 0 = {Call$CallDetail@7260}
    ▶ f id = "2_3"
    ▶ f source = {Integer@7243} 2
    ▶ f target = {Integer@7246} 3
    ▶ f detectPoint = {DetectPoint@7255} "CLIENT"
    ▶ f componentId = {Integer@7246} 3
```

serviceId=2的服务
调用了
serviceId=3的服务

TopologyQuery

拉勾教育
— 互联网人实战大学 —

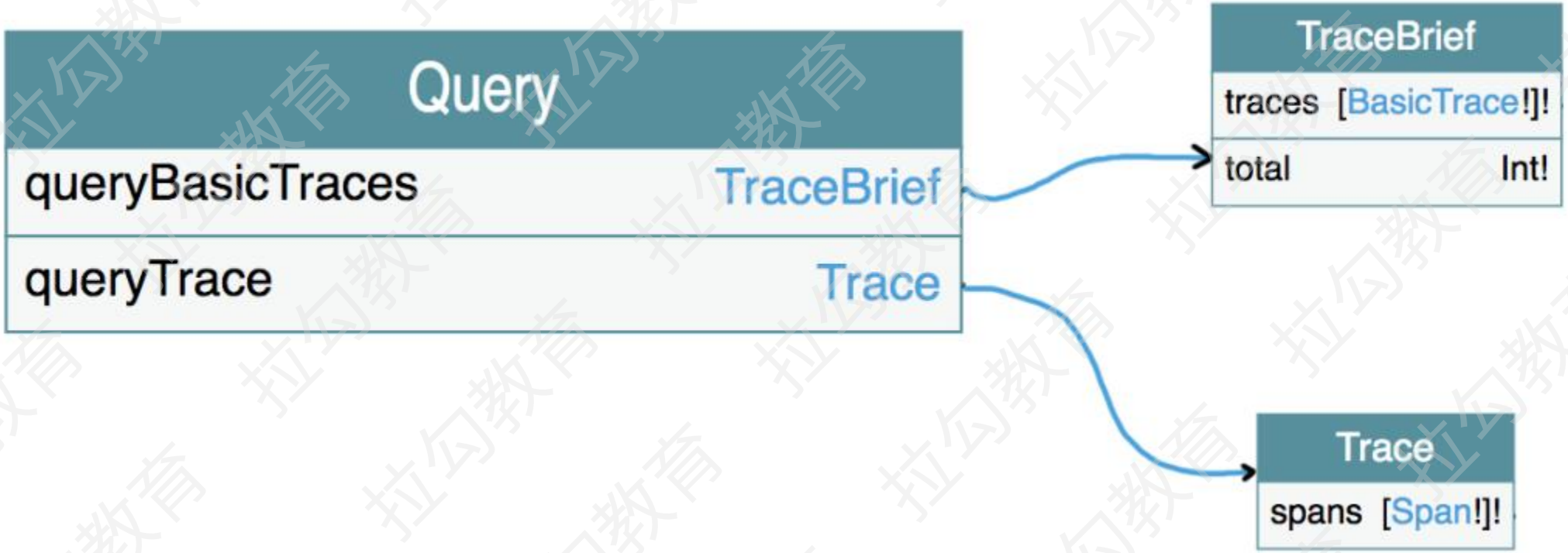
```
topology = {Topology@7482}
  nodes = {ArrayList@7484}  size = 3
    0 = {Node@7487}
      id = 1
      name = "User"
      type = "USER"
      isReal = false
    1 = {Node@7488}
      id = 2
      name = "demo-webapp"
      type = "Tomcat"
      isReal = true
    2 = {Node@7481}
      id = 3
      name = "demo-provider"
      type = "Dubbo"
      isReal = true
```

TopologyQuery


```
▼ topology = {Topology@7482}
  ▶ nodes = {ArrayList@7484}  size = 3
  ▼ calls = {ArrayList@7483}  size = 2
    ▼ 0 = {Call@7419}
      ▶ source = {Integer@7243} 2
      ▶ target = {Integer@7246} 3
      ▶ sourceComponents = {ArrayList@7491}  size = 1
      ▶ targetComponents = {ArrayList@7492}  size = 1
      ▶ id = "2_3"
      ▶ detectPoints = {ArrayList@7493}  size = 2
        sourceComponentIDs = null
        targetComponentIDs = null
    ▼ 1 = {Call@7490}
      ▶ source = {Integer@7242} 1
      ▶ target = {Integer@7243} 2
        sourceComponents = {ArrayList@7494}  size = 0
      ▶ targetComponents = {ArrayList@7495}  size = 1
      ▶ id = "1_2"
      ▶ detectPoints = {ArrayList@7496}  size = 1
        sourceComponentIDs = null
        targetComponentIDs = null
```

TraceQuery

TraceQuery


Query
queryBasicTraces
TraceBrief
queryTrace
Trace
TraceBrief
traces [BasicTrace!]!
total
Int!
Trace
spans [Span!]!

TraceQuery

拉勾教育
— 互联网人实战大学 —

- **serviceId、serviceInstanceId、endpointId 字段**

  TraceSegment 关联的 Service、ServiceInstance、Endpoint

- **traceId 字段**：指定 TraceSegment 的 traceId
- **queryDuration 字段**：指定查询的时间跨度
- **minTraceDuration 和 maxTraceDuration 字段**

  指定 TraceSegment 耗时范围，只查询耗时在 minTraceDuration~maxTraceDuration 之间的 Trace

- **traceState 字段**：Trace 的状态信息，枚举，可选值有 ALL、SUCC、ERROR 三个值
- **queryOrder 字段**：查询结果的排序方式，枚举，可选值有 BY_DURATION、BY_START_TIME 两个值
- **paging 字段**：分页信息，类似于 SQL 语句中的 limit 部分，指定了此次查询的起始位置以及结果条数

TraceQuery

拉勾教育
— 互联网人实战大学 —

```
SearchSourceBuilder sourceBuilder = SearchSourceBuilder.searchSource();
BoolQueryBuilder boolQueryBuilder = QueryBuilders.boolQuery();
sourceBuilder.query(boolQueryBuilder);
List<QueryBuilder> mustQueryList = boolQueryBuilder.must();
if (startSecondTB != 0 && endSecondTB != 0) { //查询时间范围，即过滤 time_bucket字段

mustQueryList.add(QueryBuilders.rangeQuery(SegmentRecord.TIME_BUCKET).gte(startSecon
dTB).lte(endSecondTB));
}
if (minDuration != 0 || maxDuration != 0) { //查询TraceSegment的耗时范围，即过滤 latency字段
  RangeQueryBuilder rangeQueryBuilder =
QueryBuilders.rangeQuery(SegmentRecord.LATENCY);
  if (minDuration != 0) { rangeQueryBuilder.gte(minDuration); }
  if (maxDuration != 0) { rangeQueryBuilder.lte(maxDuration); }
  boolQueryBuilder.must().add(rangeQueryBuilder);
}
if (!Strings.isNullOrEmpty(endpointName)) { //过滤 endpoint_name字段
  String matchCName =
MatchCNameBuilder.INSTANCE.build(SegmentRecord.ENDPOINT_NAME);
  mustQueryList.add(QueryBuilders.matchPhraseQuery(matchCName, endpointName));
```

TraceQuery

拉勾教育
— 互联网人实战大学 —

```
if (serviceId != 0) { //查询 TraceSegment所属的Service，即过滤 service_id字段
    boolQueryBuilder.must().add(QueryBuilders.termQuery(SegmentRecord.SERVICE_ID,
serviceId));
}
if (serviceInstanceId != 0) { //查询 TraceSegment所属的ServiceInstance，即过滤
service_instance_id字段

boolQueryBuilder.must().add(QueryBuilders.termQuery(SegmentRecord.SERVICE_INSTANCE_I
D, serviceInstanceId));
}
if (endpointId != 0) { //查询 TraceSegment所属的 Endpoint，即过滤endpoint_id字段
    boolQueryBuilder.must().add(QueryBuilders.termQuery(SegmentRecord.ENDPOINT_ID,
endpointId));
}
if (!Strings.isNullOrEmpty(traceId)) { //查询 TraceSegment所属的 traceId，即过滤 trace_id字段
    boolQueryBuilder.must().add(QueryBuilders.termQuery(SegmentRecord.TRACE_ID, traceId));
}
switch (traceState) { //查询 TraceSegment覆盖的逻辑是否发生异常，即过滤 is_error字段
    case ERROR:
        mustQueryList.add(QueryBuilders.matchQuery(SegmentRecord.IS_ERROR,
```

TraceQuery

拉勾教育
— 互联网人实战大学 —

```
BooleanUtils.TRUE));
        break;
    case SUCCESS:
        mustQueryList.add(QueryBuilders.matchQuery(SegmentRecord.IS_ERROR,
BooleanUtils.FALSE));
        break;
}
switch (queryOrder) { //查询得到的多个 TraceSegment的排序字段，可以按照start_time字段或是
latency字段逆序排序
    case BY_START_TIME:
        sourceBuilder.sort(SegmentRecord.START_TIME, SortOrder.DESC);
        break;
    case BY_DURATION:
        sourceBuilder.sort(SegmentRecord.LATENCY, SortOrder.DESC);
        break;
}
sourceBuilder.size(limit); //指定此次查询返回的Document个数
sourceBuilder.from(from); //指定查询的起始位置
//执行上述 SearchRequest请求，查询的是别名为segment的Index
SearchResponse response = getClient().search(SegmentRecord.INDEX_NAME, sourceBuilder);
```

TraceQuery

拉勾教育
— 互联网人实战大学 —

```
SearchSourceBuilder sourceBuilder = SearchSourceBuilder.searchSource();
//精确匹配 trace_id字段
sourceBuilder.query(QueryBuilders.termQuery(SegmentRecord.TRACE_ID, traceId));
//一条Trace中TraceSegment的个数上限默认是200，application.yml文件中有相应配置项可调整
sourceBuilder.size(segmentQueryMaxSize);
//执行 SearchRequest请求，查询的依旧是别名为segment的Index
SearchResponse response = getClient().search(SegmentRecord.INDEX_NAME, sourceBuilder);
```

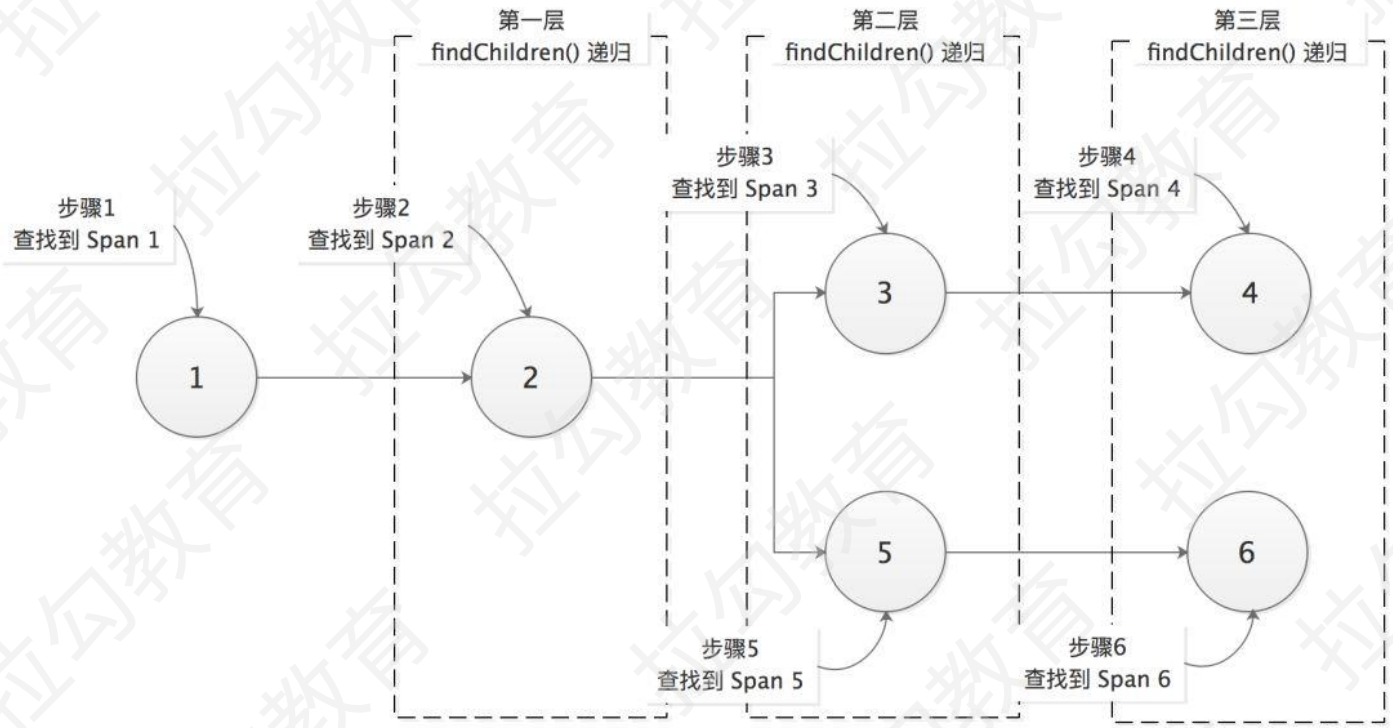
TraceQuery

拉勾教育
— 互联网人实战大学 —

- 创建 Trace 返回值，收集全部 Span 对象
- 排序 Span

```
List<Span> sortedSpans = new LinkedList<>();
//查找该Trace中最顶层的rootSpan，即第一个 Span
List<Span> rootSpans = findRoot(trace.getSpans());
rootSpans.forEach(span -> {
  List<Span> childrenSpan = new ArrayList<>();
  childrenSpan.add(span);
  //这里会递归查找当前span的子Span，并添加到sortedSpans这个List中
  findChildren(trace.getSpans(), span, childrenSpan);
  sortedSpans.addAll(childrenSpan);
});
//重新设置 Trace.spans字段
trace.getSpans().clear();
trace.getSpans().addAll(sortedSpans);
return trace;
```

TraceQuery



childrenSpan集合的变化过程：
步骤1、childrenSpan集合：1
步骤2、childrenSpan集合：1、2
步骤3、childrenSpan集合：1、2、3
步骤4、childrenSpan集合：1、2、3、4、
步骤5、childrenSpan集合：1、2、3、4、5
步骤6、childrenSpan集合：1、2、3、4、5、6

拉勾教育

— 互联网人实战大学 —

Next：第30讲《server-alarm 插件核心剖析，如何避免收到告警信息》

拉勾教育

—互联网人实战大学—

关注拉勾「教育公众号」
获取更多课程信息

L / A / G / O / U