### 쿼리 명령어 구현

이 절에서는 [SampleDriverQueryCommand.java](https://github.com/logpresso/logpresso-app-examples/blob/main/logpresso-sample-app/src/main/java/com/logpresso/sonar/sample/query/SampleDriverQueryCommand.java) 코드와 [SampleSubnetGroupsCommand.java](https://github.com/logpresso/logpresso-app-examples/blob/main/logpresso-sample-app/src/main/java/com/logpresso/sonar/sample/query/SampleSubnetGroupsCommand.java) 코드를 통해 쿼리 명령어를 구현하는 방법을 설명합니다.

#### 드라이버 커맨드

SampleDriverQueryCommand 클래스는 DriverQueryCommand 클래스를 상속하고, FieldOrdering 인터페이스를 구현합니다.

public abstract class SampleDriverQueryCommand extends DriverQueryCommand implements FieldOrdering { protected SampleParams params; public SampleDriverQueryCommand(SampleParams params) { this.params = params; } protected abstract void run(ConnectProfile profile) throws IOException; @Override public void run() { for (ConnectProfile profile : params.getProfiles()) { if (isCancelRequested()) return; try { run(profile); } catch (Throwable t) { String msg = String.format("%s (profile %s) error - %s", getName(), profile.getName(), t.getMessage()); throw new IllegalStateException(msg, t); } } }

쿼리 명령어 중에서 위와 같이 run()을 구현하여 스레드를 직접 실행하는 명령어를 드라이버 커맨드라고 지칭합니다. 하나의 쿼리에서 2개 이상의 드라이버 커맨드가 존재할 수 없습니다. 드라이버 커맨드는 가장 앞에 배치되고, 파이프로 이어지는 다른 명령어는 드라이버 커맨드의 스레드에 의해 onPush()가 호출되면서 실행됩니다. join 혹은 union 등 서브 쿼리에서 드라이버 커맨드를 추가로 사용할 수 있습니다.

쿼리 명령어는 취소 구현을 누락하지 않도록 주의해야 합니다. 루프(Loop)에서는 isCancelRequested()를 호출하여 현재 사용자나 시스템에 의해 취소가 요청된 상태인지 지속적으로 확인하고, 취소된 경우에는 즉시 수행을 중단해야 합니다.

생성자의 매개변수로 SampleParams 객체가 전달되는데, 이는 ConnectProfileParams를 상속한 클래스이기 때문에 접속 프로파일 객체 목록을 가져올 수 있습니다.

public String toString() { String s = getName(); if (params.getName() != null) s += Strings.doubleQuote(params.getName()); return s;}

모든 쿼리 명령어(QueryCommand)는 반드시 toString()을 구현해야 합니다. 쿼리 명령어의 문자열 표현은 쿼리 파서가 다시 파싱할 수 있는 정규화된 형태로 쿼리 명령어 구문을 반환해야 합니다. 이는 분산 쿼리 실행 계획을 생성하는 단계에서 쿼리 명령어 객체를 다시 쿼리 문자열로 변환하는 과정이 존재하기 때문입니다. 이 구현을 누락하면 실행 계획이 정상적으로 표시되지 않고, 분산 쿼리가 실패할 수 있습니다.

#### 쿼리 명령어의 구현

SampleSubnetGroupsCommand는 SampleDriverQueryCommand를 상속하고 아래의 메소드를 구현합니다:

* getName(): 쿼리 명령어의 이름
* getFieldOrder(): 필드 표시 순서. 쿼리 결과에 필드가 존재하지 않는 경우에는 해당 필드가 무시됩니다. 즉, 여기에서는 출력될 수 있는 모든 필드 이름 집합에 대하여 완전한 필드 순서를 정의하면 됩니다. 화면에서는 쿼리 명령어의 순서 상 FieldOrdering 인터페이스를 구현하는 마지막 명령어의 필드 출력 순서가 적용됩니다.
* run(): 드라이버 커맨드의 스레드가 실행할 기능을 구현합니다.

예제 코드는 RestApiClient 클래스를 이용하여 REST API를 호출하고, pushPipe() 메소드를 호출하여 명령어의 결과를 출력합니다. pushPipe()의 매개변수는 Row 클래스를 사용하는데, 이는 Map 인터페이스의 구현이므로 가변적인 키/값 쌍을 전달할 수 있습니다. 단, 값에는 아래의 타입만 허용됩니다:

|  |  |  |
| --- | --- | --- |
| 자바 타입 | 로그프레소 타입 | 설명 |
| java.lang.String | string | 문자열 |
| java.lang.Short | short | 16비트 정수 |
| java.lang.Integer | int | 32비트 정수 |
| java.lang.Long | long | 64비트 정수 |
| java.lang.Float | float | 32비트 실수 |
| java.lang.Double | double | 64비트 실수 |
| java.lang.Boolean | bool | 불리언 |
| java.util.Date | date | 날짜 및 시각 |
| java.net.Inet4Address | ipv4 | IPv4 주소 |
| java.net.Inet6Address | ipv6 | IPv6 주소 |
| java.util.List | list | 리스트 |
| java.util.Map | map | 맵. 키는 문자열 타입만 허용됨 |
| byte | binary | 바이트 배열 |

1초에 수십 만 건 이상의 처리 성능이 요구되는 경우에는 pushPipe(VectorizedRowBatch) 메소드를 호출하여 데이터를 출력하는 것을 권장합니다. VectorizedRowBatch의 값에는 아래의 벡터 타입을 사용할 수 있습니다.

|  |  |  |
| --- | --- | --- |
| 벡터 타입 | 로그프레소 타입 | 설명 |
| org.araqne.log.api.ObjectVector | object | 문자열 등 모든 타입 허용 |
| org.araqne.log.api.IntVector | int | 32비트 정수 배열을 최적화 처리 |
| org.araqne.log.api.LongVector | long | 64비트 정수 배열을 최적화 처리 |
| org.araqne.log.api.DoubleVector | double | 64비트 실수 배열을 최적화 처리 |
| org.araqne.log.api.BooleanVector | bool | 불리언 배열을 최적화 처리 |

약 1000건 단위로 레코드를 벡터화하여 배치 처리하면 pushPipe() 함수 호출 오버헤드가 1000배 감소하고, 잠금 경쟁으로 인한 성능 저하가 최소화되며, 가비지 컬렉션이 감소하므로 높은 성능을 구현할 수 있습니다.