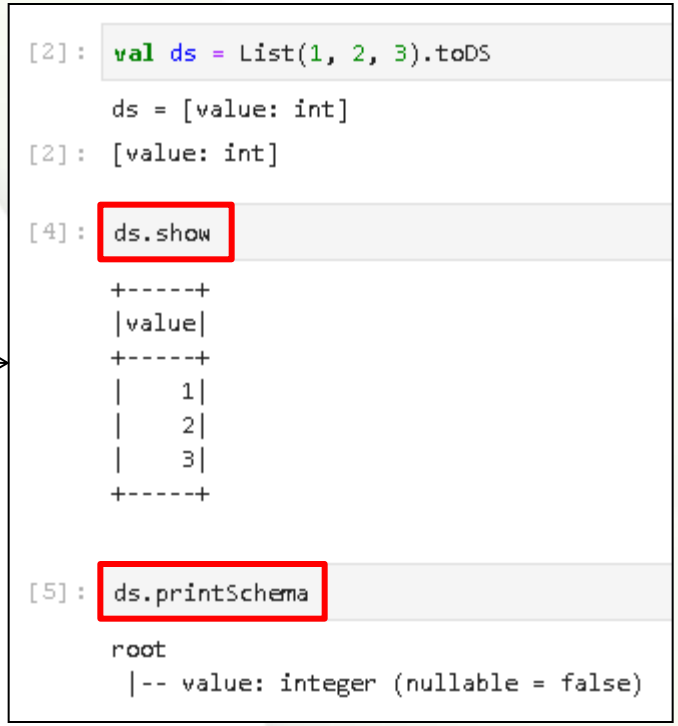
DataSet & DataFrame with SparkSQL

Spark SQL

- RDD의 문제
  - 값은 표현하지만 메타데이터(스키마)에 대한 표현 방법이 없음

- Spark SQL은 RDD의 단점을 보완할 수 있도록 또 다른 유형의 데이터 모델과 API를 제공하는 스파크 모듈

- 데이터를 다루는 방법
  - SQL 사용 (ANSI-SQL, Hive-QL 문법 지원)
  - DataSet 또는 DataFrame 사용



DataFrame & DataSet

- 스파크 1.3 버전에서 처음 DataFrame 소개
  - RDD와 다른 형태의 SQL과 유사한 연산 제공
  - 풍부한 API와 옵티마이저 기반의 높은 성능 제공
  - 작업의 특성에 따라 RDD에 비해 코드가 복잡해 지는 문제 노출
  - 컴파일 타임 오류 체크 기능 미지원

- 스파크 1.6 버전에서 처음 DataSet 소개
  - DataFrame의 장점을 유지하면서 컴파일 타임 오류 체크 등의 기능을 보완
  - DataFrame에 RDD의 장점을 통합

연산 구분

- 액션 연산 VS 트랜스포메이션 연산
  - 액션 → 실제 데이터 처리 수행 및 결과 생성
  - 트랜스포메이션 → 새로운 데이터셋 생성
  - 트랜스포메이션 연산은 액션 연산이 호출될 때까지 수행되지 않음

- 타입연산 vs 비타입연산
  - 비타입 연산 → 실제 데이터 타입이 아닌 Row와 Column 타입의 객체로 감싸서 처리하는 연산 (DataFrame의 연산)
  - 타입 연산 → 실제 데이터 타입에 따라 연산 (DataSet의 연산)

데이터 생성

- 외부 데이터소스로부터 데이터프레임 생성
  - 파일, 데이터베이스 등의 외부 저장소의 데이터 사용
  - read 함수 또는 각 데이터 포맷별 전용 함수 사용

```
val df = spark.read
               .format("json")
               .option("allowComments", "true")
               .load("file:///home/sparkdev/work/data-files/github-employees.json")
```

```
df.show
```

```
+--------------+
|      username|
+--------------+
|AbrahamSalloum|
|      aclindsa|
|  adamschwartz|
|       ahsojar|
|      AiMadobe|
+--------------+
only showing top 5 rows
```

데이터 생성

- 기존 RDD 및 로컬 컬렉션으로부터 데이터프레임 생성

- 리플렉션을 통해 데이터의 스키마를 자동으로 추론하거나,

```
case class Person(name: String, age: Int, job: String)

val row1 = Person("John Doe", 25, "Engineer")
val row2 = Person("Jane Doe", 45, "Sales Manager")
val row3 = Person("Smith Doe", 27, "Clerk")
val row4 = Person("Alice Doe", 43, "Sales Manager")
val row5 = Person("Anderson Doe", 37, "Developer")
val row6 = Person("Risa Doe", 29, "Clerk")

val data = List(row1, row2, row3, row4, row5, row6)
val df2 = spark.createDataFrame(data)
```

```
df2.show
```

```
+------------+---+-------------+
|        name|age|          job|
+------------+---+-------------+
|    John Doe| 25|     Engineer|
|    Jane Doe| 45|Sales Manager|
|   Smith Doe| 27|        Clerk|
|   Alice Doe| 43|Sales Manager|
|Anderson Doe| 37|    Developer|
+------------+---+-------------+
only showing top 5 rows
```

데이터 생성

- 개발자가 직접 명시적인 스키마 정보를 코드로 작성해서 지정

```
import org.apache.spark.sql.Row
import org.apache.spark.sql.types._
```

```
val sf1 = StructField("name", StringType, nullable = true)
val sf2 = StructField("age", IntegerType, nullable = true)
val sf3 = StructField("job", StringType, nullable = true)
val schema = StructType(List(sf1, sf2, sf3))
val rows = sc.parallelize(List(Row("John Doe", 25, "Engineer"),
                               Row("Jane Doe", 45, "Sales Manager"),
                               Row("Smith Doe", 27, "Clerk"),
                               Row("Alice Doe", 43, "Sales Manager"),
                               Row("Anderson Doe", 37, "Developer"),
                               Row("Risa Doe", 29, "Clerk")))
val df3 = spark.createDataFrame(rows, schema)
```

```
df3.show
```

```
+------------+---+-------------+
|        name|age|          job|
+------------+---+-------------+
|    John Doe| 25|     Engineer|
|    Jane Doe| 45|Sales Manager|
|   Smith Doe| 27|        Clerk|
|   Alice Doe| 43|Sales Manager|
|Anderson Doe| 37|    Developer|
+------------+---+-------------+
only showing top 5 rows
```

DataFrame과 DataSet 구분

- DataFrame은 org.apache.spark.sql.Row 타입의 객체로 구성된 DataSet

- DataSet은 DataSet[T]로 선언
  - DataSet[Int], DataSet[String] 등은 일반적인 DataSet으로 분류
  - DataSet[Row]에 대해 DataFrame으로 분류

- 일반적인 DataSet과 DataFrame은 사용할 수 있는 트랜스포메이션 연산에 차이가 있음

주요 연산

연산	설명
show	저장된 데이터를 화면에 출력
head, first	첫 번째 행의 데이터 반환
take	처음 n개 행의 데이터 반환
count	전체 데이터 개수 반환
collect, collectAsList	모든 행의 데이터를 배열 또는 리스트로 반환
describe	숫자형 컬럼에 대해 평균, 표준편차, 최소값, 최대값 등 기초 통계량 정보 반환
cache, persist	작업 중인 데이터를 메모리 등에 저장. persist는 저장 방법 선택 가능
printSchema,columns, dtyps, schema	스키마 관련 정보 출력 또는 반환
createOrReplaceTempView	데이터프레임을 테이블로 변환. 변환 후에는 SQL을 사용해서 데이터 처리 가능
explain	실행 계획 정보 출력

주요 연산

연산	설명
col("컬럼명") $"컬럼명" '컬럼명	컬럼 객체를 반환하는 연산자 및 표현식
!==, ===	컬럼 비교 연산자
alias, as	컬럼 또는 데이터프레임에 별칭 부여
isin	컬럼의 값이 전달인자 값에 포함되어 있는지 여부 확인
when, otherwise	if ~ else와 같은 분기 처리
max, mean, sum, round, sqrt, ...	다양한 집합, 수학 연산
collect_list, collect_set	컬럼의 값을 모아서 리스트 또는 셋 형식의 컬럼 반환
count, countDistinct	컬럼에 속한 데이터의 개수 반환 (전체 또는 중복제거)
grouping, grouping_id	cube의 수행 결과에 각 결과값에 그룹화 수준 계산
array_contains	배열 형식의 컬럼에서 특정 값의 포함 여부 확인
size	배열 형식의 컬럼에서 배열의 크기 (요소의 개수) 반환
sort_array	배열 형식의 컬럼에서 배열 요소 정렬

주요 연산

연산	설명
explode, posexplode	배열 컬럼에 포함된 요소를 여러 개의 행으로 변환
current_date, unix_timestapm, to_date	현재 시간, yyyy-mm-dd HH:mm:ss 형식의 문자열 → 날짜 형식 yyyy-mm-dd 형식의 문자열 → 날짜 형식
add_months, data_add, last_day	월단위 연산 일단위 연산 월의 마지막 날짜
window	주로 날짜 형식에 대해 일정 크기의 윈도우(범위, 급간)를 만들어서 집계 연산에 사용
array	여러 컬럼의 값을 하나의 배열로 변환
desc, asc	sort와 함께 사용되며 정렬의 방향 지정
desc_nulls_first, desc_nulls_last, asc_nulls_first, asc_nulls_last	정렬 방향 함께 null 데이터의 위치 지정
split, length	문자열 컬럼의 값을 분할하거나 문자 개수 계산

주요 연산

연산	설명
rownum, rank	전체 데이터를 몇 개의 윈도우로 구분하고 윈도우 내에서 행번호, 순위 계산
udf	컬럼 처리를 위한 사용자 정의 함수 등록
select, drop	특정 컬럼만 포함 또는 제외한 새로운 데이터프레임 반환
filter, where	데이터 선택 조건 지정
agg	컬럼에 대해 sum, max 등의 집합 연산 수행
groupBy	특정 컬럼의 값을 기준으로 그룹화
groupByKey	특정 함수의 처리 결과를 기준으로 그룹화
distinct	행 전체 값을 대상으로 중복된 값을 제외한 결과 반환
dropDuplicates	지정된 컬럼에서 중복된 값을 제외한 결과 반환
intersect	두 데이터에 모두 속하는 행으로 구성된 데이터 반환
except	앞 데이터에서 뒤의 데이터에 포함되지 않는 데이터 반환
join	지정된 컬럼의 값을 기준으로 두 데이터 결합 inner, outer, full 등의 join 방식 지정
crossJoin	두 데이터의 카르테시안 곱의 결과 반환

주요 연산

연산	설명
na	결측치를 다루는 DataFrameNaFunctions 객체 반환 na().drop, na().fill, na().replace 등으로 처리
orderBy	지정된 컬럼을 기준으로 데이터 정렬
stat	특정 컬럼값에 대해 자주 사용되는 통계수치 제공 stat.corr, stat.cov, stat.crosstab 등으로 처리
withColumn	새 컬럼 추가
withColumnRenamed	기존 컬럼의 이름 변경
write	로컬파일시스템, HDFS, 데이터베이스 등의 외부 저장소에 데이터를 저장할 수 있는 DataFrameWriter 반환 df.write.format("csv").save("path") 등의 형식으로 사용
unionByName	위치가 아닌 컬럼 이름을 기준으로 두 개의 데이터 병합
to_json, from_json	데이터와 json 형식 사이의 변환