# Week3,4 요약

| 생성일 | @2022년 2월 8일 오후 2:09 |
| --- | --- |
| 태그 | |

## 스파크 SQL

- 스파크 이전에 Hive를 통해 빅데이터에 SQL을 적용하였으나, 스파크를 통해서도 가능하게 되었다.
- 수백개의 Hive job으로 구성된 파이프라인을 단일 Spark Job으로 대체하였다.
- Spark SQL은 Online Transaction Processing(OLTP) 데이터베이스가 아닌 Online Analytic Processing(OLAP)로 동작
- Spark SQL은 Hive 메타스토어를 사용하여 조회할 파일 수를 최소화하고 메타데이터를 참조한다.

### 카탈로그

- 스파크 SQL에서 가장 높은 추상화 단계
- 테이블에 저장된 데이터에 대한 메타데이터 뿐만 아니라 데이터베이스, 테이블, 함수 그리고 뷰에 대한 정보를 추상화
- 스파크 SQL을 사용하는 또 다른 방식의 인터페이스

### 테이블

- 테이블은 명령을 실행한 데이터 구조라는 점에서 DataFrame과 논리적으로 동일하나 테이블은 항상 데이터를 가지고 있다.
- 임시 테이블의 개념이 없으며 데이터를 가지지 않은 뷰만 존재한다. (테이블을 제거하면 모든 데이터가 제거)
- 테이블은 1) 테이블의 데이터와 2)테이블에 대한 데이터, 즉 메타데이터를 저장
- 스파크는 데이터 뿐만 아니라 파일에 대한 메타 데이터도 관리할 수 있음 (`saveAsTable`)
- 스파크는 외부테이블의 메타데이터를 관리하지만 데이터 파일은 관리하지 ㅇ않는다.
- 테이블을 다루는 문법은 SQL과 동일

### 뷰

- 뷰는 기존 테이블에 여러 트랜스포메이션 작업을 지정하는 단순 쿼리 실행 계획
- 뷰를 사용하면 쿼리 로직을 체계화하거나 재사용하기 편하게 만들 수 있음

### 기타 기능

- 스파크 SQL에는 구조체, 리스트, 맵 세가지 핵심 복합 데이터 타입이 존재
- 스파크 SQL은 다양한 고급 함수를 제공하고 사용자가 직접 정의할 수도 있음
- 서브쿼리를  사용하여 쿼리 안에 쿼리를 지정할 수도 있음.
    - 스파크에는 상호연관 서브쿼리(서브쿼리의 정보를 보완하기 위해 쿼리의 외부 범위에 있는 일부 정보를 사용)와 비상호연관 서프쿼리(외부 범위에 있는 정보 미사용)가 있음.
- 스파크는 값에 따라 필터링할 수 있는 조권절 서브쿼리도 지원

## Dataset

- Dataset은 구조적 API의 기본 데이터 타입.
- Dataset은 자바 가상 머신을 사용하는 언어인 스칼라와 자바에서만 사용 가능
- 모든 DataFrame은 Row타입의 Dataset을 의미

- Dataset API를 사용한다면 스파크는 데이터셋에 접근할 때마다 Row 포맷이 아닌 사용자 정의 데이터타입으로 변환

### Dataset을 사용해야 하는 이유

- DataFrame 기능만으로는 수행할 연산을 표현할 수 없는 경우
  - 복잡한 비즈니스 로직을 SQL이나 DataFrame 대신 단일 함수로 인코딩해야 하는 경우
- 성능 저하를 감수하더라도 타입 안정성을 가진 데이터 타입을 사용하고 싶은 경우
  - 데이터타입이 유효하지 않은 작업에서 데이터를 제어하고 구조화 가능
- 단일 노드의 워크로드와 스파크 워크로드에서 전체 로우에 대한 다양한 트랜스포메이션을 재사용할 경우

<핵심> 성능과 타입 안정성 중 하나는 반드시 희생할 수 밖에 없음.

→ 대량의 DataFrame기반의 ETL 트랜스포메이션의 첫번째나 마지막 단계에서 사용!

## 저수준 API

- 스파크에는 두 종류의 저수준 API가 있다.
  - 분산 데이터 처리를 위한 RDD
  - 브로드캐스트 변수와 어큐뮬레이터처럼 분산형 공유 변수를 배포하고 다루기 위한 API
- 스파크를 잘 알고 있는 숙련된 개발자라 하더라도 구조적 API 위주로 사용하는 것이 좋으나 저수준 API가 필요할 수도 있음.
  - 고수준 API에서 제공하지 않는 기능이 필요한 경우 ex) 클러스터의 물리적 데이터의 배치를 아주 세밀하게 제어해야 하는 상황
  - RDD를 사용해 개발된 기존 코드를 유지해야 하는 경우
  - 사용자가 정의한 공유 변수를 다뤄야 하는 경우
  - 이전 버전의 스파크에서 자체 구현한 파티셔너를 사용하는 경우
  - 데이터 파이프라인이 실행되는 동안 변숫값을 갱신하고 추적해야 하는 경우
- 스파크 클러스터에서 연산을 수행하는 데 필요한 도구인 `SparkSession`을 이용해 SparkContext에 접근할 수 있음.

### RDD

- RDD는 불변성을 가지며 병렬로 처리할 수 있는 파티셔닝 된 레코드
- RDD의 모든 코드는 객체이므로 완벽하게 제어할 수 있으나 모든 값들을 다루거나 값 사이의 상호작용 과정을 반드시 수동을 정의해야 하고 이로 인해 최적화를 하기 위한 훨씬 많은 작업이 필요
- RDD는 다음 다섯 가지 주요 속성으로 구분
  - 파티션의 목록
  - 각 조각을 연산하는 함수
  - 다른 RDD와의 의존성 목록
  - 부가적으로 키-값 RDD를 위한 파티셔너
  - 부가적으로 각 조각을 연산하기 위한 기본 위치 목록
- RDD는 크게 제네릭 RDD타입과 키 기반의 집계가 가능한 키-값 RDD타입이 있다.
- RDD는 트랜스포메이션과 액션을 제공하나 Row라는 개념이 없음.
  - 트랜스포메이션: distinct, filter, map, flatMap, sortBy, randomSplit
  - 액션: reduce, count (countApprox, countApproxDistinct, countByValue, countByValueApprox), first, max, min, take
- RDD 역시 메모리에 있는 데이터만을 대상으로 캐시하거나 저장할 수 있음.

- 체크포인팅: 나중에 저장된 RDD를 참조할 때는 원본 데이터소스를 다시 계싼해 RDD를 생성하지 않고 디스크에 저장된 중간 결과 파티션을 참조 (반복적인 연산 수행 시 매우 유용)
- RDD에는 데이터를 키-값 형태로 다룰 수 있는 다양한 메서드가 있다.
  - keyBy, lookup, sampleByKey, flatMap 등
  - countByKey, groupByKey, reduceByKey, aggregate, aggregateByKey, combineByKey, foldByKey
  - cogroup, join, coalesce, repartition 등등

## 분산형 공유변수

- 분산형 공유변수는 클러스터에서 실행할 때 특별한 속성을 가진 사용자 정의 함수에서 주로 사용
- 분산형 공유변수는 디버깅이나 최적화 작업하기에 매우 유용함
- 어커뮬레이터: 모든 태스크의 데이터를 공유 결과에 추가할 수 있음
- 브로드캐스트 변수: 모든 워커 노드에 큰 값을 저장하여 재전송 없이 많은 스파크 액션에서 재사용 가능