EBOOK

데이터 엔지니어링 Big Book

기술 블로그 컬렉션 - 코드 샘플, 노트북 포함

databricks

목차

databricks

섹션 01

Databricks 기반 데이터 엔지니어링 소개

databricks

요즘 기업에서는 데이터가 다양한 비즈니스 관련 이니셔티브에 전략적 자산으로서 어떤 역할을 하는지 자각하고 있습니다. 예를 들어 매출 신장, 고객 경험 개선, 효율적인 운영이나 제품 또는 서비스 개선 등 다방면을 아우릅니다. 그런데 이러한 이니셔티브에 맞춰 데이터에 액세스하고 데이터를 관리하기는 점점 더 복잡해지는 상황입니다. 대부분의 경우, 데이터 볼륨과 데이터 유형이 폭증하면서 그에 따라 복잡성도 악화되었으며 기업에서 수집하는 데이터는 전체의 약 80%가 비정형, 반정형 형식인 것으로 추산됩니다. 데이터 컬렉션은 계속 늘어나므로, 전체 데이터의 73%는 분석이나 의사 결정용으로 사용되지 못하고 맙니다. 데이터 엔지니어링 팀에서는 이런 비율을 낮추고 더 많은 데이터를 활용하기 위해 데이터를 효율적이고 안정적으로 제공할 수 있도록 데이터 파이프라인을 구축해야 할 책임을 맡고 있습니다. 다만 이처럼 복잡한 데이터 파이프라인을 구축하는 과정에는 다양한 어려움이 뒤따릅니다.

- 데이터 레이크로 데이터를 가져오려면 데이터 엔지니어가 반복적인 데이터 수집 태스크를 핸드 코딩하는 데 엄청난 시간을 할애해야 함
- 데이터 플랫폼은 계속 바뀌기 때문에 데이터 엔지니어가 복잡한 확장형 인프라를 구축, 유지 관리했다가 또다시 구축하는 데 시간을 들여야 함
- 실시간 데이터의 중요성이 강화되면서 지연이 짧은 데이터 파이프라인이 필요해지며, 이는 구축과 유지 관리가 더욱 난해함
- 마지막으로, 파이프라인을 모두 개발한 뒤에도 데이터 엔지니어는 SLA에 부합하기 위해 끊임없이 성능에 중점을 두고 파이프라인과 아키텍처를 조율해야 함

Databricks를 이용하면 어떤 면에서 유리할까요?

데이터 엔지니어가 Databricks 레이크하우스 플랫폼을 이용하면 데이터 수집, 변환, 처리, 예약, 제공 등에 엔드 투 엔드 데이터 엔지니어링 솔루션을 활용할 수 있습니다. 레이크하우스 플랫폼이 파이프라인을 구축, 유지 관리하고 데이터 레이크에서 직접 ETL 워크로드를 실행하는 복잡한 작업을 자동화해주므로 데이터 엔지니어는 가치 있는 인사이트를 도출할 품질과 안정성에만 집중하면 됩니다.

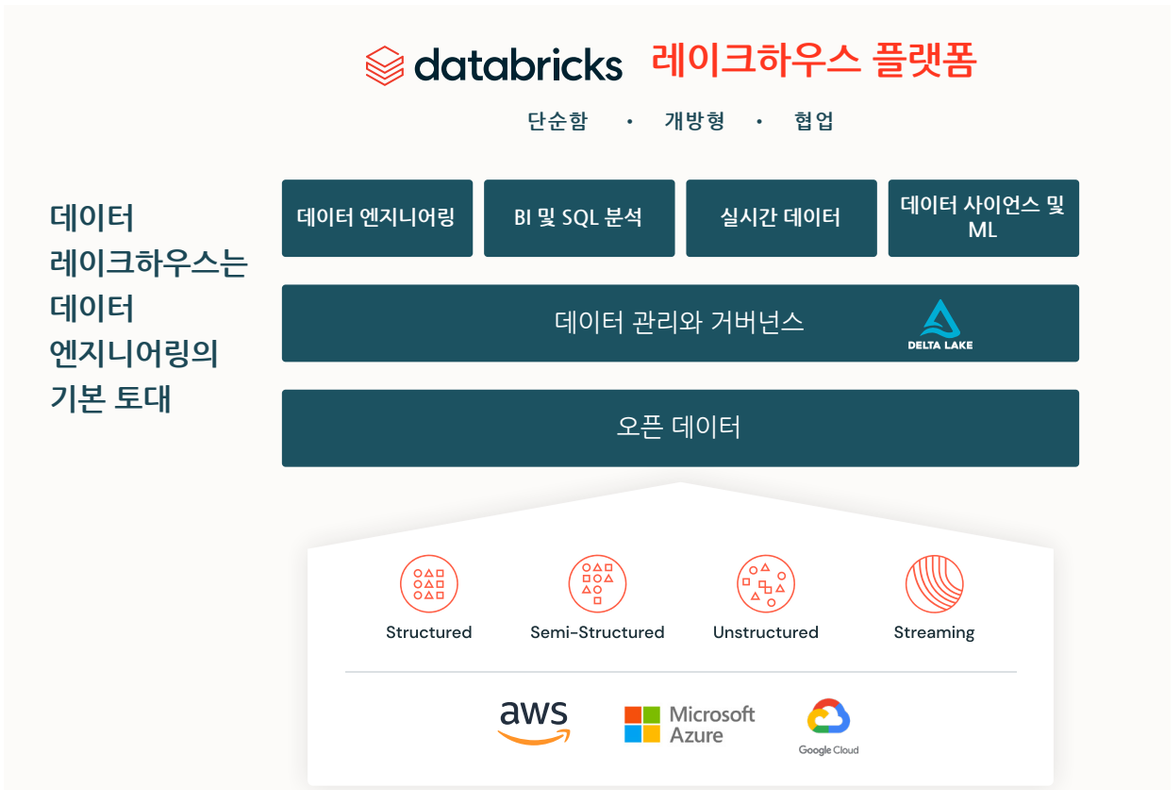

그림 1
Databricks 레이크하우스 플랫폼은 데이터, 분석과 AI를 하나의 공용 플랫폼에 통합해 모든 데이터 사용 사례에 활용하도록 지원

순조로운 데이터 엔지니어링을 위한 Databricks만의 주요 차별화 요소

레이크하우스 아키텍처를 간소화함으로써 데이터 엔지니어는 데이터 파이프라인을 구축할 때 엔터프라이즈급, 기업에서 바로 사용 가능한 방식으로 접근해야 합니다. 이를 위해 데이터 엔지니어링 솔루션 팀에서는 다음과 같은 8가지 주요 차별화 기능을 도입해야만 합니다.

지속적, 또는 예약된 데이터 수집

자동 진화(auto-evolving) 스키마를 사용해 페타바이트급 데이터를 수집할 수 있으므로 데이터 엔지니어가 분석, 데이터 사이언스나 머신 러닝용으로 사용할 빠르고 안정적이며 확장 가능한 자동 데이터를 제공할 수 있습니다. 예를 들면 다음과 같습니다.

- 파일이나 스트리밍 소스(Kafka, DBMS 및 NoSQL 등)에서 데이터가 입수되는 대로, 점차적이고 효율적인 형태로 처리
- 정형, 비정형 데이터 형식에 대해 스키마를 자동으로 추론하고 열 변경 탐지
- 데이터가 도착하는 대로 자동으로, 효율적으로 추적하며 사람이 개입할 필요 없음
- 데이터 열을 복구하여 데이터 손실 예방

선언적 ETL 파이프라인

데이터 엔지니어가 개발에 투자하는 시간과 수고가 절약되므로, 그 대신 데이터 파이프라인 내에서 SQL이나 Python을 사용해 비즈니스 로직과 데이터 품질 검사를 구현하는 데 주력할 수 있습니다. 이를 달성하는 방법은 다음과 같습니다.

- 인텐트 중심 선언적 개발을 사용하여 “방법”을 단순화하고 해결하고자 하는 “대상”을 정의
- 자동으로 양질의 데이터 계보를 생성하고 데이터 파이프라인 전체에서 테이블 종속성 관리
- 누락된 종속성이나 구문 오류를 자동으로 확인하고 데이터 파이프라인 복구 관리

데이터 품질 검증 및 모니터링

데이터 레이크하우스 전체에서 데이터 안정성이 개선되므로 데이터 팀이 다운스트림 이니셔티브에 사용할 정보를 신뢰할 수 있게 해줍니다. 그 방법은 다음과 같습니다.

- 정해진 데이터 기대치를 사용해 파이프라인 내에서 데이터 품질과 무결성 관리 정의
- 미리 정의된 정책을 사용하여 데이터 품질 오류 해결(실패, 삭제, 알림, 격리)
- 데이터 파이프라인 전체에서 캡처, 추적 및 보고된 데이터 품질 지표 활용

내결함성 및 자동 복구

일시적 오류를 처리하고, 파이프라인 운영 중 발생하는 가장 보편적인 오류 상태로부터 복구합니다. 이때 다음과 같이 빠르고 확장 가능한 자동 복구 기능을 사용합니다.

- 내결함성 메커니즘을 사용하여 데이터 상태를 일관성 있게 복구
- 체크포인팅으로 소스에서 진행 상태를 자동으로 추적하는 기능
- 데이터 파이프라인 상태를 자동으로 복구 및 복원하는 기능

데이터 파이프라인 가시성

데이터플로우 그래프 대시보드에서 데이터 파이프라인의 전반적인 상태를 모니터링하고 파이프라인 전체의 상태를 시각적으로 추적하여 성능, 품질과 지연을 살핍니다. 데이터 파이프라인 가시성 기능의 예를 들면 다음과 같습니다.

- 양질의 충실도 높은 계보 다이어그램을 통해 데이터 플로우 방식을 파악하여 영향 분석에 사용
- 행 레벨에서 데이터 파이프라인의 성능과 상태를 세분화하여 로깅
- 데이터 파이프라인 작업을 지속적으로 모니터링하여 지속적 운영 보장

배치 및 스트림 데이터 처리

데이터 엔지니어가 비용 관리를 통해 데이터 지연을 조정할 수 있게 해줍니다. 복잡한 스트림 처리는 몰라도 되고, 복구 로직을 구현할 필요도 없습니다.

- 자동 프로비저닝된 탄력적 Apache Spark™ 기반 컴퓨팅 클러스터에서 데이터 파이프라인 워크로드를 실행하여 확장 및 성능 보장
- 작업을 병렬 처리하고 데이터 이동을 최소화하는 성능 최적화 클러스터 이용

자동 배포 및 운영

손쉬운 자동 데이터 파이프라인 배포와 롤백을 지원하므로 다운타임이 최소화되어 분석과 머신 러닝 사용 사례에 활용할 데이터를 안정적이고 예측 가능한 형태로 제공하도록 보장합니다. 여기에는 다음과 같은 장점이 있습니다.

- 완전하고 매개변수화된 자동 배포를 통해 지속적인 데이터 제공 보장
- 각종 주요 클라우드 공급업체 전체에서 데이터 파이프라인 배포 전체 오케스트레이션, 테스트, 모니터링 제공

예약 파이프라인 및 워크플로우

데이터 처리 태스크를 단순하고 명확하며 안정적인 형태로 오케스트레이션하여 데이터 및 머신 러닝 파이프라인에 적합하며, 여러 개의 비 인터랙티브(non-interactive) 작업을 Databricks 컴퓨팅 클러스터에서 DAG(Directed Acyclic Graph) 형태로 실행하는 기능을 제공합니다.

- Databricks UI와 API를 사용하여 DAG에서 간편하게 태스크 오케스트레이션
- UI 또는 API 및 다양한 기능(예: 모니터링 관련 이메일 알림)을 통해 작업 내에 여러 개의 태스크를 만들어 관리
- API를 포함한 모든 태스크를 Databricks 외부에서 및 각종 클라우드(종류 무관)에서 오케스트레이션

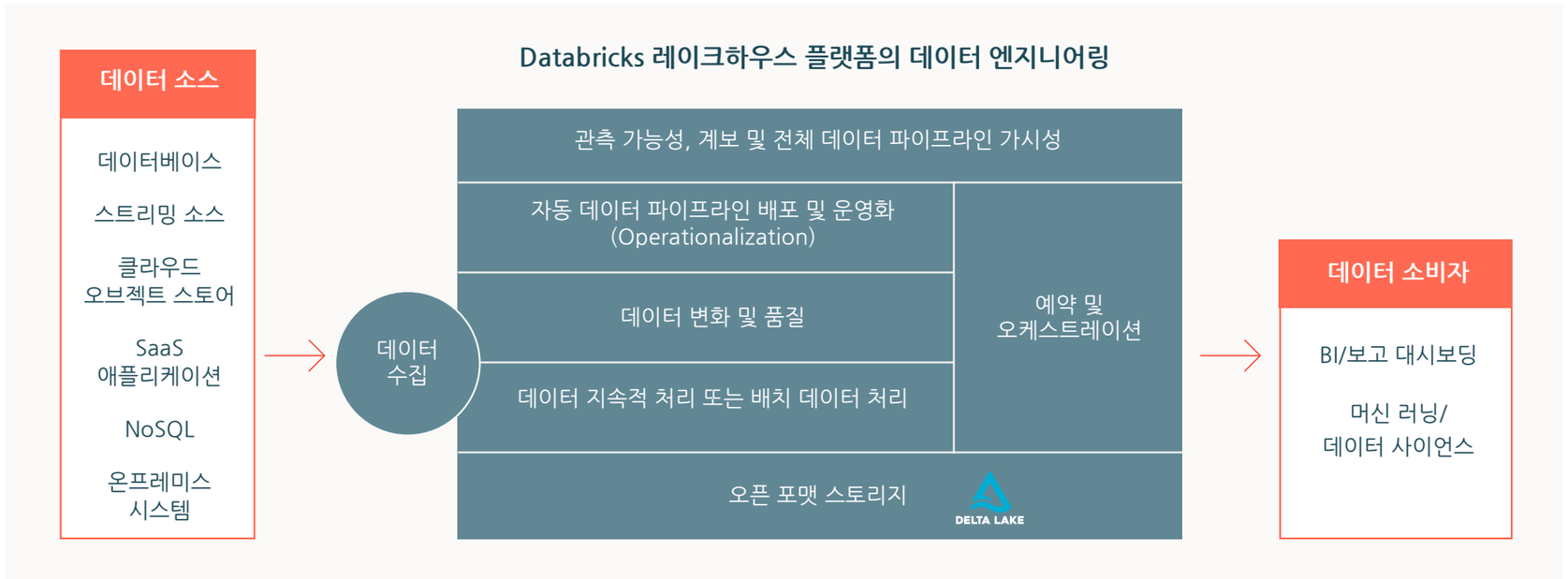

그림 2
Databricks 참조 아키텍처의 데이터 엔지니어링

결론

데이터 기반을 추구하는 기업이 늘어나면서, 데이터 엔지니어링이 성공을 위한 관심 활동의 초점으로 떠올랐습니다. 안정적이고 믿을 수 있는 데이터를 제공하려면 데이터 엔지니어가 수작업으로 전체 ETL 수명 주기를 개발하고 유지 관리하는 데 시간을 할애할 필요가 없어야 합니다. 데이터 엔지니어링 팀에는 효율적이고 확장 가능한 방식으로 ETL 개발을 간소화하고, 데이터 안정성을 개선하며 운영을 관리할 방안이 필요합니다.

앞서 설명한 8가지 주요 차별화 기능은 각종 데이터 종속성을 자동화, 유지 관리하고 모니터링에 기본 내장된 품질 관리 기능을 활용하며 자동 복구를 통해 파이프라인 운영에 가시성을 제공하여 ETL 수명 주기 관리를 간소화해줍니다. 따라서 데이터 엔지니어링 팀은 안정적인 엔드 투 엔드 형태의, 프로덕션에 바로 투입할 수 있는 데이터 파이프라인을 손쉽고 신속하게 구축하는 데만 집중할 수 있습니다. 이 경우 SQL이나 Python만 사용하여 배치 및 스트리밍에 집중해 분석, 데이터 사이언스나 머신 러닝에 적합한 가치 있는 데이터를 창출하게 됩니다.

사용 사례

다음 섹션에서는 실제 상황에서 도출한 데이터 엔지니어링의 엔드 투 엔드 사용 사례를 활용하여 모범 사례를 소개합니다. 데이터 수집과 데이터 처리부터 분석과 머신 러닝에 이르기까지, 원시 데이터를 실행 가능한 데이터로 바꾸는 방법을 알려드리겠습니다. 데이터 세트와 코드 샘플을 준비해 두었습니다. 이것을 이용해 Databricks 레이크하우스 플랫폼에서 데이터 수명 주기의 모든 측면을 전면 탐색하며 실습해 보시기 바랍니다.

섹션 02

Databricks 레이크하우스 플랫폼의 실제 사용 사례

데이터 레이크하우스를 사용한 실시간 POS 분석

CrowdStrike Falcon 이벤트를 위한 사이버 보안 레이크하우스 구축

현대적 데이터 레이크하우스로 의료 데이터 가치 재발견

규제 보고서 전송의 적시성과 신뢰성

Databricks 레이크하우스 플랫폼을 사용한 대규모 AML 솔루션

실시간 AI 모델을 구축하여 게임 내 유해 행위 탐지

Northwestern Mutual(인사이트 플랫폼)에서 확장 가능한 오픈 레이크하우스 아키텍처를 통한 혁신

Databricks 데이터 팀이 3가지 클라우드와 50개 이상 지역에 레이크하우스를 구축한 방법

섹션 2.1

데이터 레이크하우스를 사용한 실시간 POS 분석

글: BRYAN SMITH 및 ROB SAKER

2021년 9월 9일

공급망 중단(상품 수급량 감소, 창고 용량 부족 등)이 발생한 데다, 설상가상으로 소비자 기대치가 급속히 변화하면서 원활한 옴니채널 경험을 당연히 기대하게 되자 소매업체에서도 운영을 위한 기존의 데이터 관리 방식을 다시 생각할 수밖에 없게 되었습니다. 팬데믹 이전만 해도, 리테일 기업 중 전체의 71%는 옴니채널 목표를 달성하는 데 가장 큰 장애물로 '인벤토리를 실시간으로 파악하기 어렵다'라는 점을 들었습니다. 팬데믹으로 인해 온라인과 오프라인 매장 내 경험을 통합해야 한다는 수요는 전보다 더 늘어났을 뿐이므로, 리테일 기업에는 정확한 제품 가용성을 제시하고 신속하게 주문 변경을 관리해야 한다는 압박이 전보다 더욱 가중되고 있습니다. 뉴 노멀 시대 소비자의 수요에 부합하려면 실시간 정보에 대한 액세스를 개선하는 것이 무엇보다 중요합니다.

이 블로그에서는 리테일 분야에서의 실시간 데이터 요구 사항, 그리고 데이터 레이크하우스를 사용해 대규모 POS 데이터의 실시간 스트리밍을 전송할 때 따르는 문제를 해결하는 법을 알아봅니다.

POS 시스템

POS 시스템은 오래전부터 매장 내 인프라의 중심 역할을 해왔습니다. 이 시스템은 리테일 업체와 고객 사이에 발생하는 상품과 서비스 거래를 기록하는 역할을 합니다. POS는 이러한 거래를 지속하기 위해 보통 제품 재고 수준을 추적하고, 단위 수량이 위험 수위 미만으로 떨어지지 않도록 보충하기도 합니다. 매장 내 운영에서 POS가 가지는 중요한 의미는 아무리 강조해도 지나치지 않습니다. 이는 판매 및 재고 운영을 위한 시스템이므로, 비즈니스 애널리스트는 무엇보다 이 시스템의 데이터에 액세스하는 것을 중시합니다.

지금까지는 각 매장과 기업 본사 사이의 연결에 한계가 있었기 때문에 POS 시스템(단말기 인터페이스만이 아니라)을 매장 내에 실물 형태로 배치해야 했습니다. 이런 시스템이 한가한 시간대에 본사 컴퓨터에 접속해 요약 데이터를 전송하면 이 데이터가 데이터 웨어하우스에서 통합됩니다. 이것으로 하루 전의 리테일 운영 실적을 조회할 수 있게 되는 것입니다. 그러나 이 데이터는 다음 날 밤 사이클이 시작되기까지 속절없이 오래된 정보가 되어갈 뿐입니다.

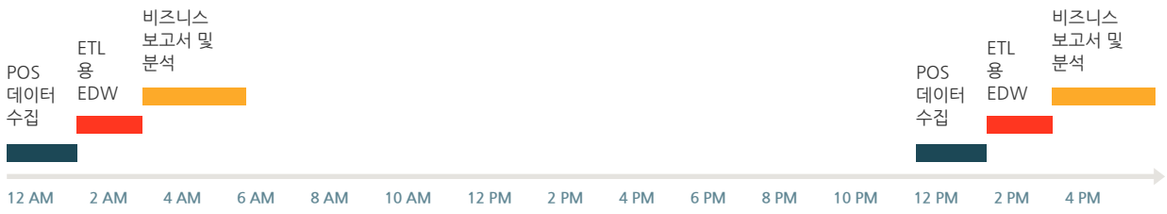

그림 1
기존의 배치 중심 ETL 패턴을 사용한 재고 가용성 파악

최근에는 연결성이 개선되어 중앙 집중형, 클라우드 기반 POS 시스템으로 전환하는 리테일 업체가 늘어났습니다. 그런가 하면 매장 내 시스템과 기업 백오피스를 연결하는 준 실시간(near real-time) 통합을 개발 중인 업체도 많습니다. 정보를 준 실시간으로 확보하면 리테일 업체에서 품목의 예상 가용성을 꾸준히 업데이트할 수 있습니다. 따라서 비즈니스 경영진이 하루 지난 재고 상태 데이터와 자신의 지식을 비교하는 방식을 버리고, 현재 재고 상태에 대한 지식을 바탕으로 행동을 취할 수 있게 됩니다.

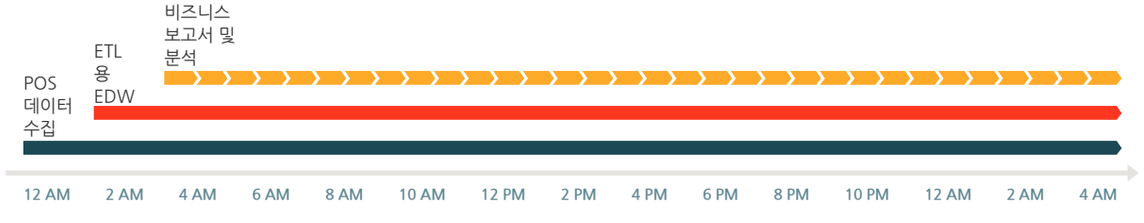

그림 2
스트리밍 ETL 패턴을 사용한 재고 가용성 파악

준 실시간 인사이트

준 실시간 인사이트가 매장 활동에 큰 파급력을 미치기는 하지만, 기존의 익일 프로세스에서 정보를 지속적으로 스트리밍하는 방식으로 전환하면 그 나름의 문제점이 뒤따릅니다. 이는 다른 종류의 데이터 처리 워크플로우를 고안해야 하는 데이터 엔지니어에게만 국한되는 것이 아니라, 정보 소비자에게도 발생하는 문제입니다. 이 글에서는 최근 이 과정을 시작한 여러 고객에게서 배운 점을 공유하고, 레이크하우스 패턴을 통해 이용할 수 있는 주요 패턴과 기능을 이용하면 어떤 면에서 유익한지 살펴봅니다.

요점 1

범위를 신중하게 고려

POS 시스템은 단순히 판매와 재고 관리에만 국한되는 것이 아니라 결제 처리, 매장 신용 관리, 청구, 발주, 회원 혜택 프로그램 관리, 직원 일정 관리, 시간 추적은 물론 급여대장까지 아우르는 다양한 기능을 제공할 수 있습니다. 매장 내 기능 면에서는 실로 만능 도구라고 해도 과언이 아닙니다.

따라서 POS 내부에 저장된 데이터는 보통 크고 복잡한 데이터베이스 구조에 널리 퍼져 있습니다. 운이 좋은 경우, POS 시스템에서 데이터 액세스 레이어를 제공합니다. 이 경우 이 데이터를 좀 더 해석하기 쉬운 구조를 통해 액세스할 수 있습니다. 하지만 그렇지 않은 경우라면, 데이터 엔지니어가 불투명한 일련의 테이블을 뒤져 가치 있는 데이터와 그렇지 않은 데이터를 판별해야 합니다.

데이터의 노출 방식과는 관계없이, 고전적인 지침이 불변의 진리인 법입니다. 즉 각자의 해법에 적당한 설득력 있는 비즈니스 사유를 파악한 다음 이것을 활용하여 처음에 사용하는 정보 자산의 범위를 제한하면 됩니다. 그와 같은 사유는 보통 유력한 비즈니스 후원자가 제시하는데, 이 인물은 특정 비즈니스 문제를 해결해야 할 책임을 맡고 있으므로 제때 정보를 이용할 수 있고 없고가 성공을 좌우하는 중대한 요소라고 생각합니다.

이 부분을 설명하기 위해 요즘 대다수의 리테일 기업이 직면한 중대한 문제점 하나를 예로 들어봅시다. 옴니채널 솔루션 지원 말입니다. 그와 같은 솔루션은 BOPIS(Buy-Online, Pickup In-Store, 온라인 구매 후 매장에서 픽업)를 지원하므로, 매장 재고에 관해 합당한 수준의 정확한 정보를 주로 활용합니다. 그런데 초기 범위를 이 한 가지 요구 사항으로만 제한한다면, 모니터링과 분석 시스템에 적용할 정보 요구 사항이 대폭 줄어들게 됩니다. 우선 실시간 인벤토리 솔루션을 제공하여 비즈니스에서 가치를 인식하면 그다음에 범위를 넓혀 다른 여러 요구 사항을 감안할 수 있습니다. 예를 들어 프로모션 모니터링, 사기 행위 탐지 등을 포함해 각 반복 작업에 활용하는 정보 자산의 폭을 넓히는 것입니다.

요점 2

전송을 데이터 생성 패턴과 긴급도에 맞추기

프로세스마다 POS 내에서 데이터를 생성하는 형태가 각기 다릅니다. 예를 들어 판매 트랜잭션의 경우, 새 레코드를 관련 테이블에 추가하여 흔적을 남길 가능성이 큽니다. 반품은 여러 가지 경로를 따라 과거 판매 기록의 업데이트를 트리거하기도 하고, 신규, 역판매 기록을 입력하고/거나 반품 전용 구조에 새 정보를 입력하는 등의 다양한 형태를 띱니다. POS 내에서 이벤트별 정보가 정확히 어떻게, 어디에 저장되는지 알아내려면 공급업체 문서, 비공식적인 내부 지식은 물론 개별적인 조사 작업을 동원해야 할 수도 있습니다.

이와 같은 패턴을 파악하면 특정 종류의 정보에 맞춰 데이터 전송 전략을 세우는 데 도움이 됩니다. 빈도가 높고 세분화된 입력 중심 패턴의 경우 지속적 스트리밍 방식이 가장 적합할 수 있습니다. 빈도가 낮은 대규모 이벤트는 배치 중심, 일괄 데이터 스타일 전송에 가장 잘 맞을지 모릅니다. 단, 이런 데이터 전송 방식이 주어진 범위의 양극단이라면 POS가 캡처하는 이벤트는 대부분 그 중간에 속할 가능성이 큽니다.

데이터 아키텍처에 데이터 레이크하우스 방식으로 접근하면 대표적인 장점은 여러 가지 데이터 전송 모드를 동시에 이용할 수 있다는 점입니다. 성격상 지속적 전송에 적합한 데이터의 경우, 스트리밍을 이용하면 됩니다. 일괄 전송에 더 적합한 데이터라면 배치 프로세스를 사용하면 됩니다. 그리고 그 중간에 속하는 데이터라면, 의사결정에 필요한 데이터의 적시성을 중점적으로 고려해 이것을 근거로 이후의 경로를 정하면 됩니다. 이런 모든 전송 모드에는 ETL 구현에 대한 일관된 접근 방식으로 접근하면 됩니다. 이것은 이전 구현 중 흔히 Lambda 아키텍처라고 하는 구현의 초기 버전을 여러 차례 실패하게 만든 원흉입니다.

요점 3

단계별 데이터 저장

매장 POS 시스템에서 도착하는 데이터는 빈도, 형식, 때를 맞춘 가용성 기대치 등이 모두 각기 다릅니다. 레이크하우스 내에서 보편적으로 쓰이는 브론즈, 실버 및 골드 설계 패턴을 활용하면 1차 정리, 형식 변경과 데이터 지속성을 특정 비즈니스 결과물에 맞춘 복잡한 필수 변환과 별도로 분리할 수 있습니다.

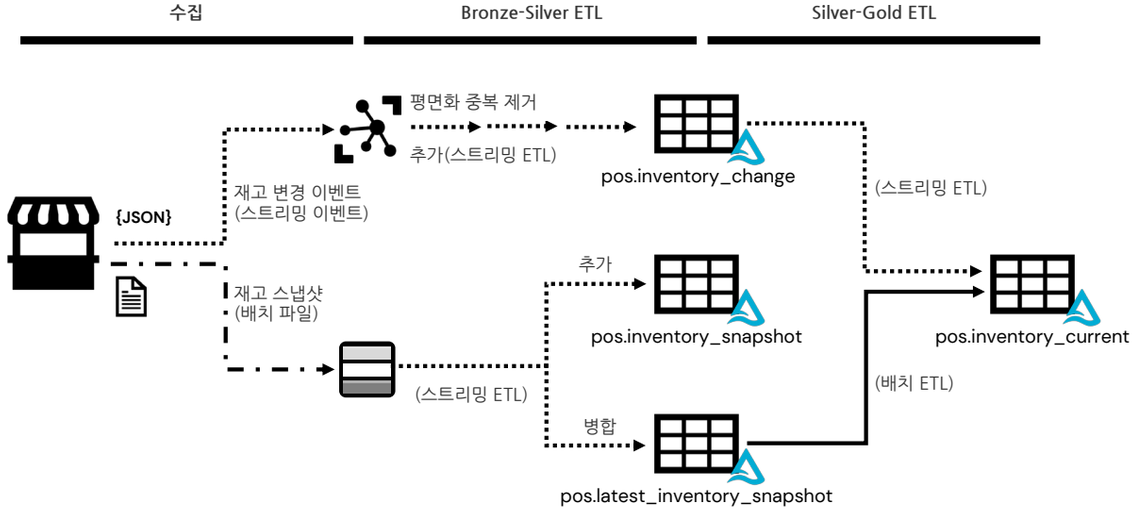

그림 3
데이터 지속성 브론즈, 실버 및 골드 패턴을 활용하여 현재 재고를 계산하는 데이터 레이크하우스 아키텍처

요점 4

기대치 관리

준 실시간 분석으로 바꾸려면 기업 차원에서의 전환이 필요합니다. Gartner에서는 이것을 스트리밍 분석 성숙도(Streaming Analytics Maturity) 모델을 통해 설명했는데, 이 경우 스트리밍 데이터 분석 작업이 일상적인 업무 구조 속에 통합되게 됩니다. 이런 결과를 하루아침에 기대해서는 안 됩니다.

그보다는 데이터 엔지니어에게 충분한 시간을 주고 실제 매장 위치에서 중앙집중형, 클라우드 기반 백오피스로 스트리밍 전송할 때 생기는 본질적인 문제점을 파악하도록 해야 합니다. 연결성과 시스템 신뢰도를 개선하고, 여기에 더욱 탄탄한 ETL 워크플로우를 함께 사용하면 데이터를 제때, 더 안정적이고 일관성 있는 형태로 저장할 수 있습니다. 이렇게 하려면 대개 시스템 엔지니어와 애플리케이션 개발자의 파트너십을 강화하여 배치 전용 ETL 워크플로우 시절에는 흔히 찾아보기 어렵던 수준으로 통합을 적극 지원해야 합니다.

또한 비즈니스 애널리스트도 데이터가 끊임없이 업데이트되면서 발생하는 본질적인 노이즈에 익숙해져야 할 것입니다. 예를 들어 몇 초 전에 실행된 쿼리가 잠시 후 약간 다른 결과를 반환하는 등 주어진 데이터 세트에서 진단과 검증 작업을 수행하는 방법도 다시 익혀야 할 것입니다. 일일 집계 데이터 형태로 표시하면 보통 숨겨지기 마련인 데이터 내부의 다양한 문제점을 심층적으로 인식할 줄 알아야 하기도 합니다. 이 모든 일을 제대로 해내려면 결과에서 탐지된 신호에 대한 분석과 대응을 둘 다 조정해야 합니다.

이 모든 문제는 성숙도 초반의 몇 단계에서만 발생합니다. 후반 단계에서는 기업이 스트림 내에서 의미 있는 신호를 감지하는 능력이 발달하면서 결과적으로 더욱 자동화된 감지 및 대응 역량을 갖추게 되기 때문입니다. 이때 데이터 스트림의 가치가 극대화됩니다. 단, 우선 모니터링과 거버넌스를 적용하고 검증을 거친 뒤에만 이러한 기술에 운영을 맡겨야 할 것입니다.

POS 스트리밍 구현

POS 데이터에 레이크하우스 아키텍처를 적용하는 방법을 설명하기 위해 데모 워크플로우를 개발하여 이 플로우에서 준 실시간으로 재고를 계산해 보았습니다. 여기에서는 별개의 POS 시스템이 2개 있어 판매, 재고 보충, 결품 데이터와 관련된 재고 관련 정보를 전송하고, 스트리밍 재고 변경 피드의 일환으로 BOPIS 트랜잭션(한 시스템에서 시작, 다른 시스템에서 처리)을 다루는 것으로 가정하였습니다. 진열된 제품 단위 수를 정기적으로(스냅샷) 계수하여 이를 POS에서 캡처해 일괄 전송합니다. 이러한 데이터를 1개월의 기간에 대하여 시뮬레이션한 다음 10배 속도로 재생해 재고 변동 상황을 더욱 정확하게 파악하는 것입니다.

ETL 프로세스(그림 3 참조)에는 스트리밍과 배치 기법이 섞여 있습니다. 2단계 방식을 적용하여 실버 계층을 나타내는 Delta 테이블에서 캡처한 데이터를 최소한만 변환했는데, 이는 최신 재고 계산에 필요한 비즈니스 중심 방식을 사용한 기술 중심적인 첫 번째 방식과 구분됩니다. 두 번째 단계는 기존의 구조적 스트리밍(structured streaming)을 사용해 구현했는데, 이 부분은 새로운 Delta Live Tables 기능이 정식으로 출시되면 그때 자세히 살펴보겠습니다.

이 데모에서는 데이터 수집에 Azure IOT Hubs와 Azure Storage를 사용하지만 적절한 대체 기술을 사용하면 AWS 및 GCP 클라우드에서도 비슷한 결과를 낼 수 있습니다.

무료 Databricks 노트북으로 실험 시작

- POS 01: 환경 설정
- POS 02: 데이터 생성
- POS 03: ETL 수집
- POS 04: 현재 재고

섹션 2.2

CrowdStrike Falcon 이벤트를 위한 사이버 보안 레이크하우스 구축

글: AEMRO AMARE, ARUN PAMULAPATI, YONG SHENG HUANG, JASON POHL

2021년 5월 20일

보안 팀에서는 엔드포인트 데이터를 위협 탐지, 위협 헌팅, 인시던트 조사에 사용하며 규정 준수 요구 사항에 부합하는 데도 엔드포인트 데이터가 필요합니다. 이런 데이터는 일 단위로는 테라바이트급, 연 단위로는 페타바이트급에 달할 수 있습니다. 대부분의 기업에서는 엔드포인트 로그를 수집, 저장하고 분석하는 데 애를 먹습니다. 데이터 볼륨이 워낙 어마어마한 탓에 비용과 복잡성 면에서 부담이 크기 때문입니다. 하지만 굳이 이렇게 어려워할 일은 아닙니다.

이 블로그에서는 2부작 시리즈를 통해 Databricks와 함께 페타바이트급 엔드포인트 데이터를 운영화(operationalize)하여 비용 효율적인 방식으로 고급 분석을 사용해 보안 태세를 개선하는 방법을 알아보겠습니다. 1부(이 블로그)에서는 데이터 수집 아키텍처와 SIEM(Splunk)과의 통합에 대해 알아봅니다. 블로그 글을 마치며 노트북을 제공하므로, 데이터를 바로 분석에 사용할 수 있습니다. 2부에서는 구체적인 사용 사례, ML 모델을 만드는 법, 자동 데이터 보강과 분석 방법을 논합니다. 2부를 마치면 노트북을 구현하여 엔드포인트 데이터를 사용해 위협을 탐지, 조사할 수 있게 됩니다.

여기에서는 CrowdStrike의 Falcon 로그를 예시로 사용하였습니다. Falcon 로그에 액세스하려면, Falcon Data Replicator(FDR)를 사용하여 원시 이벤트 데이터를 CrowdStrike 플랫폼에서 클라우드 스토리지(예: Amazon S3)로 푸시해야 합니다. 이 데이터는 Databricks 레이크하우스 플랫폼을 사용해 나머지 보안 텔레메트리와 함께 수집, 변환, 분석하고 저장할 수 있습니다. 고객은 CrowdStrike Falcon 데이터를 수집, Python 기반 실시간 탐지를 적용, Databricks SQL을 사용해 과거 데이터 검색을 수행하고 Databricks Add-on for Splunk와 같은 SIEM 도구에서 쿼리할 수 있습니다.

CrowdStrike 데이터 운영화의 문제점

CrowdStrike Falcon -데이터가 종합적인 이벤트 로깅 상세 정보를 제공하기는 하지만, 준실시간으로 비용 효율성까지 따져 가며 대량의 복잡한 사이버 보안 데이터를 수집, 처리하고 운영화하기란 버거운 업무입니다. 이와 관련된 잘 알려진 문제점을 몇 가지 소개합니다.

- **실시간 대규모 데이터 수집:** FDR이 준 실시간으로 클라우드 스토리지에 작성한 원시 데이터 파일(처리된 것도 있고 처리되지 않은 것도 있음)을 놓치지 않고 파악하기 어렵습니다.
- **복잡한 변환:** 데이터 형식이 '반정형'입니다. 각각의 로그 파일에 줄마다 수백 가지 비결정적 페이로드를 포함하며, 시간이 흐르면서 이벤트 구조가 바뀔 가능성도 있습니다.
- **데이터 거버넌스:** 이런 종류의 데이터는 중요할 수 있으며, 액세스는 필수 인원 원칙에 따라 꼭 필요한 사용자에게만 허용해야 합니다.

databricks

- **전체 보안 분석 간소화:** 이처럼 이동 속도가 빠른 대용량 데이터 세트를 대상으로 데이터 엔지니어링, ML과 분석 작업을 하려면 확장 가능한 도구가 있어야 합니다.
- **협업:** 효과적인 협업을 통해 데이터 엔지니어, 사이버 보안 애널리스트와 ML 엔지니어가 보유한 각 분야 전문성을 활용할 수 있습니다. 따라서 협업 플랫폼을 확보하면 사이버 보안 분석과 대응 워크로드 효율성이 개선됩니다.

그 결과 여러 기업의 보안 엔지니어는 비용과 운영 효율성을 균형 있게 관리해야 한다는 곤란한 상황에 직면하고 맙니다. 선택은 둘 중 하나입니다. 고가의 상용 시스템에 종속(lock-in)될 수밖에 없다고 수긍하거나, 아니면 확장성과 성능을 확보하기 위해 고군분투하는 와중에 자체적으로 엔드포인트 보안 도구를 구축하기 위해 엄청난 공을 들여야 하는 것입니다.

Databricks 사이버 보안 레이크하우스

Databricks는 보안 팀과 데이터 사이언티스트에게 업무를 능률적이고 효과적으로 수행할 수 있다는 새로운 희망을 심어줍니다. 그뿐만 아니라, 빅데이터 및 정교한 위협 등 점점 커지는 난관에 맞설 유용한 도구도 제공합니다.

데이터 레이크와 데이터 웨어하우스의 장점만을 결합한 오픈 아키텍처인 레이크하우스는 데이터에 구조를 점진적으로 추가하는 멀티 홉(multi-hop) 데이터 엔지니어링 파이프라인을 간소화해줍니다. 멀티 홉 아키텍처의 장점은 데이터 엔지니어가 원시 데이터로 시작하는 파이프라인을 구축하여 이것을 모든 것이 이동하는 원천인 “단일 정보 출처(Single source of truth, SSOT)”로 삼을 수 있다는 사실입니다. CrowdStrike의 반정형 원시 데이터는 몇 년 동안 저장해 두었다가 나중에 엔드 투 엔드 스트리밍 방식으로 변환, 집계하여 데이터를 정제할 수 있으며, 컨텍스트별 구조를 도입해 여러 가지 서로 다른 가상의 상황에서 보안 리스크를 분석, 탐지할 수도 있습니다.

- **데이터 수집:** Auto Loader (AWS | Azure | GCP) 를 사용하면 CrowdStrike FDR이 원시 데이터 스토리지에 새 파일을 쓰자마자 즉시 데이터를 읽을 수 있습니다. Auto Loader는 클라우드 알림 서비스를 활용하여 클라우드에 새 파일이 도착하면 바로 증분 방식으로 처리합니다. 또한 새 파일 수신 여부를 알려주는 알림 서비스를 자동으로 구성하여 수신하고, 초당 파일 개수 수백만 개까지 확장할 수도 있습니다.
- **통합 스트림 및 배치 처리:** Delta Lake는 데이터 레이크에 데이터 관리와 거버넌스를 적용하는 개방적 방식으로, Apache Spark™의 분산형 컴퓨팅 성능을 방대한 데이터와 메타데이터에 제공합니다. Databricks Delta Engine은 초당 수백만 개의 레코드를 처리하는 고도로 최적화된 엔진입니다.
- **데이터 거버넌스:** With Databricks Table Access Control (AWS | Azure | GCP)을 사용하는 관리자는 사용자의 업무 부서에 따라 Delta 테이블에 서로 다른 액세스 권한을 부여할 수 있습니다.

databricks

- **보안 분석 도구:** Databricks SQL을 이용하여 비정상적인 패턴이 탐지되면 자동 알림을 보내는 인터랙티브 대시보드를 생성할 수 있습니다. 마찬가지로 Tableau, Microsoft Power BI 및 Looker와 같이 도입률이 높은 도구와도 간편하게 통합됩니다.
- **Databricks 노트북 협업:** Databricks 협업 노트북을 이용하면 여러 보안 팀이 실시간으로 협업할 수 있습니다. 여러 사용자가 여러 가지 언어로 쿼리를 실행할 수 있고 시각화를 공유할 수도 있으며 같은 워크스페이스 내에 코멘트를 작성해 조사가 차질 없이 진전되도록 해줍니다.

CrowdStrike Falcon 데이터용 레이크하우스 아키텍처

CrowdStrike의 Falcon 데이터와 같은 사이버 보안 워크로드에는 다음과 같은 레이크하우스 아키텍처를 권장합니다. Auto Loader와 Delta Lake는 클라우드 스토리지에서 가져온 원시 데이터를 읽어 Delta 테이블에 쓰는 프로세스를 간소화하며, 비용이 저렴하고 DevOps 작업도 최소한만 필요합니다.

이 아키텍처에서는 반정형 CrowdStrike 데이터를 랜딩 영역에 있는 고객 측 클라우드 스토리지에 로드합니다. 그런 다음 Auto Loader가 클라우드 알림 서비스를 사용해 자동으로 새 파일 처리와 수집을 고객 브론즈 테이블에 트리거하며, 이것이 모든 다운스트림 작업의 SSOT 역할을 하게 됩니다. Auto Loader는 체크포인트를 사용해 처리한 파일과 처리하지 않은 파일을 추적하여 데이터 처리 중복을 방지합니다.

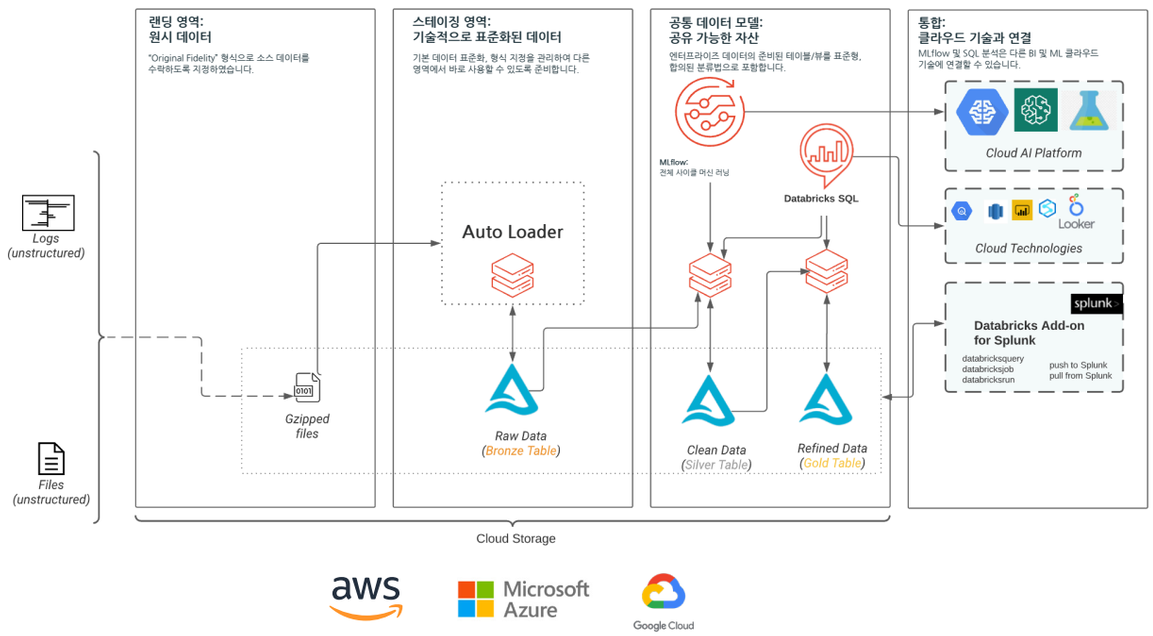

그림 1
CrowdStrike Falcon 데이터용 레이크하우스 아키텍처

브론즈에서 실버 단계로 이동하면서 스키마가 추가되어 데이터에 구조를 제공합니다. 읽기 작업을 SSOT에서 수행하므로 각종 이벤트 유형을 모두 처리할 수 있으며, 스키마가 각각의 해당 테이블에 쓰일 때 올바른 스키마를 적용할 수 있습니다. 실버 계층에서 스키마를 적용하는 기능이 ML과 분석 워크로드 구축을 위한 건실한 기반이 되어줍니다.

골드 단계의 경우 대시보드와 BI 도구에서 빠른 쿼리와 성능을 제공하기 위해 데이터를 집계합니다(사용 사례 및 데이터 볼륨에 따라 다름). 예기치 못한 추세가 탐지되면 알림이 트리거되도록 설정할 수 있습니다.

또 한 가지 옵션 기능으로 Databricks Add-on for Splunk를 들 수 있습니다. 이것을 사용하면 보안 팀이 Databricks의 비용 효율적인 모델과 AI의 파워를 유리하게 활용하면서도, 친숙한 Splunk를 포기하지 않아도 됩니다. 고객은 Splunk 대시보드 내에서나 검색창에서 이 애드온 기능을 사용하여 Databricks에 애드혹 쿼리를 실행할 수 있습니다. 또한 사용자가 Splunk 대시보드를 통해서나, Splunk 검색에 대한 대응으로 Databricks에서 노트북이나 작업을 시작할 수도 있습니다. Databricks 통합은 양방향이므로 고객이 노이즈를 포함한 데이터를 요약하거나 Splunk Enterprise Security에 표시되는 Databricks 탐지를 실행할 수 있습니다. 그뿐만이 아니라, Databricks 노트북 내에서 Splunk 검색을 실행하여 중복 데이터가 필요할 가능성을 방지하기도 합니다.

Splunk와 Databricks를 통합하면 고객이 비용을 절감하고, 분석할 데이터 소스를 확대하며 좀 더 탄탄한 분석 엔진에서 얻은 결과를 제공할 수 있으며 현재 스태프가 매일같이 사용하는 도구를 바꿀 필요도 없습니다.

코드 설명

Auto Loader는 파일 기반 데이터 수집의 가장 복잡한 부분을 추상화하므로 코드 몇 줄만으로 원시-브론즈(raw-to-Bronze) 수집 파이프라인을 만들 수 있습니다. 아래에 Delta 수집 파이프라인의 Scala 코드를 예로 들어 보았습니다. CrowdStrike Falcon 이벤트 레코드에는 다음과 같은 공통의 필드 네임이 하나 있습니다.

```
val crowdstrikeStream = spark.readStream
  .format(“cloudFiles”)
  .option(“cloudFiles.format”, “text”)    // text file doesn’t need schema
  .option(“cloudFiles.region”, “us-west-2”)
  .option(“cloudFiles.useNotifications”, “true”)
  .load(rawDataSource)
  .withColumn(“load_timestamp”, current_timestamp())
  .withColumn(“load_date”, to_date($”load_timestamp”))
     .withColumn(“eventType”,  from_json($”value”,  “struct”,  Map.empty[String,  String]))
.selectExpr(“eventType.event_simpleName”,”load_date”,”load_timestamp”, “value” )
  .writeStream
  .format(“delta”)
  .option(“checkpointLocation”, checkPointLocation)
  .table(“demo_bronze.crowdstrike”)
```

원시-브론즈 계층에서, 원시 데이터에서는 이벤트 이름만 추출합니다. 여기에 로드 타임스탬프와 날짜 열을 추가하면 사용자가 원시 데이터를 브론즈 테이블에 저장할 수 있습니다. 브론즈 테이블은 이벤트 이름과 로드 날짜를 기준으로 분할되므로, 브론즈-실버(Bronze-to Silver) 작업의 성능이 더욱 우수합니다. 특히 이벤트 날짜 범위의 수가 많지 않은 경우 이 특징이 두드러집니다. 다음으로, 브론즈-실버 스트리밍 작업이 브론즈 테이블에서

이벤트를 읽어와 스키마를 하나 적용한 다음 이벤트 이름에 따라 수백 개의 이벤트 테이블에 쓰기 작업을 수행합니다. 아래는 Scala 코드 예시입니다.

```
spark
  .readStream
  .option("ignoreChanges", "true")
  .option("maxBytesPerTrigger", "2g")
  .option("maxFilesPerTrigger", "64")
  .format("delta")
  .load(bronzeTableLocation)
  .filter($"event_simpleName" === "event_name")
  .withColumn("event", from_json($"value", schema_of_json(sampleJson)) )
  .select($"event.*", $"load_timestamp", $"load_date")
  .withColumn("silver_timestamp", current_timestamp())
  .writeStream
  .format("delta")
  .outputMode("append")
  .option("mergeSchema", "true")
  .option("checkpointLocation", checkPoint)
  .option("path", tableLocation)
  .start()
```

각각의 이벤트 스키마는 스키마 레지스트리에 저장해도 되고, 스키마 하나를 여러 개의 데이터 기반 서비스에서 공유해야 하는 경우 Delta 테이블에 저장하면 됩니다. 위의 코드에서는 브론즈 테이블에서 읽은 샘플 JSON 문자열을 사용하였으며, 스키마는 schema_of_json()을 사용하여 JSON으로부터 추론하였습니다. 이후, 이 JSON 문자열은 from_json()을 사용해 구조체(struct)로 변환됩니다. 그런 다음 구조체가 평면화되어 타임스탬프 열을 추가해야 하게 됩니다. 이러한 단계에서는 DataFrame을 제공하며 여기에 이벤트 테이블에 추가할 필수 열을 모두 포함합니다. 마지막으로, 추가 모드를 사용해 이 정형 데이터를 이벤트 테이블에 씁니다.

또한 여러 이벤트를 스트림 한 개를 포함한 여러 테이블에 팬아웃할 수도 있습니다. 마이크로배치를 처리할 함수를 하나 정의하여 foreachBatch를 사용하면 됩니다. foreachBatch()를 사용하면 기존의 배치 데이터 소스를 필터링 및 여러 테이블에 쓰기 작업에 다시 사용할 수 있습니다. 단, foreachBatch()는 최소 한 번(at-least-once) 쓰기만 보장합니다. 따라서 한 번만(exactly-once) 시맨틱을 적용하려면 수동 구현이 필요합니다.

이 단계에서는 정형 데이터를 Databricks 노트북과 작업에서 지원되는 언어라면 무엇으로든 쿼리할 수 있습니다. (Python, R, Scala 및 SQL 등) 실버 계층 데이터는 ML과 사이버 공격 분석에 사용하면 편리합니다.

다음 스트리밍 파이프라인은 실버-골드(Silver-to-Gold)가 되겠습니다. 이 단계에서는 데이터를 대시보딩과 알림 용도로 집계할 수 있습니다. 이 블로그 시리즈의 2부에서는 Databricks SQL을 사용해 대시보드를 구축하는 방법을 좀 더 자세히 알려드립니다.

다음 예고

더 많은 블로그 글을 통해 ML을 적용하고 Databricks SQL을 사용하여 이 사용 사례에서 더욱 많은 가치를 창출하는 방법을 알려드리겠습니다.

여기 주어진 노트북을 각자의 Databricks 배포에서 사용하면 됩니다. 노트북의 섹션마다 설명이 있습니다. 궁금한 점은 이메일 cybersecurity@ databricks.com으로 문의하세요. 이 노트북의 이해를 돕고 손쉽게 배포하는 방법에 관한 여러분의 문의 사항과 제안을 기다립니다.

무료 Databricks 노트북으로 실험을 시작하세요.

databricks

섹션 2.3

현대적 데이터 레이크하우스로 의료 데이터 가치 재발견

글: MICHAEL ORTEGA, MICHAEL SANKY, AMIR KERMANY

2021년 7월 19일

의료 서비스 및 생명과학 부문에서 데이터 웨어하우스와 데이터 레이크 문제를 극복하는 법

환자 한 사람이 매년 도출하는 의료 데이터는 약 80MB입니다. 이 값을 환자 수천 명, 평생으로 환산하면, 귀중한 인사이트가 담긴 환자 데이터는 페타바이트급으로 늘어납니다. 이러한 인사이트를 잘 활용하면 임상 운영을 간소화하고 신약 연구 개발 속도를 높여 결과적으로 환자의 건강 상태 개선에도 도움이 될 수 있습니다. 하지만 우선은 데이터를 다운스트림 분석과 AI에 맞게 준비하는 과정을 거쳐야 합니다. 유감스럽게도 대부분의 의료 서비스, 생명과학 기업은 단순히 데이터를 수집, 정리, 정형화하는 데만 지나치게 많은 시간을 할애합니다.

환자 한 명이 매년 80MB+의 의료 데이터 생성

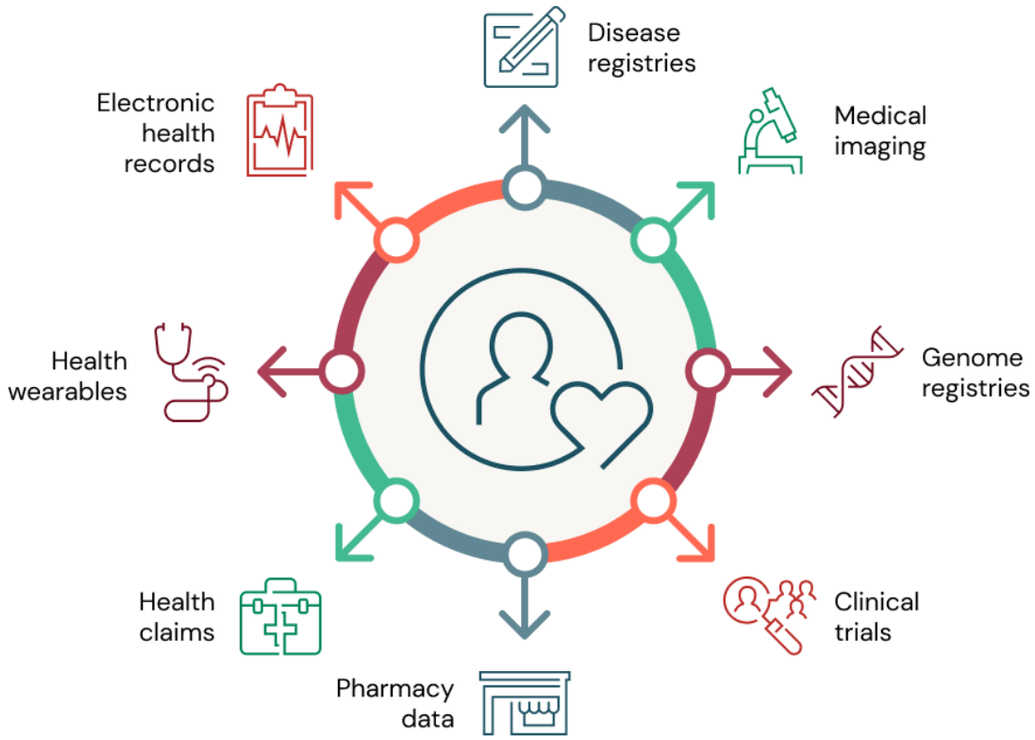

그림 1
의료 데이터가 기하급수적으로 증가하면서, 환자 한 명이 일 년에 80MB 이상의 데이터 생성

의료 서비스 및 생명과학 분야의 데이터 분석과 관련된 문제점

의료 서비스 업계에 속하는 기업에서 데이터 준비, 분석과 AI가 어려운 데는 수많은 이유가 있지만 그중 대다수는 데이터 웨어하우스 기반인 레거시 데이터 아키텍처에 투자했다는 사실과 밀접한 관련이 있습니다. 이 업계에서 확인되는 가장 보편적인 4대 문제점을 소개합니다.

문제점 #1: 볼륨

급증하는 의료 데이터에 맞춘 확장

유전체학은 아마 의료 서비스 부문의 데이터 볼륨 폭증을 입증하는 가장 좋은 예일 것입니다. 유최초의 게놈 염기 서열을 분석하는 데 10억 달러가 넘는 비용이 들었습니다. 엄청난 비용 때문에 초창기 연구는(지금도 많은 프로젝트에 해당하는 사실이지만) 유전형 분석 위주였습니다. 이는 사람의 게놈에서 매우 미미한 부분, 보통 0.1% 안팎을 차지하는 특정 변이를 찾는 과정을 말합니다. 이것이 WES(Whole Exome Sequencing)로 발전했는데, 이 경우 게놈의 단백질 암호화 부분을 다룹니다. 다만 그래봐야 이것도 게놈 전체의 2%에도 미치지 못하는 수준입니다. 요즘은 D2C(Direct-to-Consumer) 방식으로 WGS 검사를 판매하는데, WGS 30개당 300달러도 안 됩니다. 집단 유전학 수준에서는, UK Biobank에서 올해 연구용으로 전장 유전체(whole genome)를 200,000개 이상 공개할 예정입니다. 유전체학뿐만이 아닙니다. 영상, 의료용 웨어러블 및 전자 의료 기록도 엄청난 수준으로 증가하고 있습니다.

인구 건강 분석, 신약 개발과 같은 이니셔티브에서는 무엇보다도 확장이 관건입니다. 그런데 유감스럽게도 대부분의 레거시 아키텍처는 온프레미스 기반이며, 피크 용량에 맞춰 고안되었습니다. 이 방식의 경우 사용량이 적을 때 미사용 컴퓨팅 파워가 생기고(나아가 비용이 낭비됨) 업그레이드가 필요할 때 신속하게 확장할 수도 없습니다.

문제점 #2: 다양성

다양한 의료 데이터 분석

의료 서비스 및 생명과학 기업은 엄청나게 다양한 종류의 데이터를 대량으로 취급하는데, 저마다 나름의 미묘한 차이가 있습니다. 의료 데이터는 80% 이상이 비정형 데이터라는 것이 정설이지만, 대부분의 기업은 아직도 정형 데이터와 기존의 SQL 기반 분석 중심으로 고안된 데이터 웨어하우스에 주의를 집중하고 있습니다. 비정형 데이터 중에는 종양학, 면역학, 신경학 등의 분야에서 질병을 진단하고 경과를 측정하는 데 매우 중요한(비용이 가장 빨리 증가하는 분야임) 이미지 데이터, 그리고 환자의 의료 및 사회적 기록을 완전히 파악하는 데 중요한 임상 기록상의 내러티브 텍스트 데이터가 대표적입니다. 이러한 데이터 유형을 무시하거나 간과한다는 것은 불가능한 일입니다.

설상가상으로, 의료 시스템 에코시스템의 상호연결성이 점점 강화되어 이해관계자가 새로운 데이터 유형이라는 난관에 직면해야만 하는 상황입니다. 예를 들어 의료 서비스 제공자 측에서는 청구 데이터가 있어야 위험분담계약(RSA)를 관리, 판정할 수 있고, 지불자의 경우 사전 승인을 받는 등의 프로세스를 지원하고 품질 측정치를 도출하려면 임상 데이터가 필요합니다. 이와 같은 기업은 보통 이러한 새로운 데이터 유형을 지원하는 데 필요한 데이터 아키텍처와 플랫폼이 미비합니다.

기업에 따라 비정형 데이터와 고급 분석을 지원하기 위해 데이터 레이크에 투자한 경우도 있지만, 이 경우 일련의 새로운 문제가 발생합니다. 이 환경에서는 데이터 팀이 관리해야 할 시스템이 두 개가 되며(데이터 웨어하우스, 데이터 레이크) 사일로화된 도구를 오가며 데이터가 복사되므로 데이터 품질과 관리 문제가 생깁니다.

문제점 #3: 속도

실시간 환자 인사이트를 위한 스트리밍 데이터 처리

의료 서비스는 대개 생사를 좌우하는 상황과 관련이 있습니다. 상황이 매우 급박하게 변할 수 있고, 배치 데이터 처리만으로는(매일같이 수행한다고 해도) 역부족일 때가 많습니다. 인터벤션(중재적) 치료에 성공하려면 무엇보다도 초 단위까지 정확한 최신 정보를 확보하는 것이 관건입니다. 병원과 국민 보건 체계에서는 생명을 구하기 위해 스트리밍 데이터를 패혈증을 예측하거나 중환자실(ICU) 침상 수요를 실시간으로 예측하는 등 다방면으로 사용합니다.

이외에 데이터 속도도 의료 서비스 디지털 혁명을 이루는 중요한 요소입니다. 요즘은 개인에게 전에 없이 많은 정보에 액세스 권한이 주어지며, 누구나 실시간으로 자신의 치료에 영향력을 행사할 수 있습니다. 예를 들어 지속해서 혈당 수준을 모니터링하여 제공하는 Livongo와 같은 웨어러블 기기가 모바일 앱으로 실시간 데이터를 스트리밍해 앱에서 개인별 추천 행동을 제안하는 방식입니다.

이처럼 초창기에 성공을 거둔 사례가 몇 가지 있지만, 대부분의 기업은 데이터 아키텍처를 설계할 때 스트리밍 데이터 속도를 수용해야 한다는 점을 고려하지 않았습니다. 따라서 안정성 문제가 발생하고 실시간 데이터를 과거 데이터와 통합하기 어려워 혁신에 차질을 빚습니다.

문제점 #4: 정확성

의료 서비스 데이터 및 AI에서 신뢰 구축

마지막으로 꼭 지적해야 할 부분은, 의료 서비스 부문은 임상 및 규제 표준에 따라 극히 높은 수준의 데이터 정확도를 확보해야 한다는 것입니다. 의료 서비스 기업에는 수준 높은 공공 보건 규정 준수 요건이 부과되며 이를 반드시 준수해야 합니다. 내부에서 데이터를 민주화하려면 거버넌스가 필요합니다.

또한 임상 업무에 인공 지능(AI)과 머신 러닝(ML)과 같은 기술을 도입하려면 양질의 모델 거버넌스가 필요합니다. 유감스럽게도 대부분의 기업에는 데이터 사이언스 워크플로우 용도로 별도의 플랫폼이 있으며, 이는 데이터 웨어하우스와는 단절되어 있습니다. 이 때문에 AI 기반 애플리케이션에서 신뢰성과 재현성을 구축하려 하면 심각한 문제가 생깁니다.

레이크하우스로 의료 데이터 가치 재발견

레이크하우스 아키텍처를 이용하면 의료 서비스 및 생명과학 기업에서 이와 같은 문제를 극복하는 데 유리합니다. 클라우드 데이터 레이크의 저렴한 비용, 확장성과 유연성에 데이터 웨어하우스의 성능과 거버넌스를 합친 최신 데이터 아키텍처를 제공하기 때문입니다. 레이크하우스를 사용하면 오픈 환경에 각종 데이터(유형 불문)를 저장하고 각종 분석과 ML을 지원할 수 있습니다.

의료 서비스 및 생명과학 부문을 위한 레이크하우스 구축

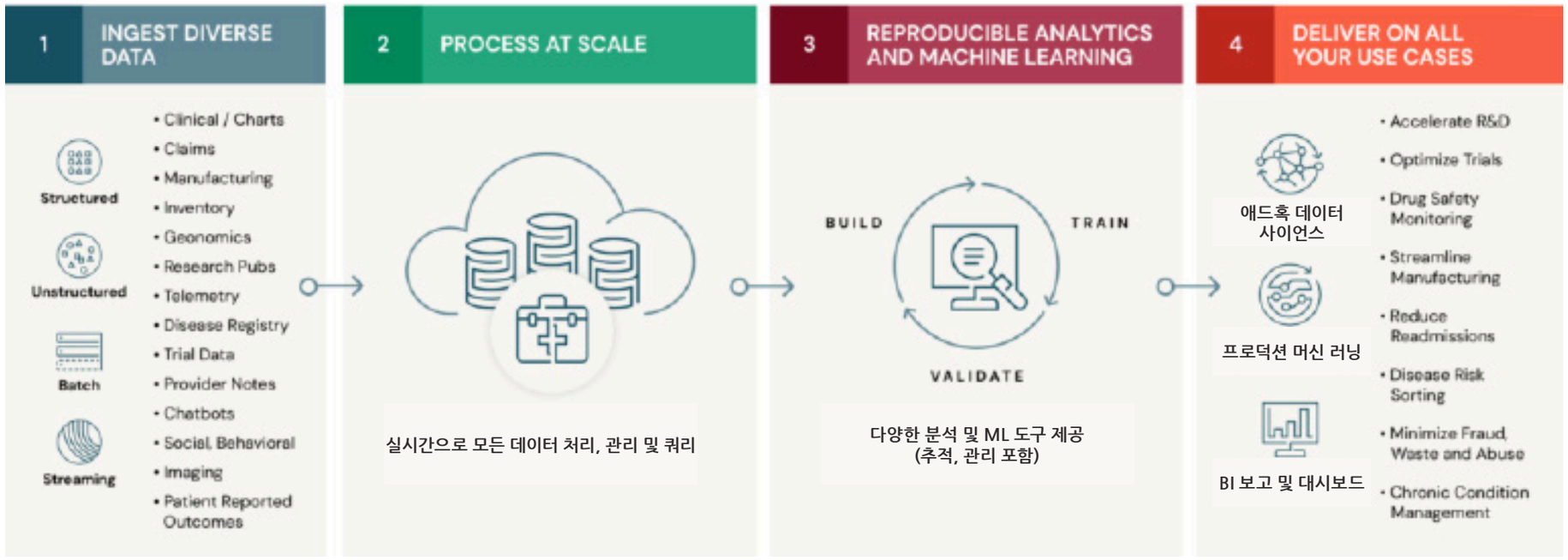

그림 2
각종 의료 서비스 및 생명과학 데이터 분석 사용 사례를 최신 레이크하우스 아키텍처로 제공

구체적으로 설명하자면, 레이크하우스는 의료 서비스 및 생명과학 기업에 다음과 같은 장점을 제공합니다.

- **각종 의료 데이터를 모두 대규모로 구성.** Databricks 레이크하우스 플랫폼의 핵심은 데이터 레이크에 안정성과 성능을 제공하는 오픈 소스 데이터 관리 계층인 Delta Lake 입니다. Delta Lake는 기존의 데이터 웨어하우스와는 달리 유형을 가리지 않고 모든 정형, 비정형 데이터를 지원합니다. 또한 Databricks에서는 의료 데이터를 간편하게 수집하도록 돕기 위해 전자 파일 형태의 의료 기록과 유전체학 등 분야별 데이터 유형에 따른 커넥터를 구축했습니다. 이러한 커넥터를 퀵스타트 솔루션 액셀러레이터 세트 형태로, 업계 표준 데이터 모델과 함께 패키지로 제공합니다. 또한 Delta Lake는 데이터 캐싱과 인덱싱을 위한 기본 내장 최적화를 제공하므로 데이터 처리 속도가 대폭 빨라집니다. 이와 같은 기능을 확보하면 원시 데이터를 모두 한곳에 저장했다가 큐레이션하여 환자 의료 정보를 총체적으로 파악할 수 있습니다.
- **환자 분석 및 AI 모두 지원.** 모든 데이터가 레이크하우스 하나에 중앙집중화되면 데이터를 직접 활용하여 우수한 환자 분석 및 예측 모델을 구축할 수 있습니다. Databricks에서는 이러한 기능을 기반으로 각종 분석 및 AI 도구를 포함한 협업형 워크스페이스를 제공하며 광범위한 프로그래밍 언어(예를 들어 SQL, R, Python, Scala)를 지원합니다. 이렇게 하면 데이터 사이언티스트, 엔지니어, 임상 정보학자 등 다양한 사용자 집단이 협력하여 의료 데이터 전체를 분석, 모델링, 시각화하는 데 큰 도움이 됩니다.

- **실시간 환자 인사이트 제공.** 레이크하우스는 스트리밍 및 배치 데이터에 통합 아키텍처를 제공합니다. 두 가지 서로 다른 아키텍처를 지원하지 않아도 되고, 안정성 문제로 애를 먹을 일도 없습니다. 또한 레이크하우스 아키텍처를 Databricks에서 실행하면 워크로드에 따라 자동으로 크기가 조정되는 클라우드 기반 플랫폼을 이용할 수 있습니다. 이렇게 하면 스트리밍 데이터를 손쉽게 수집하고 페타바이트급의 과거 데이터와 혼합해 모집단 규모로 준 실시간 인사이트를 얻을 수 있습니다.
- **데이터 품질 보장 및 규정 준수.** 레이크하우스에는 데이터 정확성 문제를 해결하기 위해 기존의 데이터 레이크에는 없었던 몇 가지 기능을 포함하였습니다. 예를 들어 스키마 적용, 감사, 버전 관리, 세분화된 액세스 제어 등이 대표적입니다. 레이크하우스의 중요한 장점은 분석과 ML을 둘 다 이 신뢰할 수 있는 데이터 소스에서 수행할 수 있다는 점입니다. 이외에도 Databricks가 제공하는 ML 모델 추적과 관리 기능을 이용하면 여러 환경에 걸쳐 결과를 재현하고 규정 준수 표준에 부합하는 것도 어렵지 않습니다. 이 모든 기능은 HIPAA를 준수하는 분석 환경에서 제공됩니다.

이 레이크하우스는 의료 서비스 및 생명과학 데이터 관리를 위한 최선의 아키텍처입니다. 이 아키텍처를 Databricks에서 제공하는 다양한 기능과 함께 사용하면 신약 개발부터 만성 질환 관리 프로그램에 이르기까지 광범위하고 파급력 높은 사용 사례를 지원할 수 있습니다.

의료 서비스 및 생명과학 레이크하우스 구축 시작하기

앞서 언급했듯이, Databricks에서는 다양한 솔루션 액셀러레이터를 제공하여 의료 서비스 및 생명과학 기업에서 각자의 요구 사항에 맞는 레이크하우스 구축을 손쉽게 시작할 수 있도록 돕고 있습니다. Databricks 솔루션 액셀러레이터에는 샘플 데이터, 미리 작성한 코드와 단계별 지침 등이 들어 있습니다(Databricks 노트북 안에 포함).

새로운 솔루션 액셀러레이터: Lakehouse for Real-World Evidence. 제약회사에서는 실제 데이터를 통해 임상시험 환경을 벗어난 상황에서 환자의 건강과 의약품 효능과 관련된 새로운 인사이트를 얻습니다. 이 액셀러레이터를 사용하면 Databricks에 Lakehouse for Real-World Evidence를 구축하는 데 도움이 됩니다. 환자 모집단의 샘플 EHR 데이터를 수집하는 법, OMOP 공용 데이터 모델을 사용해 그 데이터를 구조화한 다음 대규모로 분석을 실행하여 약 처방 패턴 조사와 같은 문제를 해결하는 법을 안내받으실 수 있습니다.

무료 Databricks 노트북으로 실험을 시작하세요.

Databricks의 의료 서비스 및 생명과학 솔루션에 관해 자세히 알아보세요.

databricks

섹션 2.4

규제 보고서 전송의 적시성과 신뢰성

글: ANTOINE AMEN, FAHMID KABIR

2021년 9월 17일

리스크 관리, 규정 준수 업무는 점점 더 복잡해지고 있고 비용도 점점 인상되고 있습니다. 2008년 글로벌 금융 위기 이후 규제 변동량이 500% 늘어나면서 그 과정에서 규제 관련 비용도 크게 증가했습니다. 규정 위반 및 SLA 위반과 관련된 벌금 규모를 감안하면(2019년 은행에 부과된 AML 관련 벌금이 100억 달러에 달하며 신기록 수립) 데이터가 불완전하더라도 처리 보고서는 계속 진행해야 합니다. 그런가 하면, 데이터 품질 불량 전력이 있어도 "관리 부실"로 인해 벌금이 부과됩니다. 그 결과 수많은 금융 서비스 기관(FSI)은 데이터 품질 불량과 엄격한 SLA 사이에서 데이터 신뢰성이냐, 데이터 적시성이냐를 두고 균형을 잡아야 하는 상황에 직면하게 됩니다.

이 규제 보고 솔루션 액셀러레이터에서는 Delta Live Tables를 이용하면 실시간으로 규제 데이터를 수집, 처리하도록 보장하므로 규제 SLA를 준수하는 데 어떤 면에서 유리한지 설명해 드립니다. 애널리스트는 Delta Sharing과 Delta Live Tables를 함께 사용하여 전송되는 규제 데이터의 품질을 실시간으로 확인할 수 있습니다. 이 글에서는 금융 서비스 업계의 데이터 모델을 클라우드 컴퓨팅의 유연성과 결합하여 높은 수준의 거버넌스 표준을 지원하면서도 개발 오버헤드는 줄이는 레이크하우스 아키텍처의 장점을 알려드립니다. 지금부터는 FIRE 데이터 모델이 무엇인지, Delta Live Tables를 통합하여 건실한 데이터 파이프라인을 구축하려면 어떻게 해야 하는지 설명하겠습니다.

FIRE 데이터 모델

금융 규제 데이터 표준(FIRE)은 금융 분야의 규제 시스템끼리 세분화된 데이터를 전송하는 데 필요한 공통의 규격을 규정한 표준입니다.

규제 데이터란 규제 기관에 제출할 자료, 요구 사항과 계산의 기반이 되는 데이터를 가리키며 정책, 모니터링, 감독 관리 등의 용도로 쓰입니다. FIRE 데이터 표준은 European Commission, Open Data Institute 및 Open Data Incubator FIRE data standard for Europe에서 Horizon 2020 자금 지원 프로그램을 통해 지원됩니다. Databricks에서는 이 솔루션의 일부분으로 FIRE 데이터 모델을 Apache Spark™ 운영 파이프라인으로 해석하는 기능이 있는 PySpark 모듈을 제공하였습니다.

Delta Live Tables

Databricks에서는 최근 신제품 Delta Live Tables 출시 소식을 발표했습니다. 이 제품은 데이터 파이프라인 오케스트레이션용으로, 이를 사용하면 엔터프라이즈급으로 안정적인 데이터 파이프라인을 손쉽게 구축하고 관리할 수 있습니다. 여러 가지 기대치를 평가하고 잘못된 레코드를 실시간으로 폐기하거나 모니터링하는 기능 등 FIRE 데이터 모델을 Delta Live Tables에 통합하면 확실한 장점을 제공합니다. 다음 아키텍처에서 설명한 바와 같이, Delta Live Tables를 사용하면 클라우드 스토리지에 도착하는 세분화된 규제 데이터를 **수집**하고 콘텐츠를 **스키마화**하여 레코드가 FIRE 데이터 규격과 일치하는지 일관성을 **검증**합니다. 이 글을 계속 읽어보시면 Delta Sharing을 사용해 여러 규제 시스템 사이에서 세분화된 정보를 안전하고 확장 가능하며 투명하게 교환하는 방식을 설명한 데모 내용을 확인하실 수 있습니다.

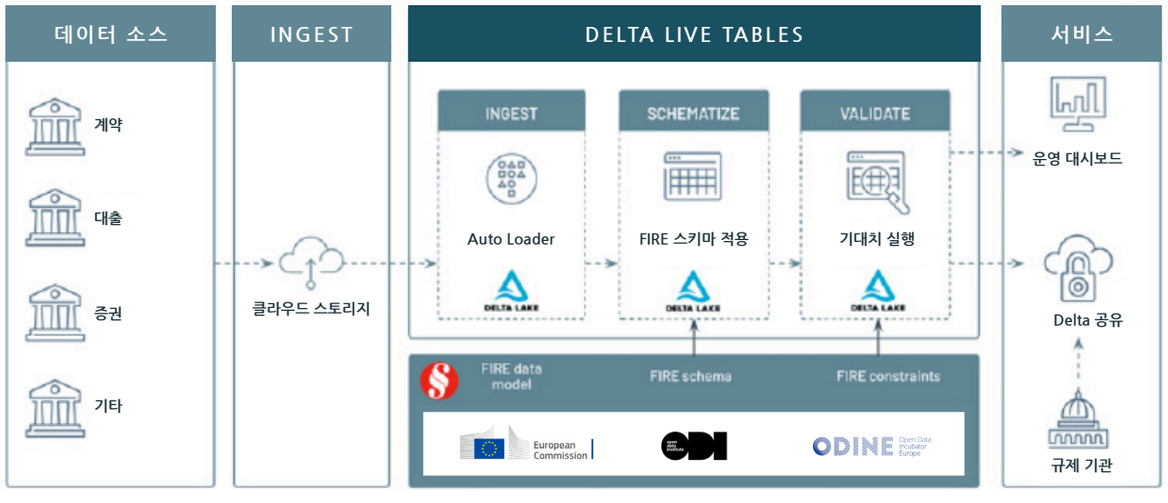

그림 1

스키마 적용

데이터 형식 중에 정형으로 "보이는" 것이 있더라도(예: JSON 파일) 스키마 적용은 엔지니어링에 좋은 관례에만 그치는 것이 아닙니다. 엔터프라이즈 환경에서는(특히 규정 준수라는 부문에서) 스키마를 적용하면 누락된 필드를 예상하고(있는 경우) 예기치 못한 필드는 폐기하며 데이터 유형을 완전히 평가(예: 날짜를 문자열이 아니라 날짜 개체로 취급)하도록 보장할 수 있습니다. 또한 결과적으로 예기치 못하게 데이터 폭주(데이터 드리프트)가 발생할 경우에 대비해 시스템의 내구력을 테스트하는 효과도 있습니다. 이 글에서는 FIRE PySpark 모듈을 사용하여 주어진 FIRE 엔터티(이 예제에서는 collateral entity)를 처리하는 데 필요한 Spark 스키마를 프로그램 방식으로 가져와서 원시 레코드 스트림에 적용합니다.

```
from fire.spark import FireModel

fire_model = FireModel().load("collateral")

fire_schema = fire_model.schema
```

아래의 예시에서는 수신되는 CSV 파일에 스키마를 적용합니다. @dlt 어노테이션을 사용해 이 프로세스를 데코레이팅하면 Delta Live Tables에 대한 진입점을 정의하고, 마운트된 디렉터리에서 원시 CSV 파일을 읽을 수 있으며 스키마화한 레코드를 브론즈 계층에 쓸 수 있습니다.

```
@dlt.create_table()
def collateral_bronze():
  return (
    spark
      .readStream
      .option("maxFilesPerTrigger", "1")
      .option("badRecordsPath", "/path/to/invalid/collateral")
      .format("csv")
      .schema(fire_schema)
      .load("/path/to/raw/collateral")
```

기대치 평가

스키마 적용과 그 스키마의 제약 조건 적용은 별개의 사안입니다. FIRE 엔터티의 스키마 정의를 보면(부수적 스키마 정의 예시 참조) 주어진 필드가 필수 필드인지 아닌지 탐지할 수 있습니다. 열거 개체를 보고는 그 값이 일관적인지 확인합니다(예: 통화 코드). FIRE 모델은 스키마의 기술적 제약 조건 외에 최솟값, 최댓값, 금액과 maxItem 등의 비즈니스 기대치도 보고합니다. 이 모든 기술적, 비즈니스 제약 조건을 FIRE 모델에서 프로그램 방식으로 가져와 일련의 Spark SQL 표현식으로 해석하는 것입니다.

```
from fire.spark import FireModel

fire_model = FireModel().load("collateral")

fire_constraints = fire_model.constraints
```

Delta Live Tables를 사용하면 사용자가 여러 개의 기대치를 한꺼번에 평가할 수 있으므로 잘못된 레코드는 삭제하고, 단순히 데이터 품질을 모니터링할 수도 있고 파이프라인 전체를 중단할 수도 있습니다. 이 글에서는 기대에 어긋나는 레코드를 모두 삭제하고자 합니다. 이것은 나중에 격리 테이블에 저장합니다(이 블로그에 제공한 노트북의 보고 내용 참조).

```
@dlt.create_table()

@dlt.expect_all_or_drop(fire_constraints)

def collateral_silver():

  return dlt.read_stream("collateral_bronze")
```

겨우 코드 몇 줄만으로 구문론 면에서나(유효한 스키마) 시맨틱 면에서나(유효한 기대치) 올바른 실버 테이블을 완성했습니다. 아래 표시한 것과 같이, 규정 준수 담당자는 처리 중인 레코드 수를 실시간으로 파악할 수 있습니다. 이 예시의 경우, collateral entity의 완료율을 정확히 92.2%로 맞추었습니다(나머지 7.8%는 격리 작업이 처리).

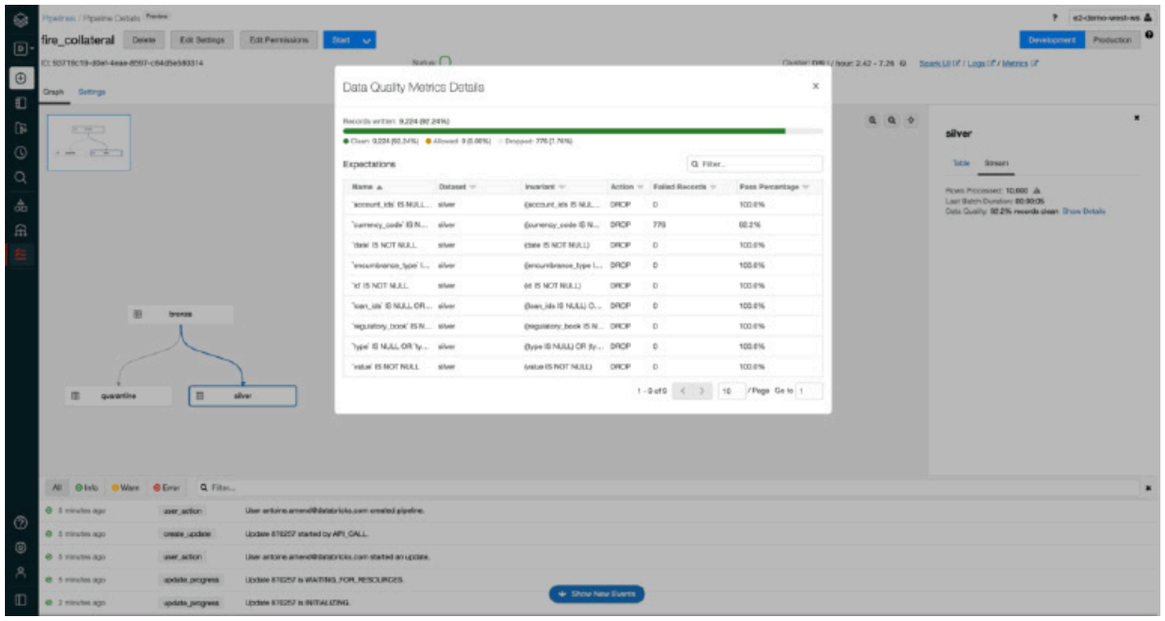

그림 2

운영 데이터 스토어

Delta Live Tables는 실제 데이터를 Delta 파일로 저장할 뿐만 아니라, 운영 지표도 시스템/이벤트 아래에 “delta” 형식으로 저장합니다. 이 글에서는 레이크하우스 아키텍처의 일반적인 패턴을 따릅니다. 즉 Auto Loader를 사용해 새 운영 지표를 “구독”하여 새 지표가 출현하는 대로 시스템 이벤트를 배치 형식이나 실시간으로 처리하는 것입니다. Delta Lake에는 트랜잭션 로그가 있어 각종 데이터 업데이트 최신 정보를 입수하므로, 기업에서 자체적으로 체크포인팅 프로세스를 구축하고 유지 관리하지 않아도 새 지표에 액세스할 수 있습니다.

```
input_stream = spark \
    .readStream \
    .format(“delta”) \
    .load(“/path/to/pipeline/system/events”)

output_stream = extract_metrics(input_stream)

output_stream \
    .writeStream \
    .format(“delta”) \
    .option(“checkpointLocation”, “/path/to/checkpoint”) \
    .table(metrics_table)
```

이용 가능한 지표가 모두 운영 스토어 한 곳에 중앙집중화되어 있으므로, 애널리스트는 Databricks SQL을 사용하여 단순한 대시보딩 기능을 만들거나, 좀 더 복잡한 알림 방식을 만들어 데이터 품질 문제를 실시간으로 탐지할 수 있습니다.

	entity	expectation_name	expectation_value
1	adjustment	[id] is mandatory	`id` IS NOT NULL
2	adjustment	[date] is mandatory	`date` IS NOT NULL
3	adjustment	[col] is mandatory	`col` IS NOT NULL
4	adjustment	[contribution_amount] is mandatory	`contribution_amount` IS NOT NULL
5	adjustment	[currency_code] is mandatory	`currency_code` IS NOT NULL
6	adjustment	[currency_code] not allowed value	(`currency_code` IS NULL) OR (`currency_code` IN ('AED', 'AFN', 'ALL', 'AMD', 'ANG', 'AOA', 'ARS', 'AUD', 'AWG', 'AZN', 'BAM', 'BBD', 'BDT', 'BGN', 'BHD', 'BIF', 'BMD', 'BND', 'BOB', 'BOV', 'BRL', 'BSD', 'BTN', 'BWP', 'BYN', 'BZD', 'CAD', 'CDF', 'CHE', 'CHF', 'CHW', 'CLF', 'CLP', 'CNY', 'COP', 'COU', 'CRC', 'CUC', 'CUP', 'CVE', 'CZK', 'DJF', 'DKK', 'DOP', 'DZD', 'EGP', 'ERN', 'ETB', 'EUR', 'FJD', 'FKP', 'GBP', 'GEL', 'GHS', 'GIP', 'GMD', 'GNF', 'GTQ', 'GYD', 'HKD', 'HNL', 'HRK', 'HTG', 'HUF', 'I...

그림 3

Delta Lake 형식의 불변성 측면에 Delta Live Tables가 제공하는 데이터 품질 투명성을 함께 활용하면 금융 기관이 규정 준수를 위해 필요한 데이터 버전(볼륨, 품질 양쪽 다 일치하는 버전)으로 “시간 이동”할 수 있습니다. 이 예시에서는 격리 테이블에 저장된 7.8%의 잘못된 레코드를 재생하면 실버 테이블에 연결된 다른 Delta 버전이 생성됩니다. 이 버전을 여러 규제 기관에서 공유할 수 있습니다.

```
DESCRIBE HISTORY fire.collateral_silver
```

규제 데이터 전송

데이터의 품질과 볼륨을 둘 다 완벽히 확신할 수 있으면 금융 기관에서는 엔터프라이즈 데이터 교환을 위한 오픈 프로토콜인 Delta Sharing을 사용하여 여러 규제 시스템을 오가며 안전하게 정보를 교환할 수 있습니다. Delta Lake는 본질적으로 오픈 소스이기 때문에 최종 사용자를 모두 같은 플랫폼 안에 제한하거나 복잡한 ETL 파이프라인에 의지해 데이터를 사용할 필요 없이(예컨대 SFTP 서버를 통해 데이터 파일에 액세스), 데이터 사용자가 Python, Spark에서 기본적으로 스키마화된 데이터에 액세스하거나 MI/BI 대시보드(예: Tableau 또는 power BI)를 통해 직접 액세스할 수 있게 해줍니다.

물론 실버 테이블을 있는 그대로 공유할 수도 있지만, 미리 정의된 데이터 품질 임곗값에 부합하는 경우에만 규제 데이터를 공유하도록 하는 비즈니스 규칙을 사용하고자 할 수도 있습니다. 이 예시에서는 실버 테이블을 다른 버전에서 복제하며, 대상 위치도 내부 네트워크 및 최종 사용자가 액세스할 수 있는 위치와는 분리된 특정 위치(일명 비무장지대, 즉 DMZ)로 설정합니다.

```
from delta.tables import *

deltaTable = DeltaTable.forName(spark, "fire.collateral_silver")
deltaTable.cloneAtVersion(
  approved_version,
  dmz_path,
  isShallow=False,
  replace=True
)

spark.sql(
  "CREATE TABLE fire.colleral_gold USING DELTA LOCATION '{}'"
    .format(dmz_path)
)
```

Delta Sharing 오픈 소스 솔루션은 권한을 관리하는 데 공유 서버를 주로 이용하지만, Databricks에서는 Unity Catalog를 사용하여 액세스 제어 정책을 중앙 집중화 및 적용하고, 사용자에게 온전한 감사 로그 기능을 제공하며 SQL 인터페이스를 통해 액세스 관리를 간소화합니다. 아래의 예시에서는 규제 테이블을 포함하는 SHARE, 데이터를 공유할 RECIPIENT를 만들었습니다.

```
-- DEFINE OUR SHARING STRATEGY
CREATE SHARE regulatory_reports;

ALTER SHARE regulatory_reports ADD TABLE fire.collateral_gold;
ALTER SHARE regulatory_reports ADD TABLE fire.loan_gold;
ALTER SHARE regulatory_reports ADD TABLE fire.security_gold;
ALTER SHARE regulatory_reports ADD TABLE fire.derivative_gold;

-- CREATE RECIPIENTS AND GRANT SELECT ACCESS
CREATE RECIPIENT regulatory_body;

GRANT SELECT ON SHARE regulatory_reports TO RECIPIENT regulatory_body;
```

권한이 부여된 규제 기관이나 사용자는 이 프로세스를 통해 교환된 개인 액세스 토큰을 사용하여 기본 데이터에 액세스할 수 있습니다. Delta Sharing에 대한 자세한 내용은 제품 페이지를 방문하여 Databricks 담당자에게 문의하세요.

규정 준수 검증 테스트

이러한 노트북 시리즈와 Detla Live Tables 작업에서는 규제 데이터 수집, 처리, 검증 및 전송 부문에서 레이크하우스 아키텍처의 장점을 보여드리고 있습니다. 구체적으로는, 규제 파이프라인의 일관성, 무결성과 적시성을 보장해야 하는 기업의 요구 사항을 다루었습니다. 이 요구 사항은 공용 데이터 모델(FIRE)과 유연한 오케스트레이션 엔진(Delta Live Tables)을 함께 사용하면 손쉽게 부합할 수 있습니다. 마지막으로 FSI에서 Delta Sharing 기능을 이용하면 다양한 규제 시스템 사이에서 교환되는 규제 데이터에 투명성과 신뢰성을 보장할 수 있고 그러면서도 보고 요건에 부합하므로 운영 비용을 절감하고 새로운 표준에 맞춰 적응할 수도 있다는 사실을 설명했습니다. 여기에 첨부한 노트북을 사용하여 FIRE 데이터 파이프라인을 익혀보세요. 솔루션 액셀러레이터 허브에서는 금융 서비스 부문에 적합한 최신 솔루션 관련 정보를 확인할 수 있습니다.

무료 Databricks 노트북으로 실험을 시작하세요.

databricks

섹션 2.5

Databricks 레이크하우스 플랫폼을 사용한 대규모 AML 솔루션

글: SRI GHATTAMANENI, RICARDO PORTILLA, ANINDITA MAHAPATRA

2021년 7월 16일 금요일

금융 범죄 솔루션 구축에 따르는 주요 문제 해결

자금세탁방지법(Anti-money Laundering, AML) 규정 준수는 이론의 여지 없이 전 세계에서 금융 기관 감독을 담당하는 규제 기관의 주요 안건 중 하나입니다. 수십 년을 거치며 AML이 발전하고 점점 더 복잡해지면서 최신 자금세탁 기법과 테러리스트 자금 지원 수법에 맞서기 위해 고안한 규제 요건도 함께 복잡해졌습니다. 1970년에 제정된 미국 은행 비밀법(Bank Secrecy Act)은 금융 기관에서 금융 거래를 모니터링하고 의심스러운 금융 관련 활동을 관련 권한 당국에 신고하기 위해 적절한 관리 수단을 마련하기 위한 지침과 프레임워크를 규정한 법입니다. 이 법률에 따라 금융 기관이 자금세탁과 금융 테러리즘에 맞서는 방식의 프레임워크가 형성되었습니다.

자금세탁방지법이 복잡한 이유

최신 AML 업무는 지난 십 년간의 이전 업무수행 방식과 완연히 다릅니다. 디지털 뱅킹으로의 전환이 이루어지고, 금융 기관(FI)에서 처리하는 거래량이 일일 수십억 건에 달하게 되면서 자금세탁 범위도 역대 가장 크게 확대되었으며, 거래 모니터링 시스템을 더욱 엄격하게 강화하고 강력한 KYC(Know Your Customer) 솔루션을 적용해도 이를 제재하기는 역부족이었습니다. 이 블로그에서는 FI 고객과 협력하여 레이크하우스 플랫폼에서 엔터프라이즈급 AML 솔루션을 구축한 경험을 공유합니다. 이 솔루션과 플랫폼 모두 강력한 감독 기능을 제공하며, 혁신적이고 확장 가능한 솔루션을 통해 최신 온라인 자금세탁 위협의 현실에 맞게 대응하게 해줍니다.

레이크하우스를 사용한 AML 솔루션 구축

하루에 수십억 건의 거래를 처리해야 하는 운영상의 부담은 여러 가지 소스와 컴퓨팅 파워 집약적인 차세대 AML 솔루션에서 가져온 데이터를 저장해야 하기 때문에 발생합니다. 이러한 솔루션은 강력한 리스크 분석, 보고 기능을 제공하면서도 지능형 머신 러닝 모델을 지원하여 오탐률을 줄이고 다운스트림 조사 효율을 개선해주는 효과가 있습니다. FI에서는 이미 온프레미스에서 클라우드로 전환하여 인프라와 확장 문제를 해결하기 위한 단계를 밟기 시작했습니다. 이로써 보안, 민첩성을 개선하고 대량의 데이터를 저장하는 데 필요한 규모의 경제를 확보하고자 한 것입니다.

그런데 이것만으로는 저가의 Object storage에 수집하여 저장한 대량의 정형, 비정형 데이터의 의미를 어떻게 이해해야 하는가, 라는 문제를 해결할 수 없습니다. 클라우드 공급업체에서 저렴한 비용으로 데이터를 저장할 방안을 제공하기는 하지만, 다운스트림 AML 리스크 관리 및 규정 준수 활동을 위해 데이터를 이해하려면 다운스트림 사용을

염두에 두고 데이터를 고품질의, 성능에 맞는 형식으로 저장하는 것부터 시작해야 합니다. Databricks 레이크하우스 플랫폼이 바로 그런 기능을 제공합니다. 데이터 레이크의 장점인 저렴한 스토리지 비용에 데이터 웨어하우스의 강력한 트랜잭션 기능을 결합하여 FI에서 진정한 의미의 최신 AML 플랫폼을 구축할 수 있게 지원하는 것입니다.

AML 애널리스트는 위에서 간단하게 언급한 데이터 스토리지 문제 외에도 몇 가지 분야별 난관에 직면하게 됩니다.

- 이미지, 텍스트 데이터 및 네트워크 링크와 같은 비정형 데이터 파싱 TTV(Time-to-value) 단축
- DevOps의 중요 ML 기능(예: 엔터티 확인, 컴퓨터 비전 및 엔터티 메타데이터 그래프 분석 등) 지원 부담 완화
- AML 거래와 강화 테이블에 분석 엔지니어링과 대시보딩 계층을 도입하여 사일로 허물기

다행히 Databricks를 이용하면 Delta Lake를 활용해 엔터티 관계를 구축하는 데 정형 데이터와 비정형 데이터를 둘 다 저장하고 조합하므로 이 문제를 해결하는 데 도움이 됩니다. 또한, Databricks Delta Engine을 사용하면 최신 Photon 컴퓨팅을 사용해 테이블에서 BI 쿼리 속도를 높여주므로 효율적인 액세스가 제공됩니다. 이 모든 기능 외에도 레이크하우스의 주인공은 ML이므로, 애널리스트와 데이터 사이언티스트가 서브샘플링이나 데이터를 공유 대시보드로 옮기는 작업에 시간을 낭비할 필요 없이 공격자보다 한발 앞설 수 있습니다.

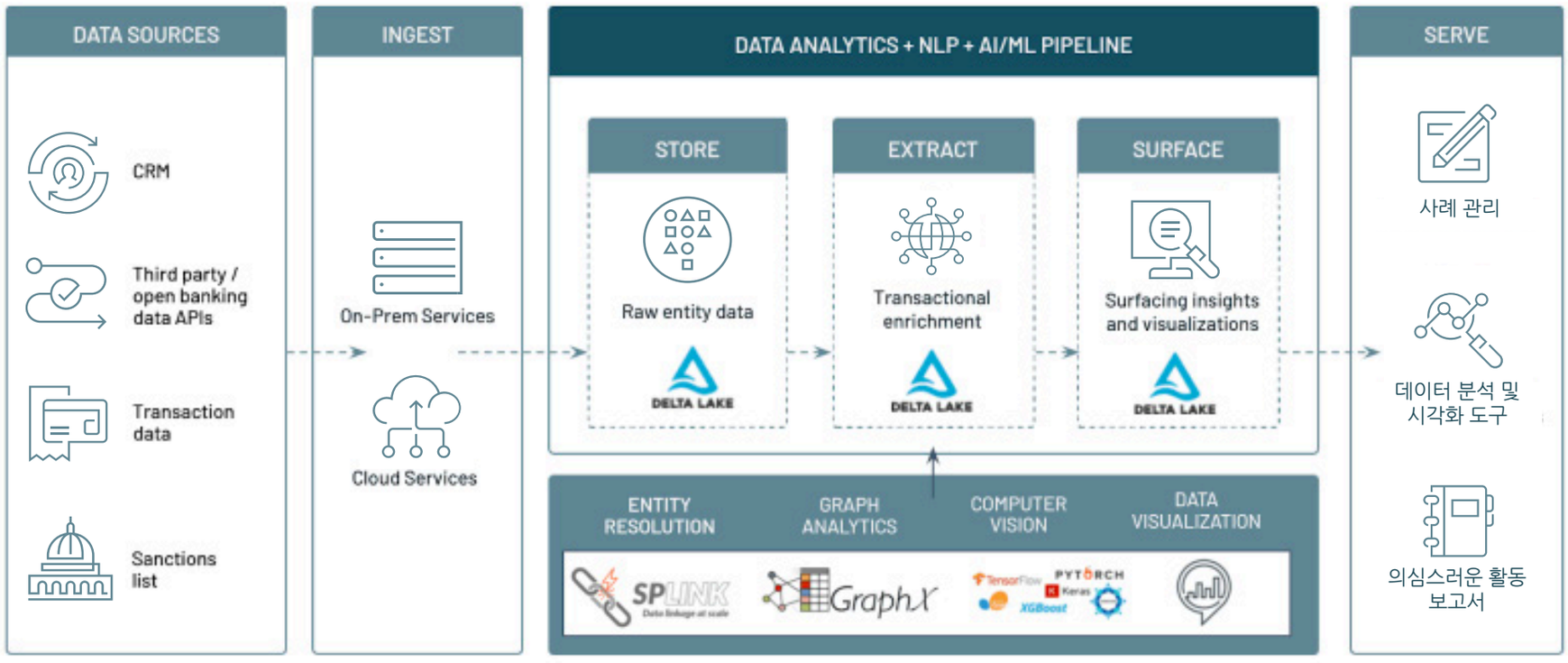

그림 1

그래프 기능으로 AML 패턴 탐지

AML 애널리스트가 사례의 일부분으로 사용하는 주요 데이터 소스는 보통 *거래 데이터*입니다. 이 데이터는 테이블 형식이고 SQL을 사용하면 손쉽게 액세스할 수 있는 것이 사실이지만, SQL 쿼리를 포함하며 3개 이상의 계층으로 이루어진 심층적인 거래망을 추적하기는 점점 부담스러워집니다. 이 때문에 다양하고 유연한 언어와 API를 확보하여, 불법적인 거래에 연루된 의심스러운 인물들로 이루어진 연결형 네트워크와 같은 간단한 개념을 표현할 수 있어야 합니다. 다행히 이것은 GraphFrames를 사용하면 간단히 해결할 수 있습니다. 이 그래프 API는 Databricks Runtime for Machine Learning에 미리 설치되어 있습니다.

이 섹션에서는 위조 ID 및 계층화/구조화 등의 AML 수법을 탐지하는 데 그래픽 분석을 어떤 식으로 활용할 수 있는지 보여드립니다. 여기에서는 여러 개의 거래로 구성된 데이터 세트 하나와 거래에서 파생된 엔터티를 활용하여 이러한 패턴의 유무를 탐지하기 위해 Apache Spark™, GraphFrames와 Delta Lake를 이용합니다. 유지된 패턴은 Delta Lake에 저장하여 이와 같은 조사 결과의 골드 레벨 집계 버전에 Databricks SQL을 적용합니다. 이렇게 하면 최종 사용자에게 그래프 분석 기능을 제공할 수 있습니다.

시나리오 1 — 위조 ID

앞서 언급했듯이 위조 ID가 있다면 경계 대상인 상황일 가능성이 있습니다. 그래프 분석을 사용하면 거래의 엔터티 전체를 일괄 분석하여 리스크 수준을 알아낼 수 있습니다. 이 글에서 제시한 분석에서는 이 작업을 세 단계로 나누어 수행하였습니다.

- 거래 데이터를 기반으로 엔터티 추출
- 주소, 전화번호 또는 이메일을 기반으로 엔터티 간 링크 생성
- GraphFrames로 연결한 구성 요소를 사용해 여러 개의 엔터티(ID 및 위에 언급한 다른 여러 속성으로 식별)가 하나 이상의 링크를 통해 연결되어 있는지 판별

엔터티 간에 연결이 몇 개나 되는지에 따라(예: 공통 속성) 리스크 점수를 높거나 낮게 할당합니다. 그런 다음 점수가 높은 그룹을 기준으로 알림을 만들면 됩니다. 아래에 이 개념을 간략하게 나타내었습니다.

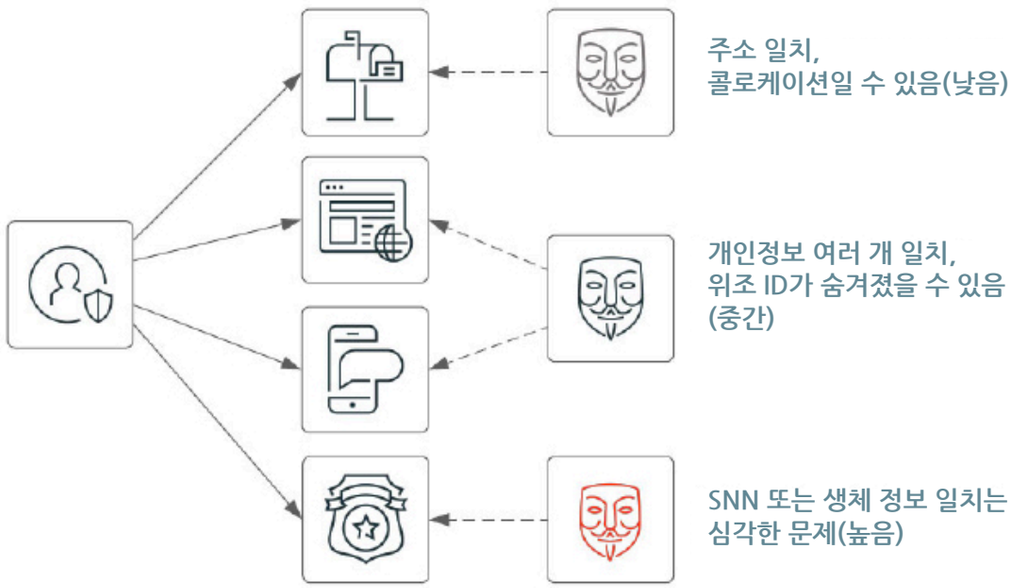

그림 2

우선 주소, 이메일과 전화번호를 사용해 ID 그래프를 만들어 개개인을 연결하여 이러한 속성 중에 일치하는 것이 있는지 확인합니다.

```
e_identity_sql = '''
select entity_id as src, address as dst from aml.aml_entities_synth  where address is not
null
UNION
select entity_id as src, email as dst from aml.aml_entities_synth  where email_addr is not
null
UNION
select entity_id as src, phone as dst from aml.aml_entities_synth  where phone_number is not
null
'''

from graphframes import *
from pyspark.sql.functions import *
aml_identity_g = GraphFrame(identity_vertices, identity_edges)
result = aml_identity_g.connectedComponents()

result \
 .select("id", "component", 'type') \
 .createOrReplaceTempView("components")
```

다음으로, 쿼리를 실행해 두 엔터티의 ID와 점수가 겹치는 시점이 언제인지 파악합니다. 이러한 쿼리 그래프 구성요소의 결과에 따르면 일치하는 속성이 하나뿐인(예를 들어 주소 하나만 일치) 코호트가 도출될 것을 예상할 수 있으며, 이 경우 크게 걱정할 일은 아닙니다. 다만, 일치하는 속성이 많을수록 경계 수준을 올려야 합니다. 아래에 표시한 것과 같이, 속성 세 개가 모두 일치하는 사례에 플래그를 지정하면 SQL 애널리스트에게 모든 엔터티에 실행한 그래프 분석 결과를 매일 제공할 수 있습니다.

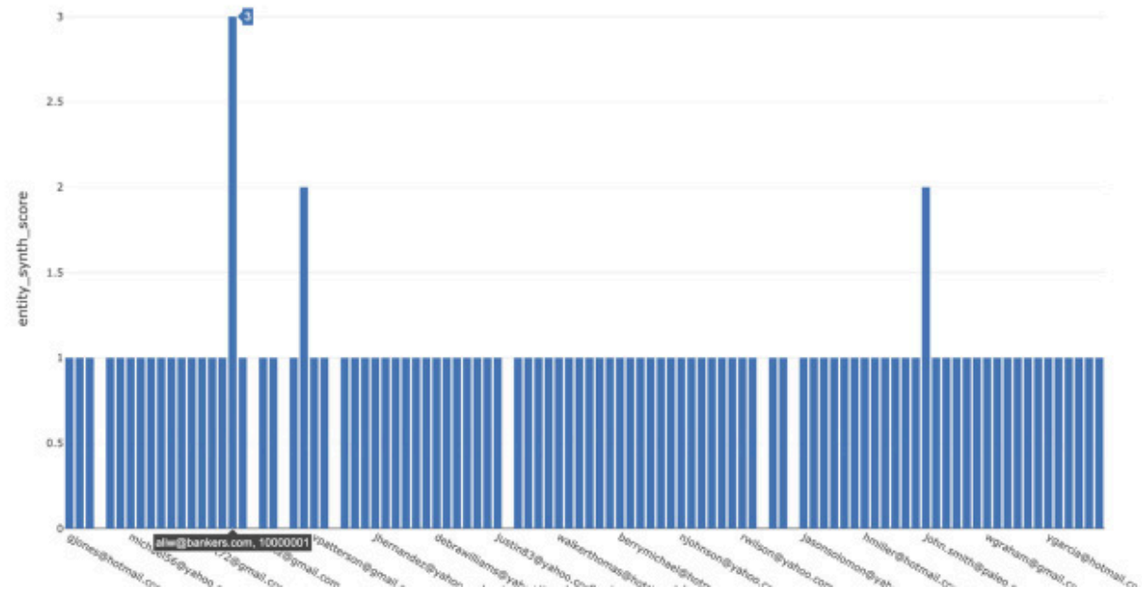

그림 3

시나리오 2 — 구조화

또 다른 보편적인 패턴으로 일명 *구조화(structuring)*라는 것이 있습니다. 이것은 여러 엔터티가 공모하여 "암암리에" 소액을 여러 은행에 나누어 보내고, 이를 집계한 더 큰 금액을 최종 기관에 우회하여 보내는 방식을 말합니다(아래 맨 오른쪽 그림 참조). 이 경우 관계자 모두 10,000달러 미만을 유지했는데, 이는 보통 권한 당국에 알림이 가는 금액 한도입니다. 그래프 분석을 사용하면 이 작업을 손쉽게 해결할 수 있을 뿐만 아니라, *모티프 검색 기법*을 자동화하여 다른 네트워크 순열로 확장하고, 같은 방식으로 다른 의심스러운 거래 위치를 확인할 수도 있습니다.

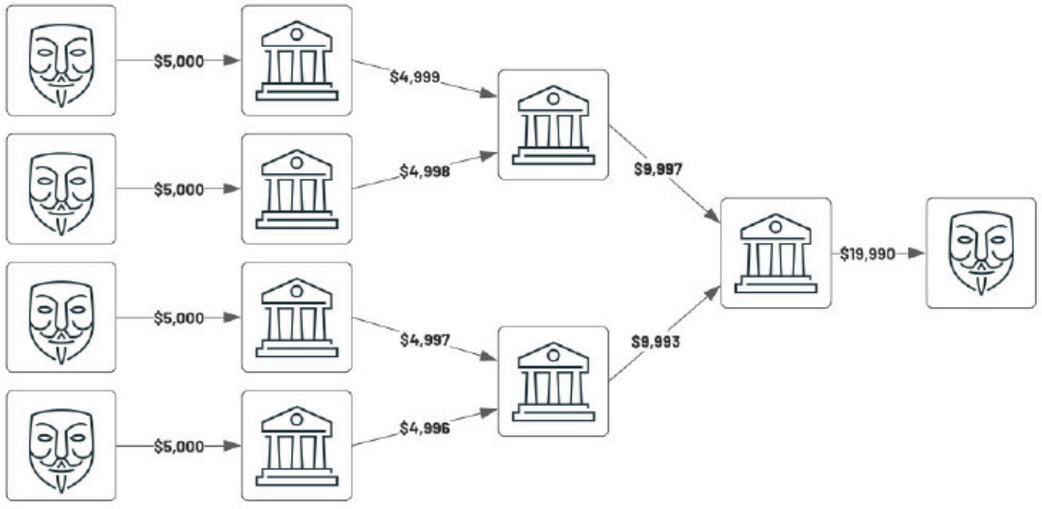

그림 4

이제 기본적인 모티프 검색 코드를 작성하여 그래프 기능을 사용해 위와 같은 상황을 탐지해 보겠습니다. 여기서 출력되는 것은 반정형 JSON입니다. 비정형 유형을 포함한 모든 데이터 유형을 레이크하우스에서 간편하게 액세스할 수 있으며, 이러한 결과는 SQL 보고용으로 저장하겠습니다.

```
motif = "(a)-[e1]->(b); (b)-[e2]->(c); (c)-[e3]->(d); (e)-[e4]->(f); (f)-[e5]-
>(c); (c)-[e6]->(g)"
struct_scn_1 = aml_entity_g.find(motif)

joined_graphs = struct_scn_1.alias("a") \
 .join(struct_scn_1.alias("b"), col("a.g.id") == col("b.g.id")) \
 .filter(col("a.e6.txn_amount") + col("b.e6.txn_amount") > 10000)
```

모티프 검색을 사용한 결과 흥미로운 패턴이 추출되었습니다. 총 네 가지 엔터티를 통해 자금이 이동하는데 10,000 한도 미만을 유지한 것입니다. 그래프 메타데이터는 도로 정형 데이터 세트에 조인하여 AML 애널리스트가 심층 조사할 인사이트를 생성하였습니다.

	top_entity_id	first_entity	second_entity	third_entity	fourth_entity
1	1	Brenda Thomas	Teresa Gibson	Mary Strong	Robert Wilkinson
2	3	Lindsey Barber	Joshua Harris	Mary Strong	Robert Wilkinson
3	5	Bruce White	Kathleen Elliott	Victor Arias	Robert Wilkinson
4	7	Jeffrey Lara	Amy Campbell	Victor Arias	Robert Wilkinson

그림 5

시나리오 3 — 리스크 점수 전달

리스크가 높은 것으로 확인된 엔터티는 자신이 속한 집단에 영향(네트워크 효과)을 미칩니다. 따라서 이들과 상호작용하는 모든 엔터티의 리스크 점수도 영향권을 반영해 조정해야 합니다. 반복 방식을 사용하여 거래 흐름을 심층적인 부분까지 따라갈 수 있으며, 네트워크에서 영향을 받은 다른 엔터티의 리스크 점수를 조정하면 됩니다. 앞서 언급했듯이 그래프 분석을 실행하면 SQL 조인을 여러 번 반복할 필요가 없고 복잡한 비즈니스 로직도 필요 없으므로 메모리 제약 사항으로 인한 성능 저하도 발생하지 않습니다. 그래프 분석과 Pregel API는 바로 이러한 목적을 위해 개발되었습니다. Pregel은 원래 Google에서 개발한 API로, 사용자가 각종 꼭짓점에서 그에 상응하는 이웃에게 메시지를 반복해서 "전달"하도록 허용하여 각 단계에서 꼭짓점 상태(여기서는 리스크 점수)를 업데이트합니다. Pregel API를 사용한 동적 리스크 방식은 다음과 같이 표현할 수 있습니다.

그림 6

왼쪽 아래 다이어그램에 네트워크의 시작 상태와 이후의 반복 두 번을 표시하였습니다. 리스크 점수가 10점인 공격자 한 사람(노드 3)부터 시작했다고 칩시다. 해당 노드와 거래하고(노드 4, 5 및 6) 자금을 받은 모든 인물을 처벌하고자 합니다. 예를 들어 공격자의 리스크 점수의 절반을 전달하여 이를 각자의 기본 점수에 추가하는 것입니다. 다음 반복에서는 노드 4, 5와 6의 다운스트림인 모든 노드에서 점수를 조정합니다.

노드 #	반복 #0	반복 #1	반복 #2
1	0	0	0
2	0	0	0
3	10	10	10
4	0	5	5
5	0	5	5
6	0	5	5
7	0	0	5
8	0	0	0
9	0	0	2.5
10	0	0	0

GraphFrame에서 Pregel API를 사용하면 이 연산을 수행하고 수정된 점수를 다운스트림의 다른 애플리케이션에서 사용하도록 유지할 수 있습니다.

```
from graphframes.lib import Pregel

ranks = aml_entity_g.pregel \
    .setMaxIter(3) \
    .withVertexColumn(
       “risk_score”,
       col(“risk”),
       coalesce(Pregel.msg()+ col(“risk”),
       col(“risk_score”))
    ) \
    .sendMsgToDst(Pregel.src(“risk_score”)/2 )  \
    .aggMsgs(sum(Pregel.msg())) \
    .run()
```

주소 일치

여기서 간략하게 다루고자 하는 패턴은 텍스트와 실제 거리 뷰 이미지를 대조하는 주소 일치입니다. AML 애널리스트가 파일에 기재된 엔터티와 연결된 주소의 타당성을 확인해야 할 때가 종종 있습니다. 이 주소는 상업용 건물인가? 아니면 주택가나 단순한 사서함인가? 단, 사진을 분석하는 작업은 번거롭고 시간이 오래 걸리며 획득, 정리하고 검증까지 손이 많이 가는 일일 때가 많습니다. 이럴 때 레이크하우스 데이터 아키텍처를 사용하면 Python과 ML 런타임을 PyTorch 및 미리 훈련한 오픈 소스 모델을 사용해 이 태스크의 대부분을 자동화할 수 있습니다. 아래에 사람의 눈으로 확인한 유효한 주소를 예로 들었습니다. 검증을 자동화하기 위해, 미리 훈련한 VGG 모델을 사용해보겠습니다. 여기에는 주택을 탐지하는 데 사용할 수 있는 유효한 개체가 수백 개 해당합니다.

그림 7

이번에는 아래의 코드를 사용하여(매일 실행하도록 자동화 가능) 모든 이미지에 레이블을 연결해 보겠습니다. 이미지 참조와 레이블을 모두 SQL 테이블에 로드하였으므로 쿼리도 더욱 단순합니다. 아래 코드를 보면 일련의 이미지를 그 안에 포함된 사물에 대하여 쿼리하기가 얼마나 간단한지 알 수 있습니다. Delta Lake를 사용해 그러한 비정형 데이터를 쿼리할 수 있으면 애널리스트에게는 엄청난 시간 절약 효과가 있으며, 검증 프로세스를 일 단위, 주 단위가 아니라 분 단위로 단축할 수 있습니다.

```
from PIL import Image
from matplotlib import cm

img = Image.fromarray(img)
...

vgg = models.vgg16(pretrained=True)
prediction = vgg(img)
prediction = prediction.data.numpy().argmax()
img_and_labels[i] = labels[prediction]
```

요약을 시작하면서 몇 가지 흥미로운 카테고리를 발견했습니다. 아래 분석 결과에서 보듯이, 주택 주소에서 탐지될 것으로 예상한 항목인 *파티오, 이동식 주택*과 *스쿠터* 등의 뻔한 레이블이 있었습니다. 그런데 CV 모델은 한 이미지의 주변 사물 중에서 태양열 집열기에 레이블을 지정했습니다. (*참고: 이 경우 맞춤형 이미지 세트에서 훈련하지 않은 오픈 소스 모델로만 제한하였으므로 태양열 집열기 레이블은 정확하지 않습니다.*) 이미지를 심층 분석한 결과, 드릴다운을 통해 i) 실제로 태양열 집열기가 있는 것이 아니고, 더 중요한 사실은 ii) 이 주소는 실제 주택이 아니라는 사실(그림 7의 일대일 비교 참조)을 곧바로 확인할 수 있었습니다. Delta Lake 형식을 사용하면 비정형 데이터에 참조를 저장하여 (레이블도 포함) 아래와 같은 분류 분석을 단순하게 쿼리할 수 있습니다.

Address Validation

```
%sql

select label, count(1) from aml.address_tags group by label
```

▸ (2) Spark Jobs

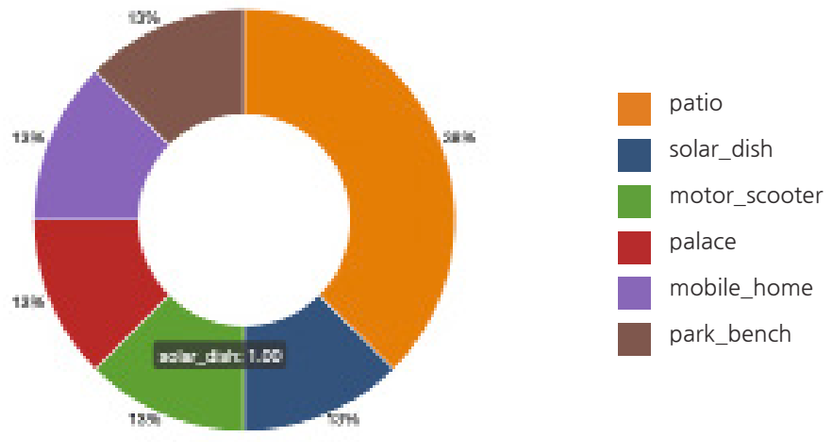

그림 8

Image Name	Rendered Image	Main Object	Risk Level
img_0.jpg		Patio	Low
img_1.jpg		Solar Dish	High

그림 9

엔터티 확인

이 글에서 다룰 AML 문제 중 마지막으로 엔터티 확인이라는 카테고리에 주안점을 두고자 합니다. 이 문제는 대다수의 오픈 소스 라이브러리에서도 다루기 때문에, 몇몇 기본적인 엔터티 퍼지 매칭의 경우 Splink를 강조하기로 했습니다. Splink는 대규모로 연결을 설정하며 여러 가지 구성을 제공하여 일치하는 열과 차단 규칙을 지정하게 해줍니다.

이 예시의 거래에서 파생한 엔터티라는 맥락에서는 아주 간단하게 Delta Lake 트랜잭션을 Splink에 맞게 입력할 수 있습니다.

```
settings = {
  “link_type”: “dedupe_only”,
  “blocking_rules”: [
      “l.txn_amount = r.txn_amount”,
  ],
  “comparison_columns”: [
      {
          “col_name”: “rptd_originator_address”,
      },
      {
          “col_name”: “rptd_originator_name”,
      }
  ]
}

from splink import Splink
linker = Splink(settings, df2, spark)
df2_e = linker.get_scored_comparisons()
```

Splink의 작동 원리는 엔터티 속성이 서로 매우 유사한 트랜잭션을 식별하는 데 사용할 수 있는 일치 확률을 할당하여 보고된 주소, 엔터티 이름이나 거래 금액과 관련해 알림을 발생시키는 것입니다. 계정 정보 매칭을 위한 엔터티 확인에는 손이 무척 많이 가므로, 이 태스크를 자동화해주고 Delta Lake에 정보를 저장해주는 오픈 소스 라이브러리를 확보하면 훨씬 생산적으로 사례를 해결할 수 있게 됩니다. 엔터티 매칭에는 사용 가능한 방식 면에서 선택의 폭이 넓지만, 이 글에서는 LSH(Locality-Sensitive Hashing)를 사용하여 작업에 적절한 알고리즘을 찾는 것을 적극 권장합니다. LSH와 그 장점에 대한 자세한 내용은 이 블로그 게시물을 참조하세요.

위에서 보고한 바와 같이, NY Mellon 은행 주소에서 몇 가지 불일치가 발견되었습니다. “Canada Square, Canary Wharf, London, United Kingdom”이라는 주소는 “Canada Square, Canary Wharf, London, UK”와 유사합니다. 중복을 제거한 레코드를 도로 Delta 테이블에 저장하면 이를 나중에 AML 조사에 사용할 수 있습니다.

unique_id_l	unique_id_r	rptd_originator_address_l	rptd_originator_address_r
223254	223256	Canada Square, Canary Wharf, London, United Kingdom	Canada Square, Canary Wharf, London, UK

그림 10

AML 레이크하우스 대시보드

Databricks SQL은 레이크하우스에서 기존 데이터 웨어하우스와 관련한 간극을 메우는 역할을 하고 있습니다. 예를 들어 데이터 관리를 간소화하고 새로운 쿼리 엔진 Photon으로 성능을 강화하며 사용자 동시성을 보장합니다. 이런 특징이 중요한 이유는 대부분의 기업에는 금융 범죄에 맞서는 데 도움이 되는 무수히 많은 사용 사례(예: 테러 자금 조달 금지(CFT) 등)를 지원하기 위해 고가의 상용 AML 소프트웨어를 도입할 예산이 없기 때문입니다. 현재 시중에서는 위의 그래프 분석을 수행할 수 있는 전용 솔루션, 웨어하우스에서 BI를 해결하는 전용 솔루션, ML 전용 솔루션을 구할 수 있습니다. AML 레이크하우스 디자인은 이 세 가지를 전부 합쳐줍니다. AML 데이터 플랫폼 팀에서는 Delta Lake를 클라우드 스토리지 수준의 저렴한 비용으로, 오픈 소스 기술을 손쉽게 통합하여 그래프 기술, 컴퓨터 비전 및 SQL 분석 엔지니어링에 따라 구성된 보고서를 작성할 수 있습니다. 그림 11에 AML 보고 기능을 구체적으로 표시하였습니다.

첨부한 노트북으로 트랜잭션 개체, 엔터티 개체는 물론 구조화 가능성, 위조 ID 및 주소 분류 등 요약도 생성했습니다. 이 경우 미리 훈련한 모델을 사용하였습니다. 아래 Databricks SQL 시각화에서는 Photon SQL 엔진을 사용하여 이에 대한 요약을 실행하고 기본 내장 시각화를 사용해 몇 분 만에 보고 대시보드를 만들었습니다. 두 테이블에 모두 ACL이 있으므로 (대시보드 자체에도 있음) 사용자가 경영진이나 데이터 팀과 쉽게 공유할 수 있습니다. 또한 이 보고서를 정기적으로 실행하는 스케줄러도 기본 내장 형태로 제공됩니다. 이 대시보드는 AML 솔루션에 내장된 AI, BI 및 분석 엔지니어링의 정점입니다.

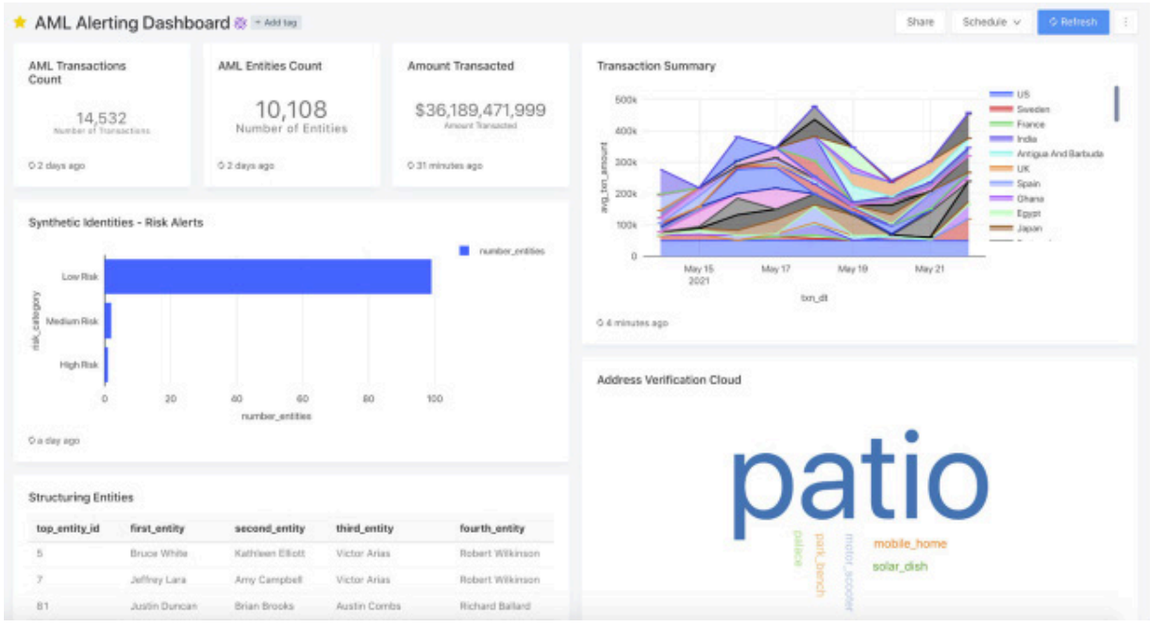

그림 11

오픈 뱅킹 혁신

오픈 뱅킹이 대세로 자리 잡으면서 FI에서는 소비자, FI와 타사 서비스 제공업체가 API를 통해 데이터를 공유하는 방식으로 전보다 나아진 고객 경험을 제공할 수 있게 되었습니다. Open Banking Europe 규정의 일부로 EU 지역의 금융 서비스를 혁신한 지불 서비스 지침(Payment Services Directive, PSD2)이 좋은 예입니다. 그 결과 FI에 여러 은행과 서비스 제공업체의 더 많은 데이터에 액세스할 수 있게 되었습니다. 예컨대 고객 계좌, 거래 데이터 등이 대표적입니다. 최근 미국 애국법(Patriot Act) 제314(b)조에 따른 금융범죄단속반(FinCEN) 최신 지침에 따라 이러한 경향이 사기 및 금융 범죄 분야로도 확대되었습니다. 즉 이 지침에 해당하는 FI에서는 이제 다른 FI 및 여타 국내 및 해외 지점과 개인, 법인, 조직 등 자금 세탁에 연루되었을 것으로 의심되는 대상에 관한 정보를 공유할 수 있게 된 것입니다.

정보 공유 조항은 투명성 면에서나 미국 내 금융 시스템을 자금 세탁과 테러 자금 지원으로부터 보호하는 데는 유익하지만, 이런 정보 교환은 반드시 적절한 데이터 및 보안 보호 기능을 포함한 프로토콜을 사용하여 이루어져야 합니다. 정보 공유 보안 확보를 해결하기 위해, 최근 Databricks에서 Delta Sharing 출시 소식을 발표했습니다. 이것은 데이터 공유를 위한 안전한 오픈 프로토콜입니다. 이제 데이터 생산자와 소비자가 친숙한 오픈 소스 API (예를 들어 pandas, Spark 등)를 사용해 안전한 오픈 프로토콜로 데이터를 공유하면서도 데이터 트랜잭션 전체에 대한 완벽한 감사를 유지하여 FinCEN 규정을 준수할 수 있게 된 것입니다.

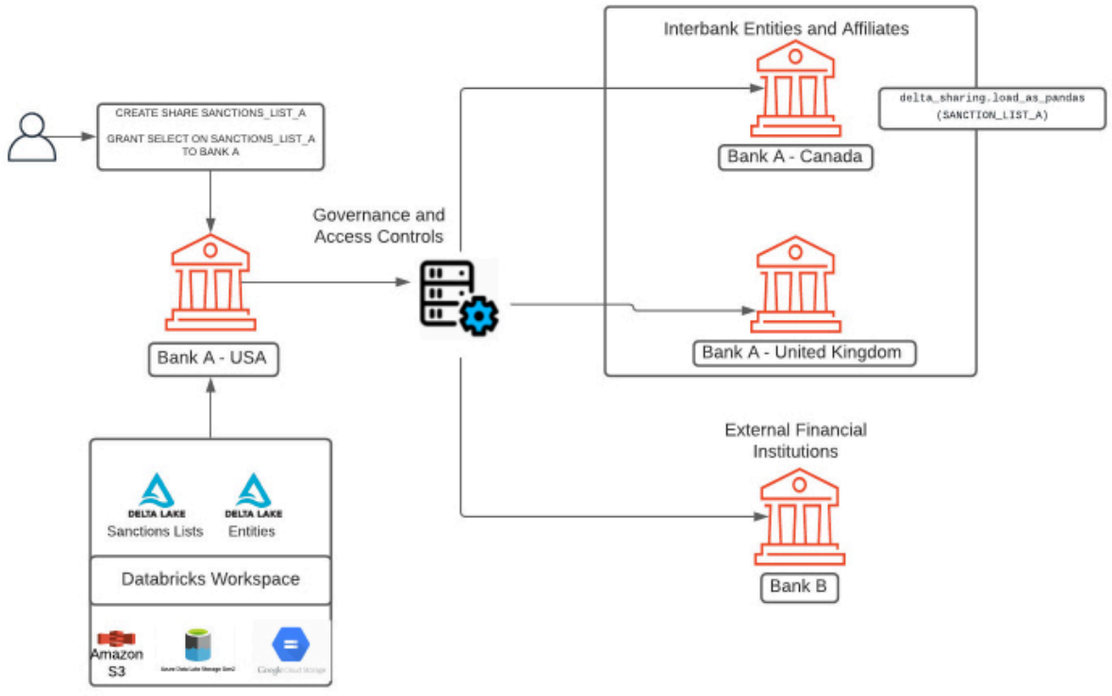

그림 12

결론

레이크하우스 아키텍처는 확장성이 매우 우수한 다목적 플랫폼으로, 애널리스트의 AML 분석을 지원하는 역할을 합니다. 레이크하우스에서는 기본 내장 대시보드를 통해 퍼지 매치, 이미지 분석부터 BI까지 다양한 사용 사례를 지원하며, 이 모든 기능은 기업에서 상용 AML 솔루션을 구매하는 선택지에 비해 총소유비용(TCO)을 낮춰주는 효과가 있습니다. Databricks의 금융 서비스 팀은 금융 서비스 분야의 다양한 비즈니스 문제를 해결하기 위해 노력 중이며, 데이터 엔지니어링 및 데이터 사이언스 전문가들이 AML과 같은 솔루션 액셀러레이터를 통해 Databricks 여정을 시작하도록 돕습니다.

무료 Databricks 노트북으로 실험 시작

- AML 그래프 이론 소개
- AML 컴퓨터 비전 소개
- AML 엔터티 확인 소개

섹션 2.6

실시간 AI 모델을 구축하여 게임 내 유해 행위 탐지

글: DAN MORRIS, DUNCAN DAVIS

2021년 6월 16일 수요일

대규모 멀티플레이어 온라인 비디오 게임(MMO), 멀티플레이어 온라인 배틀 아레나 게임(MOBA)은 물론 여타 많은 형태의 온라인 게임에서는 여러 플레이어가 실시간으로 상호작용을 주고받으며 협조하거나 경쟁하면서 공통의 목표인 '승리'를 향해 나아갑니다. 이러한 상호성은 게임플레이 역학에 핵심적이지만, 동시에 유해 행위가 발생하는 주된 틈새 역할도 합니다. 유해 행위란 온라인 비디오 게임 분야 전체에 만연한 문제입니다.

유해 행위는 다양한 형태로 나타납니다. 예를 들어 그리핑(의도적인 방해 행위), 사이버 불링, 성희롱 등이 다양한 강도로 나타나는데 이를 위 오른쪽에 Behaviour Interactive에서 제시한 매트릭스로 표시하였습니다. 이 표는 멀티플레이어 게임 "Dead by Daylight"에서 확인된 다양한 상호작용 유형을 정리한 것입니다.

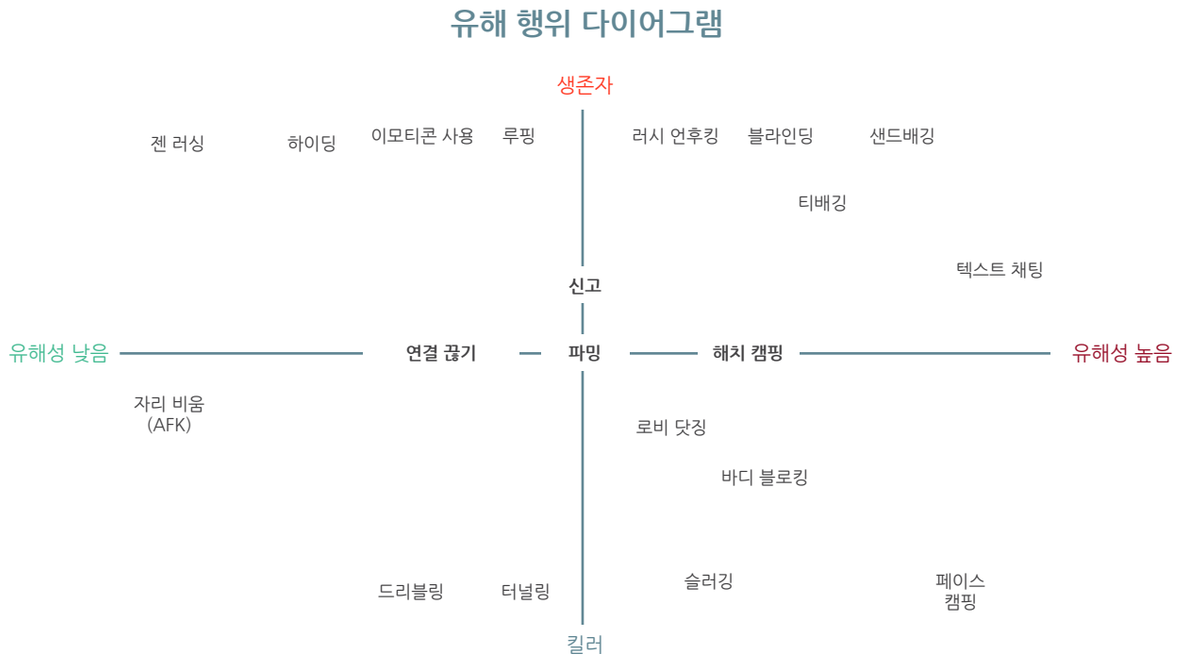

그림 1
플레이어가 경험하는 유해 행위 매트릭스

유해 행위가 게이머와 커뮤니티에 미치는 개인적인 고통도 아무리 강조해도 지나치지 않은 문제이지만, 이는 수많은 게임 스튜디오의 수익에도 손해를 끼치는 원인입니다. 예를 들어, 미시간 주립대학교에서 실시한 연구 따르면 최근에 유해 행위를 경험해본 플레이어는 전체의 80%에 달했으며, 그중 20%는 이러한 상호작용 때문에 게임을 그만두었다고 합니다. 마찬가지로, 네덜란드 틸뷔르흐대학교에서 실시한 연구에서는 게임을 처음 시작했을 때 불쾌하거나 유해한 경험을 한 플레이어는 게임을 그만두고 다시 돌아오지 않을 가능성이 그렇지 않은 경우에 비해 세 배 높은 것으로 드러났습니다. 대부분의 게임 스튜디오에서는 플레이어 유지를 최우선 선결과제로 삼고 있는 데다, 특히 게임 제공 방식이 실물 미디어에서 수명이 긴 서비스 형태로 바뀌는 와중에 이 중요성이 더욱 부각되고 있으므로 유해성은 반드시 억제해야 할 문제임이 분명합니다.

이처럼 고객 이탈과 관련된 문제에 더하여, 회사에 따라 개발 초창기에(심하면 출시 이전부터) 유해 행위와 관련한 문제에 직면하는 경우도 있습니다. 예를 들어 Amazon의 Crucible은 텍스트나 음성 채팅 없이 테스트 릴리스를 내놓았는데, 이는 일부분 유해한 게이머와 상호작용을 모니터링하거나 관리할 수단이 시스템 내에 아직 마련되지 않았다는 데 기인했습니다. 이 사실만 보아도 게임 분야의 규모가 너무 커진 나머지 그러한 행동은 이제 보고서를 통해서나, 방해 행위에 개입하는 등의 방식으로 관리하는 팀원들의 능력으로는 감당할 수 없게 되었다는 실태가 드러납니다. 이런 상황이므로, 스튜디오 측에서 게임 개발 수명 주기 초반부터 분석을 통합한 다음 지속적으로 유해한 상호 작용을 관리할 수 있도록 방법을 염두에 두고 설계하는 것이 매우 중요합니다.

게임 내 유해 행위는 다면적인 문제임이 틀림없으며, 비디오 게임 문화의 일부분으로 자리 잡았습니다. 따라서 한 가지 방식을 보편적으로 적용해 해결할 수는 없습니다. 그렇기는 하지만, 유해 행위의 빈도와 자연어 처리(NLP)를 사용한 자동 탐지 기능을 감안했을 때 게임 채팅 내에서 유해 행위 문제를 해결하기만 해도 큰 파급력을 낼 수 있습니다.

Databricks 게임 솔루션 액셀러레이터에 유해 행위 탐지 도입

이 솔루션 액셀러레이터에서는 Jigsaw에서 제공한 악성 댓글 데이터와 Dota 2 게임 매치 데이터를 예로 들어 NLP와 기존 레이크하우스를 사용해 실시간으로 악성 댓글을 탐지하는 데 필요한 단계를 소개합니다. 이 솔루션 액셀러레이터에서 사용한 NLP는 John Snow Labs의 Spark NLP로, 이것은 Apache Spark™에서 네이티브 구축한 오픈 소스, 엔터프라이즈급 솔루션입니다.

이 솔루션 액셀러레이터에서 진행할 단계는 다음과 같습니다.

- Delta Lake를 사용하여 Jigsaw 및 Dota 2 데이터를 테이블에 로드
- 멀티 레이블 분류(Spark NLP)를 사용하여 악성 댓글 분류
- MLflow를 사용하여 실험을 추적하고 모델 등록
- 배치 및 스트리밍 데이터에 추론 적용
- 유해 행위가 게임 매치 데이터에 미치는 영향 조사

제작 중인 게임 내 채팅에서 유해 행위 탐지

이 솔루션 액셀러레이터를 사용하면 전보다 더욱 간편하게 게임에 유해 행위 탐지 기능을 통합할 수 있게 됩니다. 예를 들어 아래의 참조 아키텍처에는 스트림, 파일, 음성 또는 운영 데이터베이스와 같은 다양한 소스에서 게임 데이터를 가져와 Databricks를 사용해 데이터를 수집, 저장하고 피처 테이블에 큐레이션하여 머신 러닝(ML) 파이프라인, 게임 내 ML, 분석용 BI 테이블에 사용하고 심지어 커뮤니티 중재를 위해 쓰이는 도구를 사용해 직접적으로 상호작용하는 데 쓰는 법을 나타내었습니다.

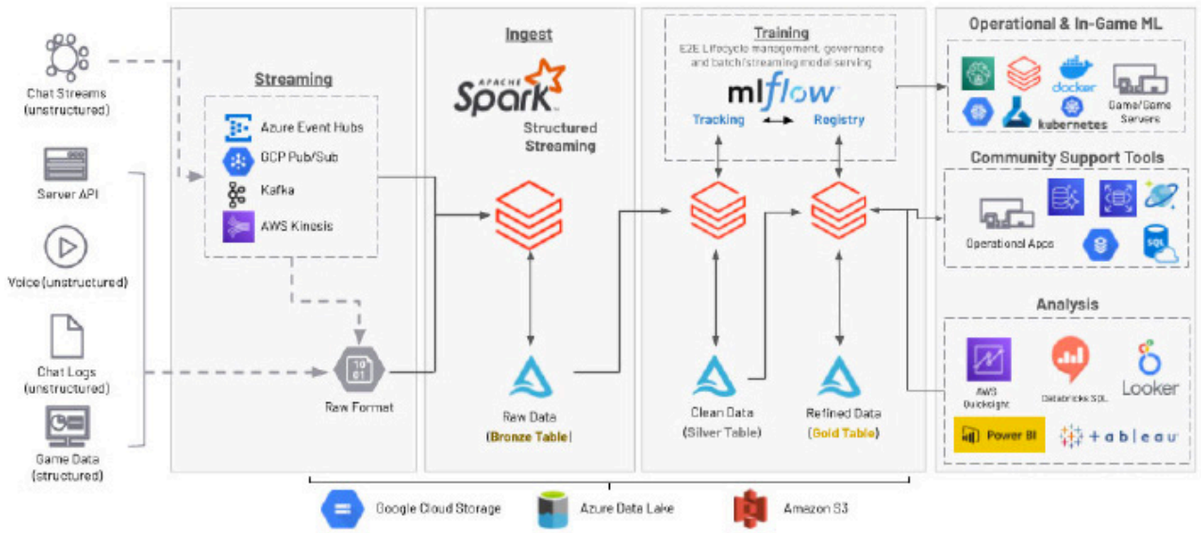

그림 2
유해 행위 탐지 참조 아키텍처

databricks

커뮤니티 내 유해 행위를 탐지할 실시간, 확장 가능한 아키텍처를 확보하면 커뮤니티 관계 관리자(CRM)가 워크플로를 간소화할 기회를 얻을 수 있고, 수백만 건에 달하는 상호작용을 필터링하여 감당할 만한 크기의 워크로드로 만들 수 있습니다. 마찬가지로, 심각한 유해 행위 이벤트에 대하여 실시간으로 알림을 제공할 수도 있고, 플레이어를 음소거하거나 인시던트를 CRM에게 신속하게 알리는 등 대응을 자동화하면 플레이어 유지율에 직접적인 영향을 미칠 수 있습니다. 또한 출처가 서로 다른 대규모 데이터 세트를 처리할 줄 아는 플랫폼이 있으면 이를 여러 보고서와 대시보드를 통해 브랜드 인지도를 모니터링하는 데 쓸 수도 있습니다.

시작하기

이 솔루션 액셀러레이터에서는 게임 내 채팅에서 악성 댓글을 실시간 탐지하도록 지원하여 온라인 게임 분야에서 유해성 상호작용을 꾸준히 관리할 수 있도록 지원하고자 합니다. 오늘 바로 이 솔루션 액셀러레이터를 Databricks 워크스페이스에 직접 가져와서 시작하세요.

액셀러레이터를 가져오면 노트북이 생겼을 것입니다. 이 안에 바로 프로덕션으로 이동할 수 있는 파이프라인이 두 개 있습니다.

- 하나는 Google Jigsaw에서 가져온 실제 데이터 세트(영문)로 훈련한 멀티 레이블 분류를 사용하는 ML 파이프라인입니다. 이 모델은 텍스트에서 유해 행위의 형식을 분류하고 레이블을 지정합니다.

- 다른 하나는 유해 행위 모델을 활용하는 실시간 스트리밍 추론 파이프라인입니다. 이 파이프라인 소스를 간편하게 수정하여 각종 공용 데이터 소스에서 가져온 채팅 데이터를 모두 수집할 수 있습니다.

이런 두 가지 파이프라인을 사용하면 큰 수고를 들이지 않고 유해 행위를 이해하고 분석하는 작업을 시작할 수 있습니다. 이 솔루션 액셀러레이터는 게임 메카닉 및 커뮤니티와 관련된 데이터를 사용하여 모델을 구축, 맞춤 설정하고 개선할 기반 역할을 하기도 합니다.

무료 Databricks 노트북으로
실험을 시작하세요.

섹션 2.7

Northwestern Mutual(인사이트 플랫폼)에서 확장 가능한 오픈 레이크하우스 아키텍처를 통한 혁신

글: MADHU KOTIAN

2021년 7월 15일 목요일

최근 대부분의 빅데이터 기업 이니셔티브에서는 디지털 혁신을 가장 중시해 왔습니다. 특히 레거시 풋프린트 비중이 큰 기업일수록 더욱 그러한 경향이 두드러집니다. 디지털 혁신의 기본적인 구성 요소 중 하나는 데이터, 그리고 그와 관련한 데이터 스토어입니다. Northwestern Mutual은 160여 년 전통의 금융 기업으로 개인 및 기업 고객을 대상으로 금융 서비스를 제공하고 있습니다. 매출 310억 달러 이상, 460만여 고객, 금융 전문가 9,300여 명이 관여하며 이 정도로 다양한 출처를 총망라해 엄청난 규모의 데이터를 보유하고 있는 기업은 많지 않습니다.

지금은 데이터 수집도 쉽지 않은 시대입니다. 기업에서 다루는 데이터 포인트의 입수되는 형식, 시간 프레임이 제각기 다르고 유입되는 방향도 다르며 볼륨 자체도 전에 없는 수준이기 때문입니다. 그래서 데이터를 이해하기 위해 분석용으로 준비하고자 합니다. 이 글에서는 데이터 수집 프로세스, 예약 프로세스, 데이터 스토어와의 여정을 혁신하고 현대화하는 데 어떤 참신한 방법을 도입했는지 알려드리겠습니다. 이 프로젝트를 통해 배운 것이 하나 있다면 효과적인 방식이란 다면적이어야 한다는 것입니다. 그런 의미에서, 기술적 조율 외에 팀원 온보딩 계획도 소개하고자 합니다.

당면 과제

우리 회사에서는 혁신을 본격적으로 시작하기에 앞서, 여러 비즈니스 파트너와 협력하여 기술적 제약 사항을 파악하고 비즈니스 케이스의 문제 설정(problem statement)을 작성했습니다.

이렇게 확인한 비즈니스 고충 사항으로는 통합 데이터 부족이 대표적이었습니다. 고객과 비즈니스 데이터가 다양한 사내외 팀과 데이터 소스에서 입수되는 형태이기 때문입니다. 실시간 데이터의 가치는 깨달았으나, 제때 비즈니스 의사 결정을 내리는 데 도움이 될 수 있는 프로덕션/실시간 데이터를 충분히 접하지 못하고 있었습니다. 또한 비즈니스팀이 구축한 데이터 스토어 때문에 데이터 사일로가 생겼고, 나아가 데이터 지연 문제가 생기면서 데이터 관리 비용이 늘고 불필요한 보안 제약 사항까지 가중되었다는 사실도 깨달았습니다.

그뿐만 아니라, 회사 현황과 관련한 기술적 문제도 있었습니다. 수요가 늘어나고 필요한 데이터가 많아지면서 인프라 확장성 제한, 데이터 지연, 데이터 사일로 관리 비용, 데이터 크기와 볼륨 한계, 데이터 보안 문제 등 다양한 곤란을 겪고 있었던 것입니다. 이처럼 문제가 점점 커져만 가는 상황에서, 갈 길이 멀다는 사실, 그리고 혁신의 여정에 도움을 줄 적절한 파트너가 절실하다는 사실을 확실히 깨달았습니다.

솔루션 분석

경쟁력을 유지하면서 고객에게 더 좋은 서비스를 제공하고, 사내 프로세스를 최적화하려면 데이터 기반 기업으로 거듭나야 했습니다. 다양한 선택지를 살펴보고 POC도 여러 차례 거친 결과 최종 추천안을 선택했습니다. 앞으로의 진행 전략에 필수적인 내용은 다음과 같이 결정되었습니다.

- 데이터 수집, 데이터 관리 및 분석 요구 사항에 적합한 포괄적 솔루션
- 개발자와 비즈니스 애널리스트가 SQL을 사용하여 분석을 수행하도록 효과적으로 지원하는 현대적 데이터 플랫폼
- S3 기반의 ACID 트랜잭션을 지원하고 역할 기반 보안을 사용할 수 있는 데이터 엔진
- PII/PHI 정보를 효과적으로 보호하는 시스템
- 데이터 처리 및 분석 수요에 따라 자동으로 확장할 수 있는 플랫폼

우리 회사 기존 인프라는 MSBI Stack 기반이었습니다. 수집에는 SSIS, 데이터 스토어에는 SQL Server, 표 형식 모델에는 Azure Analysis Service, 대시보딩과 보고에는 Power BI를 사용했습니다. 이 플랫폼은 처음에는 비즈니스 요구 사항에 부합했지만, 데이터 볼륨과 데이터 처리 수요가 늘어나면서 확장과 관련한 문제가 있었고, 데이터 분석 기대치에 한계가 지어지기도 했습니다. 데이터 요구 사항이 가중되자 로드 지연과 특정 비즈니스 요구 사항용 데이터 스토어로 인한 데이터 레이턴시 문제가 생기면서 데이터 사일로와 데이터 스프롤(sprawl, 무질서한 확산) 문제가 발생했습니다.

데이터가 여러 개의 데이터 스토어에 분산된 관계로 보안을 확보하기도 어려웠습니다. 당시 약 300건의 ETL 작업을 처리하는 데 7시간이 넘게 걸려 일과 업무 시간을 크게 빼앗기고 있었습니다. 각종 변동 사항이나 신규 개발 상품의 TTM은 4~6주 정도였습니다(복잡성에 따라 다름).

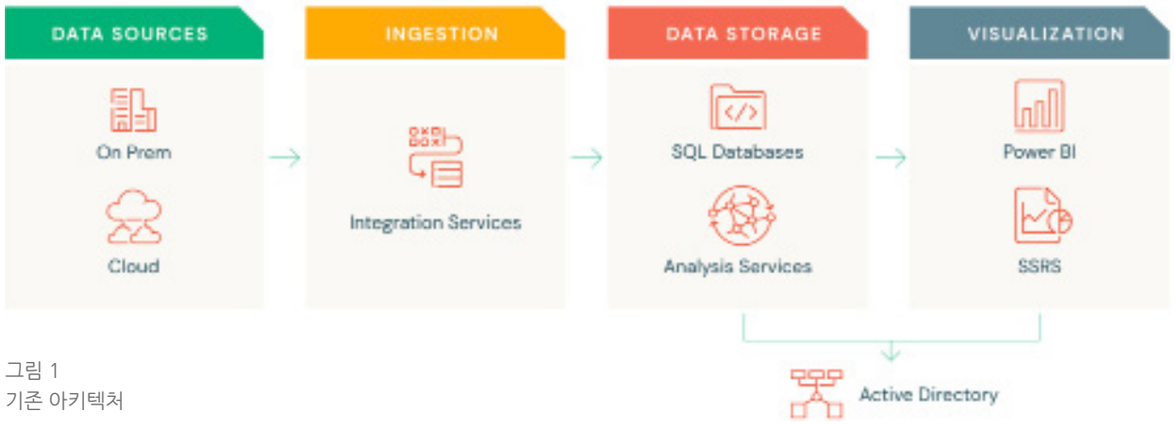

그림 1
기존 아키텍처

우리는 시중의 여러 솔루션을 평가한 끝에, Databricks를 선택하여 오픈 레이크하우스 아키텍처에서 하나의 통합형 데이터 관리 솔루션을 제공하는 데 도움을 받기로 했습니다.

Databricks는 Apache Spark™를 기반으로 개발되었기 때문에 우리 측에서 데이터 수집과 메타데이터 관리에 사용할 맞춤형 프레임워크를 구축하는 데 Python을 사용할 수 있었습니다. 이 플랫폼 덕분에 노트북을 사용하여 유연하게 애드혹 분석 및 여타 데이터 탐색을 수행할 수 있었습니다. 기존의 데이터 레이크를 기반으로 Databricks Delta Lake 스토리지 계층을 구축하자 다양한 데이터베이스 관리 기능(ACID 트랜잭션, 메타데이터 거버넌스, 시간 이동 등)을 유연하게 구현할 수 있었으며 꼭 필요한 보안 컨트롤 구현도 예외가 아니었습니다. Databricks 덕분에 클러스터 관리/확장이 한결 손쉬워졌고, 엔지니어 및 비즈니스 사용자의 억눌렸던 수요에도 효과적으로 대응할 수 있었습니다.

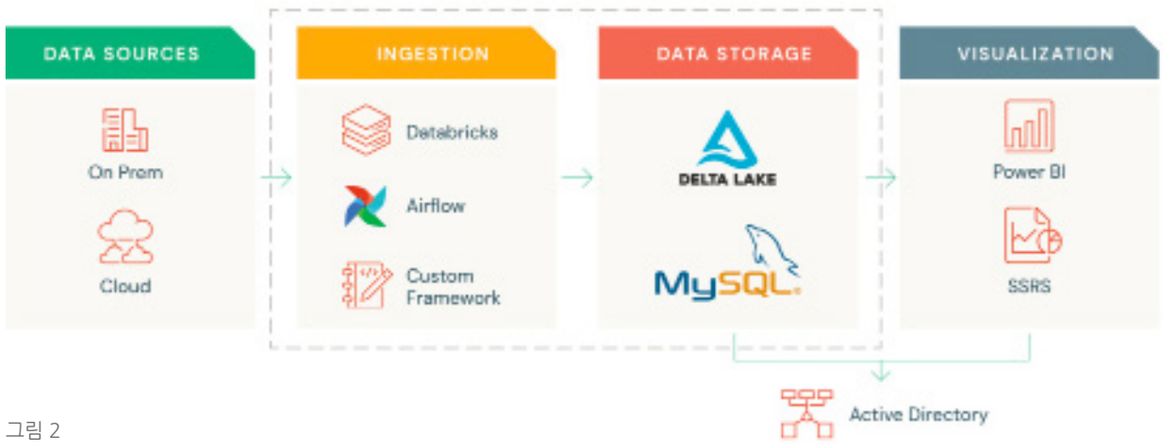

그림 2
Databricks를 사용한 아키텍처

마이그레이션 전략 및 리소스 온보딩

우선은 적은 인원의 엔지니어 팀을 꾸려 기존 스크럼 팀(scrum team)을 바탕으로 가상 팀을 만들어 배정했습니다. 이들의 목표는 서로 다른 POC를 실행하고 권장 솔루션을 구축하여 모범 사례를 개발한 다음, 각자의 원래 소속 팀으로 돌아가 온보딩을 지원하는 것이었습니다. 기존 팀원을 활용했던 것이 유리하게 작용했습니다. 이들은 기존 시스템에 대한 지식이 있었고, 현재의 수집 플로우/비즈니스 규칙을 알 뿐만 아니라, 적어도 한 가지 프로그래밍 지식(데이터 엔지니어링 + 소프트웨어 엔지니어링 지식)에 일가견이 있는 인력이었기 때문입니다. 이 팀은 우선 Python부터 배우고, Spark와 Delta의 복잡한 세부 정보를 익힌 다음 Databricks 팀과 긴밀히 협력하여 솔루션/접근 방식의 유효성을 검증했습니다. 이 가상팀이 미래를 만들어 나가는 동안, 나머지 개발자는 비즈니스 우선순위를 해결했습니다.

우리 회사 개발자는 대부분 MSBI Stack 엔지니어였기 때문에, 우리가 세운 대처 방안은 일단 우리 쪽 개발자, 비즈니스 사용자와 현장 자문이 차질 없이 사용할 수 있는 데이터 플랫폼을 제공하는 것이었습니다.

- 그래서 데이터 로드와 혁신 요구 사항에 모두 적합한 수집 프레임워크를 구축했습니다. 여기에는 보안 컨트롤을 기본 내장하여 각종 메타데이터와 소스 시스템 기밀을 모두 유지 관리하도록 하였습니다. 이 수집 프로세스에서는 소스, 대상과 필수 변환을 포함한 JSON 파일을 허용했습니다. 따라서 단순한 변환, 복잡한 변환이 둘 다 지원되었습니다.
- 예약에는 결국 Airflow를 사용하게 되었지만 DAG가 워낙 복잡한 관계로 Airflow를 기반으로 자체적인 맞춤 프레임워크를 구축했는데, 여기서는 작업 정보 및 그와 관련된 상호종속성을 포함한 YAML 파일을 허용했습니다.
- Delta를 사용한 스키마 레벨 변경 관리를 위해서는, 자체적인 맞춤 프레임워크를 구축했습니다. 이 프레임워크는 개발자가 데이터 스토어에 비상 임시 권한(break-glass)으로 액세스하지 않아도 되도록 다양한 데이터베이스 유형 작업(DDL)을 자동화해주었습니다. 이는 데이터 스토어에 다양한 감사 컨트롤을 구현하는 데도 도움이 되었습니다.

또한 팀원들은 보안 팀과도 협력하여 데이터 보안 기준을 모두 숙지하고 이에 부합하도록 만전을 기했습니다(전송 중인 데이터 암호화, 미사용 데이터 암호화, 열 레벨 암호화를 통한 PII 정보 보호 등).

이러한 프레임워크가 모두 준비된 뒤에는 코호트 팀이 엔드 투 엔드 워크플로우(모든 변환을 포함해 소스에서 대상까지)를 배포하고 Power BI에서 Delta Lake를 향하는 새 보고서/대시보드 세트를 생성했습니다. 이 시점에서의 목표는 전체적 프로세스의 성과를 테스트하고, 데이터를 검증하여 현장 사용자에게 피드백을 받는 것이었습니다. 이러한 피드백을 바탕으로 제품과 성과/검증 테스트 결과를 점진적으로 개선했습니다.

동시에, 개발자 온보딩에 사용할 교육 가이드와 방법(how-to) 안내서를 작성하기도 했습니다. 얼마 지나지 않아 코호트 팀원을 각자의 소속 팀으로 돌려보내기로 했습니다. 단, 몇 명만 플랫폼 인프라 지원(DevOps)을 위해 남겼습니다. 각각의 스크럼 팀이 각자 맡은 역량/기능을 관리하고 비즈니스에 전달할 책임을 맡겼습니다. 팀원들은 각자 소속 팀으로 복귀한 뒤, 팀 작업 속도를 조정하여 마이그레이션 업무 때문에 밀린 일을 포함하는 작업에 돌입했습니다. 팀 리더마다 구체적인 지침과 적절한 목표를 부여하여 다양한 스프린트(Sprint)/프로그램 기간마다 마이그레이션 목표를 달성하도록 지도했습니다. 코호트 그룹에 파견되었던 팀원들은 각 팀 전담 전문가 역할을 하면서 소속 팀이 새 플랫폼에 온보딩하는 과정을 도왔습니다. 이들이 그때그때 생기는 질문에 답하고 도움을 제공할 수 있었습니다.

새로운 플랫폼을 점진적으로 구축해 나가는 동안 기존 플랫폼은 검증과 확인을 위해 남겨두었습니다.

성공의 시작

전환 과정 전체에 약 1년 반이 걸렸습니다. 프레임워크를 전부 직접 구축하고, 비즈니스 우선순위를 관리하며 보안 기대치를 맞추고 팀원을 재정비, 플랫폼 마이그레이션까지 해야 했던 것을 고려하면 실로 대단한 성과입니다. 전반적인 로드 시간은 7시간에서 겨우 2시간으로 눈에 띄게 줄었습니다. TTM은 기존의 4주~6주에서 약 1~2주로 크게 단축되었습니다. 이것이 정말 중대한 개선점이었는데, 이 부분은 앞으로 우리 비즈니스에 여러 가지 방식으로 확대 적용될 것이라고 확신하고 있습니다.

우리의 여정은 아직 끝나지 않았습니다. 플랫폼을 보강하려 계속 노력 중이며, 다음으로는 레이크하우스 패턴을 확장하고자 합니다. 지금은 플랫폼을 E2로 마이그레이션하고 Databricks SQL을 배포하는 작업을 추진 중입니다. 또한 전략을 가다듬어 비즈니스 사용자가 애드혹 분석을 직접 수행할 수 있도록 셀프서비스 플랫폼을 제공하고자 하며, 사용자가 직접 자기 데이터를 가져와 사내에 보유한 통합형 데이터와 함께 분석하는 기능도 지원할 계획입니다. 이번 프로젝트를 통해 통합형, 확장할 수 있는 오픈 플랫폼을 사용하면 큰 장점이 있다는 사실을 알았습니다. 앞으로도 우리의 요구 사항이 늘어나고 역량이 성장할 텐데, Databricks라는 파트너가 있어 든든합니다.

Northwestern Mutual의 레이크하우스 전환 사례에 대해 자세히 알아보세요.

MADHU KOTIAN 소개

Madhu Kotian은 Northwestern Mutual.의 엔지니어링 부사장(투자 상품 데이터, CRM, 앱 및 보고 담당)입니다. 정보 기술(IT) 업계에 25년 이상 몸담아 왔으며 특히 데이터 엔지니어링, 인력 관리, 프로그램 관리, 아키텍처, 설계, 애자일 방식을 사용한 개발과 유지 관리 경력을 보유한 전문가입니다. 또한 데이터 웨어하우스 방법론, 데이터 통합 및 분석 구현 분야의 전문가이기도 합니다.

섹션 2.8

Databricks 데이터 팀이 3가지 클라우드와 50개 이상 지역에 레이크하우스를 구축한 방법

글: JASON POHL, SURAJ ACHARYA

2021년 7월 14일

Databricks의 내부 로깅 인프라는 오랜 시간을 거치며 발전해 왔습니다. 그 과정에서, 여러 가지 클라우드와 지역을 망라하여 고도로 사용성이 높은 로그 파이프라인을 유지 관리하는 법에 대해 꽤 많은 것을 알게 되었습니다. 이 블로그에서는 레이크하우스 플랫폼을 사용하여 실시간 지표를 수집, 관리하는 방법과 여러 클라우드를 활용하여 퍼블릭 클라우드 장애로부터 복구하는 방법에 대한 인사이트를 제공합니다.

Databricks는 설립 당시만 해도 지원하는 클라우드가 퍼블릭 클라우드 하나뿐이었습니다. 지금은 전 세계 50여 개 지역에서 세 가지 주요 퍼블릭 클라우드(AWS, Azure, GCP)를 모두 지원할 정도로 서비스 규모가 커졌습니다. Databricks는 매일 고객을 대신해 수백만 개의 가상 머신을 스핀업(spin up)합니다. Databricks 데이터 플랫폼 팀은 10명 이하의 엔지니어로 구성되어 있으며, 매일 0.5PB의 데이터를 처리하는 로깅 텔레메트리 인프라를 구축, 유지 관리하는 업무를 담당합니다. 오케스트레이션, 모니터링과 사용량은 서비스 로그를 통해 수집하며 이를 인프라에서 처리하여 적시에 정확한 지표를 제공합니다. 이 데이터는 결국 자사에서 보유한 페타바이트급 Delta Lake에 저장됩니다. 우리 데이터 플랫폼 팀은 Databricks를 사용해 클라우드 간 처리를 수행하기 때문에 경우에 따라 데이터를 페더레이션하고, 지역별 클라우드 서비스 중단으로 인한 복구를 완화하며 라이브 인프라 중단을 최소화할 수 있습니다.

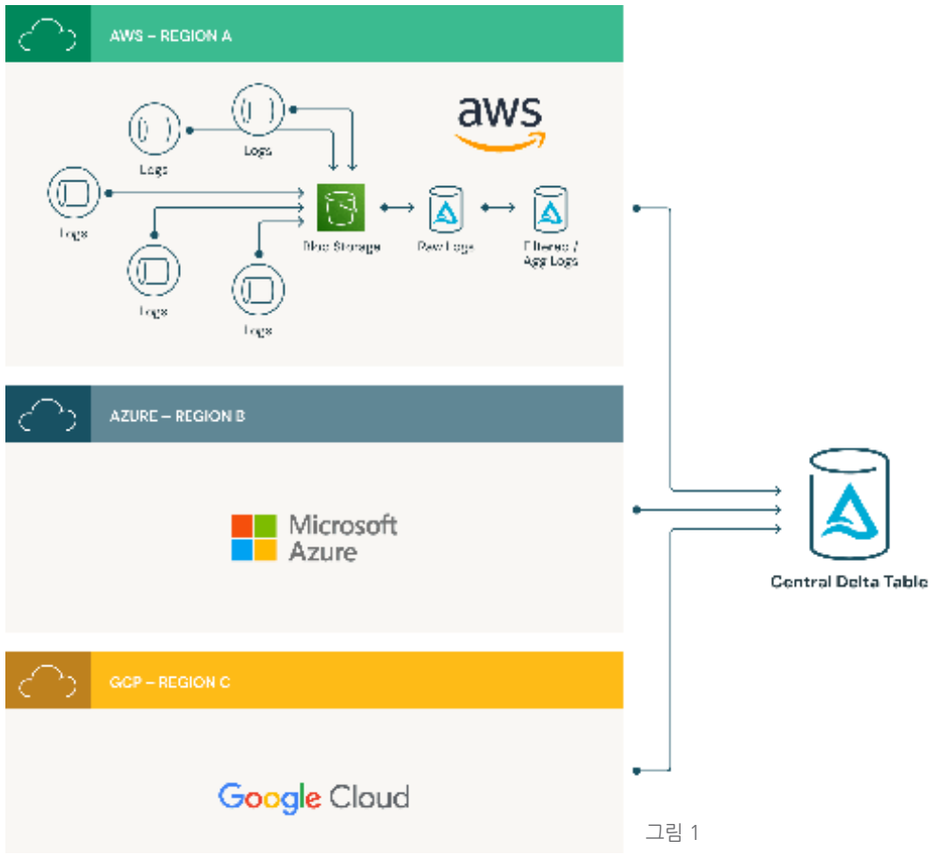

그림 1

파이프라인 아키텍처

클라우드 지역마다 자체적인 인프라 및 데이터 파이프라인이 있습니다. 이를 통해 로그 데이터를 캡처, 수집하여 지역별 Delta Lake에 유지하는 것입니다. 제품 텔레메트리 데이터의 경우 모든 클라우드 지역에 똑같이 복제되는 같은 프로세스를 이용해 제품 전체에서 및 파이프라인 내부에서 수집됩니다. 로그 데몬이 이 텔레메트리 데이터를 캡처하고, 그런 다음 이러한 로그를 지역별 클라우드 스토리지 버킷(S3, WASBS, GCS 등)에 쓰는 것입니다. 그런 다음, 예약된 파이프라인이 Auto Loader(AWS | Azure | GCP)를 사용하여 로그 파일을 수집하고 해당 데이터를 지역별 Delta 테이블에 씁니다. 또 다른 파이프라인이 지역별 Delta 테이블에서 데이터를 읽어와 필터링한 다음 한 클라우드 지역 내 중앙 집중형 Delta 테이블에 쓰게 됩니다.

databricks

Delta Lake 이전

Delta Lake 전에는 소스 데이터를 중앙 집중형 레이크에 있는 자체 테이블에 작성한 다음, 뷰를 하나 만들어 그러한 모든 테이블을 하나로 통합했습니다. 이 뷰는 런타임에 계산해야만 했으며, 지역을 많이 추가할수록 효율성이 점점 떨어졌습니다.

```
CREATE OR REPLACE VIEW all_logs AS
SELECT * FROM (
  SELECT * FROM region_1.log_table
  UNION ALL
  SELECT * FROM region_2.log_table
  UNION ALL
  SELECT * FROM region_3.log_table
  ...
);
```

재해 복구

지금은 Delta Table 하나만으로 50여 가지 다른 지역에서 동시에 쓰기 명령을 허용하고 있습니다. 그러면서 데이터에 대한 쿼리도 동시에 처리합니다. 덕분에 중앙 테이블 쿼리가 다음과 같이 매우 쉬워집니다.

```
SELECT * FROM central.all_logs;
```

트랜잭션 기능은 Delta Lake에서 처리합니다. 중앙 Delta Lake에서는 개별적인 지역별 테이블 사용을 중단했고, UNION ALL 뷰도 사용을 중지했습니다. 다음 코드는 지역별 Delta Lake에서 중앙 Delta Lake로 송신하도록 승인된 데이터를 로드하기 위해 실행한 구문을 간단하게 나타낸 것입니다.

```
spark.readStream.format(“delta”)
  .load(regional_source_path)
  .where(“egress_approved = true”)
  .writeStream
  .format(“delta”)
  .outputMode(“append”)
  .option(“checkpointLocation”, checkpoint_path)
  .start(central_target_path)
```

재해 복구

클라우드 간 서비스를 운영하는 장점은 특정 재해 복구 시나리오에서 유리하다는 것입니다. 드물기는 하지만 특정 클라우드 지역의 컴퓨팅 서비스가 일시 중단되는 경우도 있습니다. 이런 일이 일어나면 클라우드 스토리지에는 액세스할 수 있지만 새 VM을 스핀업하는 데 지장이 생깁니다. Databricks의 데이터 파이프라인 코드는 소스와 대상 경로의 구성을 허용하도록 고안했기 때문에, 다른 지역에서 데이터가 저장되는 곳으로 데이터 파이프라인을 신속하게 배포하고 실행할 수 있습니다. 클러스터가 생성되는 클라우드는 데이터를 읽거나 쓰는 클라우드와는 무관합니다.

Databricks에서는 몇몇 데이터 세트를 스토리지 서비스 장애에 대비해 안전하게 보호하는데, 이를 위해 여러 클라우드 제공업체에 데이터를 끊임없이 복제합니다. 이 작업은 이 블로그에서 설명한 것과 같이 Delta 딥 클론 기능을 이용하면 간편하게 수행할 수 있습니다. 즉, 테이블에서 클론 명령이 실행될 때마다 마지막으로 실행된 시점 이후의 점진적 변경 사항만 적용하여 클론을 업데이트합니다. 이것은 여러 지역은 물론 클라우드 종류에도 구애받지 않고 데이터를 복제하는 효율적인 방법입니다.

라이브 데이터 파이프라인 중단 최소화

데이터 파이프라인은 관리형 서비스의 생명선이자, 운영을 멈추지 않는 글로벌 비즈니스의 일부분이기도 합니다. 따라서 유지 관리, 업그레이드나 데이터 백필 때문에 오랫동안 파이프라인을 일시 중지할 수는 없습니다. 최근에는 보통 기본 테이블에 쓰는 데이터의 하위 집합을 다른 퍼블릭 클라우드에 쓰도록 필터링하기 위해 파이프라인을 분기해야 했습니다. 이 작업은 일상 업무를 중단하지 않고 해낼 수 있었습니다.

아키텍처 변경 사항을 라이브 시스템에 배포하면서도 중단을 초래하지 않은 작업 과정은 다음과 같습니다.

첫째, 딥 클론을 수행하여 기본 테이블을 다른 클라우드의 새 위치에 복제했습니다. 이렇게 하면 데이터와 트랜잭션 로그 둘 다 일관성을 보장하는 방식으로 복사할 수 있습니다.

둘째, 새 구성을 파이프라인에 릴리스하여 데이터의 대부분은 종전과 같이 중앙의 기본 테이블에 쓰게 하되, 데이터의 하위 집합은 다른 클라우드에 새로 복제된 테이블에 쓰도록 설정했습니다. 이러한 변경 사항은 새로운 구성을 배포하기만 하면 손쉽게 적용되며, 테이블에는 수신해야 하는 새 변경 사항에 대해서만 업데이트가 수신됩니다.

다음으로, 같은 딥 클론 명령을 다시 실행했습니다. Delta Lake는 원래 기본 테이블에서 점진적인 변경 사항만 캡처하여 새로 클론된 테이블에 복사합니다. 이렇게 하면 새 테이블에 사실상 1단계와 2단계 사이에 데이터에 적용된 모든 변경 사항이 백필(backfill) 됩니다.

마지막으로, 기본 테이블에서 데이터 하위 집합을 삭제하고 복제된 테이블에서는 데이터의 대부분을 삭제하면 됩니다.

이제 두 테이블 모두 원래 포함해야 하는 데이터를 나타내며, 여기에 트랜잭션 기록 전체가 동반됩니다. 게다가 이 모든 작업을 파이프라인의 최신 상태를 해치지 않고 완료했습니다.

요약

Databricks는 각각의 클라우드 서비스 상세 정보는 모두 추상화합니다. 클러스터 관리자를 사용한 인프라 스핀업을 위해서든, Auto Loader를 사용한 데이터 수집 목적이든, Delta Lake로 클라우드 스토리지에 트랜잭션 쓰기를 수행하기 위해서든 마찬가지입니다. 이렇게 하면 코드 베이스 하나만 가지고 여러 퍼블릭 클라우드에서 컴퓨팅과 스토리지를 연결하여 데이터 페더레이션과 재해 복구를 둘 다 지원할 수 있다는 점에서 유리합니다. 이러한 클라우드 간 기능을 이용하면 컴퓨팅과 스토리지를 자사와 고객에게 가장 유익한 위치로 유연하게 이동할 수 있습니다.

섹션 03

고객 사례

Atlassian

ABN AMRO

J.B. Hunt

섹션 3.1

Atlassian

Atlassian은 협업, 개발 및 문제 추적 소프트웨어를 제공하는 선도 기업입니다. Atlassian은 전 세계 150,000여 고객사(포춘 100대 기업 중 85곳 포함)를 보유하고 있으며 Jira, Confluence, Bitbucket, Trello 등 여러 제품을 포함해 협업 범위를 넓혀 나가고 있습니다.

사용 사례

Atlassian에서는 Databricks 레이크하우스 플랫폼을 사용하여 전사적으로 데이터를 민주화하고 운영 비용을 절감합니다. Atlassian에서는 현재 고객 경험 우선주의에 입각한 사용 사례를 여러 건 보유하고 있습니다.

고객 지원 및 서비스 경험

Atlassian 고객은 대부분 서버 기반이지만(Jira와 Confluence 등의 제품 사용), 이러한 고객을 클라우드로 옮겨 심층적인 인사이트를 활용해 고객 지원 경험을 보강하기로 했습니다.

마케팅 개인화

이와 같은 인사이트를 개인별 맞춤형 마케팅 이메일 제공에도 사용하면 신기능, 신제품 참여도를 높일 수 있을 것으로 전망했습니다.

오남용 방지 및 사기 탐지

이상 탐지, 예측 분석을 통해 라이선스 오남용과 사기 행동을 예측할 수 있습니다.

Atlassian에서는 여러 부서가 서로 원활하게 협업하여 끊임없이 변화하는 목표를 달성하도록 지원할 수 있도록 실무팀을 지원하고자 합니다. 간소한 레이크하우스 아키텍처를 활용하면 대량의 사용자 데이터를 수집하고, 고객 요구 사항 예측 역량을 키우는 데 필요한 분석을 실행해 나아가 고객 경험을 개선하는 데 유리할 것입니다. 사용이 간편한 클라우드 분석 플랫폼 하나만 사용하면 실행 가능한 인사이트를 바탕으로 기존 도구를 신속하게 개선하고 새로운 협업 도구를 구축하는 데도 도움이 됩니다.

Rohan Dhupelia
데이터 플랫폼 선임 관리자, Atlassian

솔루션 및 장점

Atlassian에서는 Databricks 레이크하우스 플랫폼을 사용하여 사내외 양쪽에서 모두 대규모 데이터 민주화를 추진 중입니다. 이 회사는 데이터 웨어하우징 패러다임을 버리고 Databricks 기반 표준화로 전환하여 전사적으로 좀 더 데이터 기반을 강화했습니다. HR, 마케팅부터 재무 및 연구 개발(R&D)에 이르기까지 3,000명이 넘는 사내 사용자가(전체의 과반수) Databricks SQL과 같은 오픈 기술을 통해 한 달에 한 번씩 플랫폼에서 입수한 인사이트에 액세스합니다. 또한 Atlassian은 이 플랫폼을 고객에게 더욱 개인화된 지원 및 서비스 경험을 제공하는 데 사용하기도 합니다.

- Delta Lake가 HR, 마케팅, 재무, 영업, 지원 및 R&D 등 다방면의 사용자 3,000여 명이 액세스하는 페타바이트급 데이터의 단일 레이크하우스 기반 역할을 함
- Databricks SQL 기반 BI 워크로드로 더 많은 사용자에게 대시보드 보고 지원
- MLflow를 통해 MLOps를 간소화하고 빠르게 제공
- 데이터 플랫폼 통합으로 편리한 거버넌스 제공, 자체 관리형 클러스터로 자율성 지원

클라우드급 아키텍처, 팀 간 협업을 통한 생산성 향상에 자사 고객 데이터 전체에 액세스하여 분석과 ML에 활용하는 역량까지 갖추게 될 테니 Atlassian에는 엄청난 영향이 발생할 것으로 전망됩니다. 이미 확인된 결과만 해도 다음과 같습니다.

- IT 운영 비용(특히 컴퓨팅 비용) 60% 절감: Spark 작업 50,000여 개를 EMR에서 Databricks로 옮기되, 최소한의 수고와 로우코드(low-code) 변경만으로 해결
- 개발 주기가 짧아져 제공 시간 30% 단축
- 전사적으로 셀프서비스 지원을 늘려 데이터 팀 의존도 70% 경감

자세한 정보

databricks

섹션 3.2

ABN AMRO

ABN AMRO는 저명한 은행입니다. 비즈니스 현대화를 추진하려 했지만, 레거시 인프라와 데이터 웨어하우스 때문에 다양한 소스의 데이터에 액세스하기 복잡했고, 데이터 프로세스와 워크플로우도 비효율적이어서 차질을 빚고 있었습니다. 지금 ABN AMRO는 Azure Databricks를 만나 데이터와 AI를 민주화하여, 500여 명의 엔지니어, 연구진과 애널리스트로 구성된 팀이 협업을 통해 비즈니스 운영을 개선하고 전사적으로 새로운 GTM 역량을 키우는 데 주력하고 있습니다.

사용 사례

ABN AMRO는 Databricks 레이크하우스 플랫폼을 사용하여 전역적인 규모로 금융 서비스 혁신을 제공하고 운영 전반에 자동화와 인사이트를 제공합니다.

개인 맞춤형 금융

ABN AMRO는 실시간 데이터와 고객 인사이트를 활용하여 고객의 요구 사항에 맞는 제품과 서비스를 제공합니다. 예를 들어, 머신 러닝을 활용하여 자동화된 마케팅 캠페인의 큰 틀 안에서 표적화 메시징을 지원하여 참여와 전환을 유도합니다.

리스크 관리

이 은행은 데이터 기반 의사 결정을 활용하여 회사와 고객 양측을 위해 리스크를 완화하는 데 주력하고 있습니다. 예를 들어 보고서와 대시보드를 활용해 내부 의사 결정권자와 경영진이 리스크를 잘 파악하도록 돕고, 그러한 리스크가 ABN AMRO 비즈니스를 저해하지 않도록 방지합니다.

사기 탐지

예측 분석을 활용해 악의적 활동을 차단하는 것을 목표로 삼고, 사기성 행위가 고객에게 영향을 미치기 전에 미리 파악합니다. 이를 통해 대응하고자 하는 악의적 활동 중에는 자금 세탁, 가짜 신용카드 앱 등의 문제가 있습니다.

Databricks 덕분에 업무 방식이 바뀌었습니다. 회사 차원에서 데이터, AI 혁신을 성공으로 이끄는 데 유리한 위치를 확보했거든요. 데이터 전문가에게 잘 통제된, 확장 가능한 방식으로 고급 데이터 기능을 지원할 수 있었기 때문입니다.

Stefan Groot
분석 엔지니어링 책임자, ABN AMRO

솔루션 및 장점

지금 ABN AMRO는 Azure Databricks를 만나 데이터와 AI를 민주화하여, 500여 명의 엔지니어, 연구진과 애널리스트로 구성된 팀이 협업을 통해 비즈니스 운영을 개선하고 전사적으로 새로운 GTM 역량을 키우는 데 주력하고 있습니다.

- Delta Lake로 빠르고 안정적인 데이터 파이프라인 지원, 다운스트림 분석을 위해 정확하고 완전한 데이터 피드 제공
- Power BI와 통합한 덕분에 SQL 분석이 간편해졌고, 보고서와 대시보드를 통해 500명 이상의 비즈니스 사용자에게 인사이트 제공
- MLflow로 고객 경험을 개선하는 새로운 모델 배포 기간 단축, 나아가 새로운 사용 사례를 2개월 내로 전달

10배 단축
TTM(Time to Market) — 사용 사례 배포 기간 2개월 미만

100건 이상
내년에 제공할 사용 사례

500명 이상
지원하는 비즈니스 사용자와 IT 사용자

자세한 정보

databricks

섹션 3.2

J.B. HUNT

Databricks 덕분에 시중에서 가장 혁신적인 디지털 화물 마켓플레이스의 기반을 마련할 수 있었습니다. AI를 활용해서 현실적으로 가장 우수한 운송업체 경험을 제공할 수 있었거든요.

Joe Spinelle
엔지니어링 및 기술 사업부 이사,
J.B. Hunt

자세한 정보

J.B.Hunt에서는 북미에서 가장 효율적인 디지털 운송 네트워크를 구축하겠다는 목표를 표방하며 화물 물류를 간소화하여 최선의 운송업체 경험을 제공하고자 했습니다. 다만 레거시 아키텍처, AI 역량 부족에 빅데이터를 안전하게 다룰 능력 미흡 등으로 목표 달성에 중대한 차질을 빚었습니다. 하지만 Databricks 레이크하우스와 Immuta를 구현한 뒤부터는 공급망 효율성 개선, 운전자 생산성 향상 등 광범위한 운영상의 솔루션을 제공할 수 있게 되어 결과적으로 IT 인프라 비용을 대폭 절약하고 매출이 증가하게 되었습니다.

사용 사례

J.B. Hunt에서는 자사에서 보유한 Carrier 360 플랫폼을 통해 업계를 선도하는 화물 운송업체 분석을 제공하는 데 Databricks를 이용하여 비용은 절감하면서 동시에 운전자 생산성과 안전성은 강화하는 효과를 얻고 있습니다. 이 기업의 사용 사례에는 화물 물류, Customer 360 맞춤 설정 등이 있습니다.

솔루션 및 장점

J.B. Hunt는 Databricks 레이크하우스 플랫폼을 사용하여 북미에서 가장 안전하고 효율적인 화물 마켓플레이스를 구축함으로써 물류를 간소화하고 운송업체 경험을 최적화하며 원가를 절감하는 효과를 누리고 있습니다.

- Carrier 360 플랫폼을 통한 실시간 경로 최적화, 운전자 추천 등을 위해 Delta Lake의 데이터 페더레이션, 민주화 기능 활용
- 노트북을 사용하여 데이터 팀의 생산성을 강화해 더 많은 사용 사례를 더 빨리 제공
- MLflow를 사용하여 운전자 경험을 개선하는 새 모델 배포 속도 향상

$270만
IT 인프라 비용 절약, 수익성 향상

5%
물류 개선으로 매출 증가

99.8% 빨라진 속도
더 나은 운송업체 경험을 위한 추천

databricks

Databricks 소개

Databricks는 데이터 및 AI 회사입니다. Comcast, Condé Nast, H&M을 비롯한 전 세계 5,000여 개 기업을 위시하여, 포춘 500대 기업 중 40% 이상이 Databricks 레이크하우스 플랫폼을 사용하여 데이터, 분석 및 AI를 통합합니다. Databricks 본사는 샌프란시스코에 있으며 전 세계에 지사를 운영하고 있습니다. Databricks는 Apache Spark™, Delta Lake와 MLflow 원작자가 창립한 회사로서 데이터 팀이 온 세상의 까다로운 문제를 해결하도록 돕고자 합니다. Databricks에 대해 더 알고 싶으시다면 Twitter, LinkedIn, Facebook을 팔로우해 주세요.

무료 체험 시작

개인별 맞춤형 데모 문의
databricks.com/contact

databricks