O'REILLY®
Technical Guide

Understanding ETL

Data Pipelines for Modern Data Architectures

Matt Palmer

databricks

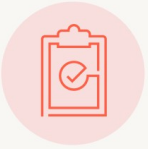
Delta Live Tables

Reliable data pipelines made easy

Delta Live Tables (DLT) is the first ETL framework that uses a simple declarative approach to building reliable data pipelines. DLT automatically manages your infrastructure at scale so data analysts and engineers can spend less time on tooling and focus on getting value from data.

Accelerate ETL Development

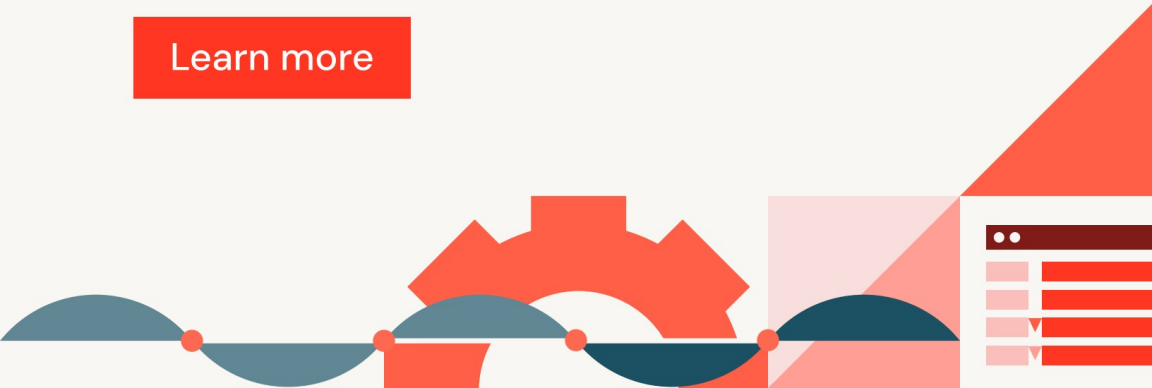
Automatically manage your infrastructure



Have confidence in your data

Simplify batch and streaming

Learn more



ETL 이해

최신 데이터를 위한 데이터 파이프라인 아키텍처

초기 출시 eBook을 사용하면 작가가 집필하는 그대로 편집되지 않은 원시 콘텐츠인 가장 초기 형태의 책을 얻을 수 있으므로 해당 타이틀이 공식 출시되기 훨씬 전에 이러한 기술을 활용할 수 있습니다.

매트 팔머

베이징•보스턴 파넘 세바스토폴 도쿄•

O'REILLY®

Matt Palmer의 ETL 이해

미국에서 인쇄되었습니다.

출판사: O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly 도서는 교육, 비즈니스 또는 판매 홍보용으로 구입할 수 있습니다. 대부분의 타이틀에 대해 온라인 버전도 제공됩니다 (http://oreilly.com). 자세한 내용은 기업/기관 영업부(800-998-9938 또는 Corporate@oreilly.com)에 문의하세요.

편집자: Gary O'Brien 및 Aaron Black
프로덕션 편집자: 크리스틴 브라운
인테리어 디자이너: David Futato
표지 디자이너: 카렌 몽고메리
일러스트레이터: 케이트 덜리아

2024년 4월: 초판

초기 릴리스의 개정 내역

2023-10-23: 첫 번째 출시

http://oreilly.com/catalog/errata.csp?isbn=9781098159252를 참조하세요. 릴리스 세부정보를 확인하세요.

이 작업은 O'Reilly와 Databricks 간의 협력의 일부입니다. 편집 독립성 선언문을 참조하세요 .

978-1-098-15925-2

[LSI]

목차

머리말

데이터 엔지니어링의 빵과 버터

귀하가 데이터 엔지니어이거나 데이터 공간의 모든 종류의 실무자(데이터 과학자 및 데이터 분석가인 귀하를 보세요!)라면 ETL이라는 용어에 익숙할 것입니다(또는 적어도 이 용어가 등장하는 것을 들어본 적이 있을 것입니다). ETL은 "Extract Transform Load"를 의미하며, 이 용어는 데이터 엔지니어가 구현, 즉 데이터를 시스템으로 가져와 다양한 방식으로 사용할 수 있도록 준비하는 임무를 맡은 가장 기본적인 프로세스를 나타냅니다. 비즈니스 리더가 지난 해의 마케팅 데이터를 이해하는 데 도움이 되는 대시보드를 구축해야 합니까? ETL이 필요합니다. 기계 학습 모델의 다음 반복을 훈련하기 위해 대규모 데이터 세트를 사용하고 싶으십니까? ETL이 필요합니다.

규정 준수 요구 사항을 충족하기 위해 데이터를 안전하게 저장하고 보관해야 합니까? 짐작하셨겠지만, ETL이 필요합니다. 실제로 데이터로 무엇이든 하고 싶을 때마다 안정적인 프로세스나 파이프라인이 필요할 가능성이 높습니다(데이터 세계의 배관공인 데이터 엔지니어의 일반적인 비유에 기반). 이러한 근본적인 사실은 비즈니스 인텔리전스(BI) 및 분석의 기존 워크로드와 해당 분야의 가장 최첨단의 새로운 발전에도 적용됩니다.

AI의 멋진 신세계

데이터 세계에서는 많은 추세가 왔다가 사라지는 것을 보았습니다. 일부는 공간을 변화시켰고 일부는 일시적인 유행으로 판명되었습니다. 가장 최근의 트렌드는 의심할 여지 없이 생성적 AI(Generative AI)입니다.
어디에서나 AI, LLM(대형 언어 모델) 또는 챗봇에 대해 이야기하는 사람이 있는 것 같습니다. OpenAI의 ChatGPT 출시로 인해 최근 AI에 대한 관심이 높아졌습니다.

이는 언론의 관심과 연구자들 사이에서 확장됩니다. 이제 많은 사람들은 이를 회사가 만들거나 뒤쳐져야 하는 전략적 투자로 간주합니다. 많은 경우 LLM(Large Language Models)의 큰 잠재력은 회사의 독점 데이터 또는 전문 지식에서 발견됩니다.

LLM을 교육하는 데 사용되는 경우 이 데이터는 흥미로운 방식으로 비즈니스 가치를 실현할 수 있습니다. 해당 데이터의 모양, 출처, 생성 방식, 위치를 살펴보면 AI에 활용하기 시작하려면 이동하고, 준비하고, 준비해야 한다는 것이 매우 빨리 분명해집니다. 단일 위치에 통합됩니다. 즉, 추출, 변환 및 로드됩니다. 예, 짐작하셨겠지만, 오늘날 기술 세계에서 일어나는 가장 흥미로운 일조차도 무엇보다도 ETL에 의존하고 있습니다.

변화하는 데이터 환경

생성 AI의 최신 추세 외에도 지난 10년 동안 다른 추세가 데이터 환경을 변화시켰습니다. 한 가지 예는 스트리밍 데이터의 지배력이 커지고 있다는 것입니다. 오늘날 기업은 센서, 웹사이트, 모바일 애플리케이션 등을 통해 실시간으로 방대한 양의 데이터를 생성하고 있습니다. 이러한 현실에서는 실시간 의사 결정에 사용할 수 있도록 데이터를 실시간으로 수집하고 처리해야 합니다. 이제 데이터 엔지니어는 일괄 처리를 넘어 방대한 양의 스트리밍 데이터를 처리할 수 있는 지속적으로 실행되는 파이프라인을 구축하고 관리해야 합니다.

또 다른 예는 데이터 레이크하우스 아키텍처의 출현입니다.
데이터 레이크하우스는 새로운 패러다임입니다(본 논문에서 처음 소개) 데이터 웨어하우스와 데이터 레이크를 통합하는 것을 목표로 합니다. Delta Lake 와 같은 새로운 스토리지 기술 사용 데이터 레이크에 신뢰성과 성능을 추가하는 레이크하우스는 데이터 웨어하우스의 고성능 트랜잭션을 통해 데이터 레이크에서 찾을 수 있는 저렴하고 확장 가능한 데이터 스토리지라는 두 가지 장점을 모두 제공합니다. 이러한 통합을 통해 분석 워크로드(일반적으로 데이터 웨어하우스에서 데이터 분석가가 수행)와 함께 AI 워크로드(일반적으로 데이터 레이크에서 데이터 과학자가 수행)를 모두 실행하고 중복 이중 아키텍처, 일관되지 않은 데이터 거버넌스 및 데이터 복제와 관련된 복잡성을 제거할 수 있습니다.

ETL은 한동안 사용되어 왔던 용어이고 여전히 여전히 관련성이 있지만, ETL을 다시 방문하여 현대 데이터 환경에서 이를 어떻게 구현할 수 있는지 살펴봐야 합니다. ETL은 배치 및 스트리밍 데이터 수집 및 처리를 어떻게 처리할 수 있습니까? 어떻게

데이터 레이크하우스 아키텍처에서 구현될 수 있나요? 이 가이드는 이러한 질문에 대해 조명하고 독자가 최근 추세의 맥락에서 ETL을 이해하는 데 도움이 될 것입니다.

ELT(및 기타 맛)는 어떻습니까?

ETL 외에도 ELT와 같은 용어를 접했다면 두려워하지 마세요!
이것은 오타가 아닙니다. ELT는 Extract Transform Load 프로세스의 순열로, Load와 Transform의 순서가 ETL과 다릅니다.

이 가이드에서 논의된 많은 고려 사항과 원칙은 ETL과 ELT(역방향 ETL과 같은 다른 순열은 물론) 모두에 공통적이므로 앞으로는 ETL이라는 용어를 이러한 모든 항목을 설명하는 일반적인 용어로 사용할 것입니다. 유사한 프로세스. ETL이 구현하려는 프로세스를 정확하게 설명하지 않더라도 관측 가능성, 문제 해결, 확장 및 최적화에 대한 모범 사례와 마찬가지로 데이터 수집, 변환 및 조정을 이해하는 것이 여전히 중요합니다.

따라서 계속 읽어보시고 이 가이드가 도움이 되기를 바랍니다.

제1장

데이터 수집

초기 출시 독자를 위한 참고 사항 초기 출시 eBook을 사용

하면 가장 초기 형태의 책, 즉 저자가 집필하는 그대로의 편집되지 않은 원본 콘텐츠를 얻을 수 있으므로 이러한 타이틀이 공식 출시되기 훨씬 전에 이러한 기술을 활용할 수 있습니다.

이것이 마지막 책의 첫 번째 장이 될 것입니다.

이 책의 내용 및/또는 예제를 개선할 수 있는 방법에 대한 의견이 있거나 이 장에 누락된 자료가 있는 경우 gobrien@oreilly.com으로 편집자에게 문의하십시오.

본질적으로 데이터 수집에는 소스에서 지정된 대상으로 데이터를 전송하는 작업이 포함됩니다. 주요 목표는 스테이징, 처리, 분석 및 AI/ML에 적합한 환경으로 데이터를 안내하는 것입니다. 대규모 조직에서는 내부적으로(팀 간) 데이터를 이동하는 데 중점을 둘 수 있지만, 우리 대부분의 경우 데이터 수집은 외부 소스에서 가져와 내부 대상으로 전달하는 것을 강조합니다.

비즈니스와 제품 개발 모두에서 데이터가 핵심적인 중요성을 갖는 시대에 정확하고 시의적절한 데이터의 중요성은 아무리 강조해도 지나치지 않습니다. 이렇게 데이터에 대한 의존도가 높아짐에 따라 팀은 의사 결정 프로세스를 개선하고 뛰어난 제품을 제작하며 기타 다양한 작업을 수행하기 위해 정보를 추출하는 수많은 "소스"가 생겨났습니다. 예를 들어 마케팅팀

Meta, Google(광고 및 분석 포함), Snapchat, LinkedIn, Mailchimp 등과 같은 여러 광고 및 분석 플랫폼에서 데이터를 검색해야 합니다.

그러나 시간이 지남에 따라 API와 데이터 소스는 수정됩니다. 열이 추가되거나 제거될 수 있고 필드 이름이 변경될 수 있으며 새 버전이 오래된 버전을 대체할 수 있습니다. 단일 소스의 변경 사항을 처리하는 것이 가능할 수 있지만 5개, 10개, 심지어 100개 등 여러 소스의 변경 사항을 저글링하는 것은 어떻습니까? 시급한 과제는 "신진 데이터 팀이 어떻게 일관되고 확장 가능한 방식으로 이러한 다양한 소스를 효율적으로 처리할 수 있는가?"입니다. 데이터 엔지니어로서, 특히 모든 부서의 요구 사항이 지속적으로 증가하는 경우 안정적이고 간단한 데이터 액세스를 제공한다는 평판을 어떻게 보장할 수 있습니까?

데이터 수집: 현재와 과거 데이터 수집에 관해 지

난 10년 동안 많은 변화가 있었지만 수집에 영향을 미치는 두 가지 가장 큰 변화는 소스의 양과 속도에서 비롯됩니다.

이를 수용하기 위해 업계에서는 클라우드로의 이동, 웨어하우스에서 데이터 레이크, 레이크하우스로의 이동, 스트리밍 기술의 단순화 등 여러 가지 변화를 겪었습니다.

이는 추출-변환-로드 워크플로우에서 추출-로드-변환으로의 전환으로 나타났습니다. 주요 차이점은 이제 모든 데이터가 대상 시스템에 로드된다는 것입니다.

ETL 및 ELT라는 용어에 대해 지나치게 현학적으로 표현하는 것은 삼가지만, 거의 모든 최신 데이터 엔지니어링 워크플로우에는 거의 모든 데이터를 클라우드에 스테이징하는 작업이 포함된다는 점을 강조하고 싶습니다.
주목할만한 예외는 수백 조 행의 고도로 세분화된 데이터(예: IoT 또는 센서 데이터)가 매일 처리되는 경우이며, 스테이징 전에 데이터를 집계하거나 삭제하는 것이 합리적입니다.

끊임없는 변화에도 불구하고 추출의 근본적인 진실은 데이터가 소스에서 가져와서 대상에 기록된다는 것입니다. 그러므로 추출에 관한 논의는 바로 여기에 집중되어야 한다.

소스 및 타겟

대부분의 수집은 추출과 연관되어 있지만 로드와도 긴밀하게 연결되어 있습니다. 결국 모든 소스에는 대상이 필요합니다.
이 가이드에서는 귀하가

저장된 웨어하우스 또는 데이터 레이크 - 스토리지는 이 장의 주요 주제가 아닙니다. 따라서 스토리지가 뒷자리에 있는 동안 우리는 여전히 스테이징 모범 사례를 조명하고 이상적인 스토리지 구현의 특징을 강조할 것입니다.

"완벽한" 솔루션이 존재하지 않을 수도 있다는 점을 염두에 두고 아키텍처 설계를 위한 툴킷을 제공하는 것이 우리의 임무입니다. 소스를 평가하고 데이터 수집의 고유한 매듭을 풀기 위한 프레임워크를 살펴보겠습니다. 우리의 높은 수준의 접근 방식은 정보를 바탕으로 적절한 결정을 내릴 수 있도록 풍경을 한눈에 볼 수 있도록 설계되었습니다.

소스

데이터를 수집할 때 가장 먼저 고려해야 할 사항은 데이터 소스의 특성입니다. 앞서 언급한 것처럼 둘 이상의 데이터 소스가 있을 가능성이 높습니다(그렇지 않다면 축하합니다. 생활이 훨씬 쉬워졌습니다). 별도로 가중치를 부여합니다.

엄청난 양의 데이터 소스와 비즈니스 요구 사항의 특성으로 인해(소스를 제거하라는 요청을 받은 적이 거의 없지만 소스를 추가하는 것은 또 다른 목요일에 불과합니다), 그렇지 않은 하나 이상의 소스를 접하게 될 가능성이 높습니다. 단일 솔루션에 적합합니다.

삶의 썰물과 흐름에서와 마찬가지로 데이터 엔지니어링은 적응성의 필요성, 변화하는 요구 사항과 비전통적인 소스 수용, 정확성 사이에서 춤을 춥니다. 신뢰를 쌓는 데는 몇 주, 몇 달, 몇 년이 걸리지만 하루 만에 무너질 수도 있습니다. 그렇다면 소스를 선택할 때 중요한 것은 무엇입니까?

소스 검토 실용적

인 가이드로서 우리는 소스 데이터의 특성과 비즈니스 가치를 얻는 방법을 이해하는 데 도움이 되는 오랜 시간에 걸쳐 검증된 질문을 제시하는 접근 방식을 취합니다.

매우 중요한 입장을 취하는 것이 좋습니다. 소스 데이터가 필요하지 않거나 다른 소스가 비즈니스에 더 적합할 가능성이 항상 있습니다. 귀하는 조직의 데이터 전문가이며 가정을 확인하고 다시 확인하는 것이 귀하의 임무입니다. 행동과 복잡성에 편견을 갖는 것은 정상이지만 본질주의와 단순성을 고려하는 것은 필수적입니다.

소스를 조사할 때 업스트림 데이터에 대해 소프트웨어 엔지니어와 함께 작업할 가능성이 높지만 다운스트림 고려 사항도 마찬가지로 중요하다는 점을 명심하세요. 이를 무시하면 비용이 많이 들 수 있습니다.

몇 주 동안 작업을 수행할 때까지 귀하의 오류가 나타나지 않을 수 있기 때문입니다.

다음은 물어볼 몇 가지 질문입니다.

우리는 누구와 함께 일할 것인가?

인공지능 시대에 우리는 실제 지능을 최우선으로 생각합니다.

모든 데이터 파이프라인에서 가장 중요한 부분은 서비스를 제공할 사람들입니다. 관련된 이해관계자는 누구입니까? 주요 동기는 무엇입니까? OKR(목표 및 주요 결과) 또는 조직의 명령은 인센티브를 조정하고 프로젝트를 빠르게 진행하는 데 유용할 수 있습니다.

데이터는 어떻게 사용되나요?

"누가"와 밀접하게 연관되어 데이터를 어떻게 사용할 것인지는 후속 결정의 지침이 될 것입니다. 이는 이해관계자의 요구 사항을 확인하고 이해관계자가 해결하려고 하는 "문제 뒤에 있는 문제"를 배우는 방법입니다. 모호함을 피하기 위해 기술적인 예/아니요 요구 사항 목록을 적극 권장합니다.

빈도는 얼마나 됩니까?

나중에 더 자세히 논의하겠지만 대부분의 실무자는 즉시 일괄 처리와 스트리밍으로 전환합니다. 모든 데이터는 배치 또는 스트림으로 처리될 수 있지만 일반적으로 스트리밍하려는 데이터의 특성을 말합니다. 우리는 데이터가 경계가 있는지 없는지, 즉 2020년 인구 조사 ACS 데이터 세트처럼 끝나는지, 아니면 연속적인지, 즉 파이버 캐비닛의 로그 데이터인지 먼저 고려하는 것을 옹호합니다.

한계를 고려한 후 사용 가능한 최소 주파수에 따라 소스에서 가져올 수 있는 빈도에 대한 엄격한 제한이 설정됩니다. API가 매일 업데이트되는 경우 보고 빈도에 엄격한 제한이 있습니다. 경계, 속도 및 비즈니스 요구 사항에 따라 데이터 추출 빈도가 결정됩니다.

예상되는 데이터 양은 얼마입니까?

데이터 볼륨은 더 이상 데이터 저장 능력을 제한하는 요소가 아닙니다. 결국 "스토리지는 저렴하고" 컴퓨팅 비용이 많이 들 수 있지만 그 어느 때보다 저렴합니다(수백만 배). 그러나 볼륨은 데이터 작성 및 처리 방법과 원하는 솔루션의 확장성을 밀접하게 알려줍니다.

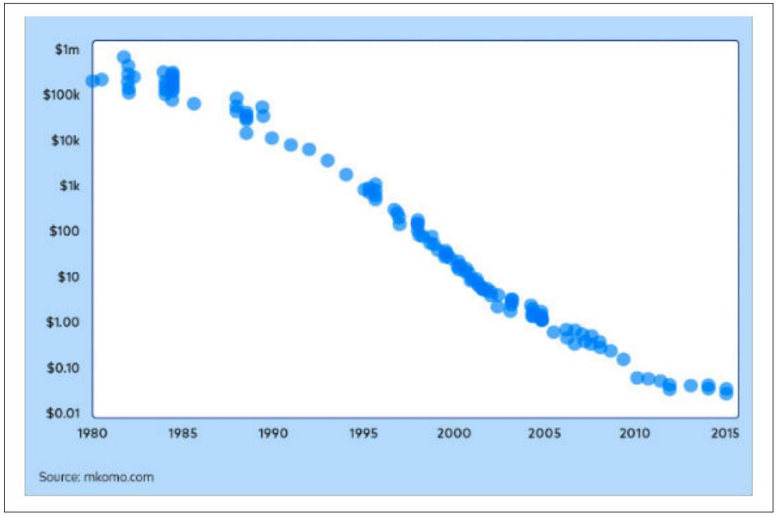


그림 1-1. 1980년부터 2015년까지 GB당 하드 드라이브 비용

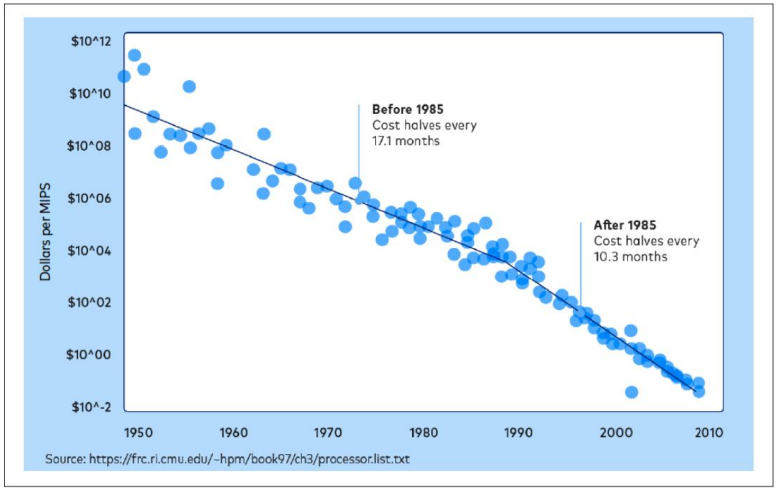


그림 1-2. 컴퓨팅 비용, 초당 수백만 명령(MIPS)

형식은 무엇입니까?

결국 저장 형식을 선택하게 되지만 입력 형식은 중요한 고려 사항입니다. 데이터는 어떻게 전달되나요? API를 통한 JSON 페이로드를 통해 이루어지나요? 아마도 FTP 서버일까요? 운이 좋다면 이미 관계형 데이터베이스 어딘가에 있을 것입니다. 스키마는 어떤 모습인가요? 스키마가 있나요?

끝없이 많은 데이터 형식은 우리의 생계를 흥미롭게 해주지만 동시에 어려운 일이기도 합니다.

품질은 어떻습니까?

데이터 세트의 품질에 따라 변환이 필요한지 여부가 크게 결정됩니다. 데이터 엔지니어로서 사용자를 위한 일관된 데이터 세트를 보장하는 것이 우리의 임무입니다. 누락된 특성을 보완하기 위해 데이터를 과도하게 처리하거나 외부 소스에서 강화해야 할 수도 있습니다.

우리는 마지막 질문에 답하기 위해 이러한 특성을 사용할 것입니다.

데이터는 어떻게 저장되나요?

앞서 언급했듯이 이 보고서는 데이터의 고정된 목적지를 가정합니다. 그럼에도 불구하고 데이터 저장에는 몇 가지 주요 고려 사항이 있습니다. 스테이지를 준비할지 여부(정말 질문인가요?), 비즈니스 요구 사항 및 이해 관계자 적합성이 가장 중요합니다.

탄탄하다.

소스 체크리스트

만나는 모든 소스에 대해 다음 안내 질문을 고려하십시오.
소스 수가 쌓이면 어려워 보일 수도 있지만 기억하세요. 이것은 글쓰기 작업이 아닙니다. 이는 각 소스의 과제를 해결하고 목표를 달성하는 적절한 솔루션을 스케치하는 데 도움이 되는 프레임워크입니다.

반복적으로 느껴질 수도 있지만 이 기초 작업은 장기적인 시간과 리소스를 절약해 줍니다.

질문	메모
우리는 누구와 협력할 것인가? 엔지니어링(결제)	
데이터는 어떻게 사용되나요?	재무 보고 및 분기별 전략 수립
소스가 여러개인가요?	예
형식은 무엇입니까?	반구조화된 API(스트라이프 및 내부)
빈도는 얼마나 됩니까?	시간별
볼륨은 얼마나 됩니까?	약 100,000개의 기존 풀이 있는 일일 약 1,000개의 새 행
어떤 처리가 필요합니까?	열 이름 변경, 보조 소스로부터의 강화 등 데이터 정리
데이터는 어떻게 저장되나요?	Databricks를 통해 델타 테이블에 준비된 데이터 저장

목적지

엔드투엔드 시스템에는 바로 그런 고려 사항이 필요하지만 대부분의 독자는 스토리지 기술이 이미 선택된 기존 시스템에 파이프라인을 구축하는 임무를 맡게 될 것이라고 가정합니다.

데이터 스토리지 기술을 선택하는 것은 이 가이드에서 중점을 두는 범위를 벗어나지만, 데이터 시스템에서 생성되는 총 가치에 매우 중요하므로 대상에 대해 간략하게 살펴보겠습니다. 따라서 목적지를 분석(또는 고려)할 때 소스와 유사한 체크리스트를 활용하는 것이 좋습니다. 일반적으로 소스보다 대상이 훨씬 적으므로 연습이 훨씬 간단합니다.

대상 검토 대상의 주요 차

이점은 이해관계자가 우선순위에 있다는 것입니다. "목적지에는 독특한 특성이 있습니다. 즉, 이해관계자를 중심으로 회전합니다. 이러한 대상은 BI, 분석 및 AI/ML 애플리케이션을 직접적으로 지원하거나 사용자 지향 앱은 말할 것도 없이 단계적 데이터를 처리할 때 간접적으로 지원합니다. 우리는 동일한 체크리스트를 권장하지만, 엔지니어링 목표를 향해 노력하는 동시에 이해관계자의 요구 사항을 충족하는지 확인하기 위해 이해관계자를 향해 약간 프레임을 지정합니다.

우리는 이것이 항상 가능한 것은 아니라는 점을 충분히 인식하고 있습니다. 엔지니어로서 귀하의 역할은 가장 적합한 솔루션을 만드는 것입니다. 비록 그것이 중간 지점에 안착하거나 명확한 답이 없다는 것을 인정하는 것을 의미하더라도 말입니다. 기술의 놀라운 발전에도 불구하고 특정 논리적 딜레마에는 직접적인 해결책이 없습니다.

수집된 데이터 준비 우리는 데이

터 수집에 대한 데이터 레이크 접근 방식을 옹호합니다. 이를 위해서는 분석을 위해 데이터 웨어하우스에 로드하기 전에 대부분의 데이터를 S3, GCP, Azure와 같은 Cloud Storage 시스템에 수집해야 합니다. 한 단계 더 나아가면 레이크하우스가 있습니다. Unity Catalog 및 Delta와 같은 메타데이터 관리 시스템을 활용하여 격동적인 데이터 바다에 구조를 가져오고 이를 통해 웨어하우스 없이 대부분의 분석을 수행할 수 있는 기능을 제공합니다.

데이터 준비를 위한 일반적이고 효과적인 방법은 Delta, Iceberg 또는 Hudi를 포함한 메타데이터 중심의 쪽모이 세공 기반 파일 형식을 활용하는 것입니다. 쪽모이 세공 마루 바닥 - 압축된 원주형

대규모 데이터 세트용으로 설계된 형식 - 이러한 형식은 메타데이터 레이어를 통합하여 시간 여행, ACID 준수 등과 같은 기능을 제공합니다.

세 가지 서로 다른 품질 계층에서 단계적 데이터를 처리하는 메달리온 아키텍처와 이러한 형식을 통합하면 전체 데이터 기록의 보존이 보장됩니다. 이를 통해 새 열 추가, 손실된 데이터 검색 및 기록 데이터 백필이 용이해집니다.

메달리온 아키텍처의 미묘한 차이는 데이터 변환을 다루는 장에서 자세히 설명할 것입니다. 현재 논의에서는 선택한 Cloud Storage 제공업체 내의 '스테이징 레이어'로 모든 데이터를 전달할 수 있는 가능성을 고려하는 것이 적절합니다.

변경 데이터 캡처(CDC)

변경 데이터 캡처(CDC)는 소스 데이터베이스의 변경 사항을 캡처하고 추적하여 다운스트림 시스템을 업데이트하는 데이터 엔지니어링의 설계 패턴입니다. CDC는 전체 데이터베이스를 일괄 로드하는 대신 변경된 데이터만 전송하여 속도와 리소스 사용량을 모두 최적화합니다.

이 기술은 대상 시스템의 데이터가 최신 상태이고 소스와 동기화되도록 보장하므로 실시간 분석 및 데이터 웨어하우징에 매우 중요합니다. 증분 업데이트를 활성화함으로써 CDC는 데이터 파이프라인 전체에서 데이터 가용성과 일관성을 향상합니다.

데이터 엔지니어링 기초에서 Joe Reis와 Matt Housely가 간단히 설명하면 "CDC는… 소스 데이터베이스 시스템에서 변경 사항을 수집하는 프로세스입니다."

CDC는 불필요하게 복잡하고 시간이 많이 소요될 수 있는 프로세스인 SCD 유형 1 및 2 테이블 생성과 같은 분석 및 엔지니어링 패턴에서 중요해졌습니다. CDC를 기본적으로 지원하는 플랫폼이나 솔루션을 선택하면 일반적인 작업을 신속하게 수행할 수 있어 가장 중요한 일에 집중할 수 있습니다. 한 가지 예는 SCD 유형 1에 대한 기본 지원을 제공하는 Databricks의 DLT(Delta Live Tables)입니다 . 2, 일괄 및 스트리밍 파이프라인 모두에서.

목적지 체크리스트

다음은 목적지를 선택할 때 고려해야 할 질문에 대한 간단한 체크리스트입니다.

질문	메모
우리는 누구와 협력할 것인가?	마케팅 - 분석
데이터는 어떻게 사용되나요?	재무 보고 및 분기별 계획
목적지가 여러 개인가요?	데이터는 먼저 GCP에 스테이징된 후 BQ에 기록됩니다.
형식은 무엇입니까?	GCP의 쪽모이 세공 마루. BQ의 구조화/반구조화.
빈도는 얼마나 됩니까?	일괄
볼륨은 얼마나 됩니까?	~1,000개 행/일 새 데이터, ~100,000개의 기존 데이터
어떤 처리가 필요합니까?	데이터 양식/dbt를 사용한 다운스트림 처리
데이터는 어떻게 저장되나요?	Google BigQuery의 소스 테이블

섭취 고려사항

이 섹션에서는 중요한 데이터 특성에 대해 간략히 설명합니다. 이 동안
목록은 완전하지 않으며 특정 상황, 측면의 영향을 받습니다.
빈도, 양, 형식 및 처리와 같은 것이 기본으로 나타납니다.
우려.

빈도

우리는 이미 빈도의 첫 번째 고려 사항을 언급했습니다.
즉, 데이터 세트가 제한되어 있거나 제한되어 있지 않아야 합니다. 범위
비즈니스 요구 사항에 따라 데이터 수집 빈도가 결정됩니다.
배치 또는 스트리밍 형식 중 하나입니다.

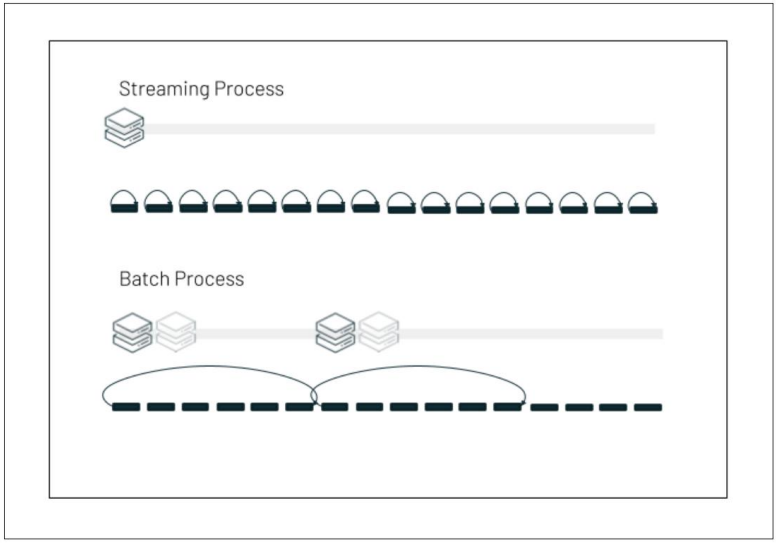


그림 1-3. 지연 시간은 "배치" 또는 "스트리밍" 프로세스를 정의하는 속성입니다. 임의의 지연 시간 임계값을 지나면 데이터가 '스트리밍된' 것으로 간주됩니다.

가장 적절한 빈도와 호환되는 수집 솔루션을 선택하는 데 도움이 되도록 배치, 마이크로 배치, 스트리밍을 우리의 생각과 함께 제시하겠습니다.

일괄

일괄 처리는 데이터를 한 번에 모두 처리하는 것이 아니라 일괄적으로 처리하는 행위입니다. 소스를 반복하는 for 루프와 마찬가지로 배치에는 단순히 제한된 데이터세트를 청크하고 각 구성 요소를 처리하거나 데이터가 도착할 때 제한되지 않은 데이터세트를 처리하는 작업이 포함됩니다.

마이크로 배치

마이크로 배치는 단순히 배치 처리에서 "다이얼을 낮추는 것"입니다.
일반적인 배치 수집이 매일 작동하는 경우 마이크로 배치는 시간 단위 또는 분 단위로 작동할 수 있습니다. 물론 당신은 이렇게 말할 수도 있습니다. “Matt, 이게 도대체 현학적인 말인가요? 지연 시간이 100ms인 마이크로 배치 파이프라인은 제게는 스트리밍과 매우 유사해 보입니다.”

우리는 동의한다!

이 장(및 보고서)의 나머지 부분에서는 지연 시간이 짧은 마이크로 배치 솔루션을 스트리밍 솔루션으로 지칭하겠습니다.

SSS(Spark 구조적 스트리밍)입니다. "기술적으로" 마이크로 배치인 반면, 수백 밀리초의 대기 시간은 SSS를 효과적으로 실시간 솔루션으로 만듭니다.

스트리밍 스

트리밍은 데이터 세트가 생성될 때 제한이 있거나 제한되지 않은 데이터 세트를 지속적으로 읽는 것을 의미합니다.

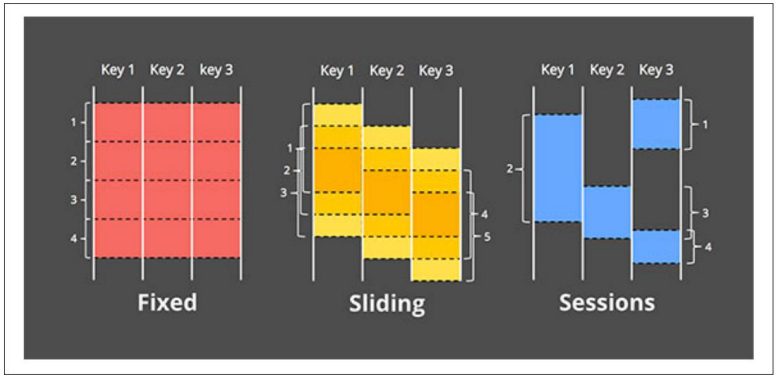


그림 1-4. [자막이 나올 예정]

스트리밍에 대해 자세히 논의하지는 않지만 추가 연구가 필요합니다. 스트리밍 데이터 소스를 포괄적으로 이해하려면 Streaming 101, Streaming Databases 및 The Fundamentals of Data Engineering과 같은 리소스를 살펴보는 것이 좋습니다.

행동 양식. 제한되지 않은 데이터를 스트리밍하는 일반적인 방법은 다음과 같습니다.

윈도우잉 시간
: 적 경계를 기준으로 데이터 소스를 유한한 덩어리로 분할하는 것입니다.

고정 창 데이터는 기
: 본적으로 "마이크로 배치"되어 작은 고정 창에서 대상으로 읽혀집니다.

슬라이딩 창 고정 창
: 과 비슷하지만 경계가 겹치는 점이 있습니다.

세션 이벤

트 시퀀스가 비활성 간격으로 분리되는 동적 창 - 세션에서 "창"은 데이터 자체에 의해 정의됩니다.

시간에 구애받지 않

음 시간이 중요하지 않고 종종 배치 워크로드를 활용하는 데이터에 적합합니다.

실제 이벤트 시간과 이벤트 시간을 구별하는 것이 중요합니다.

불일치가 자주 발생하기 때문에 처리 시간이 걸립니다.

메시지 서비스. "메시지 서비스"라고 하면 스트리밍 데이터를 전달하고 전송하는 "전송 계층" 또는 시스템을 의미합니다.

Kafka

2011년 LinkedIn에서 시작된 Apache Kafka는 메시지 대기열 시스템으로 시작했지만 빠르게 분산 스트리밍 플랫폼으로 발전했습니다. Kafka의 디자인은 높은 처리량과 확장성을 허용하지만 본질적인 복잡성은 많은 사람들에게 여전히 장애물로 남아 있습니다.

Redpanda

Kafka의 대안으로 개발된 Redpanda는 단순화된 구성 및 설정으로 비슷한 성능을 자랑합니다. Redpanda는 Java가 아닌 C++를 기반으로 하며 Kafka API와 호환됩니다.

스트림 처리 엔진. 스트림 처리는 실시간 데이터(스트림)를 분석하고 이에 대한 조치를 취하는 것입니다. Kafka의 수명을 고려할 때 가장 인기 있고 잘 알려진 세 가지 스트림 처리 도구는 다음과 같습니다.

플링크

가동 중지 시간을 최소화하면서 무제한 및 제한된 데이터 세트를 모두 지속적으로 처리하는 오픈 소스 엔진입니다.

Apache Flink는 인메모리 계산을 통해 짧은 대기 시간을 보장하고 단일 실패 지점을 제거하여 고가용성을 제공하며 수평적으로 확장됩니다.

Spark 구조적 스트리밍 실시간 데이

터 처리를 처리하도록 설계된 Apache Spark 에코시스템의 일부입니다. 이는 스트리밍 데이터에 Spark의 DataFrame 및 DataSet API의 친숙함과 강력한 기능을 제공합니다. 불꽃

데이터 처리 분야에서 Apache Spark의 인기와 Databricks와 같은 도구에서 엔진의 편재성을 고려하면 매력적인 옵션일 수 있습니다.

Kafka Streams
: Kafka를 기반으로 구축된 라이브러리로 상태 저장 처리 기능을 제공하지만 Java와의 연결이 제한될 수 있습니다.

스트림 처리 단순화. 비교적 새로운 여러 솔루션은 성능과 단순한 개발 주기에 초점을 맞춘 간단한 클라이언트를 제공하여 스트림 처리를 단순화합니다.

관리형 플랫폼
: Databricks를 예로 들어: Delta와 같은 도구 활용
라이브 테이블 또는 간단히 Spark Streaming 작업을 실행
Databricks 런타임은 복잡성을 강력하게 추상화하고 스트리밍 시스템 구축 프로세스를 대폭 단순화할 수 있습니다.
항목.

Confluent-Kafka
: Kafka 기능을 Python에 도입하려는 시도이지만 Java에 비해 아직 초보적입니다.
Confluent-Kafka는 psycopg2가 Postgres 클라이언트 라이브러리인 것과 같은 방식으로 단순한 클라이언트 라이브러리입니다.

Bytewax
: 스트림 처리를 처리하는 보다 직관적이고 Python적인 방법을 제공하여 더 많은 개발자가 더 쉽게 액세스할 수 있도록 함으로써 격차를 해소하는 것을 목표로 하는 라이브러리입니다. Rust를 기반으로 구축된 Bytewax는 성능이 뛰어나고 Flink보다 단순하며 피드백 루프가 짧고 배포/확장성이 쉽습니다.

Apache Beam 또는 Estuary Flow와 같은 스트림 처리를 통합하거나 스트림 처리를 데이터베이스(스트리밍 데이터베이스)와 직접 결합하려는 새로운 도구의 인기가 높아지고 있습니다.
스트리밍 시스템을 권장합니다 및 스트리밍 데이터베이스 자세히 살펴보려면.

스트리밍 환경은 복잡하기는 하지만 특히 Databricks와 같은 관리형 플랫폼과 Spark Structured Streaming과 같은 대기 시간이 짧은 마이크로 배치 솔루션을 고려할 때 단순화와 사용자 친화성 측면에서 큰 진전을 보였습니다.

많은 사람들이 실시간 데이터를 궁극적인 목표로 생각하지만 우리는 "적시" 접근 방식을 강조합니다. 다른 솔루션과 마찬가지로 대기 시간은 다음과 같습니다.

무한히 최적화되지만 솔루션 비용(및 복잡성)은 그에 비례하여 증가합니다. 대부분의 사람들은 일별 또는 반일별 데이터에서 시간별/시간별 데이터로 전환하는 것이 완벽하게 수용 가능한 솔루션임을 알게 될 것입니다.

유효 탑재량

용량

볼륨은 데이터 수집 결정에서 중추적인 요소로, 처리 및 스토리지 솔루션의 확장성에 영향을 미칩니다. 데이터 볼륨을 평가할 때는 다음과 같은 요소를 고려해야 합니다.

비용 볼

: 륨이 높을수록 스토리지 및 컴퓨팅 리소스 측면에서 비용이 증가하는 경우가 많습니다. 솔루션에 따라 창고, 호수 또는 호수 주택과 관련된 보관/준비 비용을 포함하여 비용 요소를 예산 및 프로젝트 요구 사항에 맞게 조정하십시오.

지연 시간 실

: 시간, 거의 실시간 또는 일괄 데이터 수집이 필요한지 여부에 따라 지연 시간이 중요한 요소가 될 수 있습니다. 실시간 처리에는 더 많은 리소스가 필요하므로 비용이 더 많이 들 수 있습니다.

처리량/확장성 엄청난 양의 수신 데

: 이터를 처리하는 수집 도구의 능력을 아는 것이 중요합니다. 데이터 원본이 대량의 데이터를 생성하는 경우 수집 도구는 병목 현상을 일으키지 않고 해당 데이터를 수집할 수 있어야 합니다.

보존 볼륨이 높

: 을수록 데이터 보존 정책이 더욱 중요해집니다. 오래된 데이터를 오래 사용하거나 더 저렴한 장기 스토리지 솔루션으로 옮기는 전략이 필요합니다.

대규모 데이터 세트를 처리하려면 Avro 또는 Parquet와 같은 압축 형식을 사용하는 것이 중요합니다. 각각은 특히 스키마 진화와 관련하여 뚜렷한 장점과 제약 조건을 제공합니다.

구조 및 형태 데이터는 깔끔

: 한 관계형 "구조화된" 데이터부터 보다 자유로운 형식의 "비구조화된" 데이터에 이르기까지 형식과 구조가 매우 다양합니다.

중요한 것은 구조가 품질과 동일하지 않다는 것입니다. 이는 단지 스키마의 존재를 의미할 뿐입니다.

오늘날의 AI 중심 환경에서는 대규모 언어 및 기계 학습 모델의 발전으로 이러한 데이터에서 풍부한 통찰력을 얻을 수 있으므로 구조화되지 않은 데이터의 가치가 치솟고 있습니다. 그럼에도 불구하고 인간은 심층 분석에 있어 구조화된 데이터를 선호하며, 이는 거의 반세기 동안 데이터 분석의 주요 요소인 SQL(구조적 쿼리 언어)의 지속적인 인기로 인해 더욱 강조됩니다.

구조화되지 않았습니다. 앞서 언급했듯이 구조화되지 않은 데이터는 사전 정의된 스키마나 구조가 없는 데이터입니다. 대부분의 경우 이는 텍스트로 표현되지만 다른 형태의 미디어도 구조화되지 않은 데이터를 표현합니다.
비디오, 오디오, 이미지는 모두 수치적으로 분석할 수 있는 요소를 가지고 있습니다. 구조화되지 않은 데이터는 Pierce Brown 소설의 텍스트일 수 있습니다.

> “사람은 자신이 날 수 있다고 생각하지만 점프하는 것을 두려워합니다. 불쌍한 친구가 뒤에서 그를 밀어낸다.” 그는 나를 올려다본다. “좋은 친구가 함께 뛰어요.”

이러한 데이터는 종종 기계 학습이나 AI 애플리케이션에 제공되어 이해관계자의 요구 사항을 포괄적으로 이해해야 할 필요성을 강조합니다. 기계 학습의 복잡성을 고려할 때 이 비정형 데이터를 수집하기 전에 어떻게 활용될지 파악하는 것이 중요합니다.

텍스트 길이나 압축되지 않은 크기와 같은 측정항목이 모양을 측정하는 역할을 할 수 있습니다.

반 구조화. 반구조화된 데이터는 구조화된 데이터와 구조화되지 않은 데이터 사이에 있습니다. XML과 JSON은 널리 사용되는 두 가지 형식입니다. 데이터 플랫폼이 지속적으로 성숙해짐에 따라 반구조화된 데이터를 직접 처리하고 분석하는 능력도 향상되었습니다.

다음 스니펫은 Google BigQuery에서 반구조화된 JSON을 구문 분석하여 목록을 데이터 행으로 피벗하는 방법을 보여줍니다.

```
j_data AS 사용(
        선택하다 (

                JSON '{"친구": ["steph", "julie", "thomas", "tommy", "michelle", "tori",
 "larry"]}'
                ) AS my_friends_json ), l_data
                AS (
        선택하다
                JSON_EXTRACT_ARRAY(
```

```
                JSON_QUERY(j_data.my_friends_json, "$.friends"),
'$'
            ) my_friends_list로
        j_data에서
)
        l_data,

        UNNEST(l_data.my_friends_list)에서 my_friends를 my_friends ORDER BY RAND()로 선택합니다.
```

어떤 상황에서는 데이터 처리 다운스트림을 분석 계층으로 이동하는 것이 가치가 있습니다. 이를 통해 분석가와 분석 엔지니어는 데이터를 쿼리하고 저장하는 방법에 있어 더 큰 유연성을 얻을 수 있습니다. 웨어하우스의 반구조화된 데이터(또는 외부 테이블을 통해 액세스)를 사용하면 테이블 형식 데이터 조작 및 SQL의 모든 이점을 활용하면서 스키마를 변경하거나 데이터가 누락되는 유연성을 얻을 수 있습니다.

하지만 누락된 데이터로 인해 오류가 발생하지 않도록 이 데이터가 제대로 검증되었는지 주의 깊게 확인해야 합니다. NULL 데이터를 부적절하게 고려하여 잘못된 쿼리를 많이 보았습니다. 확장성에 대해서는 5장에서 더 자세히 논의하겠습니다.

JSON의 형태를 설명하는 데에는 키, 값, 중첩 요소 수에 대한 논의가 포함되는 경우가 많습니다. https://jsonlint.com 과 같은 도구를 적극 권장합니다. 그리고 https://jsoncrack.com/ 이를 위해 JSON/XML 데이터의 유효성을 검사하고 형식을 지정하는 VSCode 확장도 존재합니다.

구조화되어 있습니다. 황금색의 "이상적인" 데이터인 구조화된 소스는 고정된 스키마와 변하지 않는 키로 깔끔하게 구성됩니다. 50년 넘게 SQL은 구조화된 데이터를 쿼리하는 데 선택되는 언어였습니다. 구조화된 데이터를 저장할 때 열 수와 테이블 길이(행)에 관심을 두는 경우가 많습니다.

이러한 특성은 구체화, 증분 빌드 및 전체적으로 OLAP 및 OLTP 데이터베이스(열/행 중심)의 사용을 알려줍니다.

오늘날 많은 데이터에는 구조가 부족하지만 여전히 SQL이 분석을 위한 주요 도구입니다. 이것이 바뀔까요? 그럴 수도 있지만 앞서 보여드린 것처럼 SQL은 단순히 반구조화된 형식을 수용하도록 적응할 가능성이 더 높습니다. AI의 발전과 함께 언어 기반 쿼리 도구가 등장하기 시작했지만 SQL은 종종 중개자 역할을 합니다. SQL이 데이터 분석에서 사라지더라도 API로 계속 존재할 가능성이 높습니다.

OLAP 및 OLTP 데이터베이스

데이터 웨어하우징에서 가장 큰 선택은 클라우드 네이티브 데이터베이스를 사용할지 아니면 클라우드에 호스팅되는 기존 데이터베이스를 사용할지 여부입니다. 주요 차이점은 클라우드 네이티브 솔루션의 분산 특성과 열 중심 아키텍처입니다. 이를 더 일반적으로 OLAP(온라인 분석 처리) 및 OLTP(온라인 트랜잭션 처리)라고 합니다.

- OLAP 시스템은 대량의 데이터를 신속하게 처리하도록 설계되었습니다. 이는 일반적으로 분산 처리 및 열 기반 아키텍처를 통해 수행됩니다. 최신 클라우드 기반 데이터베이스는 OLAP 시스템입니다. 세 가지 OLAP 솔루션은 Amazon Redshift, Google BigQuery 및 Snowflake입니다. • OLTP 시스템은 여러 사용자로부터 발생하는 대량의 트랜잭션 데이터를 처리하도록 설계되었습니다. 이는 일반적으로 행 기반 데이터베이스의 형태를 취합니다. 많은 기존 데이터베이스 시스템은 Postgres, MySQL 등 OLTP입니다.

OLAP 시스템은 속도, 안정성 및 낮은 유지 관리 비용으로 인해 분석 및 데이터 과학 팀에서 가장 일반적으로 사용됩니다.
데이터 웨어하우스 선택 시 고려해야 할 사항은 다음과 같습니다.

선택한 컴퓨팅 플랫폼 이러한 기술의 통합
은 나머지 기술 스택에 따라 크게 달라집니다. 예를 들어 조직의 모든 도구가 Amazon 생태계에 있는 경우 Redshift는 Google BigQuery보다 비용 효율적이고 구현이 간단할 수 있습니다.

기능 Snowflake
는 스토리지와 컴퓨팅의 분리를 최초로 지원함으로써 명성을 얻었습니다. BigQuery는 반구조화된 데이터 및 ML 작업을 훌륭하게 지원합니다.

Redshift Spectrum은 S3의 데이터에서 외부 테이블을 생성하는 데 유용한 도구임이 입증되었습니다. 모든 창고에는 강점과 약점이 있으므로 팀의 사용 사례에 따라 평가해야 합니다.

비용
가격 구조는 플랫폼마다 크게 다릅니다. 불행하게도 가격은 불투명할 수 있습니다. 대부분의 경우 플랫폼을 사용하기 전까지는 데이터베이스가 어떻게 사용될지 실제로 알 수 없습니다.

가격 책정의 목표는 데이터베이스를 사용하여 비용을 최소화하는 방법과 그렇게 할 경우 비용이 얼마나 될지 이해하는 것입니다.

경로가 잡힙니다. 예를 들어, 비용이 스캔된 데이터 양(BigQuery)과 직접적으로 연관되어 있는 경우 지능형 분할 및 쿼리 필터링이 큰 도움이 될 수 있습니다. 비용에 대한 직접적인 대답은 없지만 창고는 특히 규모에 따라 비용이 많이 들 수 있으므로 조사해 볼 가치가 있습니다.

체재

원본 데이터는 어떤 형식으로 되어 있나요? JSON과 CSV가 일반적인 선택이지만 형식에 대한 고려 사항은 셀 수 없이 많습니다. 예를 들어 일부 오래된 SFTP/FTP 전송은 압축되어 도착할 수 있으므로 추가 추출 단계가 필요합니다.

데이터 형식에 따라 처리 요구 사항과 사용 가능한 솔루션이 결정되는 경우가 많습니다. Airbyte와 같은 도구는 CSV 소스와 원활하게 통합될 수 있지만 사용자 지정 압축 방법이나 기발한 Windows 인코딩으로 인해 문제가 발생할 수 있습니다(믿고 계십니까?).

가능하다면 친숙하고 널리 사용되는 데이터 형식을 선택하는 것이 좋습니다.
차량 수리와 마찬가지로 형식이 인기가 많을수록 리소스, 라이브러리 및 지침을 더 쉽게 찾을 수 있습니다. 하지만 우리의 경험에 따르면 복잡한 형식과 씨름하는 것은 일종의 통과 의례이지만 그것이 우리 일을 재미있게 만드는 이유 중 하나입니다!

다양성

여러 소스를 다루게 되어 다양한 페이로드를 처리하게 될 가능성이 높습니다. 데이터 다양성은 수집 솔루션을 선택하는 데 큰 역할을 합니다. 이는 서로 다른 데이터 유형을 처리할 수 있을 뿐만 아니라 스키마 변경 및 다양한 형식에 적응할 수 있을 만큼 유연해야 합니다. 다양성은 거버넌스와 관찰 가능성을 특히 어렵게 만듭니다. 이에 대해서는 5장에서 논의하겠습니다.

데이터 다양성을 고려하지 못하면 병목 현상, 대기 시간 증가, 무질서한 파이프라인이 발생하여 수집된 데이터의 무결성과 유용성이 손상될 수 있습니다.

솔루션 선택

팀을 위한 가장 좋은 도구는 소스와 타겟을 지원하는 도구입니다. 각 조직의 고유한 데이터 요구 사항을 고려하여 상황에 따라 선택을 하게 됩니다. 그래도 우린 못해

멘토, 동료, 업계 전문가의 지식을 활용하는 것의 중요성을 충분히 강조하십시오. 컨퍼런스는 비용이 많이 들 수 있지만 지식의 양은 매우 귀중할 수 있습니다. 예산이 부족한 사람들에게는 데이터 커뮤니티가 매우 중요할 수 있습니다. Slack, LinkedIn과 같은 플랫폼 및 업계 뉴스레터를 통해 찾아보세요.

수집 솔루션을 고려할 때 우리는 일반적인 고려 사항과 솔루션별 고려 사항을 고려합니다. 전자는 우리가 고려할 모든 도구에 적용되는 반면 후자는 도구 클래스에만 적용됩니다.

- 일반 고려 사항: 확장성, 구축 비용, 유지 관리 비용, 전환 비용. 일반적인 솔루션에 대해 자세히 알아보려면 Meltano의 5일 데이터 통합 가이드를 적극 권장합니다.

솔루션별 고려 사항은 일반적으로 두 가지 형식 중 하나를 취하는 도구 클래스에 따라 다릅니다.

- 선언적 솔루션은 결과를 결정합니다. 즉, Databricks를 활용하여 클라우드 스토리지 공급자로부터 데이터를 기본적으로 수집하거나 UI를 통해 새로운 Airbyte 연결을 생성합니다. • 필수적인 해결책은 행동을 지

  시합니다. 예를 들어, 저는 Stripe API를 호출하고, 데이터를 인코딩/디코딩하고, 이를 Snowflake에 점진적으로 로드하는 Lambda를 구축합니다.

이러한 각 솔루션에는 장단점이 있습니다. 각 솔루션에 대해 간략하게 설명하고 데이터 통합에 접근하기 위한 권장 방법을 제시하겠습니다.

선언적 솔루션

우리는 선언적 솔루션을 "레거시" 또는 "현대"로 분류합니다.

레거시

Think Talend, Wherescape 및 Pentaho. 이러한 도구에는 강력한 커넥터가 있으며 풍부한 커뮤니티와 광범위한 지원의 이점을 누릴 수 있습니다. 그러나 데이터 환경이 발전함에 따라 이러한 도구 중 상당수는 최신 데이터 스택(MDS)의 요구 사항을 충족하지 못하고 뒤쳐져 있습니다. 특별한 이유가 없는 한 레거시 엔터프라이즈 도구 이외의 다른 도구를 살펴보는 것이 좋습니다.

Modern 여

기 Fivetran, Stitch, Airbyte가 등장합니다.
"커넥터"를 중심으로 설계된 이 도구는 원활하게 연결할 수 있습니다.

최첨단 기술과 최고의 MDS를 활용하는 다양한 소스와 타겟.

토종의

처음 두 가지 솔루션에서는 데이터를 한 소스에서 다른 소스로 이동해야 한다는 가정을 바탕으로 작업하고 있습니다. 하지만 기본적으로 수집을 지원하는 관리형 플랫폼이 있다면 어떨까요? 예를 들어 Databricks는 메시지 버스와 클라우드 스토리지에서 기본적으로 수집할 수 있습니다.

```
스트리밍 테이블 생성 raw_data
AS 선택 *
FROM cloud_files("/raw_data", "json"); SUM(이익)을
선택하여 스트리밍 테이블 clean_data 생성...

원시 데이터에서;
```

데이터 수집에서 선언적 솔루션의 주요 매력은 무엇입니까?
"구축/유지 관리 비용"이 감소합니다.

구축/유지 비용

여기에서 돈을 벌 수 있습니다. 전담 엔지니어가 커넥터 개발 및 유지 관리를 담당합니다. 이는 무거운 작업을 전문가에게 위임한다는 의미입니다. 대부분의 유료 솔루션에는 지원 팀 또는 전담 고객 관리자가 함께 제공되어 귀하의 요구에 맞는 통찰력과 지침을 제공합니다. 이러한 전문가는 데이터 환경에 대한 조감도를 갖고 있으며 특정 데이터 문제에 대한 모호성을 탐색하거나 다른 실무자와 연결하는 데 도움을 줄 수 있습니다.

그러나 한 서비스를 다른 서비스와 구분하는 요소이자 고려해야 할 요소는 확장성입니다.

확장성 이는 기

존 구조를 기반으로 구축하는 것이 얼마나 쉬운지에 관한 것입니다. 새로운 커넥터가 Airbyte 또는 Fivetran에 추가될 가능성은 얼마나 됩니까? 동일한 프레임워크를 기반으로 직접 구축할 수 있나요?
아니면 제품 팀을 위해 몇 주/개월/년을 기다려야 합니까?
Stitch를 사용하면 충분합니까, 아니면 보완적인 솔루션이 필요합니까?
여러 솔루션을 저글링하면 비용이 늘어나고 워크플로가 복잡해질 수 있다는 점을 기억하십시오.

그러나 중요한 단점은 도구를 전환하는 데 드는 비용일 수 있습니다.

전환 비용

여기서 선언적 도구의 한계가 드러납니다.
독점 프레임워크와 특정 변경 데이터 캡처(CDC) 방법으로 인해 다른 도구로 마이그레이션하는 데 비용이 많이 듭니다. 공급업체를 고수하는 것은 필요한 절충안일 수 있지만 잠재적인 솔루션을 평가할 때 이를 고려하는 것이 중요합니다.

필수 솔루션 필수 데이터 수집

접근 방식은 사내 Singer 탭, Lambda 함수, Apache Beam 템플릿 또는 Airflow와 같은 시스템을 통해 조정된 작업일 수 있습니다.

일반적으로 상당한 리소스를 보유한 대규모 조직은 필수 방법론을 채택하는 데 가장 큰 가치를 찾습니다. 맞춤형 사내 수집 프레임워크를 유지 관리하고 확장하려면 일반적으로 여러 데이터 엔지니어 또는 전담 인력의 전문 지식이 필요합니다.

팀.

선언적 솔루션의 가장 큰 이점은 확장성입니다.

확장성 기본적으

로 선언적이라는 것은 사용자 정의라는 의미입니다. 즉, 각 탭과 대상이 비즈니스 요구 사항에 맞게 조정된다는 의미입니다. 데이터 통합 옵션을 탐색할 때 모든 기준을 충족하는 단일 도구는 없다는 사실이 금방 명백해집니다. 표준 솔루션에는 필연적으로 타협이 수반됩니다. 그러나 명령형 접근 방식을 사용하면 원하는 사양에 따라 정확하게 설계할 수 있는 자유가 있습니다.

불행하게도 이 확장성은 구축하고 유지하는 데 엄청나게 많은 비용이 듭니다.

구축/유지 비용

필수 솔루션은 복잡하고 어려운 수집 문제를 해결할 수 있지만 엔지니어는 소스와 대상을 이해해야 합니다.
Stripe ERD 살펴보기 이것은 엄청나게 시간이 많이 걸릴 수 있다는 것을 당신에게 확신시키기에 충분할 것입니다. 또한 스키마 변경, API 지원 중단 등 데이터의 진화하는 특성으로 인해 복잡성이 증폭될 수 있습니다. 단일 데이터 원본을 관리하는 것도 중요하지만 여러 원본으로 확장하는 경우에는 어떻게 되나요?

진정으로 탄력적인 필수 시스템은 모듈성, 테스트 가능성 및 명확성을 강조하면서 소프트웨어 설계의 모범 사례를 통합해야 합니다. 이러한 원칙을 무시하면 시스템 복구 시간이 단축되고 확장성이 저하되어 궁극적으로 비즈니스 운영에 영향을 미칠 수 있습니다. 따라서 우리는 강력한 데이터 엔지니어링 인프라를 갖춘 기업만이 완전히 필수적인 방식을 고려할 것을 제안합니다.

전환 비용

서로 다른 공급자의 형식 간에 잠재적인 비호환성을 고려할 때 하나의 필수 솔루션에서 다른 솔루션으로 전환하는 것이 항상 간단하지는 않을 수 있습니다. 그러나 긍정적으로 보면 Singer와 같은 공통 프레임워크 기반 플랫폼은 더 높은 호환성을 보여 잠재적으로 Fivetran 또는 Airbyte와 같은 순수 선언적 도구에 비해 더 부드러운 전환을 제공할 수 있습니다.

하이브리드 솔루션 통합에서 올

바른 균형을 맞추는 것은 종종 하이브리드 접근 방식을 채택하는 것을 의미합니다. 여기에는 대부분의 통합 작업에 Fivetran과 같은 도구를 활용하는 동시에 고유한 소스에 대한 자체 솔루션을 제작하거나 Airbyte/Meltano와 같은 플랫폼을 선택하고 지원되지 않는 데이터 소스에 대한 사용자 정의 구성 요소를 만드는 것이 포함될 수 있습니다. 또는 S3에서 Redshift로 데이터를 이동하기 위해 Databricks 클라우드 수집 또는 AWS Glue와 같은 기본 솔루션의 기능을 활용할 수도 있습니다.

오픈 소스에 기여하는 것은 하이브리드 환경에서도 보람을 느낄 수 있습니다. 결함이 없는 것은 아니지만 Airbyte 또는 Singer Tap과 같은 하이브리드 커넥터는 광범위한 커뮤니티 지원의 혜택을 받습니다. 특히 해당 부문에 대한 Airbyte의 기여는 시장 역학에 긍정적인 영향을 미쳐 Fivetran과 같은 경쟁업체가 무료 계층을 도입하도록 유도했습니다. 또한 새로운 라이브러리와 도구를 탐색하는 데 적극적인 접근 방식을 권장합니다. 예를 들어 'dlt'는 상당한 가능성을 보여주는 오픈 소스 라이브러리입니다.

자동차 선택과 유사한 데이터 통합을 고려해보세요. 모든 작업에 Formula One 자동차의 성능이 필요한 것은 아닙니다. 더 흔히 필요한 것은 신뢰할 수 있고 다재다능한 차량입니다.

그러나 Toyota는 99%의 용도를 충족하지만 F1 경주에서 Sienna를 찾을 수는 없습니다. 최적의 전략은? 믿을 수 있는 일상용 차량에 의존하되 필요할 경우 고성능을 얻을 수 있는 능력도 확보하십시오.

표 1-1. 선언적/명령적 매트릭스

	선언적	필수 하이브리드	
확장성	낮음/보통 높음		높은
구축/유지 비용 낮음		높은	보통의
가두어 넣다	높은	낮은	낮은

데이터 수집 체크리스트

Stage	주제	메모
Upstream 누구와 협력할 것인가?		엔지니어링(결제)
	데이터는 어떻게 사용되나요?	재무 보고 및 분기별 전략 수립
	소스가 여러개인가요?	예
	형식은 무엇입니까?	반구조화된 API(스트라이프 및 내부)
	빈도는 얼마나 됩니까?	마이크로배치
	볼륨은 얼마나 됩니까?	하루에 약 1,000개의 새 행이 생성됩니다. ~100,000개의 기존 풀
	어떤 처리가 필요합니까?	열 이름 바꾸기 등의 데이터 정리 및 보충 소스로부터 농축
	데이터는 어떻게 저장되나요?	다음을 통해 델타 테이블에 준비된 데이터 저장 데이터브릭스
다운스트림 우리는 누구와 협력할 것인가?		마케팅(분석)
	데이터는 어떻게 사용되나요?	재무 보고 및 분기별 전략 수립
	목적지가 여러 개인가요?	데이터는 먼저 GCP에 스테이징된 다음 GCP에 기록됩니다. 빅쿼리.
	형식은 무엇입니까?	GCP의 쪽모이 세공 마루. 구조화/반구조화 빅쿼리.
	빈도는 얼마나 됩니까?	일괄
	볼륨은 얼마나 됩니까?	하루에 약 1,000개의 새 행이 생성됩니다. ~100,000개의 기존 풀
	어떤 처리가 필요합니까?	Dataform/dbt를 사용한 다운스트림 처리
	데이터는 어떻게 저장되나요?	Google BigQuery의 소스 테이블
솔루션		
업스트림	1. 연결을 위한 FiveTran 2. 타인을 위한 오케스트레이션된 배치 API	메달리온 스테이지
GCP Databrick을 BigQuery로 다운스트림		외부 테이블

저자 소개

Matt Palmer는 Underline Infrastructure의 데이터 엔지니어입니다. 그는 Swarthmore College에서 물리학과 경제학을 전공했습니다. Matt는 분석을 사용하여 제품을 이해하고 데이터 기반 의사결정을 내리는 것을 즐깁니다. 그는 새로운 도구를 만들어 일을 더 효율적으로 만드는 것을 좋아하기 때문에 엔지니어링에 열정을 갖고 있습니다.