databricks

E-BOOK

O livro completo da engenharia de dados 2ª edição

Uma coleção de blogs técnicos, incluindo amostras de código e notebooks

Com conteúdo totalmente novo

Conteúdo

databricks

SEÇÃO

01

Introdução à Engenharia de Dados na Databricks

As organizações percebem que os dados de valor atuam como um ativo estratégico para várias iniciativas relacionadas aos negócios, como o aumento da receita, a melhoria da experiência do cliente, a operação eficiente ou a melhoria de um produto ou serviço. No entanto, o acesso e o gerenciamento de dados para essas iniciativas têm se tornado cada vez mais complexos. A maior parte da complexidade surgiu com a explosão de volumes e tipos de dados, com organizações acumulando uma estimativa de 80% dos dados em formato não estruturado e semiestruturado. À medida que a coleta de dados continua aumentando, 73% deles não são usados para análise ou tomada de decisão. Para tentar diminuir essa porcentagem e tornar os dados mais utilizáveis, as equipes de engenharia de dados são responsáveis por criar pipelines de dados para entregá-los de forma eficiente e confiável. Mas o processo de criação desses pipelines de dados complexos traz uma série de dificuldades:

- Para colocar dados em um data lake, os engenheiros de dados são obrigados a gastar muito tempo programando tarefas de ingestão de dados repetitivas
- Uma vez que as plataformas de dados mudam continuamente, os engenheiros de dados gastam tempo construindo e mantendo e, em seguida, reconstruindo, uma infraestrutura escalável complexa.
- À medida que o pipeline de dados se torna mais complexo, os engenheiros de dados são obrigados a encontrar ferramentas confiáveis para orquestrar esses pipelines.
- Com a crescente importância dos dados em tempo real, são necessários pipelines de dados de baixa latência, que são ainda mais difíceis de construir e manter.
- Por fim, com todos os pipelines escritos, os engenheiros de dados precisam se concentrar constantemente no desempenho, ajustando pipelines e arquiteturas para atender aos SLAs.

Como a Databricks pode ajudar?

Com a Plataforma Databricks Lakehouse, os engenheiros de dados têm acesso a uma solução de engenharia de dados de ponta a ponta para ingerir, transformar, processar, agendar e entregar dados. A Plataforma Lakehouse automatiza a complexidade de construir e manter pipelines e executar cargas de trabalho ETL diretamente em um data lake para que os engenheiros de dados possam se concentrar na qualidade e confiabilidade para gerar insights valiosos.

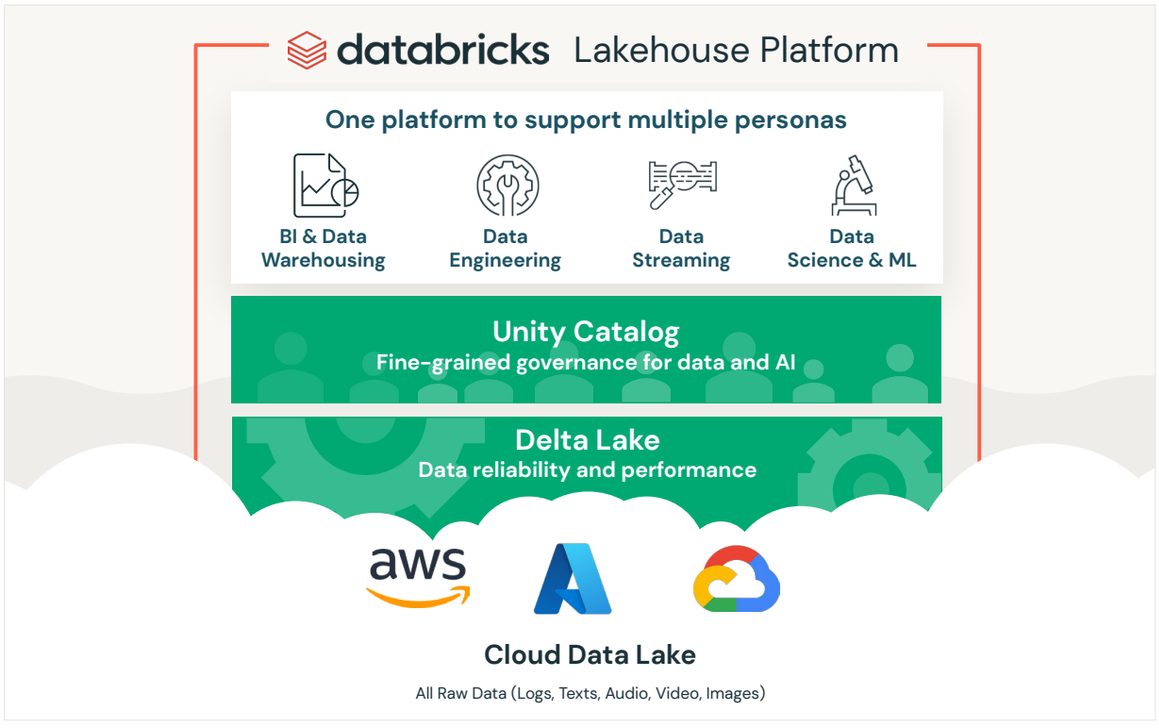


Figura 1
A Plataforma Databricks Lakehouse unifica seus dados, análises e IA em uma plataforma comum para todos os seus casos de uso de dados

Principais diferenciais para engenharia de dados bem-sucedida com Databricks

Ao simplificar uma arquitetura de lakehouse, os engenheiros de dados precisam de uma abordagem pronta e de nível empresarial para criar pipelines de dados. Para ter sucesso, uma equipe de soluções de engenharia de dados deve adotar estes oito principais recursos diferenciais:

Ingestão de dados em escala

Com a capacidade de ingerir petabytes de dados com esquemas de evolução automática, os engenheiros de dados podem fornecer dados rápidos, confiáveis, escaláveis e automáticos para análise, ciência de dados ou machine learning. Isso inclui:

- Processar dados de forma progressiva e eficiente à medida que chegam de arquivos ou fontes de streaming como Kafka, DBMS e NoSQL
- Inferir automaticamente o esquema e detectar alterações de coluna para formatos de dados estruturados e não estruturados
- Rastrear dados de forma automática e eficiente à medida que eles chegam sem intervenção manual
- Evitar a perda de dados resgatando colunas de dados

Pipelines de ETL declarativos

Os engenheiros de dados podem reduzir o tempo e o esforço de desenvolvimento e se concentrar na implementação de lógica de negócios e verificações de qualidade de dados no pipeline de dados usando SQL ou Python. Isso pode ser alcançado por meio de:

- Uso do desenvolvimento declarativo orientado por intenção para simplificar o “como” e definir “o que” resolver
- Criação automática de linhagem de alta qualidade e gerenciamento de dependências de tabela em todo o pipeline de dados
- Verificação automática de dependências ausentes ou erros de sintaxe e gerenciamento de recuperação do pipeline de dados

Processamento de dados em tempo real

Permita que engenheiros de dados ajustem a latência de dados com controles de custo sem a necessidade de conhecer o processamento de fluxo complexo ou implementar lógica de recuperação.

- Evite lidar com fontes de dados de streaming em tempo real e em batch separadamente
- Execute cargas de trabalho de pipeline de dados em compute clusters baseados em Apache Spark™ flexíveis e automaticamente provisionados para escala e desempenho
- Elimine a necessidade de gerenciar a infraestrutura e concentre-se na lógica de negócios para casos de uso downstream

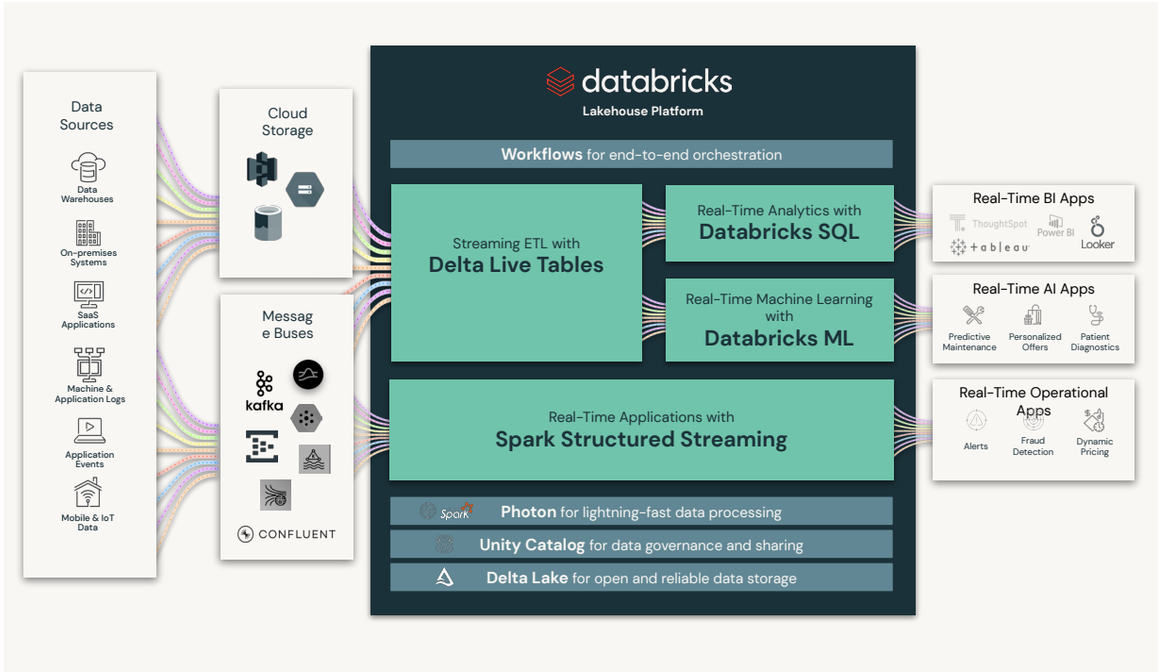


Figura 2
Um conjunto unificado de ferramentas para processamento de dados em tempo real

Orquestração unificada de fluxo de trabalho de dados

Orquestração simples, clara e confiável de tarefas de processamento de dados para pipelines de dados, análises e machine learning com a capacidade de executar múltiplas tarefas não interativas como um grafo acíclico dirigido (DAG) em clusters de compute Databricks. Orquestre tarefas de qualquer tipo (SQL, Python, JARs, Notebooks) em um DAG usando Databricks Workflows, uma ferramenta de orquestração incluída no lakehouse sem necessidade de manter ou pagar por um serviço de orquestração externo.

- Crie e gerencie facilmente múltiplas tarefas com dependências via UI, API ou do seu IDE
- Tenha total observabilidade de todas as execuções de fluxo de trabalho e receba alertas quando as tarefas falharem para solução de problemas rápida e reparo e execução eficientes
- Aproveite a alta confiabilidade do tempo de atividade de 99,95%
- Use clusters de otimização de desempenho que paralelizam jobs e minimizam o movimento de dados com reutilização de clusters

Validação e monitoramento da qualidade dos dados

Melhore a confiabilidade dos dados em todo o data lakehouse para que as equipes de dados possam confiar nas informações para iniciativas de downstream através de:

- Definição de controles de integridade e qualidade de dados dentro do pipeline com expectativas de dados definidas
- Abordagem de erros de qualidade de dados com políticas predefinidas (falha, queda, alerta, quarentena)
- Alavancagem das métricas de qualidade de dados que são capturadas, rastreadas e comunicadas para todo o pipeline de dados

Recuperação automática e tolerante a falhas

Manuseie erros transitórios e recupere-se das condições de erro mais comuns que ocorrem durante a operação de um pipeline com recuperação automática rápida e escalonável, que inclui:

- Mecanismos tolerantes a falhas para recuperar consistentemente o estado dos dados
- Capacidade de rastrear automaticamente o progresso da fonte com pontos de verificação
- Capacidade de recuperar e restaurar automaticamente o estado do pipeline de dados

Observabilidade do pipeline de dados

Monitore o status geral do pipeline de dados de um painel de gráfico de fluxo de dados e monitore visualmente a integridade do pipeline de ponta a ponta para desempenho, qualidade e latência. Os recursos de observabilidade do pipeline de dados incluem:

- Um diagrama de linhagem de alta qualidade e alta fidelidade que fornece visibilidade sobre como os dados fluem para análise de impacto
- Registro granular com desempenho e status do pipeline de dados em nível de linha
- Monitoramento contínuo dos trabalhos de pipeline de dados para garantir operação contínua

Implementações e operações automáticas

Garanta uma entrega confiável e previsível de dados para casos de uso de funções analíticas e machine learning, permitindo implementações e reversões de pipeline de dados fáceis e automáticas para minimizar o tempo de inatividade. Os benefícios incluem:

- Implementação completa, parametrizada e automatizada para a entrega contínua de dados
- Orquestração, testes e monitoramento de ponta a ponta da implementação do pipeline de dados em todos os principais provedores de nuvem

Migrações

Acelerar e reduzir os riscos da jornada de migração para o lakehouse, seja de sistemas legados no local ou de serviços de nuvens distintos.

O processo de migração começa com uma descoberta e avaliação detalhadas para obter insights sobre as cargas de trabalho da plataforma legada e estimar a migração, bem como os custos de consumo da plataforma Databricks. Obtenha ajuda com a arquitetura alvo e como a pilha de tecnologia atual é mapeada para Databricks, seguida de uma implementação em fases com base em prioridades e necessidades de negócios. Por toda parte nesta jornada, as empresas podem aproveitar:

- Ferramentas de automação da Databricks e de suas parcerias de ISV
- SIs globais e/ou regionais que criaram soluções de migração para o Brickbuilder
- Treinamento e serviços profissionais Databricks

Esta é a abordagem recomendada para uma migração bem-sucedida, em que os clientes observaram uma redução de custos de 25 a 50% e um tempo de retorno duas a três vezes mais rápido para seus casos de uso.

Governança unificada

Com o Unity Catalog, as equipes de engenharia e governança de dados se beneficiam de um catálogo de dados para toda a empresa com uma interface única para gerenciar permissões, centralizar a auditoria, rastrear automaticamente a linhagem de dados até o nível da coluna e compartilhar dados entre plataformas, nuvens e regiões. Benefícios:

- Descubra todos os seus dados em um só lugar, não importa onde estejam, e gerencie centralmente permissões de acesso refinadas usando uma interface ANSI baseada em SQL
- Aproveite a linhagem de dados automatizada em nível de coluna para realizar análises de impacto de quaisquer alterações de dados no pipeline e conduzir análises de causa raiz de quaisquer erros no pipeline de dados
- Audite centralmente os direitos e o acesso aos dados
- Compartilhe dados entre nuvens, regiões e plataformas de dados, mantendo uma única cópia dos seus dados no armazenamento em nuvem

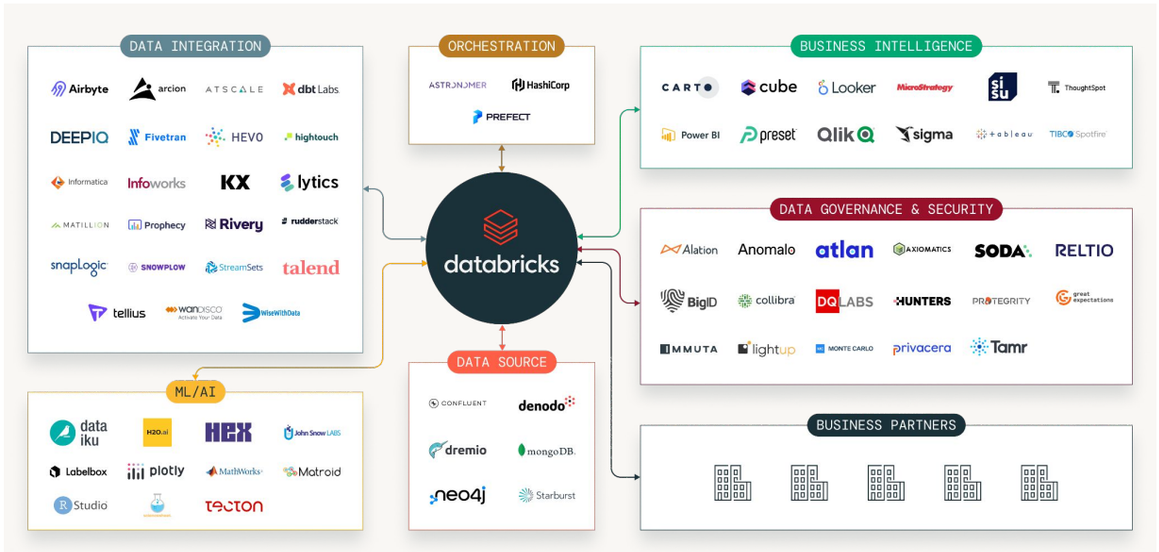


Figura 3
A Plataforma Databricks Lakehouse se integra a uma grande coleção de tecnologias

Um rico ecossistema de soluções de dados

A Plataforma Databricks Lakehouse é construída em tecnologia de código aberto e usa padrões abertos para que as principais soluções de dados possam ser aproveitadas com qualquer coisa que você crie no lakehouse. Uma grande coleção de parcerias tecnológicas torna fácil e simples integrar a tecnologia na qual você confia ao migrar para Databricks e saber que você não está preso a uma pilha fechada de tecnologia de dados.

Conclusão

À medida que as organizações se esforçam para se tornarem orientadas por dados, a engenharia de dados é um ponto focal para o sucesso. Para fornecer dados confiáveis, os engenheiros de dados não precisam gastar tempo desenvolvendo e mantendo manualmente um ciclo de vida ETL de ponta a ponta. As equipes de engenharia de dados precisam de uma maneira eficiente e escalável de simplificar o desenvolvimento de ETL, melhorar a confiabilidade dos dados e gerenciar as operações.

Conforme descrito, os oito principais recursos de diferenciação simplificam o gerenciamento do ciclo de vida de ETL, automatizando e mantendo todas as dependências de dados, aproveitando os controles de qualidade integrados com monitoramento e fornecendo visibilidade profunda das operações de pipeline com recuperação automática. As equipes de engenharia de dados agora podem se concentrar na criação fácil e rápida de pipelines de dados prontos para produção de ponta a ponta e confiáveis usando apenas SQL ou Python em batch e streaming que fornecem dados de alto valor para funções analíticas, ciência de dados ou machine learning.

Siga as práticas recomendadas comprovadas

Na próxima seção, descrevemos as práticas recomendadas para casos de uso de ponta a ponta da engenharia de dados extraídos de exemplos do mundo real. Da ingestão e do processamento em tempo real até funções analíticas e machine learning, você aprenderá como converter dados brutos em dados acionáveis.

À medida que você explora o restante deste guia, pode encontrar conjuntos de dados e exemplos de código nos vários **aceleradores de soluções Databricks**, para que você possa colocar a mão na massa enquanto explora todos os aspectos do ciclo de vida dos dados na Plataforma Databricks Lakehouse.

Comece a experimentar esses notebooks Databricks gratuitos.

databricks

SEÇÃO

02

Orientações e práticas recomendadas

SEÇÃO 2.1

As 5 principais dicas de desempenho da Databricks

de **BRYAN SMITH** e **ROB SAKER**

10 de março de 2022

Como arquitetos de soluções, trabalhamos em estreita colaboração com os clientes todos os dias para ajudá-los a obter o melhor desempenho do seu trabalho no Databricks, e muitas vezes acabamos por dar o mesmo conselho. Não é incomum conversar com um cliente e obter o dobro, o triplo ou até mais desempenho com apenas alguns ajustes. Qual é o segredo? Como fazemos isso? Aqui estão as cinco principais coisas que vemos que podem causar um enorme impacto no desempenho que os clientes obtêm do Databricks.

Aqui está um resumo:

- **Use clusters maiores.** Pode parecer óbvio, mas este é o problema número um que vemos. Na verdade, não é mais caro usar um cluster grande para uma carga de trabalho do que usar um cluster menor. É apenas mais rápido. Se há algo que você deveria aprender com estes artigos, é isso. Leia a seção 1. Sério.
- **Use o Photon**, o novo mecanismo de execução super-rápido da Databricks. Leia a seção 2 para saber mais. Você não vai se arrepender.
- **Limpe suas configurações**. Configurações transferidas de uma versão do Apache Spark™ para outra podem causar problemas enormes. Limpe tudo! Leia a seção 3 para saber mais.
- **Use o Delta Caching**. Há uma boa chance de você não estar usando o cache corretamente, se é que está usando. Consulte a seção 4 para saber mais.
- **Cuidado com a avaliação preguiçosa**. Se isso não significa nada para você e você está escrevendo código Spark, vá para a seção 5.
- **Dica bônus! O design da tabela é muito importante**. Abordaremos isso em blogs futuros, mas, por enquanto, confira o guia sobre as práticas recomendadas do Delta Lake.

1. Dê potência aos seus clusters!

Este é o erro número um que os clientes cometem. Muitos clientes criam pequenos clusters de dois workers com quatro núcleos cada, e leva uma eternidade para fazer qualquer coisa. A preocupação é sempre a mesma: não querem gastar muito dinheiro em clusters maiores. O problema é o seguinte: **na verdade, não é mais caro usar um cluster grande para uma carga de trabalho do que usar um cluster menor. É apenas mais rápido.**

O segredo é que você está alugando o cluster pela duração da carga de trabalho. Portanto, se você ativar esses dois clusters worker e levar uma hora, estará pagando por esses workers pela hora inteira. Mas se você ativar quatro clusters worker e levar apenas meia hora, na verdade, o custo será o mesmo. E essa tendência continua enquanto houver trabalho suficiente para os clusters realizarem.

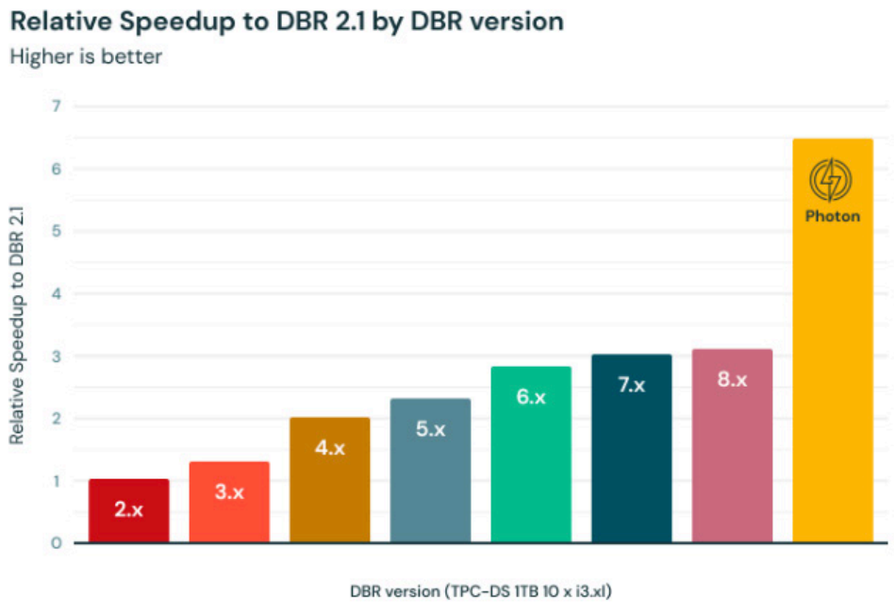
Aqui está um cenário hipotético que ilustra esse ponto:

Número workers	Custo por hora	Duração da carga de trabalho (horas)	Custo da carga de trabalho
1	US$ 1	2	US$2
2	US$ 2	1	US$ 2
4	US$ 4	0,5	US$ 2
8	US$ 8	0,25	US$ 2

Observe que o custo total da carga de trabalho permanece o mesmo, enquanto o tempo real necessário para a execução do job cai significativamente. Aumente as especificações dos clusters do Databricks e acelere suas cargas de trabalho sem gastar mais dinheiro. Não dá para ser mais simples do que isso.

2. Use o Photon

Nossos colegas da engenharia reescreveram o mecanismo de execução Spark em C++ e o apelidaram de Photon. Os resultados são impressionantes!



Além das melhorias óbvias devido à execução do mecanismo em código nativo, eles também fizeram uso de recursos de desempenho em nível de CPU e melhor gerenciamento de memória. Além disso, reescreveram o código Parquet em C++. Isso também faz com que escrever para o Parquet e o Delta (com base no Parquet) seja super-rápido.

Mas também sejamos claros sobre o que o Photon está acelerando. Ele melhora a velocidade de computação para quaisquer funções ou operações integradas, bem como grava em Parquet ou Delta. Joins? Sim! Agregações? Claro! ETL? Também! Aquela UDF (função definida pelo usuário) que você escreveu? Poxa, mas não vai ser útil nisso. O trabalho que passa a maior parte do tempo lendo um antigo banco de dados local? Também não vai ajudar nisso, infelizmente.

A boa notícia é que ele ajuda onde pode. Portanto, mesmo que parte do seu trabalho não possa ser acelerada, as outras partes serão. Além disso, a maioria dos trabalhos são escritos com operações nativas e passam muito tempo escrevendo para Delta, e o Photon ajuda muito nisso. Experimente. Você pode se impressionar com os resultados!

3. Limpe as configurações antigas

Sabe aquelas configurações do Spark que você vem carregando de versão em versão e ninguém sabe mais o que elas fazem? Elas podem não ser inofensivas. Vimos jobs passarem de horas para minutos simplesmente limpando configurações antigas. Pode ter havido uma peculiaridade em uma versão específica do Spark, um ajuste de desempenho que não envelheceu bem ou algo retirado de alguns blogs em algum lugar que nunca fez sentido. No mínimo, vale a pena revisar as configurações do Spark se você estiver nessa situação. Muitas vezes, as configurações-padrão são as melhores, e elas estão ficando cada vez melhor. Suas configurações podem estar atrasando você.

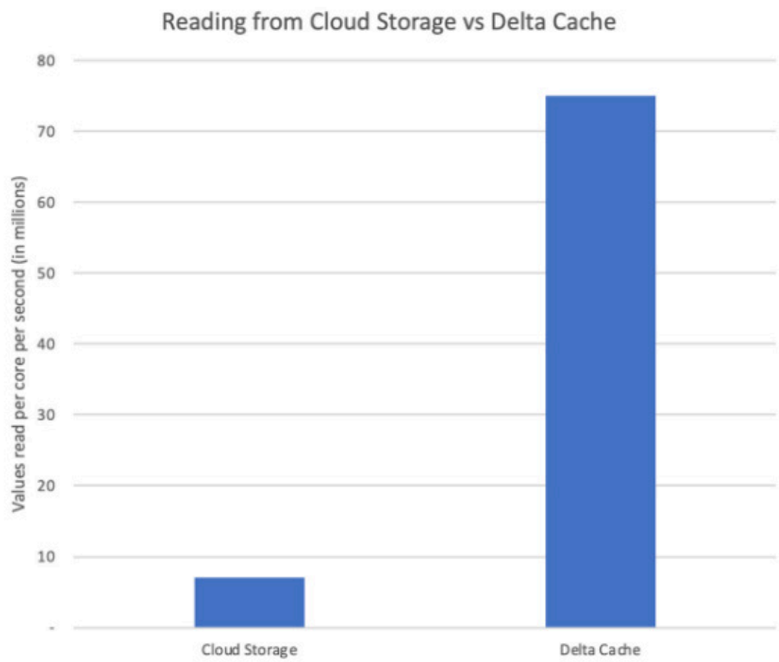
4. O Delta Cache é seu amigo

Isso pode parecer óbvio, mas você ficaria surpreso com a quantidade de pessoas que não estão usando o Delta Cache, que carrega dados do armazenamento em nuvem (S3, ADLS) e os mantém nos SSDs worker para acesso mais rápido.

Se você estiver usando Databricks SQL Endpoints, você está com sorte. Eles têm cache ativado por padrão. Na verdade, recomendamos usar a tabela CACHE SELECT * FROM para pré-carregar suas tabelas principais ao iniciar um endpoint. Isso garantirá velocidades extremamente rápidas para qualquer query nessas tabelas.

Se você estiver usando clusters regulares, use a série i3 na Amazon Web Services (AWS), a série L ou a série E no Azure Databricks ou n2 no GCP. Todos eles terão SSDs rápidos e cache habilitados por padrão.

Claro, sua milhagem pode variar. Se você estiver fazendo BI, o que envolve ler as mesmas tabelas repetidamente, o armazenamento em cache oferece um impulso incrível. No entanto, se você simplesmente ler uma tabela uma vez e escrever os resultados como em algum job ETL, poderá não obter muitos benefícios. Você conhece seu trabalho melhor do que ninguém. Vá em frente e conquiste.



5. Cuidado com a avaliação lenta

Se você é analista de dados ou cientista de dados apenas usando SQL ou fazendo BI, você pode pular esta seção. No entanto, se você estiver trabalhando em engenharia de dados e escrevendo pipeline ou processando usando Databricks/Spark, continue lendo.

Ao escrever código Spark como select, groupBy, filter, etc., você está realmente construindo um plano de execução. Você notará que o código retorna quase imediatamente ao executar essas funções. Isso porque, na verdade, ele não está fazendo nenhum cálculo. Portanto, mesmo que você tenha petabytes de dados, ele retornará em menos de um segundo.

No entanto, depois de escrever seus resultados, você perceberá que leva mais tempo. Isso se deve à avaliação lenta. Só quando você tenta exibir ou gravar resultados é que seu plano de execução é realmente executado.

```
—————————
# Construa um plano de execução
# Isso retorna em menos de um segundo, mas não funciona
df2 = (df
  .join(...)
  .select(...)
  .filter(...)
         )

# Agora execute o plano de execução para obter resultados
df2.display()
———————
```

No entanto, há um problema aqui. Cada vez que você tenta exibir ou escrever resultados, o plano de execução é executado novamente. Vejamos o mesmo bloco de código, mas estenda-o e faça mais algumas operações.

```
—————————
# Construa um plano de execução
# Isso retorna em menos de um segundo, mas não funciona
df2 = (df
  .join(...)
  .select(...)
  .filter(...)
         )

# Agora execute o plano de execução para obter resultados
df2.display()

# Infelizmente, isso executará o plano novamente, incluindo filtragem,
combinação, etc.
df2.display()

# Então será que…
df2.count()
———————
```

O desenvolvedor deste código pode muito bem estar pensando que está apenas imprimindo os resultados três vezes, mas o que na verdade ele está fazendo é iniciar o mesmo processamento três vezes. Opa. Isso é muito trabalho extra. Este é um erro muito comum que cometemos. Então, por que existe uma avaliação lenta e o que fazemos a respeito?

Resumindo, o processamento com avaliação lenta é muito mais rápido do que sem ela. O Databricks/Spark analisa o plano de execução completo e encontra oportunidades de otimização que podem reduzir o tempo de processamento em ordens de magnitude. Isso é ótimo, mas como evitamos o cálculo extra? A resposta é bastante direta: salve os resultados do cálculo que você vai reutilizar.

Vejamos o mesmo bloco de código novamente, mas desta vez vamos evitar recálculo:

```
# Construa um plano de execução
# Isso retorna em menos de um segundo, mas não funciona
df2 = (df
  .join(...)
  .select(...)
  .filter(...)
        )

# salve
df2.write.save(path)

# carregue de volta
df3 = spark.read.load(path)

# agora use
df3.display()

# não está mais fazendo nenhum cálculo extra.  Sem junções, filtros, etc.
Já está feito e salvo
df3.display()

# nem é isso
df3.count()
```

Funciona especialmente bem quando o Delta Caching está ativado. Resumindo, você se beneficia muito da avaliação lenta, mas é algo em que muitos clientes tropeçam. Portanto, esteja ciente de sua existência e salve os resultados que você reutiliza para evitar cálculos desnecessários.

Comece a experimentar com estes notebooks gratuitos da Databricks.

SEÇÃO 2.2

Como criar um perfil do PySpark

de **XINRONG MENG, TAKUYA UESHIN, HYUKJIN KWON** e **ALLAN FOLTING**

6 de outubro de 2022

No Apache Spark™, as APIs Python declarativas são compatíveis com cargas de trabalho de big data. Elas são poderosas o suficiente para lidar com os casos de uso mais comuns. Além disso, os UDFs do PySpark oferecem mais flexibilidade, pois permitem que os usuários executem código Python arbitrário no mecanismo Apache Spark™. Os usuários só precisam indicar "o que fazer"; O PySpark, como uma sandbox, encapsula "como fazer". Isso torna o PySpark mais fácil de usar, mas pode ser difícil identificar gargalos de desempenho e aplicar otimizações personalizadas.

Para resolver a dificuldade mencionada acima, o PySpark é compatível com várias ferramentas de criação de perfil, todas baseadas em cProfile, uma das implementações-padrão do criador de perfil do Python. Os PySpark Profilers fornecem informações como o número de chamadas de função, o tempo total gasto em uma determinada função e o nome do arquivo, bem como o número da linha para ajudar na navegação. Essas informações são essenciais para expor lacunas em seus programas PySpark e permitir que você tome decisões de melhoria de desempenho.

Perfil do driver

As aplicações PySpark são executadas como conjuntos independentes de processos em clusters, coordenados pelo objeto SparkContext no programa do driver. Do lado do driver, PySpark é um processo Python regular; portanto, podemos traçar o perfil dele como um programa Python normal usando cProfile conforme ilustrado abaixo:

```
import cProfile

with cProfile.Profile() as pr:
    # Seu código

pr.print_stats()
```

Perfil de worker

Executores são distribuídos em nós worker nos clusters, o que introduz complexidade porque precisamos agregar perfis. Além disso, um processo worker Python é gerado por executor para execução do PySpark UDF, o que torna o perfil mais complexo.

O criador de perfil UDF, introduzido no Spark 3.3, supera todos esses obstáculos e se torna uma ferramenta importante para criar perfil de workers para aplicações PySpark. Vamos ilustrar como usar o criador de perfil UDF com um exemplo simples de Pandas UDF.

Primeiro, um DataFrame do PySpark com 8.000 linhas é gerado, como mostrado abaixo.

```
sdf = spark.range(0, 8 * 1000).withColumn(
  'id', (col('id') % 8).cast('integer') # 1000 rows x 8 groups (if group
by 'id')
).withColumn('v', rand())
```

Depois, vamos agrupar pela coluna id, o que resulta em 8 grupos com 1.000 linhas por grupo.

O Pandas UDF plus_one é então criado e aplicado conforme mostrado abaixo:

```
import pandas as pd

def plus_one(pdf: pd.DataFrame) -> pd.DataFrame:
    return pdf.apply(lambda x: x + 1, axis=1)

res = sdf.groupby("id").applyInPandas(plus_one, schema=sdf.schema)
res.collect()
```

Observe que `plus_one` pega um DataFrame do Pandas e retorna outro DataFrame do Pandas. Para cada grupo, todas as colunas são passadas juntas como um DataFrame do Pandas para o `plus_one` UDF, e os DataFrames do Pandas retornados são combinados em um DataFrame do PySpark.

Executar o exemplo acima e `sc.show_profiles()` imprime o seguinte perfil. O perfil abaixo também pode ser inserido no disco por `sc.dump_profiles(path)`.

```
============================================================
Profile of UDF<id=271>
============================================================
         2898160 function calls (2881848 primitive calls) in 2.254 seconds

   Ordered by: internal time, cumulative time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
   ...
     8000    0.084    0.000    1.384    0.000 series.py:5516(_arith_method)
   ...
        8    0.000    0.000    2.254    0.282 <command-14168941>:1(plus_one)
   ...
```

O UDF id no perfil (271, destacado acima) corresponde ao do plano Spark para res. O plano Spark pode ser mostrado chamando `res.explain()`.

```
== Physical Plan ==
...
   FlatMapGroupsInPandas [id#238], plus_one(id#238, v#240)#271, [id#272, v#273]
...
```

A primeira linha do corpo do perfil indica o número total de chamadas monitoradas. O cabeçalho da coluna inclui

- `ncalls`, para o número de chamadas.
- `tottime`, para o tempo total gasto na função dada (excluindo o tempo gasto em chamadas para subfunções)
- `percall`, o quociente de tottime dividido por ncalls
- `cumtime`, o tempo cumulativo gasto nesta e em todas as subfunções (da invocação até a saída)
- `percall`, o quociente de cumtime dividido por chamadas primitivas
- `filename:lineno(function)`, que fornece as respectivas informações para cada função

Analisando os detalhes da coluna: `plus_one` é acionado uma vez por grupo, 8 vezes no total; `_arith_method` da série Pandas é chamado uma vez por linha, 8.000 vezes no total. `pandas.DataFrame.apply` aplica a função `lambda x: x + 1` linha por linha, sofrendo assim uma alta sobrecarga de invocação.

Podemos reduzir essa sobrecarga substituindo `pandas.DataFrame.apply` por pdf + 1, que é vetorizado no Pandas. O Pandas UDF otimizado fica assim:

```
import pandas as pd

def plus_one_optimized(pdf: pd.DataFrame) -> pd.DataFrame:
    return pdf + 1

res = sdf.groupby("id").applyInPandas(plus_one_optimized, schema=sdf.
schema)
res.collect()
```

O perfil atualizado é mostrado abaixo.

```
============================================================
Profile of UDF<id=258>
============================================================
         2384 function calls (2328 primitive calls) in 0.003 seconds

   Ordered by: internal time, cumulative time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
   ...
        8    0.000    0.000    0.003    0.000 frame.py:6857(_arith_method)
   ...
        8    0.000    0.000    0.003    0.000 <command-14168952>:1(plus_one_optimized)
   ...
```

Podemos resumir as otimizações da seguinte forma:

- Operação aritmética de 8.000 chamadas para 8 chamadas
- Total de chamadas de função de 2.898.160 para 2.384 chamadas
- Tempo total de execução de 2,300 segundos para 0,004 segundos

O breve exemplo acima demonstra como o criador de perfil UDF nos ajuda a compreender profundamente a execução, identificar o gargalo de desempenho e melhorar o desempenho geral da função definida pelo usuário.

O criador de perfil UDF foi implementado com base no criador de perfil do lado do executor, que é projetado para API PySpark RDD. O criador de perfil do lado do executor está disponível em todas as versões ativas do Databricks Runtime.

Tanto o criador de perfil UDF quanto o criador de perfil do lado do executor são executados nos workers Python. Eles são controlados pela configuração `spark.python.profile` do Spark, que é false por padrão. Podemos habilitar essa configuração do Spark em clusters do Databricks Runtime, conforme mostrado abaixo.

Conclusão

Os criadores de perfil PySpark são implementados com base em cProfile; portanto, o relatório do perfil depende da classe Stats. Os acumuladores Spark também desempenham um papel importante ao coletar relatórios de perfil dos workers Python.

Criadores de perfil poderosos são fornecidos pelo PySpark para identificar hot loops e sugerir possíveis melhorias. São fáceis de usar e essenciais para melhorar o desempenho dos programas PySpark. O criador de perfil UDF, que está disponível a partir do Databricks Runtime 11.0 (Spark 3.3), supera todos os desafios técnicos e traz insights para funções definidas pelo usuário.

Além disso, há um esforço contínuo na comunidade de código aberto Apache Spark™ para introduzir perfis de memória em executores. Consulte SPARK-40281 para obter mais informações.

Comece a experimentar com estes notebooks gratuitos da Databricks.

SEÇÃO 2.3

Pipelines de dados de streaming de baixa latência com Delta Live Tables e Apache Kafka

de **FRANK MUNZ**

9 de agosto de 2022

Delta Live Tables (DLT) é o primeiro framework ETL que usa uma abordagem declarativa simples para criar pipeline de dados confiável e gerencia totalmente a infraestrutura subjacente em escala para batch e streaming de dados. Muitos casos de uso exigem insights acionáveis derivados de dados quase em tempo real. O Delta Live Tables permite streaming de dados de pipeline de baixa latência para oferecer suporte a esses casos de uso com baixas latências, ingerindo dados diretamente de barramentos de eventos como Apache Kafka, AWS Kinesis, Confluent Cloud, Amazon MSK ou Azure Event Hubs.

Este artigo explicará como usar DLT com Apache Kafka e fornecerá o código Python necessário para ingerir a streams. A arquitetura de sistema recomendada será explicada e as configurações DLT relacionadas que valem a pena considerar serão exploradas ao longo do caminho.

Plataformas de streaming

Barramentos de eventos ou barramentos de mensagens separam os produtores de mensagens dos consumidores. Um caso de uso de streaming popular é a coleta de dados de cliques de usuários que navegam em um site em que cada interação do usuário é armazenada como um evento no Apache Kafka. O stream de eventos do Kafka é então usado para análise de dados de streaming em tempo real. Vários consumidores de mensagens podem ler os mesmos dados do Kafka e usar os dados para aprender sobre os interesses do público, taxas de conversão e motivos de rejeição. Os dados de eventos de streaming em tempo real das interações do usuário muitas vezes também precisam ser correlacionados com compras reais armazenadas em um banco de dados de cobrança.

Apache Kafka

O Apache Kafka é um barramento popular de eventos de código aberto. O Kafka usa o conceito de um tópico, um log distribuído de eventos somente para aplicativos onde as mensagens são armazenadas em buffer por um determinado período de tempo. Embora as mensagens no Kafka não sejam excluídas após serem consumidas, elas também não são armazenadas indefinidamente. A retenção de mensagens para o Kafka pode ser configurada por tópico e o padrão é de sete dias. As mensagens expiradas serão excluídas eventualmente.

Este artigo é centrado no Apache Kafka. No entanto, os conceitos discutidos também se aplicam a muitos outros barramentos de eventos ou sistemas de mensagens.

Pipelines de dados de streaming

Em um pipeline de fluxo de dados, o Delta Live Tables e suas dependências podem ser declaradas com um comando SQL Create Table As Select (CTAS) padrão e a palavra-chave DLT "live".

Ao desenvolver DLT com Python, o decorador @dlt.table é usado para criar uma Delta Live Table. Para garantir a qualidade dos dados em um pipeline, o DLT usa Expectations, que são simples cláusulas de restrição SQL que definem o comportamento do pipeline com registros inválidos.

Como as cargas de trabalho de transmissão geralmente vêm com volumes de dados imprevisíveis, o Databricks emprega dimensionamento automático aprimorado para pipelines de fluxo de dados para minimizar a latência geral de ponta a ponta e, ao mesmo tempo, reduzir custos ao desligar infraestruturas desnecessárias.

As **Delta Live Tables** são totalmente recalculadas, na ordem correta, exatamente uma vez para cada execução do pipeline.

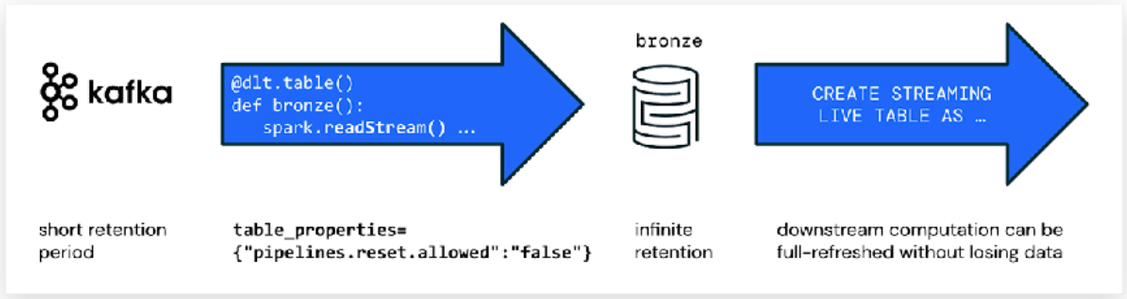
Por outro lado, as **Delta Live Tables de streaming** são estáveis, incrementalmente computadas e processam apenas dados que foram adicionados desde a última execução do pipeline. Se a query que define alterações nas tabelas de streaming ao vivo muda, novos dados serão processados com base na nova query, mas os dados existentes não são recomputados. As Live Tables de streaming sempre usam uma fonte de streaming e só funcionam em streams somente de aplicativos, como Kafka, Kinesis ou Auto Loader. As DLTs de streaming são baseadas na parte superior do Structured Streaming do Spark.

Você pode encadear vários pipelines de streaming, por exemplo, cargas de trabalho com volume de dados muito grande e requisitos de baixa latência.

Ingestão direta de engines de streaming

Delta Live Tables escritas em Python podem ingerir dados diretamente de um barramento de eventos como o Kafka usando Structured Streaming do Spark. Você pode definir um curto período de retenção para o tópico Kafka para evitar problemas de conformidade, reduzir custos e então se beneficiar do armazenamento barato, elástico e governável que o Delta oferece.

Como primeiro passo no pipeline, recomendamos a ingestão dos dados como estão em uma tabela Bronze (bruto) e evitar transformações complexas que possam descartar dados importantes. Como qualquer tabela Delta, a tabela Bronze manterá o histórico e permitirá que ele execute o GDPR e outras tarefas de conformidade.



Ingerir dados de streaming do Apache Kafka

Ao escrever pipelines DLT em Python, você usa a anotação `@dlt.table` para criar uma tabela DLT. Não existe nenhum atributo especial para marcar DLTs de streaming em Python. Basta usar `spark.readStream()` para acessar o stream. O código de exemplo para criar uma tabela DLT com o nome `kafka_bronze` que está consumindo dados de um tópico Kafka é o seguinte:

```
import dlt
from pyspark.sql.functions import *
from pyspark.sql.types import *

TOPIC = "tracker-events"
KAFKA_BROKER = spark.conf.get("KAFKA_SERVER")
# subscribe to TOPIC at KAFKA_BROKER
raw_kafka_events = (spark.readStream
    .format("kafka")
    .option("subscribe", TOPIC)
    .option("kafka.bootstrap.servers", KAFKA_BROKER)
    .option("startingOffsets", "earliest")
    .load()
    )

@dlt.table(table_properties={"pipelines.reset.allowed":"false"})
def kafka_bronze():
  return raw_kafka_events
```

`pipelines.reset.allowed`

Observe que os barramentos de eventos normalmente expiram mensagens após um determinado período de tempo, enquanto o Delta é projetado para retenção infinita.

Isso pode fazer com que os dados de origem no Kafka já tenham sido excluídos ao executar um refresh completo para um pipeline DLT. Nesse caso, nem todos os dados históricos poderiam ser preenchidos com a plataforma de mensagens, e os dados estariam faltando nas tabelas DLT. Para evitar a perda do uso de dados, use a seguinte propriedade da tabela DLT:

`pipelines.reset.allowed=false`

Configurar pipelines.reset.allowed como falso evita refreshes para a tabela, mas não impede que gravações incrementais para as tabelas ou novos dados fluam para a tabela.

Checkpointing

Se você é desenvolvedor experiente do Structured Streaming do Spark, notará a ausência de checkpointing no código acima. No Structured Streaming do Spark, o checkpointing é necessário para persistir informações de progresso sobre quais dados foram processados com sucesso e, em caso de falha, esses metadados são usados para reiniciar uma query com falha exatamente de onde parou.

Considerando que os checkpoints são necessários para a recuperação de falhas com garantias exatamente uma vez no Structured Streaming do Spark, o DLT lida com o estado automaticamente, sem qualquer configuração manual ou necessidade de checkpoint explícito.

Misturando SQL e Python para um pipeline de DLT

Um pipeline DLT pode consistir em vários notebooks, mas um notebook DLT deve ser escrito inteiramente em SQL ou Python (ao contrário de outros notebooks do Databricks em que você pode ter células de diferentes linguagens em um único notebook).

Agora, se sua preferência for SQL, você pode programar a ingestão de dados do Apache Kafka em um notebook em Python e depois implementar a lógica de transformações do seu pipeline de dados em outro notebook em SQL.

Mapeamento de esquemas

Ao ler dados da plataforma de mensagens, o stream dos dados é opaco, e um esquema deve ser fornecido.

O exemplo em Python abaixo mostra a definição do esquema de eventos de um rastreador de condicionamento físico e como a parte do valor da mensagem do Kafka é mapeada para esse esquema.

```
event_schema = StructType([ \
    StructField("time", TimestampType(),True)      , \
    StructField("version", StringType(),True), \
    StructField("model", StringType(),True)     , \
    StructField("heart_bpm", IntegerType(),True), \
    StructField("kcal", IntegerType(),True)       \
  ])
```

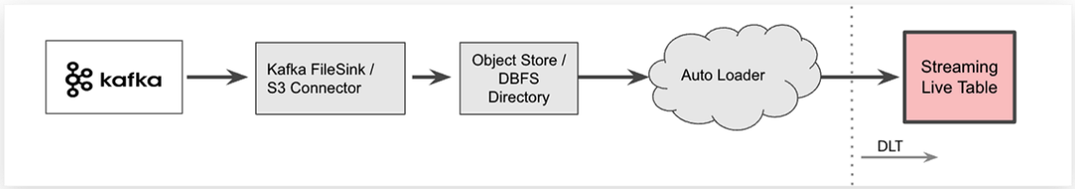
```
# temporary table, visible in pipeline but not in data browser,
# cannot be queried interactively
@dlt.table(comment="real schema for Kakfa payload",
           temporary=True)


def kafka_silver():
  return (
    # kafka streams are (timestamp,value)
    # value contains the kafka payload

    dlt.read_stream("kafka_bronze")
    .select(col("timestamp"),from_json(col("value")
    .cast("string"), event_schema).alias("event"))
    .select("timestamp", "event.*")
  )
```

Benefícios

A leitura de dados de streaming em DLT diretamente de um broker de mensagens minimiza a complexidade arquitetônica e fornece menor latência ponta a ponta, uma vez que os dados são transmitidos diretamente do broker de mensagens e nenhum intermediário ou passo está envolvido.

Ingestão de streaming com intermediário de armazenamento de objetos em nuvem

Para alguns casos de uso específicos, você pode querer descarregar dados do Apache Kafka, por exemplo, usando um conector Kafka, e armazenar seus dados de streaming em um objeto de nuvem intermediário. Em um workspace do Databricks, o armazenamento de objetos específico do fornecedor de nuvens pode então ser mapeado através do Sistema de Arquivos Databricks (DBFS) como uma pasta independente de nuvens. Depois que os dados são descarregados, o Databricks Auto Loader pode ingerir os arquivos.



O Auto Loader pode ingerir dados com uma única linha de código SQL. A sintaxe para ingerir arquivos JSON em uma tabela DLT é mostrada abaixo (está agrupada em duas linhas para facilitar a leitura).

```
-- INGEST with Auto Loader
create or replace streaming live table raw
as select * FROM cloud_files("dbfs:/data/twitter", "json")
```

Observe que o próprio Auto Loader é uma fonte de dados de streaming e todos os arquivos recém-chegados serão processados exatamente uma vez, daí a palavra-chave streaming para a tabela bruta que indica que os dados são ingeridos de forma incremental nessa tabela.

Como o descarregamento de dados de streaming para um armazenamento de objetos em nuvem introduz um passo adicional na arquitetura do sistema, também aumentará a latência ponta a ponta e criará custos adicionais de armazenamento. Lembre-se de que o conector Kafka que grava os dados do evento no armazenamento de objetos na nuvem precisa ser gerenciado, aumentando a complexidade operacional.

Portanto, a Databricks recomenda como prática acessar diretamente os dados de barramento de eventos do DLT usando o Structured Streaming do Spark, conforme descrito acima.

Outros barramentos de eventos ou sistemas de mensagens

Este artigo é centrado no Apache Kafka. No entanto, os conceitos discutidos também se aplicam a outros barramentos de eventos ou sistemas de mensagens. O DLT é compatível com qualquer fonte de dados que o Databricks Runtime suporta diretamente.

Amazon Kinesis

No Kinesis, você escreve mensagens em um stream serverless totalmente gerenciado. Igual ao Kafka, o Kinesis não armazena mensagens permanentemente. A retenção de mensagens padrão no Kinesis é de um dia.

Ao usar o Amazon Kinesis, substitua format("kafka") por format("kinesis") no código Python para ingestão de streaming acima e adicione configurações específicas do Amazon Kinesis com option(). Para obter mais informações, consulte a seção sobre a integração do Kinesis na documentação Structured Streaming do Spark.

Azure Event Hubs

Para as configurações do Azure Event Hubs, confira a documentação oficial na Microsoft e o artigo Delta Live Tables recipes: Consuming from Azure Event Hubs.

Resumo

O DLT é muito mais do que apenas o "T" no ETL. Com ele, você pode facilmente ingerir de fontes de streaming e batch, limpar e transformar dados na Plataforma Databricks Lakehouse em qualquer nuvem com qualidade de dados garantida.

Os dados do Apache Kafka podem ser ingeridos conectando-se diretamente a um broker Kafka a partir de um notebook DLT em Python. A perda de dados pode ser evitada para um refresh completo do pipeline, mesmo quando os dados de origem na camada de streaming do Kafka expiraram.

Comece agora

Se você é um cliente do Databricks, basta seguir o guia para começar. Leia as notas de versão para saber mais sobre o que está incluído nesta versão GA. Se você ainda não é um cliente do Databricks, inscreva-se para uma versão de avaliação gratuita e veja nossos preços detalhados do DLT aqui.

Participe da conversa na Comunidade Databricks, onde colegas obcecados por dados estão conversando sobre anúncios e atualizações do Data + AI Summit 2022. Aprenda. Faça contatos.

Por último, mas não menos importante, aproveite a sessão Mergulhe na engenharia de dados do Summit. Nessa sessão, eu apresento o código de outro exemplo de dados de streaming com uma transmissão ao vivo do Twitter, Auto Loader, Delta Live Tables em SQL e análise de sentimento Hugging Face.

SEÇÃO 2.4

Streaming em produção: práticas recomendadas coletadas

de **ANGELA CHU** e **TRISTEN WENTLING**

12 de dezembro de 2022

Lançar qualquer pipeline de dados ou aplicativo em um estado de produção exige planejamento, teste, monitoramento e manutenção. Os pipelines de streaming não são diferentes nesse sentido. Neste artigo, apresentamos algumas das considerações mais importantes para a implantação de pipelines e aplicações de streaming em um ambiente de produção.

Na Databricks, oferecemos duas maneiras diferentes de criar e executar pipelines e aplicações de streaming: Delta Live Tables (DLT) e Databricks Workflows. O DLT é nosso principal produto ETL totalmente gerenciado compatível com pipelines em batch e streaming. Oferece desenvolvimento declarativo, operações automatizadas, qualidade de dados, recursos avançados de observabilidade e muito mais. O Workflows permite que os clientes executem cargas de trabalho Apache Spark™ no ambiente de runtime otimizado do Databricks (ou seja, Photon) com acesso à governança unificada (Unity Catalog) e armazenamento (Delta Lake). Em relação às cargas de trabalho de streaming, tanto o DLT quanto o Workflows compartilham o mesmo mecanismo central de streaming: Structured Streaming do Spark. No caso do DLT, os clientes programam contra a API DLT e o DLT usa o mecanismo de Structured Streaming nos bastidores. No caso de Jobs, os clientes programam diretamente na API Spark.

As recomendações neste post são escritas a partir da perspectiva do mecanismo de Structured Streaming, a maioria dos quais se aplica tanto a DLT quanto a Workflows (embora o DLT cuide de alguns desses automaticamente, como Triggers e Checkpoints). Agrupamos as recomendações sob os títulos “Antes da implantação” e “Depois da implantação” para destacar quando esses conceitos precisarão ser aplicados, e estamos lançando esta série de posts com essa divisão entre os dois. Também haverá conteúdo adicional aprofundado para algumas das seções seguintes. Recomendamos a leitura de todas as seções antes de iniciar o trabalho para produzir um pipeline ou aplicação de streaming e revisitar essas recomendações à medida que você o promove do desenvolvimento ao controle de qualidade e, eventualmente, à produção.

Antes da implantação

Há muitas coisas que você precisa considerar ao criar seu aplicativo de streaming para melhorar a experiência de produção. Alguns desses tópicos, como teste de unidade, checkpoints, triggers e gerenciamento de estado, determinarão o desempenho do seu aplicativo de streaming. Outros, como convenções de nomenclatura e quantos streams executar em quais clusters, têm mais a ver com o gerenciamento de vários aplicativos de streaming no mesmo ambiente.

Teste unitário

O custo associado à localização e correção de um bug aumenta exponencialmente quanto mais longe você chega no processo de SDLC, e um aplicativo de Structured Streaming não é diferente. Quando você está transformando esse protótipo em um pipeline de produção reforçado, você precisa de um processo de CI/CD com testes integrados. Então, como você cria esses testes?

No início, você pode pensar que o teste de unidade de um pipeline de streaming requer algo especial, mas esse não é o caso. A orientação geral para pipelines de streaming não é diferente da orientação que você pode ter ouvido falar sobre jobs em batch do Spark. Começa organizando seu código para que ele possa ser testado de forma eficaz:

- Divida seu código em partes testáveis.
- Organize sua lógica de negócios em funções que chamam outras funções. Se você tem muita lógica em um foreachBatch ou implementou mapGroupsWithState ou flatMapGroupsWithState, organize esse código em várias funções que podem ser testadas individualmente.
- Não programe em dependências do estado global ou de sistemas externos.
- Qualquer função que manipule um DataFrame ou conjunto de dados deve ser organizada para assumir o DataFrame/conjunto de dados/configuração como entrada e saída do DataFrame/conjunto de dados.

Depois que seu código é separado de maneira lógica, você pode implementar testes de unidade para cada uma de suas funções. As funções independentes do Spark podem ser testadas como qualquer outra função nessa linguagem. Para testar UDFs e funções com DataFrames e conjuntos de dados, há vários frameworks de teste Spark disponíveis. Esses frameworks oferecem suporte a todas as APIs de DataFrame/conjunto de dados para que você possa criar entradas facilmente, e elas têm asserções especializadas que permitem comparar o conteúdo e os esquemas do DataFrame. Alguns exemplos são:

- O conjunto de testes Spark integrado, concebido para testar todas as partes do Spark
- spark-testing-base, que é compatível com Scala e Python
- spark-fast-tests, para testar Scala Spark 2 e 3
- chispa, uma versão Python de spark-fast-tests

Exemplos de código para cada uma dessas bibliotecas podem ser encontrados aqui.

Mas espere! Estou testando um aplicativo de streaming aqui. Não preciso fazer DataFrames de streaming para meus testes de unidade? A resposta é não. Embora um DataFrame de streaming represente um conjunto de dados sem fim definido, quando funções são executadas nele, elas são executadas em um microbatch, um conjunto discreto de dados. Você pode usar os mesmos testes de unidade que usaria para um aplicativo em batch, para streams sem estado e com estado. Uma das vantagens do Structured Streaming em relação a outros frameworks é a capacidade de usar o mesmo código de transformação para streaming e com outras operações em batch para o mesmo destino, ou "sink". Isso permite que você simplifique algumas operações, como backfilling de dados, por exemplo, em vez de tentar sincronizar a lógica entre duas aplicações diferentes, você pode simplesmente modificar as fontes de entrada e escrever no mesmo destino. Se o sink for uma tabela Delta, você poderá até mesmo fazer essas operações simultaneamente se ambos os processos forem operações somente de acréscimo.

Triggers

Agora que você sabe que seu código funciona, precisa determinar com que frequência seu stream vai procurar novos dados. É aqui que os triggers entram. Definir um trigger é uma das opções para o comando writeStream, e tem a seguinte aparência:

```
// Scala/Java
.trigger(Trigger.ProcessingTime("30 seconds"))

# Python
.trigger(processingTime='30 seconds')
```

No exemplo acima, se um microbatch for concluído em menos de 30 segundos, o mecanismo aguardará o restante do tempo antes de iniciar o próximo microbatch. Se um microbatch demorar mais de 30 segundos para ser concluído, o mecanismo iniciará o próximo microbatch imediatamente após o término do anterior.

Os dois fatores que você deve considerar ao configurar seu intervalo de trigger são quanto tempo você espera que seu stream processe um microbatch e com que frequência você quer que o sistema verifique novos dados. Você pode reduzir a latência de processamento geral usando um intervalo de trigger menor e aumentando os recursos disponíveis para a query de streaming adicionando mais workers ou usando instâncias otimizadas de compute ou memória adaptadas ao desempenho da sua aplicação. Esses recursos maiores vêm com custos maiores, portanto, se seu objetivo é minimizar os custos, um intervalo de trigger mais longo com menos compute pode funcionar. Normalmente, você não definiria um intervalo de trigger maior do que o que normalmente levaria para seu stream processar um microbatch para maximizar o uso dos recursos, mas definir um intervalo maior faria sentido se seu stream estiver sendo executado em um cluster compartilhado e você não quiser que ele use constantemente os recursos do cluster.

Se não for necessário que o stream seja executado continuamente, seja porque os dados não chegam com frequência ou porque o SLA é de 10 minutos ou mais, você poderá usar a opção Trigger.Once. Ela iniciará o start, verificará se há algo novo desde a última vez que foi executado, processará tudo em um batch grande e, em seguida, encerrará. Assim como em um stream em execução contínua ao usar o Trigger.Once, o checkpoint que garante a tolerância a falhas (veja abaixo) garantirá o processamento exatamente uma vez.

O Spark tem uma nova versão do Trigger.Once chamada Trigger.AvailableNow. Enquanto o Trigger.Once processará tudo em um grande batch, o que, dependendo do tamanho dos dados, pode não ser o ideal, o Trigger.AvailableNow dividirá os dados com base nas configurações maxFilesPerTrigger e maxBytesPerTrigger. Isso permite que os dados sejam processados em vários batches. Essas configurações são ignoradas com Trigger.Once. Você pode ver exemplos de configuração de triggers aqui.

Teste rápido: como transformar seu processo de streaming em um processo em batch que mantém automaticamente o controle de onde parou com apenas uma linha de código?

Resposta: altere seu trigger de tempo de processamento para Trigger.Once/Trigger.AvailableNow. Exatamente o mesmo código, executado em uma programação, que não perderá nem reprocessará nenhum registro.

Dê um nome ao seu stream

Você nomeia seus filhos, seus pets, agora é hora de nomear seus streams. Há uma opção do writeStream chamada .queryName que permite dar um nome legal ao seu stream. Por quê? Bem, suponha que você não dê nenhum nome. Nesse caso, você só vai precisar ir até a tab Structured Streaming na interface do usuário Spark e usar a string <no name> e a GUID incompreensível que é gerada automaticamente como identificador exclusivo do stream. Se você tem mais de um stream em execução em um cluster e todos eles têm <no name> e strings ininteligíveis como identificadores, como você encontra cada um deles? Se você exportar as métricas, como saber qual é qual?

Facilite as coisas para você e dê nomes aos seus streams. Quando estiver gerenciando-os na produção, ficará feliz por ter nomeado e, enquanto isso, nomeie suas queries em batch em qualquer código foreachBatch() que tiver.

Tolerância a falhas

Como seu stream se recupera do desligamento? Existem alguns casos diferentes em que isso pode entrar em jogo, como falhas no nó de cluster ou paralisações intencionais, mas a solução é configurar o checkpointing. Os checkpoints com logs de gravação antecipada fornecem um grau de proteção contra a interrupção do aplicativo de streaming, garantindo que ele possa retomar de onde parou pela última vez.

Os checkpoints armazenam os deslocamentos atuais e os valores de estado (por exemplo, valores agregados) para seu stream. Os checkpoints são específicos do stream. Portanto, cada um deve ser definido em seu próprio local. Isso permitirá que você se recupere com mais facilidade de desligamentos, falhas no código do aplicativo ou falhas ou limitações inesperadas do provedor de nuvem.

Para configurar checkpoints, adicione a opção checkpointLocation à sua definição de stream:

```
// Scala/Java/Python
streamingDataFrame.writeStream
  .format("delta")
  .option("path", "")
  .queryName("TestStream")
  .option("checkpointLocation", "")
  .start()
```

Para simplificar, sempre que você chamar .writeStream, deve especificar a opção de checkpoint com um local de checkpoint único. Mesmo que você esteja usando foreachBatch e o writeStream em si não especifique uma opção de caminho ou tabela, você ainda deve especificar esse checkpoint. É assim que o Structured Streaming do Spark proporciona tolerância a falhas sem complicações.

Os esforços para gerenciar os checkpoints em seu stream devem ser de pouca preocupação em geral. Como disse Tathagata Das: “A maneira mais simples de realizar o streaming analítico é não ter que raciocinar sobre o streaming.” Dito isso, uma configuração merece menção, pois ocasionalmente surgem dúvidas sobre a manutenção de arquivos de checkpoint. Embora seja uma configuração interna que não requer configuração direta, a configuração spark.sql.streaming.minBatchesToRetain (default 100) controla o número de arquivos de checkpoint que são criados. Basicamente, o número de arquivos será aproximadamente esse número vezes dois, pois há um arquivo criado anotando os deslocamentos no início dos batches (offsets, também conhecidos como write ahead logs) e outro na conclusão dos batches (commits). O número de arquivos é verificado periodicamente para limpeza como parte dos processos internos. Isso simplifica pelo menos um aspecto da manutenção do aplicativo de streaming de longo prazo para você.

Também é importante observar que algumas alterações no código do seu aplicativo podem invalidar o checkpoint. É recomendável verificar qualquer uma dessas alterações durante as revisões de código antes da implantação. Você pode encontrar exemplos de mudanças em que isso pode acontecer em Recovery Semantics after Changes in a Streaming Query. Vamos supor que você queira analisar o checkpoint com mais detalhes ou considerar se o checkpoint assíncrono pode melhorar a latência em seu aplicativo de streaming. Nesse caso, eles são abordados com maior profundidade no artigo Speed Up Streaming Queries With Asynchronous State Checkpointing.

Gerenciamento de estado e RocksDB

Aplicações de streaming com estado são aquelas em que os registros atuais podem depender de eventos anteriores, então o Spark precisa reter dados entre os microbatches. Os dados retidos são chamados de estado, e o Spark os armazenará em um armazenamento de estado e os lerá, atualizará e excluirá durante cada microbatch. As operações típicas com estado são agregações de streaming, streaming de dropDuplicates, joins stream-stream, mapGroupsWithState ou flatMapGroupsWithState. Alguns tipos comuns de exemplos em que você precisará pensar sobre o estado da sua aplicação podem ser a sessão ou a agregação por hora usando métodos de agrupamento para calcular métricas de negócios. Cada registro no armazenamento de estado é identificado por uma chave que é usada como parte do cálculo de estado, e as chaves mais exclusivas que são exigidas quanto maior a quantidade de dados de estado que serão armazenados.

Quando a quantidade de dados de estado necessários para habilitar essas operações com monitoração de estado se torna grande e complexa, isso pode degradar o desempenho de suas cargas de trabalho, levando a um aumento da latência ou até mesmo a falhas. Um indicador típico de que o armazenamento de estado é o culpado pela latência adicional é a grande quantidade de tempo gasto em pausas de coleta de lixo (GC) na JVM. Se você está monitorando o tempo de processamento do microbatch, isso pode parecer um aumento contínuo ou um tempo de processamento extremamente variável entre os microbatches.

A configuração padrão para um armazenamento de estado, que é suficiente para a maioria das cargas de trabalho de streaming gerais, é armazenar os dados de estado na memória JVM dos executores. Um grande número de chaves (normalmente milhões, consulte a seção Monitoramento e Instrumentação na parte 2 deste artigo) pode adicionar pressão de memória excessiva na memória da máquina e aumentar a frequência de atingir essas pausas de GC enquanto tenta liberar recursos.

No Databricks Runtime (agora também compatível com o Apache Spark 3.2+), você pode usar o RocksDB como um provedor de armazenamento de estado alternativo para aliviar essa fonte de pressão de memória. O RocksDB é um armazenamento de chave persistente incorporável para armazenamento rápido. Ele apresenta alto desempenho por meio de um mecanismo de banco de dados estruturado por log escrito inteiramente em C++ e otimizado para armazenamento rápido e de baixa latência.

Aproveitar o RocksDB como provedor de armazenamento de estado ainda usa memória de máquina, mas não ocupa mais espaço no JVM e cria um sistema de gerenciamento de estado mais eficiente para grandes quantidades de chaves. No entanto, isso não vem de graça, pois introduz uma etapa extra no processamento de cada microbatch. Não se deve esperar que a introdução do RocksDB reduza a latência, exceto quando ela estiver relacionada à pressão de memória do armazenamento de dados de estado na JVM. O armazenamento de estado apoiado no RocksDB ainda oferece o mesmo grau de tolerância a falhas que o armazenamento de estado regular, pois está incluído no checkpoint de stream.

A configuração do RocksDB, como a configuração do checkpoint, é mínima por design e, portanto, você só precisa declará-la em sua configuração geral do Spark:

```
spark.conf.set(
  "spark.sql.streaming.stateStore.providerClass",
  "com.databricks.sql.streaming.state.RocksDBStateStoreProvider")
```

Se você estiver monitorando seu stream usando a classe streamingQueryListener, você também notará que as métricas do RocksDB serão incluídas no campo stateOperators. Para informações mais detalhadas sobre isso, consulte a seção RocksDB State Store Metrics de "Structured Streaming in Production".

Vale a pena notar que um grande número de chaves pode ter outros impactos adversos, além de aumentar o consumo de memória, especialmente com chaves de estado ilimitadas ou sem expiração. Com ou sem RocksDB, o estado do aplicativo também é copiado em checkpoints para tolerância a falhas. Portanto, faz sentido que, se você tiver arquivos de estado sendo criados para que não expirem, você continuará acumulando arquivos no checkpoint, aumentando a quantidade de armazenamento necessária e, potencialmente, o tempo para gravá-los ou também para se recuperar de falhas. Para os dados na memória (veja a seção Monitoramento e Instrumentação na parte 2 deste artigo), esta situação pode levar a erros um tanto vagos de falta de memória, e, para os dados verificados gravados no armazenamento em nuvens você pode observar um crescimento inesperado e irracional. A menos que você tenha uma necessidade comercial de reter o estado de streaming para todos os dados que foram processados (e isso é raro), leia a documentação de Structured Streaming do Spark e implemente suas operações com estado para que o sistema possa descartar registros de estado que não são mais necessários (preste muita atenção em dropDuplicates e joins stream-stream).

Executando vários streams em um cluster

Assim que seus streams forem totalmente testados e configurados, é hora de descobrir como organizá-los em produção. É um padrão comum empilhar vários streams no mesmo cluster Spark para maximizar o uso de recursos e economizar custos. Isso é bom até certo ponto, mas há limites para a quantidade que pode ser adicionada a um cluster antes que o desempenho seja afetado. O driver precisa gerenciar todos os streams em execução no cluster, e todos os streams competirão pelos mesmos núcleos entre os workers. Você precisa entender o que seus streams estão fazendo e planejar sua capacidade adequadamente para empilhar de forma eficaz.

Aqui está o que você deve levar em consideração ao planejar empilhar vários streams no mesmo cluster:

- Seu driver deve ser grande o suficiente para gerenciar todos os seus streams. O driver está tendo dificuldade com a alta utilização da CPU e coleta de lixo? Isso significa que ele está tentando gerenciar todos os seus streams. Reduza o número de streams ou aumente o tamanho do seu driver.
- Considere a quantidade de dados que cada stream está processando. Quanto mais dados você ingere e grava em um sink, mais núcleos serão necessários para maximizar o throughput de cada stream. Você precisará reduzir o número de streams ou aumentar o número de workers dependendo da quantidade de dados que estão sendo processados. Para fontes como Kafka, você precisará configurar quantos núcleos estão sendo usados para ingerir com a opção minPartitions se não tiver núcleos suficientes para todas as partições em todos os seus streams.

- Considere a complexidade e o volume de dados dos seus streams. Se todos os fluxos estiverem fazendo manipulação mínima e apenas anexando a um sink, então cada stream precisará de menos recursos por microbatch e você poderá empilhar mais. Se os streams estiverem realizando processamento com estado ou operações com uso intensivo de computação/memória, isso exigirá mais recursos para um bom desempenho e será melhor empilhar menos streams.
- Considere scheduler pools. Ao empilhar streams, todos eles contestarão os mesmos workers e núcleos, e um stream que precisa de muitos núcleos fará com que os outros streams esperem. Os scheduler pools permitem que você tenha streams diferentes executados em diferentes partes do cluster. Isso permitirá que os streams sejam executados em paralelo com um subconjunto dos recursos disponíveis.
- Considere seu SLA. Se você tiver streams de missão crítica, isole-os como prática recomendada para que streams de menor criticidade não os afetem.

No Databricks, normalmente vemos a pilha de clientes entre 10 e 30 streams em um cluster, mas isso varia dependendo do caso de uso. Considere os fatores acima para que você possa ter uma boa experiência com desempenho, custo e manutenção.

Conclusão

Algumas das ideias que abordamos aqui certamente merecem seu próprio tempo e tratamento especial com uma discussão mais aprofundada, que você poderá aguardar em aprofundamentos posteriores. No entanto, esperamos que essas recomendações sejam úteis quando você inicia sua jornada ou busca aprimorar sua experiência de streaming de produção. Continue com a próxima publicação: "Streaming em produção: práticas recomendadas coletadas (parte 2)".

Leia o Guia de primeiros passos para Structured Streaming do Databricks

Comece a experimentar com estes notebooks gratuitos da Databricks.

databricks

SEÇÃO 2.5

Streaming em produção: práticas recomendadas coletadas (parte 2)

de **ANGELA CHU** e **TRISTEN WENTLING**

10 de janeiro de 2023

Este é o segundo artigo da nossa série de posts em duas partes intitulada "Streaming em produção: práticas recomendadas coletadas". Aqui discutimos as considerações "Após a implantação" de um pipeline de Structured Streaming. A maioria das sugestões deste artigo é relevante para os jobs de Structured Streaming e para o Delta Live Tables (nosso principal produto de ETL totalmente gerenciado compatível com pipelines em batch e streaming).

Após a implantação

Após a implantação do seu aplicativo de streaming, normalmente há três coisas principais que você precisa saber:

- Como meu aplicativo está sendo executado?
- Os recursos estão sendo usados de forma eficiente?
- Como gerencio os problemas que surgirem?

Vamos começar com uma introdução a esses tópicos, seguida de um mergulho mais profundo nesta série.

Monitoramento e instrumentação (Como meu aplicativo está sendo executado?)

As cargas de trabalho de streaming devem ser praticamente automáticas depois de implantadas na produção. No entanto, uma coisa que às vezes pode vir à mente é: "como meu aplicativo está sendo executado?". Monitorar os aplicativos pode assumir diferentes níveis e formas, dependendo do seguinte:

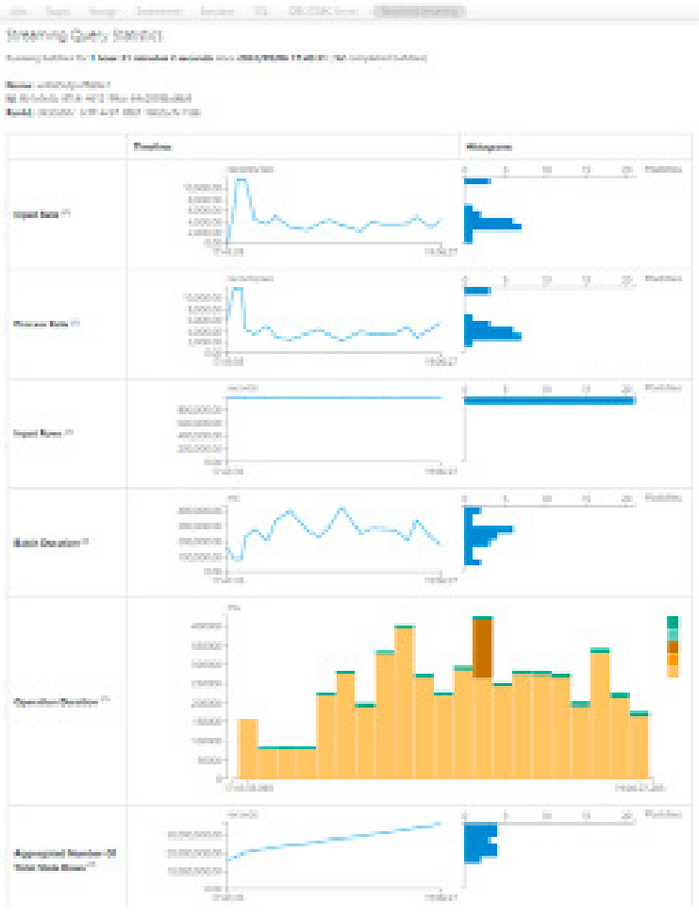
- As métricas coletadas para seu aplicativo (duração/latência do batch, throughput...)
- De onde você deseja monitorar o aplicativo

No nível mais simples, há um dashboard de streaming (A Look at the New Structured Streaming UI) e um registro integrado diretamente na interface do Spark que pode ser usado em diversas situações.

Isso se soma à configuração de alertas de falha em jobs que executam cargas de trabalho de streaming.

Se você quiser métricas mais refinadas ou criar ações personalizadas com base nessas métricas como parte da sua base de código, então `StreamingQueryListener` está mais alinhado com o que você está procurando.

Se você quiser que as métricas do Spark sejam relatadas (incluindo rastreamentos no nível da máquina para drivers ou workers), deverá usar o sink de métricas da plataforma.



UI de Structured Streaming do Apache Spark

Outro ponto a considerar é onde você deseja exibir essas métricas para observabilidade. Há um dashboard do Ganglia no nível do cluster, aplicativos de parceiros integrados, como o Datadog, para monitorar cargas de trabalho de streaming ou ainda mais opções de código aberto que você pode criar usando ferramentas como Prometheus e Grafana. Cada um tem vantagens e desvantagens a serem consideradas em relação aos requisitos de custo, desempenho e manutenção.

Se você tem volumes baixos de cargas de trabalho de streaming onde as interações na interface do usuário são suficientes ou se decidiu investir em uma plataforma de monitoramento mais robusta, deve saber como observar suas cargas de trabalho de streaming de produção. Outros artigos sobre "Monitoramento e alerta" mais adiante nesta série conterão uma discussão mais completa. Em particular, veremos diferentes medidas para monitorar aplicativos de streaming e, mais tarde, examinaremos mais profundamente algumas das ferramentas que você pode aproveitar para observabilidade.

Otimização de aplicativos (os recursos estão sendo usados de forma eficaz? Pense em "custo")

A próxima preocupação que temos após a implantação em produção é "meu aplicativo está usando recursos de forma eficaz?". Como desenvolvedores, entendemos (ou aprendemos rapidamente) a diferença entre código funcional e código bem escrito. Melhorar a maneira como seu código é executado geralmente é muito satisfatório, mas o que importa na verdade é o custo geral de executá-lo. As considerações de custo para aplicativos de Structured Streaming serão em grande parte semelhantes às de outros aplicativos Spark. Uma diferença notável é que não otimizar as cargas de trabalho de produção pode ser extremamente caro, já que essas cargas de trabalho são frequentemente aplicativos "sempre ativos" e, portanto, gastos desperdiçados podem aumentar rapidamente. Como a assistência

com otimização de custos é sempre requisitada, um post separado nesta série abordará isso. Os pontos principais em que nos concentraremos serão a eficiência de uso e dimensionamento.

Obter o dimensionamento do cluster certo é uma das diferenças mais significativas entre eficiência e desperdício em aplicativos de streaming. Isso pode ser particularmente complicado porque, em alguns casos, é difícil estimar as condições de carga total do aplicativo em produção antes que ele esteja de fato lá. Em outros casos, pode ser difícil devido a variações naturais no volume tratadas ao longo do dia, semana ou ano. Ao implantar pela primeira vez, pode ser benéfico superdimensionar ligeiramente, incorrendo em despesas extras para evitar a indução de gargalos de desempenho. Use as ferramentas de monitoramento que você escolheu implantar depois que o cluster estiver em execução por algumas semanas para garantir a utilização adequada do cluster. Por exemplo, a CPU e os níveis de memória estão sendo usados em um nível alto durante o pico de carga ou a carga geralmente é pequena e o cluster pode ser reduzido? Mantenha o monitoramento regular disso e fique atento às mudanças no volume de dados ao longo do tempo. Se isso ocorrer, um redimensionamento de cluster pode ser necessário para manter a operação econômica.

Como diretriz geral, você deve evitar operações de embaralhamento excessivas, joins ou um threshold de marca d'água excessivo ou extremo (não exceda suas necessidades), pois cada um pode aumentar o número de recursos necessários para executar seu aplicativo. Um threshold grande de marca d'água fará com que o Structured Streaming mantenha mais dados no armazenamento de estado entre os batches, levando a um aumento nos requisitos de memória em todo o cluster. Além disso, preste atenção ao tipo de VM configurada: você está usando memória otimizada para seu fluxo intenso de memória? Compute otimizado para seu stream computacionalmente intensivo? Caso contrário, observe os níveis de uso de cada um e considere tentar um tipo de máquina que possa ser mais adequado. Famílias mais recentes de servidores de provedores de nuvem com CPUs mais ideais geralmente levam a uma execução mais rápida, o que significa que você pode precisar de menos deles para cumprir seu SLA.

Solução de problemas (Como gerencio os problemas que surgirem?)

A última pergunta que nos fazemos após a implantação é “como faço para gerenciar os problemas que surgirem?”. Assim como ocorre com a otimização de custos, a solução de problemas de aplicativos de streaming no Spark costuma ser igual à de outros aplicativos, pois a maior parte da mecânica permanece a mesma nos bastidores. Para aplicativos de streaming, os problemas geralmente se enquadram em duas categorias: cenários de falha e cenários de latência.

Cenários de falha

Os cenários de falha geralmente se manifestam com o stream parando com um erro, os executores falhando ou uma falha de driver fazendo com que todo o cluster falhe. Causas comuns para isso são:

- Muitos streams em execução no mesmo cluster, sobrecarregando o driver. No Databricks, isso pode ser visto no Ganglia, em que o nó do driver aparecerá como sobrecarregado antes que o cluster falhe.
- Poucos workers em um cluster ou um tamanho de worker com uma proporção muito pequena de núcleo para memória, fazendo com que os executores falhem com um erro de falta de memória. Isso também pode ser visto no Databricks no Ganglia antes que um executor falhe, ou na interface do usuário do Spark na tab Executores.
- Usar uma coleta para enviar muitos dados ao driver, fazendo com que ele falhe com um erro de falta de memória.

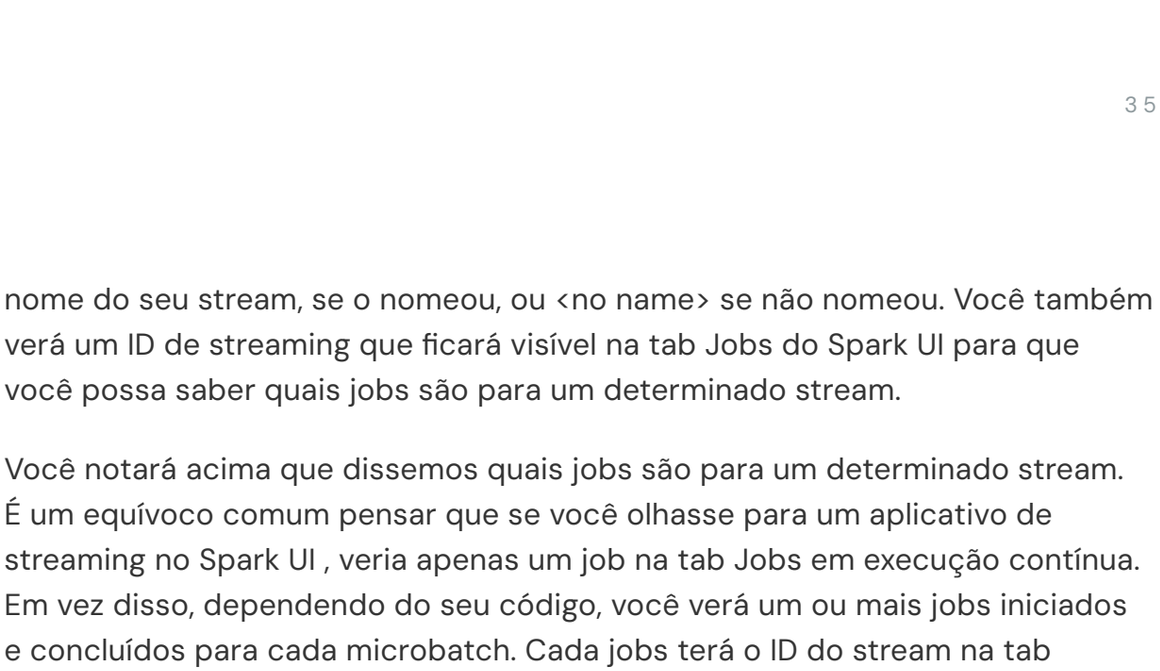
Cenários de latência

Para cenários de latência, seu stream não será executado tão rápido quanto você deseja ou espera. Um problema de latência pode ser intermitente ou constante. Muitos streams ou um cluster muito pequeno também podem ser a causa disso. Algumas outras causas comuns são:

- Distorção de dados: quando algumas tarefas acabam com muito mais dados do que o resto das tarefas. Com dados distorcidos, essas tarefas demoram mais para serem executadas do que as outras, geralmente vazando para o disco. Seu stream só pode ser executado tão rápido quanto a tarefa mais lenta.
- Executar uma query com estado sem definir uma marca d'água ou definir uma muito longa fará com que seu estado cresça muito, diminuindo a velocidade do stream ao longo do tempo e potencialmente levando a falhas.
- Sink mal otimizado. Por exemplo, executar um merge em uma tabela Delta superparticionada como parte do seu stream.
- Latência estável, mas alta (tempo de execução em batch). Dependendo da causa, adicionar mais workers para aumentar o número de núcleos atualmente disponíveis para tarefas Spark pode ajudar. Aumentar o número de partições de entrada e/ou diminuir a carga por núcleo através das configurações de tamanho do batch também pode reduzir a latência.

Assim como solucionar problemas de um job em batch, você usará o Ganglia para verificar o uso de clusters e o Spark UI para encontrar gargalos de desempenho. Há uma tab Structured Streaming específica no Spark UI criada para ajudar a monitorar e solucionar problemas de aplicativos de streaming. Nessa tab, cada stream que está sendo executado será listado, e você verá o nome do seu stream, se o nomeou, ou <no name> se não nomeou. Você também verá um ID de streaming que ficará visível na tab Jobs do Spark UI para que você possa saber quais jobs são para um determinado stream.

Você notará acima que dissemos quais jobs são para um determinado stream. É um equívoco comum pensar que se você olhasse para um aplicativo de streaming no Spark UI , veria apenas um job na tab Jobs em execução contínua. Em vez disso, dependendo do seu código, você verá um ou mais jobs iniciados e concluídos para cada microbatch. Cada jobs terá o ID do stream na tab Structured Streaming e um número de microbatch na descrição, para que você possa dizer quais jobs acompanham qual stream. Você pode clicar nesses jobs para encontrar os estágios e tarefas de execução mais longos, verificar os spills de disco e pesquisar por ID de job na tab SQL para encontrar as queries mais lentas e verificar seus planos de explicação.



Tab Jobs na UI do Apache Spark

Se você clicar em seu stream na tab Structured Streaming, verá quanto tempo as diferentes operações de streaming estão levando para cada microbatch, como adicionar um batch, planejar queries e dar commit (veja a captura de tela anterior da UI de Structured Streaming do Apache Spark). Você também pode ver quantas linhas estão sendo processadas, bem como o tamanho do seu armazenamento de estado para um stream com monitoramento de estado. Isso pode fornecer insights sobre onde estão os possíveis problemas de latência.

Vamos nos aprofundar na solução de problemas mais adiante nesta série de artigos, em que examinaremos algumas das causas e soluções para os cenários de falha e de latência, conforme descrito acima.

Conclusão

Você deve ter notado que muitos dos tópicos abordados aqui são muito semelhantes a como outros aplicativos Spark de produção devem ser implantados. Se suas cargas de trabalho são principalmente aplicativos de streaming ou processos em batch, a maioria dos mesmos princípios será aplicada. Nós nos concentramos mais em coisas que se tornam especialmente importantes ao criar aplicativos de streaming, mas, como temos certeza de que você já percebeu, os tópicos que discutimos devem ser incluídos na maioria das implantações de produção.

Na maioria dos setores do mundo hoje, as informações são necessárias mais rápido do que nunca, mas isso não será um problema para você. Com o Structured Streaming do Spark, você tem tudo pronto para fazer isso acontecer em escala na produção. Fique de olho nas discussões mais detalhadas sobre alguns dos tópicos que abordamos neste artigo e, enquanto isso, continue o streaming.

Leia a documentação sobre Structured Streaming em produção do Databricks

Comece a experimentar com estes notebooks gratuitos da Databricks.

SEÇÃO 2.6

Criação de produtos de dados geoespaciais

de **MILOS COLIC**

6 de janeiro de 2023

Os dados geoespaciais vêm impulsionando a inovação há séculos, por meio do uso de mapas, cartografia e, mais recentemente, do conteúdo digital. Por exemplo, o mapa mais antigo foi encontrado gravado em um pedaço de presa de mamute e data de aproximadamente 25.000 AEC. Isso torna os dados geoespaciais uma das fontes de dados mais antigas usadas pela sociedade para tomar decisões. Um exemplo mais recente, rotulado como o nascimento da análise espacial, é o de Charles Picquet em 1832, que usou dados geoespaciais para analisar surtos de cólera em Paris. Algumas décadas depois, John Snow, em 1854, seguiu a mesma abordagem para surtos de cólera em Londres. Esses dois indivíduos usaram dados geoespaciais para resolver um dos problemas mais difíceis de sua época e, na verdade, salvar inúmeras vidas. Avançando rapidamente para o século XX, o conceito de Sistemas de Informação Geográfica (SIG) foi introduzido pela primeira vez em 1967 em Ottawa, no Canadá, pelo Departamento de Silvicultura e Desenvolvimento Rural.

Hoje estamos no meio da revolução do setor de computação em nuvem: escala de supercomputação disponível para qualquer organização, quase que infinitamente escalável tanto para armazenamento quanto compute. Conceitos como malha de dados e marketplace de dados estão surgindo na comunidade de dados para abordar questões como federação de plataformas e interoperabilidade. Como podemos adotar esses conceitos para dados geoespaciais, análise espacial e SIGs? Adotando o conceito de produtos de dados e abordando o design de dados geoespaciais como um produto.

Neste artigo, apresentaremos um ponto de vista sobre como projetar produtos de dados geoespaciais dimensionáveis que sejam modernos e robustos. Discutiremos como a Plataforma Databricks Lakehouse pode ser usada para liberar todo o potencial dos produtos geoespaciais, que são um dos ativos mais valiosos na solução dos problemas mais difíceis de hoje e do futuro.

O que é um produto de dados? E como criar um?

A definição mais ampla e concisa de "produto de dados" foi cunhada por DJ Patil (o primeiro cientista-chefe de dados dos EUA) em *Data Jujitsu: The Art of Turning Data into Product*: "um produto que facilita um objetivo final por meio do uso de dados". A complexidade dessa definição (como admitido pelo próprio Patil) é necessária para encapsular a amplitude de produtos possíveis, incluindo dashboards, relatórios, planilhas do Excel e até extratos CSV compartilhados por e-mail. Você pode notar que os exemplos fornecidos se deterioram rapidamente em qualidade, robustez e governança.

Quais são os conceitos que diferenciam um produto de sucesso de um produto malsucedido? É a embalagem? O conteúdo? A qualidade do conteúdo? Ou é apenas a adoção do produto no mercado? A Forbes define os 10 itens essenciais para um produto de sucesso. Uma boa estrutura para resumir isso é através da pirâmide de valor.

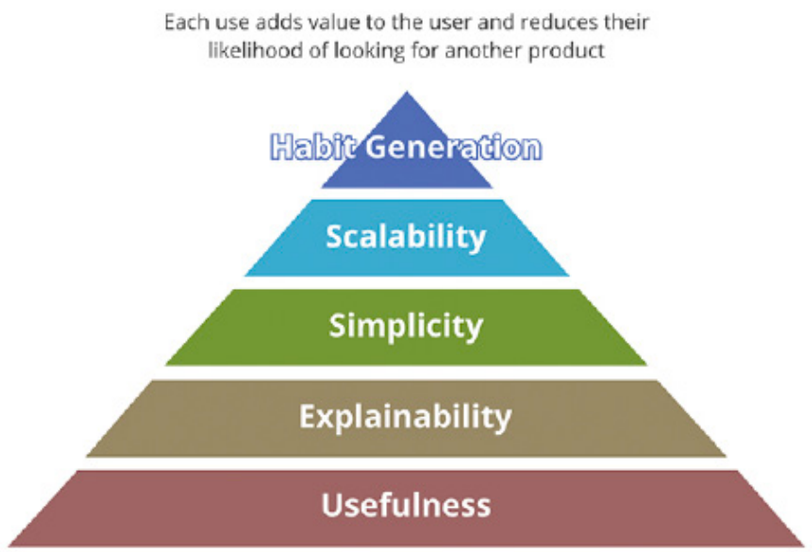


Figura 1: Pirâmide de valor do produto (fonte)

A pirâmide de valor fornece uma prioridade em cada aspecto do produto. Nem todas as perguntas de valor que fazemos sobre o produto têm o mesmo peso. Se o resultado não for útil, nenhum dos outros aspectos será importante. O resultado não é realmente um produto, mas se torna mais poluente de dados para o grupo de resultados úteis. Da mesma forma, a escalabilidade só importa depois que a simplicidade e a explicabilidade são abordadas.

Como a pirâmide de valor está relacionada aos produtos de dados? Cada saída de dados, para ser um produto de dados:

- **Deve ter uma utilidade clara.** A quantidade de dados que a sociedade está gerando é rivalizada apenas pela quantidade de poluentes de dados que estamos gerando. Esses são resultados que não têm valor e uso claros, muito menos uma estratégia sobre o que fazer com eles.
- **Deve ser explicável.** Com o surgimento da IA/ML, a explicabilidade se tornou ainda mais importante para a tomada de decisão baseada em dados. Os dados são tão bons quanto os metadados que os descrevem. Pense nisso em termos de alimentos: o sabor importa, mas um fator mais importante é o valor nutricional dos ingredientes.
- **Deve ser simples.** Um exemplo de uso indevido de produto é usar um garfo para comer cereais em vez de uma colher. Além disso, a simplicidade é essencial, mas não suficiente: além da simplicidade os produtos devem ser intuitivos. Sempre que possível, tanto o uso intencional como o não intencional dos dados devem ser óbvios.
- **Deve ser escalável.** Os dados são um dos poucos recursos que cresce com o uso. Quanto mais dados você processa, mais dados tem. Se as entradas e saídas do sistema não forem limitadas e crescerem constantemente, o sistema deve ser escalável em potência de compute, capacidade de armazenamento e capacidade de compute expressiva. Plataformas de dados em nuvem, como Databricks, estão em uma posição única para responder a todos os três aspectos.
- **Deve gerar hábitos.** No domínio dos dados, não estamos preocupados com a retenção de clientes, como é o caso dos produtos de varejo. No entanto, o valor da geração de hábitos é óbvio se aplicado às práticas recomendadas. Os sistemas e as saídas de dados devem apresentar as práticas recomendadas e promovê-las: deve ser mais fácil usar os dados e o sistema da forma pretendida do que o contrário.

Os dados geoespaciais (e, na verdade, quaisquer produtos de dados) devem aderir a todos os aspectos acima mencionados. Além dessa tarefa difícil, os dados geoespaciais têm algumas necessidades específicas.

Padrões de dados geoespaciais

Os padrões de dados geoespaciais são usados para garantir que os dados geográficos sejam coletados, organizados e compartilhados de forma consistente e confiável. Estes padrões podem incluir diretrizes para coisas como formatação de dados, sistemas de coordenadas, projeções de mapas e metadados. A adesão aos padrões facilita o compartilhamento de dados entre diferentes organizações, permitindo maior colaboração e acesso mais amplo a informações geográficas.

A Comissão Geoespacial (governo do Reino Unido) definiu o UK Geospatial Data Standards Register como um repository central para padrões de dados a serem aplicados no caso de dados geoespaciais. Além disso, a missão desse Register é:

- **"Garantir que os dados geoespaciais do Reino Unido sejam mais consistentes, coerentes e utilizáveis em uma ampla gama de sistemas."** — Esses conceitos são uma chamada para a importância da explicabilidade, utilidade e geração de hábitos (possivelmente outros aspectos da pirâmide de valores).
- **"Capacitar a comunidade geoespacial do Reino Unido para que se envolva mais com os padrões e órgãos de padrões relevantes."** — A geração de hábitos na comunidade é tão importante quanto o design robusto e crítico do padrão. Se não forem adotados, os padrões serão inúteis.
- **"Defender a compreensão e o uso dos padrões de dados geoespaciais em outros setores do governo."** — A pirâmide de valor também se aplica aos padrões. Conceitos como facilidade de adesão (útil/simplicidade), finalidade do padrão (explicabilidade/usabilidade), adoção (geração de hábito) são fundamentais para a geração de valor de um padrão.

Uma ferramenta crítica para alcançar a missão dos padrões de dados são os princípios de dados FAIR:

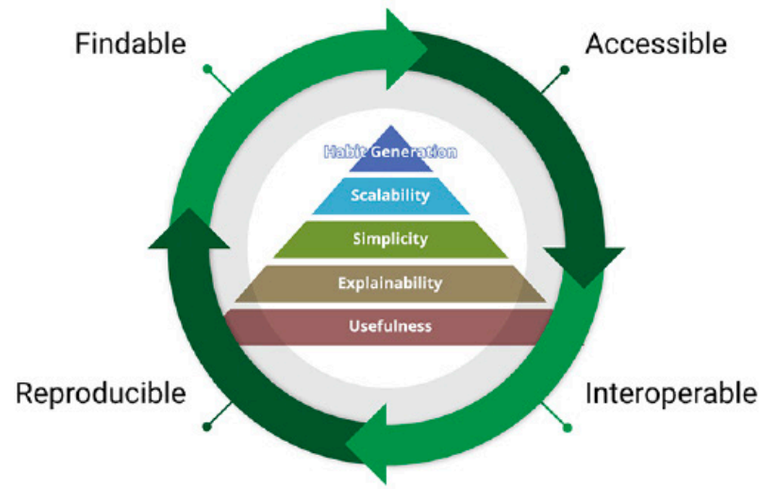
- **Findáveis** — O primeiro passo no (re)uso de dados é encontrá-los. Metadados e dados devem ser fáceis de encontrar para humanos e computadores. Metadados legíveis em máquina são essenciais para a descoberta automática de conjuntos de dados e serviços.
- **Acessíveis** — Depois que o usuário encontra os dados necessários, ele precisa saber como podem ser acessados, possivelmente incluindo autenticação e autorização.
- **Interoperáveis** — Os dados geralmente precisam ser integrados a outros dados. Além disso, os dados precisam interoperar com aplicativos ou fluxos de trabalho para análise, armazenamento e processamento.
- **Reutilizáveis** — O objetivo final dos princípios FAIR é otimizar a reutilização de dados. Para isso, os metadados e os dados devem ser bem descritos para que possam ser replicados e/ou combinados em diferentes configurações.

Compartilhamos a crença de que os princípios FAIR são cruciais para o design de produtos de dados escaláveis nos quais podemos confiar. Para ser justo, os princípios FAIR são baseados no senso comum, então por que são fundamentais para nossas considerações? *"O que vejo nos princípios FAIR não é novo por si só, mas o que eles fazem bem é articular, de forma acessível, a necessidade de uma abordagem abrangente para a melhoria de dados. Essa facilidade de comunicação é o motivo pelo qual os princípios FAIR estão sendo usados cada vez mais amplamente como um guarda-chuva para a melhoria de dados, e não apenas na comunidade geoespacial."* — Artigo "A FAIR wind sets our course for data improvement".

Para apoiar ainda mais essa abordagem, o Comitê Federal de Dados Geográficos desenvolveu o Plano Estratégico de Infraestrutura Nacional de Dados Espaciais (NSDI), que abrange os anos de 2021 a 2024 e foi aprovado em novembro de 2020. Os objetivos do NSDI são, em essência, os princípios FAIR e transmitir a mesma mensagem de projetar sistemas que promovam a economia circular de dados: produtos de dados que fluem entre as organizações seguindo padrões comuns e em cada etapa da cadeia de fornecimento de dados desbloqueiam novos valores e novas oportunidades. O fato de que esses princípios estão permeando diferentes jurisdições e são adotados em diferentes reguladores é uma prova da robustez e solidez da abordagem.

Figura 2:
Objetivos estratégicos do NDSI

Os conceitos FAIR combinam muito bem com o design do produto de dados. Na verdade, os princípios FAIR estão atravessando toda a pirâmide de valor do produto e forma um ciclo de valor. Ao adotar os princípios de pirâmide de valor e FAIR, projetamos produtos de dados com perspectiva interna e externa. Isso promove a reutilização de dados em vez do acúmulo de dados.



Por que os princípios FAIR são importantes para dados geoespaciais e produtos de dados geoespaciais? Os princípios FAIR são transcendentes aos dados geoespaciais, são, na verdade, transcendentes aos dados, são um sistema simples, mas coerente, de princípios orientadores para um bom design, e esse bom design pode ser aplicado a qualquer coisa, incluindo dados geoespaciais e sistemas geoespaciais.

Sistemas de indexação de grade

Nas soluções SIG tradicionais, o desempenho das operações espaciais é geralmente alcançado através da construção de estruturas em árvore (KD trees, ball trees, Quad trees etc.). O problema com as abordagens em árvore é que elas eventualmente quebram o princípio da escalabilidade: quando os dados são grandes demais para serem processados para construir a árvore e a computação necessária para construir a árvore é muito longa e vai contra o propósito. Isso também afeta negativamente a acessibilidade dos dados. Se não pudermos construir a árvore, não poderemos acessar os dados completos e, na verdade, não poderemos reproduzir os resultados. Neste caso, os sistemas de indexação de grade fornecem uma solução.

Os sistemas de indexação de grade são criados desde o início tendo em mente os aspectos de escalabilidade dos dados geoespaciais. Em vez de construir as árvores, eles definem uma série de grades que cobrem a área de interesse. No caso do H3 (desenvolvido pela Uber), a grade cobre a área da Terra. No caso de sistemas de indexação de grade local (por exemplo, British National Grid), eles podem cobrir apenas a área específica de interesse. Essas grades são compostas por células que possuem identificadores exclusivos. Existe uma relação matemática entre a localização e a célula na grade. Isso torna os sistemas de indexação de grade muito escaláveis e de natureza paralela.

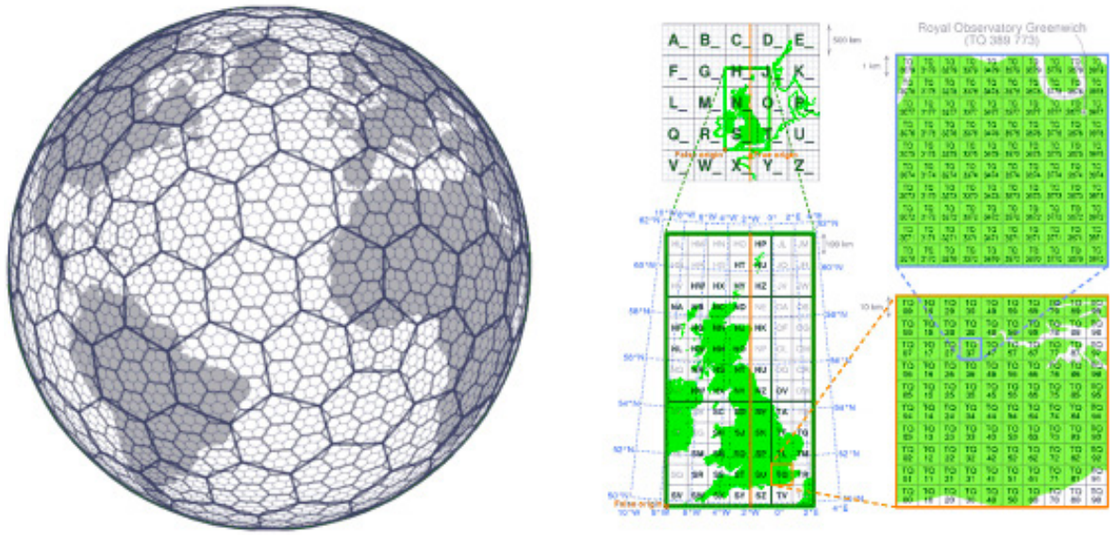


Figura 4: Sistemas de indexação de grade (H3, British National Grid)

Outro aspecto importante dos sistemas de indexação de grade é que eles são de código aberto, permitindo que os valores de indexação sejam universalmente aproveitados por produtores de dados e consumidores. Os dados podem ser enriquecidos com as informações da indexação de grade em qualquer etapa de sua jornada pela cadeia de fornecimento de dados. Isso torna os sistemas de indexação de grade um exemplo de padrões de dados orientados pela comunidade. Os padrões de dados orientados pela comunidade por natureza não exigem imposição, que adere totalmente ao aspecto de geração de hábitos da pirâmide de valor e aborda significativamente os princípios de interoperabilidade e acessibilidade da FAIR.

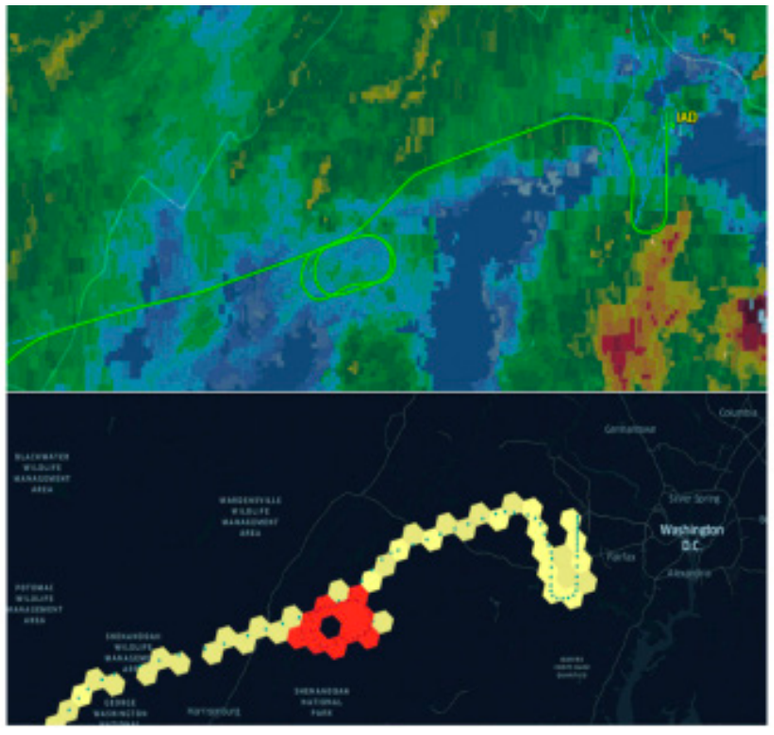


Figura 5: Exemplo de como usar H3 para expressar padrões de espera de voo

A Databricks anunciou recentemente a compatibilidade nativa para o sistema de indexação de grade H3 seguindo a mesma proposta de valor. A adoção de padrões comuns do setor, orientados pela comunidade, é a única maneira de impulsionar adequadamente a geração de hábitos e a interoperabilidade. Para fortalecer essa declaração, organizações como CARTO , ESRI e Google têm promovido o uso de sistemas de indexação de grade para projetos de SIGs escaláveis. Além disso, o projeto Mosaic do Databricks Labs é compatível com o British National Grid como o sistema de índice de grade padrão que é amplamente usado no governo do Reino Unido. Os sistemas de indexação de grade são fundamentais para a escalabilidade do processamento de dados geoespaciais e para projetar adequadamente soluções para problemas complexos (por exemplo, Figura 5: Padrões de espera de voo usando H3).

Diversidade de dados geoespaciais

Os padrões de dados geoespaciais gastam uma quantidade sólida de esforços em relação à padronização do formato dos dados, e o formato é uma das considerações mais importantes quando se trata de interoperabilidade e reprodutibilidade. Além disso, se a leitura dos seus dados é complexa, como podemos falar em simplicidade? Infelizmente, os formatos de dados geoespaciais são em geral complexos, pois os dados podem ser produzidos em vários formatos, incluindo formatos de código aberto e formatos específicos do fornecedor. Considerando apenas dados vetoriais, podemos esperar que os dados cheguem em WKT, WKB, GeoJSON, web CSV, CSV, Shape File, GeoPackage e muitos outros. Por outro lado, se estamos considerando dados rasterizados, podemos esperar que os dados cheguem em qualquer número de formatos, como GeoTiff, netCDF, GRIB ou GeoDatabase. Para obter uma lista abrangente de formatos, consulte este artigo.

O domínio de dados geoespaciais é tão diversificado e cresceu organicamente ao longo dos anos em torno dos casos de uso que estava abordando. A utilização de um ecossistema tão diversificado é um desafio enorme. Um esforço recente do Open Geospatial Consortium (OGC) para padronizar o Apache Parquet e sua especificação de esquema geoespacial GeoParquet é um passo na direção certa. A simplicidade é um dos principais aspectos do projeto de um produto bom escalável e robusto — a unificação leva à simplicidade e aborda uma das principais fontes de fricção no ecossistema — a ingestão de dados. Padronizar para GeoParquet traz muito valor que aborda todos os aspectos dos dados de FAIR e pirâmide de valor.

Figura 6: GeoParquet como formato de dados padrão geoespacial

Por que introduzir outro formato em um ecossistema já complexo? GeoParquet não é um formato novo, é uma especificação de esquema para o formato Apache Parquet que já é amplamente adotado e usado pela indústria e pela comunidade. O Parquet como formato base é compatível com colunas binárias e permite o armazenamento de carga de dados arbitrária. Ao mesmo tempo, o formato oferece suporte a colunas de dados estruturados que podem armazenar metadados junto com a carga de dados. Isso o torna uma escolha que promove a interoperabilidade e a reprodutibilidade. Por fim, o formato Delta Lake foi criado com base no Parquet e traz propriedades ACID para a mesa. As propriedades ACID de um formato são cruciais para a reprodutibilidade e para resultados confiáveis. Além disso, o Delta é o formato usado pela solução de compartilhamento de dados escalável Delta Sharing.

O Delta Sharing permite o compartilhamento de dados em escala empresarial entre qualquer nuvem pública usando Databricks (opções DIY para nuvem privada estão disponíveis usando módulos de código aberto). O Delta Sharing abstrai completamente a necessidade de APIs Rest personalizadas para expor dados a terceiros. Qualquer ativo de dados armazenado no Delta (usando o esquema GeoParquet) torna-se automaticamente um produto de dados que pode ser exposto a partes externas de forma controlada e controlada. O Delta Sharing foi criado do zero com as práticas recomendadas de segurança em mente.

Figura 7: Delta Sharing simplificando o acesso aos dados no ecossistema

Economia de dados circulares

Emprestando os conceitos do domínio de sustentabilidade, podemos definir uma economia circular de dados como um sistema no qual os dados são coletados, compartilhados e usados de forma a maximizar seu valor, minimizando o desperdício e os impactos negativos, como tempo de compute desnecessário, insights não confiáveis ou ações tendenciosas baseadas em poluentes de dados. A reutilização é o conceito-chave nesta consideração: como podemos minimizar a "reinvenção da roda"? Existem inúmeros ativos de dados que representam a mesma área, os mesmos conceitos com apenas pequenas alterações para melhor corresponder a um caso de uso específico. Isso se deve às otimizações reais ou ao fato de que foi mais fácil criar uma nova cópia dos ativos do que reutilizar os existentes? Ou era muito difícil encontrar os ativos de dados existentes ou talvez fosse muito complexo definir padrões de acesso a dados?

A duplicação de ativos de dados tem muitos aspectos negativos nas considerações FAIR e nas considerações de pirâmide de valor de dados — ter muitos ativos de dados semelhantes (mas diferentes) que representam a mesma área e os mesmos conceitos pode deteriorar as considerações de simplicidade do domínio de dados — torna-se difícil identificar o ativo de dados em que realmente podemos confiar. Também pode ter implicações muito

negativas na geração de hábitos. Surgirão muitas comunidades de nicho que se padronizarão ignorando as práticas recomendadas do ecossistema mais amplo ou, pior ainda, não padronizarão nada.

Em uma economia circular de dados, os dados são tratados como um recurso valioso que pode ser usado para criar novos produtos e serviços, além de melhorar os já existentes. Esta abordagem incentiva a reutilização e a reciclagem de dados, em vez de tratá-los como um bem descartável. Mais uma vez, utilizamos a analogia da sustentabilidade no sentido literal: argumentamos que esta é a forma correta de abordar o problema. Os poluentes de dados são um verdadeiro desafio para as organizações, tanto interna como externamente. Um artigo do The Guardian afirma que menos de 1% dos dados coletados são realmente analisados. Há muita duplicação de dados, a maioria dos dados é difícil de acessar e a obtenção do valor real é complicada demais. A economia circular de dados promove as melhores práticas e a reutilização dos ativos de dados existentes, permitindo uma interpretação e percepções mais consistentes em todo o ecossistema de dados.

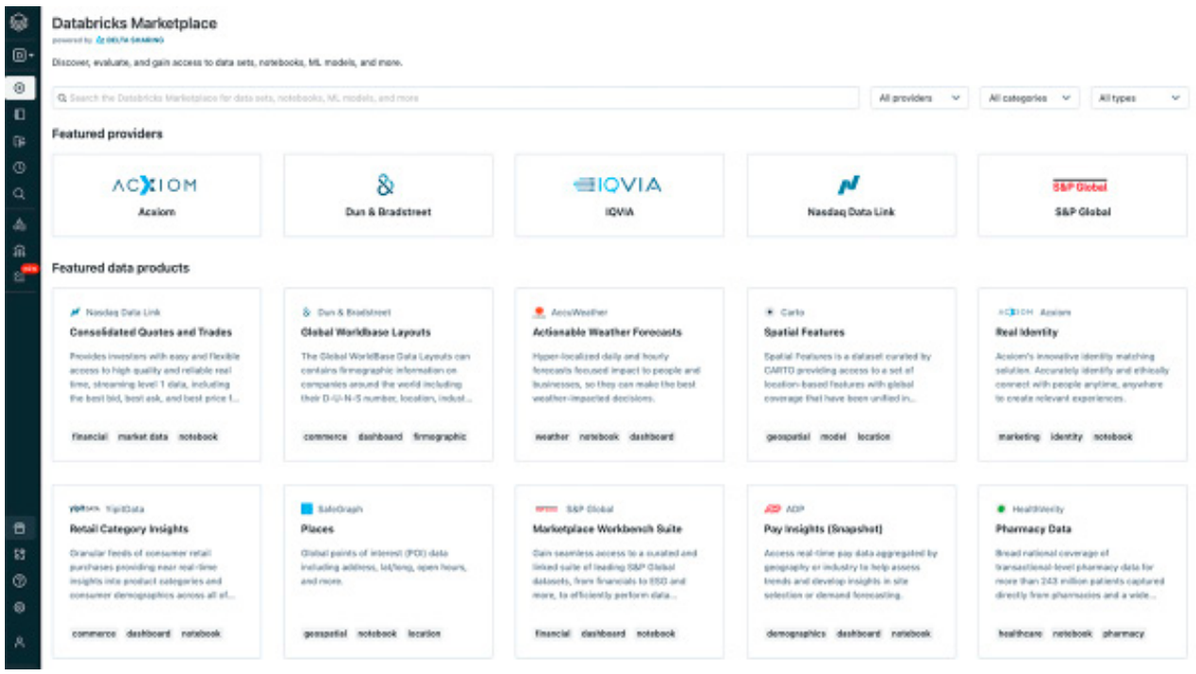


Figura 8: Databricks Marketplace

A interoperabilidade é um componente essencial dos princípios de dados FAIR e, a partir da interoperabilidade, vem à mente uma questão de circularidade. Como podemos projetar um ecossistema que maximize a utilização e a reutilização dos dados? Mais uma vez, os princípios FAIR, juntamente com a pirâmide de valor, oferecem respostas. A capacidade de localização dos dados é fundamental para a reutilização de dados e para solucionar a poluição de dados. Com ativos de dados que podem ser descobertos facilmente, podemos evitar a recriação dos mesmos ativos de dados em vários lugares com apenas uma pequena alteração. Em vez disso, obtemos um ecossistema de dados coerente com dados que podem ser facilmente combinados e reutilizados. A Databricks anunciou recentemente o Databricks Marketplace. A ideia por trás do Marketplace está alinhada com a definição original de produto de dados de DJ Patil. O Marketplace oferecerá suporte ao compartilhamento de conjuntos de dados, notebooks, dashboards e modelos de machine learning. O módulo essencial para esse mercado é o conceito de Delta Sharing: o canal dimensionável, flexível e robusto para o compartilhamento de qualquer dado, incluindo dados geoespaciais.

Projetar produtos de dados escaláveis que permanecerão no mercado é crucial. Para maximizar o valor agregado de cada produto de dados, deve-se considerar fortemente os princípios FAIR e a pirâmide de valor do produto. Sem esses princípios orientadores, aumentaremos apenas os problemas que já estão presentes no sistemas atuais. Cada produto de dados deve resolver um problema único e deve resolvê-lo de forma simples, reprodutível e robusta.

Leia mais sobre como a Plataforma Databricks Lakehouse pode ajudar você a acelerar o retorno sobre o investimento dos seus produtos de dados no e-book: Uma nova abordagem para o Data Sharing.

Comece a experimentar com estes notebooks gratuitos da Databricks.

SEÇÃO 2.7

Linhagem de dados com o Unity Catalog

de **PAUL ROOME, TAO FENG E SACHIN THAKUR**

8 de junho de 2022

Este artigo explicará a importância da linhagem de dados, alguns dos casos de uso comuns, nossa visão para uma melhor transparência de dados e compreensão de dados com linhagem de dados.

O que é linhagem de dados e por que é importante?

A linhagem de dados descreve as transformações e refinamentos de dados da fonte ao insight. A linhagem inclui a captura de todos os metadados e eventos relevantes associados aos dados em seu ciclo de vida, incluindo a origem do conjunto de dados, quais outros conjuntos de dados foram usados para criá-lo, quem o criou e quando, quais transformações foram realizadas, quais outros conjuntos de dados aproveitá-lo e muitos outros eventos e atributos. Com uma solução de linhagem de dados, as equipes de dados obtêm uma visão completa de como os dados são transformados e como fluem em seus bens de dados.

À medida que cada vez mais organizações adotam uma cultura orientada por dados e configuram processos e ferramentas para democratizar e dimensionar dados e IA, a linhagem de dados está se tornando um pilar essencial de uma estratégia pragmática de governança e gerenciamento de dados.

Para entender a importância da linhagem de dados, destacamos abaixo alguns dos casos de uso comuns que ouvimos de nossos clientes.

Análise de impacto

Os dados passam por várias atualizações ou revisões ao longo de seu ciclo de vida, e entender o impacto potencial de qualquer mudança de dados nos consumidores downstream torna-se importante do ponto de vista de gerenciamento de riscos. Com a linhagem de dados, as equipes de dados podem ver todos os consumidores downstream (aplicativos, dashboards, modelos de machine learning ou conjuntos de dados etc.) afetados por alterações de dados, entender a gravidade do impacto e notificar as partes interessadas relevantes. A linhagem também ajuda as equipes de TI a comunicar proativamente migrações de dados para as equipes apropriadas, garantindo a continuidade dos negócios.

Entendimento e transparência de dados

As organizações lidam com um influxo de dados de várias fontes, e construir uma melhor compreensão do contexto em torno dos dados é fundamental para garantir a confiabilidade dos dados. A linhagem de dados é uma ferramenta poderosa que permite que os líderes de dados promovam uma melhor transparência e compreensão dos dados em suas organizações. A linhagem de dados também capacita os consumidores de dados, como cientistas de dados, engenheiros de dados e analistas de dados, a terem conhecimento de contexto à medida que realizam análises, resultando em melhores resultados de qualidade. Finalmente, os administradores de dados podem ver quais conjuntos de dados não são mais acessados ou se tornaram obsoletos para se aposentar dos dados desnecessários e garantir a qualidade dos dados para os usuários finais.

Depuração e diagnóstico

Você pode ter todas as verificações e equilíbrios em vigor, mas algo acabará falhando. A linhagem de dados ajuda as equipes de dados a realizar uma análise de causa raiz de quaisquer erros em seu pipeline de dados, aplicativos, dashboards, modelo de machine learning etc., rastreando o erro até sua origem. Isto reduz significativamente o tempo de depuração, economizando dias ou, em muitos casos, meses de esforço manual.

Prontidão para conformidade e auditoria

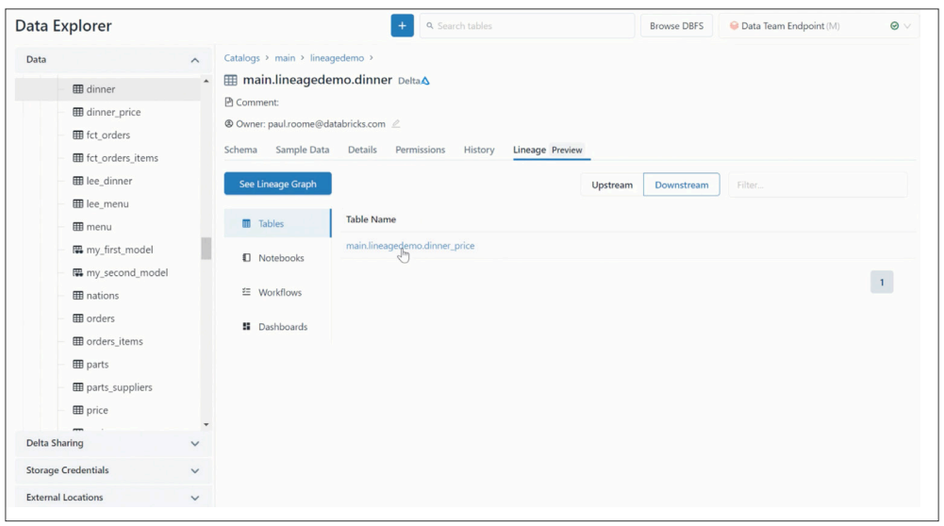
Muitas regulamentações de conformidade, como o Regulamento Geral sobre a Proteção de Dados (RGPD), a Lei de Privacidade do Consumidor da Califórnia (CCPA), a Lei de Portabilidade e Responsabilidade de Seguros de Saúde (HIPPA), o Comitê de Supervisão Bancária da Basileia (BCBS) 239 e a Lei Sarbanes-Oxley (SOX), exigem que as organizações tenham um entendimento claro e visibilidade do fluxo de dados. Como resultado, a rastreabilidade dos dados se torna um requisito fundamental para que a arquitetura de dados atenda às normas legais. A linhagem de dados ajuda as organizações a estarem em conformidade e prontas para a auditoria, aliviando assim a sobrecarga operacional de criar manualmente as trilhas de fluxos de dados para fins de relatórios de auditoria.

Transparência sem esforço e controle proativo com linhagem de dados

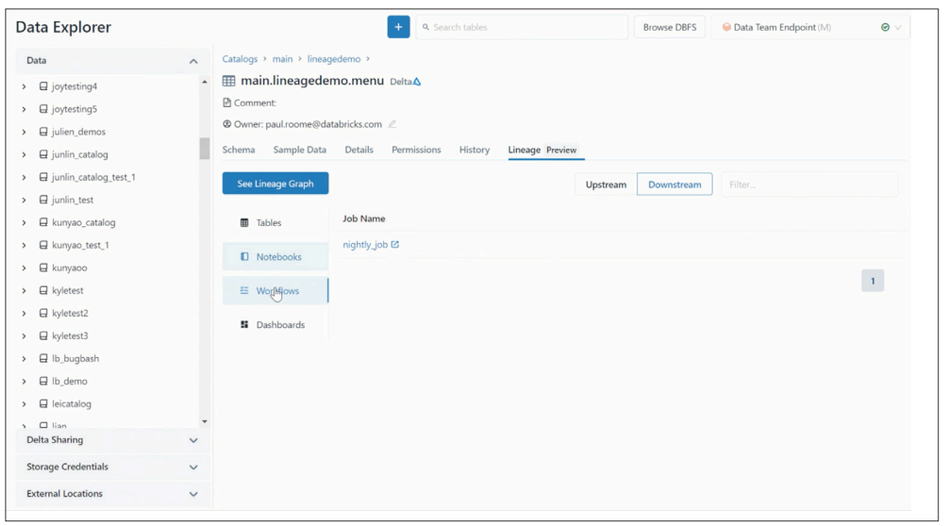
O lakehouse oferece uma arquitetura pragmática de gerenciamento de dados que simplifica substancialmente a infraestrutura de dados empresariais e acelera a inovação, unificando seus casos de uso de armazenamento de dados e IA em uma única plataforma. Acreditamos que a linhagem de dados é um facilitador essencial para uma melhor transparência de dados e compreensão de dados em seu lakehouse, superando os relacionamentos entre dados, trabalhos e consumidores, e ajudando as organizações a adotar práticas proativas de gerenciamento de dados. Por exemplo:

- Como responsável por um dashboard, você quer receber notificação da próxima vez que uma tabela da qual seu dashboard depende não tiver sido carregada corretamente?
- Como profissional de machine learning que desenvolve um modelo, você quer receber um alerta de que um recurso crítico em seu modelo será descontinuado em breve?
- Como administrador de governança, você quer controlar automaticamente o acesso aos dados com base na procedência?

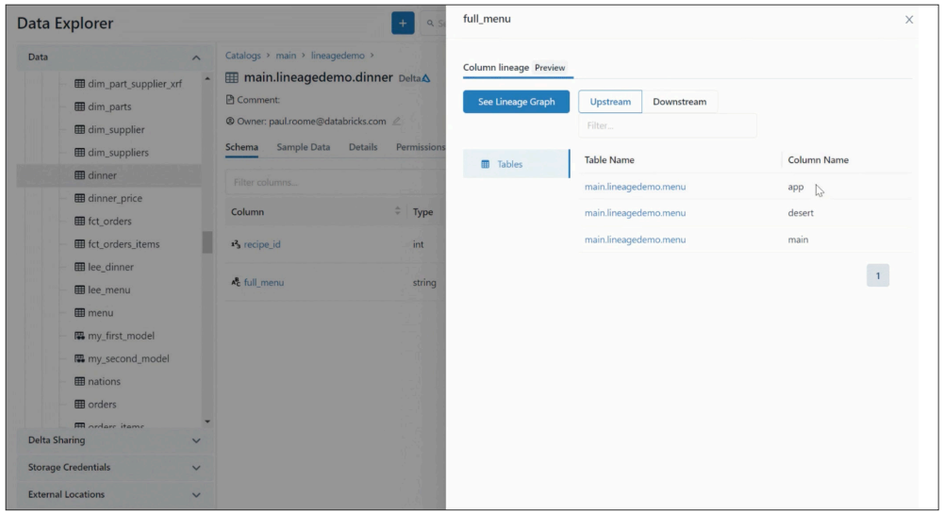
Todos esses recursos dependem da coleta automática de linhagem de dados em todos os casos de uso e personas, e é por isso que a linhagem de dados e o lakehouse são uma combinação poderosa.



Linhagem de dados para tabelas



Linhagem de dados para notebooks, workflows, dashboards



Linhagem de dados para colunas de tabela

Segurança integrada: os gráficos de linhagem no Unity Catalog reconhecem privilégios e compartilham o mesmo modelo de permissão que o Unity Catalog. Se os usuários não tiverem acesso a uma tabela, eles não poderão explorar a linhagem associada à tabela, adicionando uma camada adicional de segurança para considerações de privacidade.

Facilmente exportável via API REST: A linhagem pode ser visualizada no Data Explorer quase em tempo real e recuperada por meio da API REST para dar suporte a integrações com nossos parceiros de catálogo.

Como começar com linhagem de dados no Unity Catalog

A linhagem de dados está disponível com níveis Databricks Premium e Enterprise sem custo adicional. Se você já é um cliente Databricks, siga os guias de linhagem de dados (AWS | Azure) para começar. Se você não é um cliente Databricks, inscreva-se para uma avaliação gratuita com um workspace Premium ou Enterprise.

databricks

SEÇÃO 2.8

Ingestão fácil para lakehouse com COPY INTO

de **AEMRO AMARE, EMMA LIU, AMIT KARA** e **JASRAJ DANGE**

17 de janeiro de 2023

Uma nova arquitetura de gerenciamento de dados, conhecida como data lakehouse, surgiu de forma independente em muitas organizações e casos de uso para dar suporte à IA e ao BI diretamente em grandes quantidades de dados. Um dos principais fatores de sucesso do uso do data lakehouse para funções analíticas e machine learning é a capacidade de ingerir dados de vários tipos de forma rápida e fácil, incluindo dados de plataformas de armazenamento no local (data warehouses, mainframes), dados de streaming em tempo real e ativos de dados em massa.

Como a ingestão de dados no lakehouse é um processo contínuo que alimenta o proverbial pipeline de ETL, você precisará de várias opções para ingerir vários formatos, tipos e latências de dados. Para dados armazenados em objetos em nuvens como AWS S3, Google Cloud Storage e Azure Data Lake Storage, o Databricks oferece Auto Loader, um recurso nativamente integrado, que permite que engenheiros de dados ingiram milhões de arquivos do armazenamento em nuvens continuamente. Em outros casos de transmissão (por exemplo, sensor IoT ou dados clickstream), o Databricks fornece conectores nativos para Structured Streaming do Apache Spark para ingerir rapidamente dados de filas de mensagens populares, como Apache Kafka, Azure Event Hubs ou AWS Kinesis em baixas latências. Além disso, muitos clientes podem aproveitar ferramentas de ingestão populares que se integram ao Databricks, como o Fivetran, para ingerir facilmente dados de aplicativos corporativos, bancos de dados, mainframes e muito mais no lakehouse. Por fim, o analista pode usar o comando simples "COPY INTO" para extrair novos dados automaticamente para o lakehouse, sem a necessidade de controlar quais arquivos já foram processados.

Este artigo se concentra em COPY INTO, um comando SQL simples, mas poderoso, que permite realizar a ingestão de arquivos de batches no Delta Lake a partir de armazenamentos de objetos em nuvens. É idempotente, o que garante a ingestão de arquivos com semântica exatamente uma vez quando executado várias vezes, suportando acréscimos incrementais e transformações simples. Pode ser executado uma vez, de forma ad hoc, e pode ser agendado através do Databricks Workflows. Nas versões recentes do Databricks Runtime, COPY INTO introduziu novas funcionalidades para visualização de dados, validação, tratamento aprimorado de erros e uma nova maneira de copiar para uma tabela Delta Lake sem esquema para que os usuários possam começar rapidamente, completando a jornada do usuário de ponta a ponta para ingerir de armazenamentos de objetos de nuvens. Vamos dar uma olhada nos casos de uso populares de COPY INTO.

1. Ingerindo dados pela primeira vez

COPY INTO requer a existência de uma tabela à medida que ingere os dados em uma tabela Delta de destino. No entanto, você não tem ideia de como seus dados se parecem. Primeiro, você cria uma tabela Delta vazia.

```
CREATE TABLE my_example_data;
```

Antes de gravar seus dados, convém visualizá-los e garantir que os dados pareçam corretos. O modo COPY INTO Validate é um novo recurso no Databricks Runtime 10.3 e superior que permite visualizar e validar dados de origem antes de ingerir muitos arquivos dos repositórios de objetos na nuvem. Essas validações incluem:

- se os dados puderem ser analisados
- o esquema corresponde ao da tabela de destino ou se o esquema precisa ser evoluído
- todas as restrições de nulidade e verificação na tabela são atendidas

```
COPY INTO my_example_data
FROM 's3://my-bucket/exampleData'
FILEFORMAT = CSV
VALIDATE
COPY_OPTIONS ('mergeSchema' = 'true')
```

O padrão para validação de dados é analisar todos os dados no diretório de origem para garantir que não haja problemas, mas as linhas retornadas para visualização são limitadas. Uma alternativa é fornecer o número de linhas para visualizar após VALIDATE.

O "mergeSchema" COPY_OPTION especifica que não há problema em desenvolver o esquema da tabela Delta de destino. A evolução do esquema permite apenas a adição de novas colunas e não aceita alterações de tipo de dados para colunas existentes. Em outros casos de uso, você pode omitir essa opção se pretende gerenciar seu esquema de tabela mais estritamente, pois seu pipeline de dados pode ter requisitos rigorosos de esquema e talvez você não queira evoluir o esquema o tempo todo. No entanto, nossa tabela Delta de destino no exemplo acima é uma tabela vazia e sem colunas no momento; portanto, é preciso especificar o "mergeSchema" COPY_OPTION neste caso.

Table +

	_c0	_c1	_c2	_c3	_c4	_c5
1	first_name	last_name	product_id	subscription_id	subscription_date	last_updated
2	Aemro	Amare	101	1	2020-08-31	2022-12-21 19:23:23.609348
3	Emma	Liu	200	2	2020-08-30	2022-12-21 19:23:23.609348
4	Amit	Kara	50	3	2020-09-01	2022-12-21 19:23:23.609348
5	Burak	Yavuz	501	4	2020-09-02	2022-12-21 19:23:23.609348

Figura 1: Saída do modo COPY INTO VALIDATE

2. Configurar COPY INTO

Ao analisar os resultados de VALIDATE (veja a Figura 1), você pode notar que seus dados não têm a aparência desejada. Não ficou satisfeito por ter visualizado o conjunto de dados primeiro? A primeira coisa que você observa é que os nomes das colunas não refletem o que está especificado no cabeçalho CSV. O que é pior, o cabeçalho é mostrado como uma linha em seus dados. Você pode configurar o analisador CSV especificando FORMAT_OPTIONS. Vamos adicioná-los a seguir.

```
COPY INTO my_example_data
FROM 's3://my-bucket/exampleData'
FILEFORMAT = CSV
VALIDATE
FORMAT_OPTIONS ('header' = 'true', 'inferSchema' = 'true', 'mergeSchema' =
'true')
COPY_OPTIONS ('mergeSchema' = 'true')
```

Ao usar FORMAT OPTION, você pode dizer a COPY INTO para inferir os tipos de dados do arquivo CSV especificando a opção inferSchema; caso contrário, todos os tipos de dados padrão são STRINGs. Por outro lado, formatos de arquivo binário como AVRO e PARQUET não precisam dessa opção, pois definem seu próprio esquema. Outra opção, "mergeSchema", afirma que o esquema deve ser inferido em uma amostra abrangente de arquivos CSV em vez de apenas uma. A lista abrangente de opções específicas de formato pode ser encontrada na documentação.

A Figura 2 mostra a saída de validação de que o cabeçalho foi analisado corretamente.

Table +

	first_name	last_name	product_id	subscription_id	subscription_date	last_updated
1	Aemro	Amare	101	1	2020-08-31	2022-12-21T19:23:23.609+0000
2	Emma	Liu	200	2	2020-08-30	2022-12-21T19:23:23.609+0000
3	Amit	Kara	50	3	2020-09-01	2022-12-21T19:23:23.609+0000
4	Burak	Yavuz	501	4	2020-09-02	2022-12-21T19:23:23.609+0000

Figura 2: Saída do modo COPY INTO VALIDATE com cabeçalho habilitado e inferSchema

3. Anexar dados a uma tabela Delta

Agora que a visualização está adequada, podemos remover a palavra-chave VALIDATE e executar o comando COPY INTO.

```
COPY INTO my_example_data
FROM 's3://my-bucket/exampleData'
FILEFORMAT = CSV
FORMAT_OPTIONS ('header' = 'true', 'inferSchema' = 'true', 'mergeSchema' =
'true')
COPY_OPTIONS ('mergeSchema' = 'true')
```

COPY INTO monitora o estado dos arquivos que foram ingeridos. Ao contrário de comandos como INSERT INTO, os usuários obtêm idempotência com COPY INTO, o que significa que não obterão dados duplicados na tabela de destino ao executar COPY INTO várias vezes a partir dos mesmos dados de origem.

COPY INTO pode ser executado uma vez, de forma ad hoc, e pode ser agendado com o Databricks Workflows. Embora COPY INTO não ofereça suporte a baixas latências para ingestão nativa, você pode acioná-lo por meio de orquestradores como o Apache Airflow.

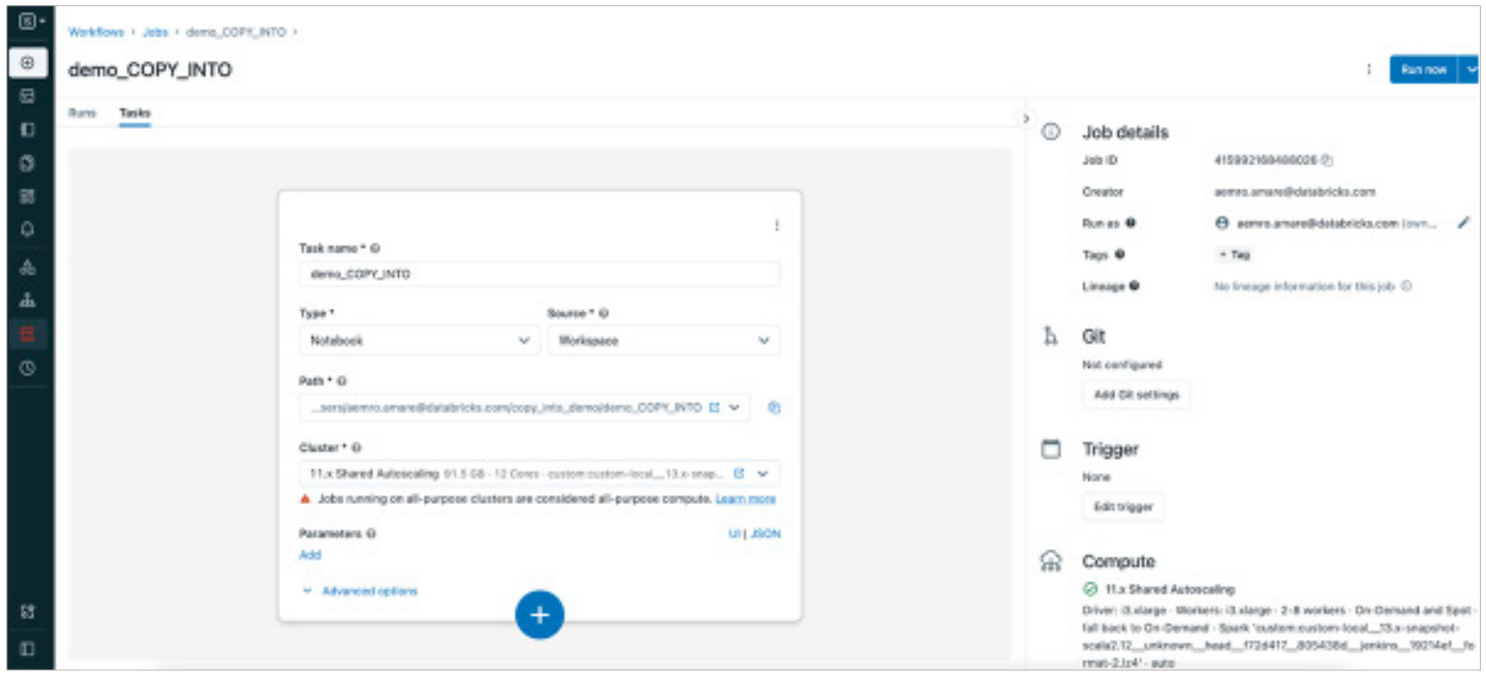


Figura 3: UI do fluxo de trabalho do Databricks para agendar uma tarefa

4. Acesso seguro aos dados com COPY INTO

COPY INTO oferece suporte ao acesso seguro de várias maneiras. Nesta seção, destacaremos duas novas opções que podem ser usadas no Databricks SQL e em notebooks de versões recentes:

Unity Catalog

Com a disponibilidade geral do Unity Catalog da Databricks, você pode usar COPY INTO para ingerir dados em tabelas gerenciadas ou externas do Unity Catalog a partir de qualquer fonte e formato de arquivo suportado pelo COPY INTO. O Unity Catalog também adiciona novas opções para configurar o acesso seguro a dados brutos, permitindo usar locais externos do Unity Catalog ou credenciais de armazenamento para acessar dados no armazenamento de objetos na nuvem. Saiba mais sobre como usar COPY INTO com o Unity Catalog.

Credenciais temporárias

E se você não tiver configurado o Unity Catalog ou o perfil da instância? Que tal usar dados de um bucket de terceiros confiável? Este é um recurso COPY INTO conveniente que permite ingerir dados com credenciais temporárias em linha para lidar com o caso de uso de ingestão em massa ad hoc.

```
COPY INTO my_example_data
FROM 's3://my-bucket/exampleDataPath' WITH (
  CREDENTIAL (AWS_ACCESS_KEY = '...', AWS_SECRET_KEY = '...', AWS_SESSION_
TOKEN = '...')
)
FILEFORMAT = CSV
```

5. Filtrar arquivos para ingestão

E quanto à ingestão de um subconjunto de arquivos onde os nomes dos arquivos correspondem a um padrão? Você pode aplicar padrões de glob — um padrão de glob que identifica os arquivos a serem carregados no diretório de origem. Por exemplo, vamos filtrar e ingerir arquivos que contenham a palavra “raw_data” no nome do arquivo abaixo.

```
COPY INTO my_example_data
  FROM  's3://my-bucket/exampleDataPath'
  FILEFORMAT = CSV
  PATTERN = '*raw_data*.csv'
  FORMAT_OPTIONS ('header' = 'true')
```

6. Ingerir arquivos em um período de tempo

Na engenharia de dados, geralmente é necessário ingerir arquivos que foram modificados antes ou depois de um timestamp específico. Dados entre dois timestamps também podem ser relevantes. As opções de formato “modifiedAfter” e “modifiedBefore” oferecidas pelo COPY INTO permitem que os usuários ingiram dados de uma janela de tempo escolhida em uma tabela Delta.

```
COPY INTO my_example_data
  FROM  's3://my-bucket/exampleDataPath'
  FILEFORMAT = CSV
  PATTERN = '*raw_data_*.csv'
  FORMAT_OPTIONS('header' = 'true', 'modifiedAfter' = '2022-09-
12T10:53:11,000+0000')
```

7. Corrigir dados com a opção de força

Como o COPY INTO é idempotente por padrão, a execução da mesma query nos mesmos arquivos de origem mais de uma vez não tem efeito sobre a tabela de destino após a execução inicial. Você deve propagar as alterações na tabela de destino porque, em circunstâncias reais, os arquivos de dados de origem no armazenamento de objetos na nuvem podem ser alterados para correção em um momento posterior. Nesse caso, é possível apagar primeiro os dados da tabela de destino antes de ingerir os arquivos de dados mais recentes da origem. Para essa operação, você só precisa definir a opção de cópia "force" como "true".

```
COPY INTO my_example_data
  FROM  's3://my-bucket/exampleDataPath'
  FILEFORMAT = CSV
  PATTERN = '*raw_data_2022*.csv'
  FORMAT_OPTIONS('header' = 'true')
  COPY_OPTIONS ('force' = 'true')
```

8. Aplicar transformações simples

E se você quiser renomear colunas? Ou se os dados de origem foram alterados e uma coluna anterior foi renomeada? Não seria bom inserir esses dados como duas colunas separadas, mas como uma única coluna. Podemos aproveitar o comando SELECT em COPY INTO para fazer transformações simples.

```
COPY INTO demo.my_example_data
  FROM ( SELECT  concat(first_name, " ", last_name) como full_name,
                 * EXCEPT (first_name, last_name)
    FROM 's3://my-bucket/exampleDataPath'
  )
  FILEFORMAT = CSV
  PATTERN = '*.csv'
  FORMAT_OPTIONS('header' = 'true')
  COPY_OPTIONS ('force' = 'true')
```

9. Tratamento de erros e observabilidade com COPY INTO

Tratamento de erros:
que tal ingerir dados com problemas de corrupção de arquivos? Exemplos comuns de corrupção de arquivos são:

- Arquivos com um formato de arquivo incorreto
- Falha ao descompactar
- Arquivos ilegíveis (por exemplo, Parquet inválido)

A opção de formato ignoreCorruptFiles de COPY INTO ajuda a ignorar esses arquivos durante o processamento. O resultado do comando COPY INTO retorna o número de arquivos ignorados na coluna num_skipped_corrupt_files. Além disso, esses arquivos corrompidos não são rastreados pelo estado de ingestão em COPY INTO, portanto, eles podem ser recarregados em uma execução subsequente uma vez que a corrupção é corrigida. Essa opção está disponível no Databricks Runtime 11.0+.

Você pode ver quais arquivos foram detectados como corrompidos executando COPY INTO no modo VALIDATE.

```
COPY INTO my_example_data
  FROM  's3://my-bucket/exampleDataPath'
  FILEFORMAT = CSV
  VALIDATE ALL
  FORMAT_OPTIONS('ignoreCorruptFiles' = 'true')
```

Observabilidade:
no Databricks Runtime 10,5, foi introduzida a coluna de metadados do arquivo para fornecer informações de metadados do arquivo de entrada, o que permite ao usuário monitorar e obter propriedades importantes dos arquivos ingeridos, como caminho, nome, tamanho e tempo de modificação, consultando uma coluna STRUCT oculta chamada _metadados. Para incluir essas informações no destino, você deve fazer referência explicitamente à coluna _metadados em sua query no COPY INTO.

```
COPY INTO my_example_data
  FROM (
     SELECT *, _metadata source_metadata FROM 's3://my-bucket/
exampleDataPath'
  )
  FILEFORMAT = CSV
```

Como ele se compara ao Auto Loader?

COPY INTO é um comando simples e poderoso para usar quando o diretório de origem contém um pequeno número de arquivos (ou seja, milhares de arquivos ou menos) e se você preferir SQL. Além disso, COPY INTO pode ser usado no JDBC para enviar dados para o Delta Lake como você achar melhor, um padrão comum de muitos parceiros de ingestão. Para ingerir um número maior de arquivos em streaming e em batch, recomendamos usar o Auto Loader. Além disso, para um pipeline de dados moderno baseado na arquitetura medallion, recomendamos usar o Auto Loader nos pipelines do Delta Live Tables, aproveitando os recursos avançados de tratamento automático de erros, controle de qualidade, linhagem de dados e definição de expectativas em uma abordagem declarativa.

Como começar?

Para começar, você pode acessar o editor de queries do **Databricks SQL**, atualizar e executar o exemplo de comandos SQL para ingerir a partir de seus armazenamentos de objetos em nuvem. Confira as opções no No. 4 para estabelecer acesso seguro aos seus dados para consultá-los no Databricks SQL. Para se familiarizar com o COPY INTO no Databricks SQL, você também pode seguir este tutorial de início rápido.

Como alternativa, você pode usar este notebook nos workspaces de Data Science e Data Engineering e Machine Learning para aprender a maioria dos recursos COPY INTO deste blog, em que os dados de origem e as tabelas Delta de destino são gerados no DBFS.

Veja mais tutoriais sobre COPY INTO aqui.

SEÇÃO 2.9

Simplificar a captura de dados de alterações com o Delta Live Tables da Databricks

por **MOJGAN MAZOUCHI**

25 de abril de 2022

Este guia demonstrará como aproveitar a captura de dados de alterações nos pipelines do Delta Live Tables para identificar novos registros e capturar alterações feitas no conjunto de dados em seu data lake. Os pipelines do Delta Live Tables permitem desenvolver pipelines de dados escaláveis, confiáveis e de baixa latência, enquanto executa a captura de dados de alterações em seu data lake com os recursos de compute mínimos necessários e o tratamento contínuo de dados fora de ordem.

> **Observação:** recomendamos seguir a Introdução ao Delta Live Tables, que explica como criar pipelines escaláveis e confiáveis usando o Delta Live Tables (DLT) e suas definições de ETL declarativas.

Histórico da captura de dados de alterações

A captura de dados de alterações (CDC) é um processo que identifica e captura alterações incrementais (exclusões, inserções e atualizações de dados) em bancos de dados, como rastreamento de status de clientes, pedidos ou produtos para aplicativos de dados quase em tempo real. O CDC proporciona a evolução dos dados em tempo real, processando-os de forma contínua e incremental à medida que ocorrem novos eventos.

Uma vez que mais de 80% das organizações planejam implementar estratégias multicloud até 2025, é fundamental escolher a abordagem certa para sua empresa que permita centralização perfeita em tempo real de todas as alterações de dados em seu pipeline de ETL em vários ambientes.

Ao capturar eventos de CDC, os usuários do Databricks podem materializar novamente a tabela de origem como Delta Table no Lakehouse e executar uma análise sobre ela, além de conseguir combinar dados com sistemas externos. O comando MERGE INTO no Delta Lake no Databricks permite que os clientes atualizem e excluam registros de forma eficiente em seus data lakes. Você pode saber mais sobre o tópico anterior aqui. Esse é um caso de uso comum em que observamos muitos dos clientes do Databricks usando o Delta Lakes para realizar e manter seus dados atualizados com dados de negócios em tempo real.

Embora o Delta Lake forneça uma solução completa para sincronização de CDC em tempo real em um data lake, agora temos o prazer de anunciar o recurso de captura de dados de alterações no Delta Live Tables que torna sua arquitetura ainda mais simples, eficiente e escalável. O DLT permite que os usuários ingiram dados de CDC sem problemas usando SQL e Python.

As soluções CDC anteriores com tabelas Delta usavam a operação MERGE INTO, que exige a ordenação manual dos dados para evitar falhas quando várias linhas do conjunto de dados de origem coincidem ao tentar atualizar as mesmas

linhas da tabela Delta de destino. Para lidar com os dados fora de ordem, era necessária uma etapa extra para pré-processar a tabela de origem usando uma implementação de ForEachBatch para eliminar a possibilidade de várias correspondências, mantendo somente a alteração mais recente para cada chave (veja o exemplo de captura dados de alterações). A nova operação APPLY CHANGES INTO em pipelines DLT lida de forma automática e perfeita com dados fora de ordem, sem a necessidade de intervenção manual de data engineering .

CDC com Delta Live Tables do Databricks

Neste blog, demonstraremos como usar o comando APPLY CHANGES INTO nos pipelines do Delta Live Tables para um caso de uso comum do CDC em que os dados do CDC vêm de um sistema externo. Há uma variedade de ferramentas de CDC disponível, como Debezium, Fivetran, Qlik Replicate, Talend e StreamSets. Embora as implementações específicas sejam diferentes, essas ferramentas geralmente capturam e registram o histórico de alterações de dados nos registros; os aplicativos downstream consomem esses registros do CDC. Em nosso exemplo, os dados são colocados no armazenamento de objetos na nuvem a partir de uma ferramenta CDC, como Debezium, Fivetran etc.

Temos dados de várias ferramentas CDC que entram em um armazenamento de objetos na nuvem ou em uma fila de mensagens como o Apache Kafka. Normalmente, o CDC é usado em uma ingestão do que chamamos de arquitetura medallion. A arquitetura medallion se refere ao design de dados usado para organizar logicamente os dados em um Lakehouse, que visa melhorar de forma incremental e progressiva a estrutura e a qualidade dos dados à medida que fluem por camada da arquitetura. O Delta Live Tables permite aplicar perfeitamente as alterações dos feeds do CDC às tabelas do seu Lakehouse; a combinação dessa funcionalidade com a arquitetura medallion permite que as alterações incrementais fluam facilmente pelas cargas de trabalho analíticas em escala. O uso do CDC em conjunto com a arquitetura medallion oferece vários benefícios aos usuários, já que só é preciso processar dados alterados ou adicionados. Assim, ele permite que os usuários mantenham as tabelas Gold atualizadas de forma econômica com os dados de negócios mais recentes.

> **OBSERVAÇÃO:** o exemplo aqui se aplica às versões SQL e Python do CDC e também sobre uma forma específica de usar as operações; para avaliar variações, consulte a documentação oficial aqui.

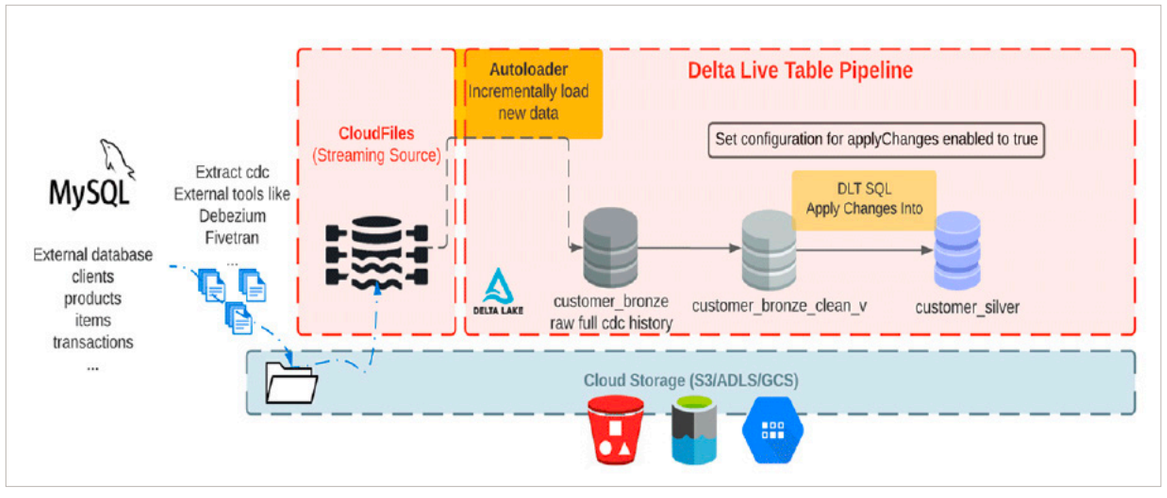
Pré-requisitos

Para aproveitar ao máximo este guia, você deve conhecer pelo menos o básico sobre:

- SQL ou Python
- Delta Live Tables
- Desenvolvimento de pipelines ETL e/ou trabalho com sistemas de Big Data
- Notebooks e clusters interativos da Databricks
- Você deve ter acesso a um Databricks Workspace com permissões para criar novos clusters, executar jobs e salvar dados em um local no armazenamento externo de objetos na nuvem ou no DBFS
- Para o pipeline que estamos criando neste blog, é preciso selecionar a edição “Advanced” do produto, que oferece suporte à aplicação de restrições de qualidade de dados

Conjunto de dados

Aqui estamos consumindo dados de CDC realistas de um banco de dados externo. Neste pipeline, usaremos a biblioteca Faker para gerar o conjunto de dados que uma ferramenta CDC como o Debezium pode produzir e trazer para o armazenamento em nuvem para a ingestão inicial no Databricks. Usando o Auto Loader, carregamos incrementalmente as mensagens do armazenamento de objetos na nuvem e as armazenamos na tabela Bronze à medida que armazena as mensagens brutas. As tabelas Bronze destinam-se à ingestão de dados que permitem o acesso rápido a uma única fonte de verdade. Em seguida, executamos APPLY CHANGES INTO da tabela de camada Bronze limpa para propagar as atualizações downstream para a tabela Silver. À medida que os dados fluem para as tabelas Silver, geralmente eles se tornam mais refinados e otimizados ("apenas o suficiente") para fornecer a uma empresa uma visão de todas as suas principais entidades de negócios. Veja o diagrama abaixo.



Este blog tem por foco um exemplo simples que exige uma mensagem JSON com quatro campos de nome, e-mail, endereço e id do cliente juntamente com os dois campos: operation (que armazena o código de operação (DELETE, APPEND, UPDATE, CREATE) e operation_date (que armazena a data e o timestamp para o registro de cada ação de operação) para descrever os dados alterados.

Para gerar um conjunto de dados de amostra com os campos acima, estamos usando um pacote Python que gera dados falsos, Faker. Você pode encontrar o notebook relacionado a esta seção de geração de dados aqui. Neste notebook, fornecemos o nome e o local de armazenamento para gravar os dados gerados lá. Estamos usando a funcionalidade DBFS do Databricks; consulte a documentação DBFS para saber mais sobre como funciona. Em seguida, usamos uma função do PySpark definida pelo usuário para gerar o conjunto de dados sintéticos para cada campo e gravamos os dados de volta ao local de armazenamento definido, que iremos nos referir em outros notebooks para acessar o conjunto de dados sintéticos.

Ingerir o conjunto de dados brutos usando o Auto Loader

De acordo com o paradigma da arquitetura medallion, a camada Bronze mantém a qualidade dos dados brutos. Neste estágio, podemos ler incrementalmente novos dados usando o Auto Loader de um local no armazenamento em nuvem. Aqui estamos adicionando o caminho para nosso conjunto de dados gerado à seção de configuração nos ajustes do pipeline, o que nos permite carregar o caminho de origem como uma variável. Portanto, agora nossa configuração nos ajustes do pipeline fica assim:

```
"configuration": {
"source": "/tmp/demo/cdc_raw"
}
```

Depois, carregamos esta propriedade de configuração em nossos notebooks.

Vamos dar uma olhada na tabela Bronze que iremos ingerir, a. Em SQL, e b. Usando Python

A. SQL

```
SET spark.source;
CREATE STREAMING LIVE TABLE customer_bronze
(
address string,
email string,
id string,
firstname string,
lastname string,
operation string,
operation_date string,
_rescued_data string
)
TBLPROPERTIES ("quality" = "bronze")
COMMENT "Novos dados de clientes ingeridos de forma incremental a partir
da zona de destino do armazenamento de objetos na nuvem"
AS
SELECT *
FROM cloud_files("${source}/customers", "json", map("cloudFiles.
inferColumnTypes", "true"));
```

B. PYTHON

```
import dlt
from pyspark.sql.functions import *
from pyspark.sql.types import *

source = spark.conf.get("source")

@dlt.table(name="customer_bronze",
                  comment = "Novos dados do cliente ingeridos
incrementalmente da zona de aterrissagem do armazenamento de objetos na
nuvem",
  table_properties={
    "quality": "bronze"
  }
)
def customer_bronze():
  return (
    spark.readStream.format("cloudFiles") \
      .option("cloudFiles.format", "json") \
      .option("cloudFiles.inferColumnTypes", "true") \
      .load(f"{source}/customers")
  )
```

As declarações acima usam o Auto Loader para criar uma tabela de streaming ao vivo chamada customer_bronze a partir de arquivos json. Ao usar o Auto Loader no Delta Live Tables, você não precisa fornecer nenhum local para esquema ou ponto de verificação, pois esses locais serão gerenciados automaticamente pelo pipeline DLT.

O Auto Loader fornece uma fonte de Structured Streaming chamada cloud_files no SQL e cloudFiles no Python, que usa um caminho e um formato de armazenamento em nuvem como parâmetros.

Para reduzir os custos de compute, recomendamos executar o pipeline DLT no modo Triggered como um micro-batch, supondo que você não tenha requisitos de latência muito baixa.

Expectativas e dados de alta qualidade

Na próxima etapa para criar um conjunto de dados de alta qualidade, diversificado e acessível, impomos critérios de expectativa de verificação de qualidade usando restrições. Atualmente, uma restrição pode ser reter, soltar ou falhar. Para mais detalhes, consulte aqui. Todas as restrições são registradas para permitir um monitoramento de qualidade simplificado.

A. SQL

```
CREATE TEMPORARY STREAMING LIVE TABLE customer_bronze_clean_v(
  CONSTRAINT valid_id EXPECT (id IS NOT NULL) ON VIOLATION DROP ROW,
  CONSTRAINT valid_address EXPECT (address IS NOT NULL),
  CONSTRAINT valid_operation EXPECT (operation IS NOT NULL) ON VIOLATION
DROP ROW
)
TBLPROPERTIES ("quality" = "silver")
COMMENT "Visualização do cliente Bronze limpa (ou seja, o que se tornará
Silver)"
AS SELECT *
FROM STREAM(LIVE.customer_bronze);
```

B. PYTHON

```
@dlt.view(name="customer_bronze_clean_v",
  comment="Visualização do cliente Bronze limpa (ou seja, o que se tornará
Silver)")

@dlt.expect_or_drop("valid_id", "id IS NOT NULL")
@dlt.expect("valid_address", "address IS NOT NULL")
@dlt.expect_or_drop("valid_operation", "operation IS NOT NULL")

def customer_bronze_clean_v():
  return dlt.read_stream("customer_bronze") \
            .select("address", "email", "id", "firstname", "lastname",
"operation", "operation_date", "_rescued_data")
```

Usar a declaração APPLY CHANGES INTO para propagar alterações na tabela de destino downstream

Antes de executar a consulta Apply Changes Into, devemos garantir que exista uma tabela de streaming de destino na qual queremos manter os dados mais atualizados. Se não existir, precisamos criar uma. As células abaixo são exemplos de criação de uma tabela de streaming de destino. Observe que, no momento da publicação deste blog, a instrução de criação da tabela de streaming de destino é necessária junto com a consulta Apply Changes Into e ambas precisam estar presentes no pipeline — caso contrário, haverá falha na query de criação da tabela.

A. SQL

```
CREATE STREAMING LIVE TABLE customer_silver
TBLPROPERTIES ("quality" = "silver")
COMMENT "Clientes limpos e combinados";
```

B. PYTHON

```
dlt.create_target_table(name="customer_silver",
  comment="Clientes limpos e mesclados",
  table_properties={
    "quality": "silver"
  }
)
```

Agora que temos uma tabela de streaming de destino, podemos propagar alterações na tabela de destino downstream usando a consulta Apply Changes Into. Enquanto o feed do CDC vem com eventos INSERT, UPDATE e DELETE, o comportamento padrão da DLT é aplicar eventos INSERT e UPDATE de qualquer registro no conjunto de dados de origem correspondente em chaves primárias e sequenciado por um campo que identifica a ordem dos eventos. Mais especificamente, ele atualiza qualquer linha na tabela de destino existente que corresponda às chaves primárias ou insere uma nova linha quando um registro correspondente não existe na tabela de streaming de destino. Podemos usar APPLY AS DELETE WHEN no SQL ou seu argumento apply_as_deletes equivalente em Python para manipular eventos DELETE.

Neste exemplo, usamos "id" como minha chave primária, que identifica exclusivamente os clientes e permite que os eventos do CDC sejam aplicados aos registros de clientes identificados na tabela de streaming de destino. Como "operation_date" mantém a ordem lógica dos eventos do CDC no conjunto de dados de origem, usamos "SEQUENCE BY operation_date" no SQL, ou seu equivalente "sequence_by = col("operation_date")" no Python para lidar com eventos de mudança que chegam fora de ordem. Lembre-se de que o valor de campo que usamos com SEQUENCE BY (ou sequence_by) deve ser exclusivo entre todas as atualizações da mesma chave. Na maioria dos casos, a sequência por coluna será uma coluna com informações de timestamp.

Por fim, usamos "COLUMNS * EXCEPT (operation, operation_date, _rescued_data)" no SQL ou seu equivalente "except_column_list"= ["operation", "operation_date", "_rescued_data"] no Python para excluir três colunas de "operation", "operation_date", "_rescued_data" da tabela de streaming de destino. Por padrão, todas as colunas são incluídas na tabela de streaming de destino, quando não especificamos a cláusula "COLUMNS".

A. SQL

```
APPLY CHANGES INTO LIVE.customer_silver
FROM stream(LIVE.customer_bronze_clean_v)
  KEYS (id)
  APPLY AS DELETE WHEN operation = "DELETE"
  SEQUENCE BY operation_date
  COLUMNS * EXCEPT (operation, operation_date,
_rescued_data);
```

B. PYTHON

```
dlt.apply_changes(
  target = "customer_silver",
  source = "customer_bronze_clean_v",
  keys = ["id"],
  sequence_by = col("operation_date"),
  apply_as_deletes = expr("operation = 'DELETE'"),
  except_column_list = ["operation", "operation_date", "_rescued_data"])
```

Para conferir a lista completa de cláusulas disponíveis, consulte aqui.

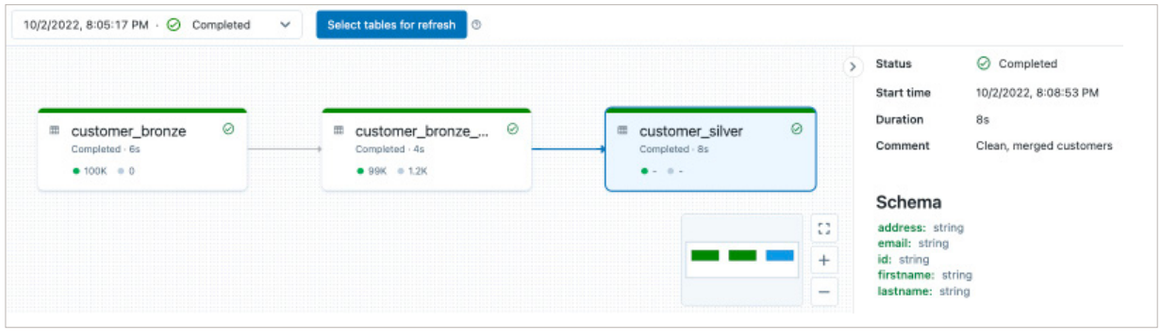
Observe que, no momento da publicação deste blog, uma tabela que leia a partir do destino de uma query APPLY CHANGES INTO ou função apply_changes deve ser uma tabela ao vivo e não pode ser uma tabela de streaming ao vivo.

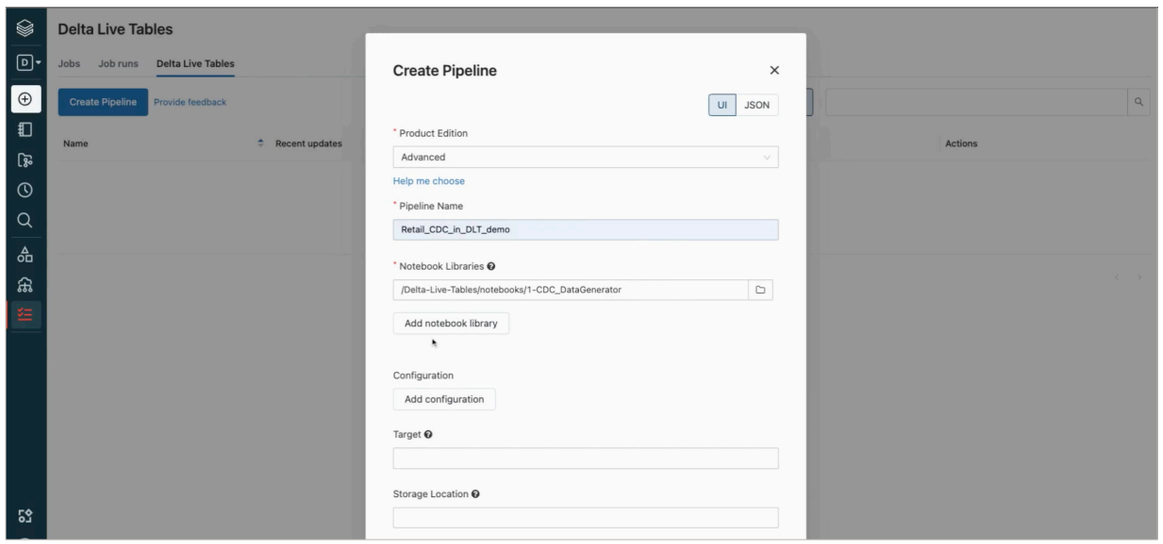
Um notebook SQL e Python está disponível para referência nesta seção. Agora que temos todas as células prontas, vamos criar um pipeline para ingerir dados do armazenamento de objetos em nuvem. Abra Jobs em uma nova tab ou janela no workspace e selecione "Delta Live Tables".

O pipeline associado a este blog tem as seguintes configurações de pipeline DLT:

```
{
     "clusters": [
        {
            "label": "default",
            "num_workers": 1
        }
    ],
    "development": true,
    "continuous": false,
    "edition": "advanced",
    "photon": false,
    "libraries": [
        {
            "notebook": {
"path":"/Repos/mojgan.mazouchi@databricks.com/Delta-Live-Tables/
notebooks/1-CDC_DataGenerator"
            }
        },
        {
            "notebook": {
"path":"/Repos/mojgan.mazouchi@databricks.com/Delta-Live-Tables/
notebooks/2-Retail_DLT_CDC_sql"
            }
        }
    ],
    "name": "CDC_blog",
    "storage": "dbfs:/home/mydir/myDB/dlt_storage",
    "configuration": {
        "source": "/tmp/demo/cdc_raw",
        "pipelines.applyChangesPreviewEnabled": "true"
    },
    "target": "my_database"
}
```

1. Selecione “Criar pipeline” para criar um novo pipeline
2. Especifique um nome, como “Pipeline de CDC de Varejo”
3. Especifique os caminhos do notebook que você já criou anteriormente, um para o conjunto de dados gerado usando o pacote Faker e outro para a ingestão dos dados gerados no DLT. O segundo caminho do notebook pode se referir ao notebook escrito em SQL ou Python, dependendo da linguagem de sua escolha.
4. Para acessar os dados gerados no primeiro notebook, adicione o caminho do conjunto de dados na configuração. Aqui armazenamos os dados em “/tmp/demo/cdc_raw/customers”, depois definimos “source” como “/tmp/demo/cdc_raw/” para referenciar “source/customers” em nosso segundo notebook.”
5. Especifique o Target (que é opcional e se refere ao banco de dados de destino), onde você pode consultar as tabelas resultantes do seu pipeline
6. Especifique o local de armazenamento no seu armazenamento de objetos (opcional) para acessar os conjuntos de dados produzidos pelo DLT e os registros de metadados do seu pipeline
7. Defina o Modo de pipeline como Acionado. No modo Acionado, o pipeline DLT consumirá novos dados na origem de uma só vez e, uma vez concluído o processamento, encerrará o recurso de compute automaticamente. Você pode alternar entre os modos Acionado e Contínuo ao editar as configurações de pipeline. Definir “continuous”: false no JSON equivale a definir o pipeline para o modo Acionado.
8. Para esta carga de trabalho, você pode desabilitar o dimensionamento automático em Opções do Autopilot e usar somente um cluster de worker. Para cargas de trabalho de produção, recomendamos habilitar o dimensionamento automático e definir o número máximo de workers necessários para o tamanho do cluster.
9. Selecione “Iniciar”
10. Seu pipeline já está criado e funcionando!





Observabilidade da linhagem do pipeline DLT e monitoramento da qualidade dos dados

Todos os logs do pipeline DLT são armazenados no local de armazenamento do pipeline. Você pode especificar o local de armazenamento somente quando estiver criando seu pipeline. Observe que, após a criação do pipeline, você não poderá mais modificar o local de armazenamento.

Você pode conferir nossa análise aprofundada anterior sobre o assunto aqui. Experimente este notebook para ver a observabilidade do pipeline e o monitoramento da qualidade dos dados no exemplo de pipeline DLT associado a este blog.

Conclusão

Neste blog, mostramos como facilitamos para os usuários a implementação eficiente da captura de dados de alterações (CDC) em sua plataforma de lakehouse com o Delta Live Tables (DLT). O DLT oferece controles de qualidade integrados com visibilidade profunda das operações do pipeline, observando a linhagem do pipeline, o esquema de monitoramento e as verificações de qualidade em cada etapa do pipeline. O DLT oferece suporte ao tratamento automático de erros e ao melhor recurso de dimensionamento automático da categoria para cargas de trabalho de streaming, o que permite que os usuários tenham dados de qualidade com os recursos ideais necessários para a carga de trabalho.

Agora os engenheiros de dados podem implementar facilmente o CDC com uma nova APPLY CHANGES INTO API declarativa com DLT em SQL ou Python. Esse novo recurso permite que seus pipelines ETL identifiquem facilmente as alterações e apliquem essas alterações em dezenas de milhares de tabelas com suporte de baixa latência.

> **Pronto para começar e experimentar você mesmo o CDC no Delta Live Tables?**
> Assista a este webinar para saber como o Delta Live Tables simplifica a complexidade da transformação de dados e do ETL e leia o documento Captura de dados de alterações com o Delta Live Tables, github oficial e siga as etapas deste vídeo para criar seu pipeline!

SEÇÃO 2.10

Práticas recomendadas para compartilhamento de dados entre governos

por **MILOS COLIC, PRITESH PATEL, ROBERT WHIFFIN, RICHARD JAMES WILSON, MARCELL FERENCZ** e **EDWARD KELLY**

21 de fevereiro de 2023

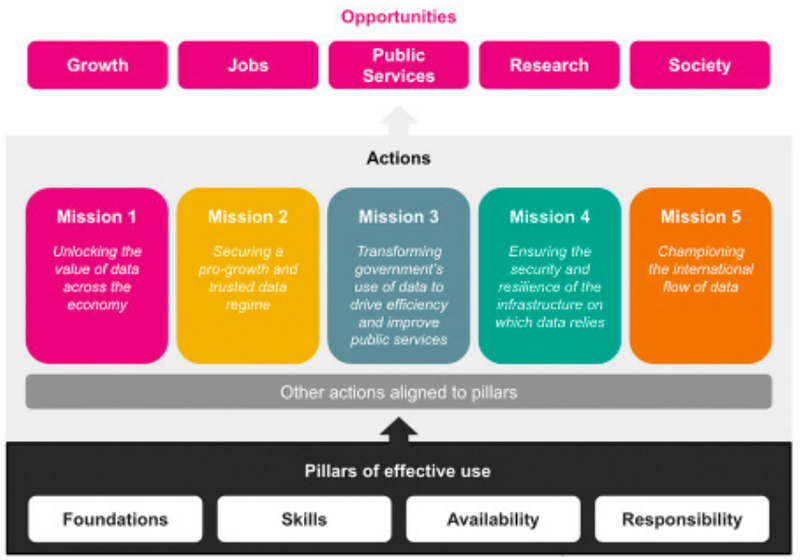
A troca de dados governamentais é a prática de compartilhar dados entre diferentes órgãos governamentais e, muitas vezes, parceiros em setores comerciais. O governo pode compartilhar dados por várias razões, como melhorar a eficiência das operações governamentais, fornecer melhores serviços ao público ou apoiar a pesquisa e a formulação de políticas. Além disso, a troca de dados no setor público pode envolver o compartilhamento com o setor privado ou o recebimento de dados do setor privado. As considerações abrangem várias jurisdições e em quase todos os setores. Neste blog, abordaremos as necessidades divulgadas como parte das estratégias de dados nacionais e como as tecnologias modernas, particularmente Delta Sharing, Unity Catalog e clean rooms, podem ajudar você a projetar, implementar e gerenciar um ecossistema de dados sustentável e preparado para o futuro.

Compartilhamento de dados e setor público

"O milagre é este: quanto mais compartilhamos, mais temos." — Leonard Nimoy."

Esta provavelmente é a citação sobre compartilhamento que se aplica mais profundamente ao tópico de compartilhamento de dados, na medida em que o objetivo de compartilhar os dados é criar novas informações, novos insights e novos dados. A importância do compartilhamento de dados é ainda mais acentuada no contexto governamental, onde a federação entre departamentos permite maior foco. Ainda assim, a mesma federação apresenta desafios em relação à integridade, qualidade, acesso, segurança, controle, equidade dos dados etc. Esses desafios estão longe de ser triviais e exigem uma abordagem estratégica e multifacetada para serem abordados de forma adequada. Tecnologia, pessoas, processos, estruturas legais, entre outros, exigem consideração dedicada ao projetar um ecossistema robusto de compartilhamento de dados.

A National Data Strategy (NDS) do governo do Reino Unido descreve cinco missões acionáveis por meio das quais podemos materializar o valor dos dados para benefício do cidadão e de toda a sociedade.



databricks

Não surpreende que cada uma das missões esteja fortemente relacionada ao conceito de compartilhamento de dados ou, mais amplamente, ao acesso a dados dentro e fora dos departamentos governamentais:

1. **Desbloquear o valor dos dados em toda a economia:** a missão principal da NDS é afirmar o governo e os reguladores como facilitadores da extração de valor dos dados por meio da adoção das práticas recomendadas. Estima-se que a economia de dados do Reino Unido esteja próxima de £125 bilhões em 2021 com uma tendência ascendente. Nesse contexto, é essencial entender que os dados abertos coletados pelo governo e fornecidos podem ser cruciais para enfrentar muitos dos desafios em todos os setores.

   Por exemplo, as seguradoras podem avaliar melhor o risco de garantir propriedades, ingerindo e integrando áreas de inundação fornecidas pela DEFRA. Por outro lado, os investidores do mercado de capitais poderiam entender melhor o risco de seus investimentos, ingerindo e integrando o Índice de taxa de inflação pela ONS. Por outro lado, é crucial que os reguladores tenham padrões bem definidos de acesso a dados e compartilhamento de dados para conduzir suas atividades regulatórias. Essa clareza realmente empodera os atores econômicos que interagem com os dados do governo.

2. **Garantir um regime de dados pró-crescimento e confiável:** o principal aspecto da segunda missão é a confiança nos dados ou, mais amplamente, a adesão às normas de qualidade dos dados. As considerações de qualidade de dados são ainda mais relevantes para casos de uso de compartilhamento e troca de dados em que levamos em conta todo o ecossistema de uma só vez, e as implicações de qualidade transcendem os limites da nossa própria plataforma. É justamente por isso que temos que adotar a "sustentabilidade de dados". O que queremos dizer com produtos de dados sustentáveis são produtos de dados que aproveitam as fontes existentes em detrimento da reinvenção dos mesmos/similares ativos, acúmulo de dados desnecessários (poluentes de dados) e que antecipam usos futuros.

   O compartilhamento de dados não controlado e ilimitado pode afetar negativamente a qualidade dos dados e prejudicar o crescimento e o valor dos dados. A qualidade de como os dados são compartilhados deve ser uma consideração fundamental das estruturas de qualidade de dados. Por essa razão, exigimos um conjunto sólido de padrões e práticas recomendadas para o compartilhamento de dados com governança e garantia de qualidade embutidas no processo e nas tecnologias. Só assim podemos garantir a sustentabilidade dos nossos dados e garantir um regime de dados de confiança pró-crescimento.

3. **Transformar o uso de dados pelo governo para impulsionar a eficiência e melhorar os serviços públicos:** — "Até 2025, os ativos de dados serão organizados e tratados como produtos, independentemente de serem usados por equipes internas ou clientes externos... Os produtos de dados evoluem continuamente de maneira ágil para atender às necessidades dos consumidores... esses produtos fornecem soluções de dados que podem ser usadas de maneira mais fácil e repetida para atender a vários desafios de negócios e reduzir o tempo e o custo do fornecimento de novos recursos orientados por IA." — A empresa orientada por dados de 2025, pela McKinsey. A IA e o ML podem ser poderosos facilitadores da transformação digital tanto para os setores público como privado.

   IA, ML, relatórios e painéis são apenas alguns exemplos de produtos e serviços de dados que extraem valor dos dados. A qualidade dessas soluções reflete-se diretamente na qualidade dos dados usados para construí-las e na nossa capacidade de acessar e aproveitar os ativos de dados disponíveis, tanto interna como externamente. Embora exista uma grande quantidade de dados disponíveis para construirmos novas soluções inteligentes para impulsionar a eficiência para melhores processos, melhor tomada de decisões e melhores políticas, também há inúmeras barreiras que podem prender os dados, como sistemas legados, silos de dados, padrões fragmentados, formatos proprietários etc. Modelar soluções de dados como produtos de dados e padronizá-los em um formato unificado nos permite abstrair essas barreiras e aproveitar verdadeiramente o ecossistema de dados.

4. **Garantir a segurança e a resiliência da infraestrutura da qual os dados dependem:** pensando na visão do ano 2025 — isso não está muito longe do presente e mesmo em um futuro não tão distante, seremos obrigados a repensar nossa abordagem aos dados, mais especificamente — qual é a nossa infraestrutura de cadeia de fornecimento digital/infraestrutura de compartilhamento de dados? Dados e ativos de dados são produtos e devem ser gerenciados como tais.

   Se os dados são um produto, precisamos de uma forma coerente e unificada de fornecer esses produtos. Se os dados forem usados em todos os setores e nos setores público e privado, precisamos de um protocolo aberto que impulsione a adoção e a geração de hábitos. Para motivar a adoção, as tecnologias que usamos devem ser resilientes, robustas, confiáveis e utilizáveis por/para todos. A dependência do fornecedor, da plataforma ou da nuvem são limites para concretizar essa visão.

5. **Promover o fluxo internacional de dados:** — a troca de dados entre jurisdições e governos provavelmente será uma das aplicações mais transformadoras dos dados em escala. Alguns dos desafios mais difíceis do mundo dependem da troca eficiente de dados entre governos — prevenção de atividades criminosas, atividades contraterroristas, metas de emissão não zero, comércio internacional, e essa lista vai longe. Algumas etapas nesta direção já são realidade: o governo federal dos EUA e o governo do Reino Unido concordaram em trocar dados para combater atividades criminais graves. Esse é um exemplo real de promoção de dados de fluxo internacional e uso de dados para o bem. É imperativo que, para esses casos de uso, abordemos o compartilhamento de dados de um ângulo de segurança em primeiro lugar. Os padrões e protocolos de compartilhamento de dados precisam aderir às práticas recomendadas de segurança e privacidade.

Embora originalmente construído com foco no governo do Reino Unido e em como integrar melhor os dados como um ativo-chave de um governo moderno, esses conceitos se aplicam em um contexto mais amplo do setor público global. No mesmo sentido, o governo federal dos EUA propôs a Estratégia Federal de Dados como uma coleção de princípios, práticas, etapas de ação e cronograma através dos quais o governo pode aproveitar todo o valor dos dados federais para missão, serviço e bem público.

Os princípios são agrupados em três tópicos principais:

- **Governança ética** — Dentro do domínio da ética, o compartilhamento de dados é uma ferramenta fundamental para promover transparência, responsabilidade e explicabilidade da tomada de decisões. É praticamente impossível manter a ética sem alguma forma de auditoria conduzida por uma parte independente. A troca de dados (e metadados) é um facilitador crítico para processos robustos contínuos que garantem que estamos usando os dados para o bem e que estamos usando os dados em que podemos confiar.
- **Design consciente** — Esses princípios estão fortemente alinhados com a ideia de sustentabilidade de dados. As diretrizes promovem o pensamento futuro em relação à usabilidade e interoperabilidade dos dados e princípios de design centrados no usuário de produtos de dados sustentáveis.
- **Cultura de aprendizagem** — O compartilhamento de dados, ou compartilhamento de conhecimento alternativo, tem um papel importante na criação de um ecossistema de aprendizagem escalável e cultura de aprendizagem. Os dados são a frente e o centro da síntese do conhecimento e, a partir de um ângulo científico, os dados comprovam o conhecimento factual. Outro componente crítico do conhecimento é o "Por quê?", e precisamos dos dados para abordar esse componente de qualquer decisão que tomamos, qual política aplicar, quem sancionar, quem apoiar com concessões, como melhorar a eficiência dos serviços governamentais, como atender melhor aos cidadãos e à sociedade.

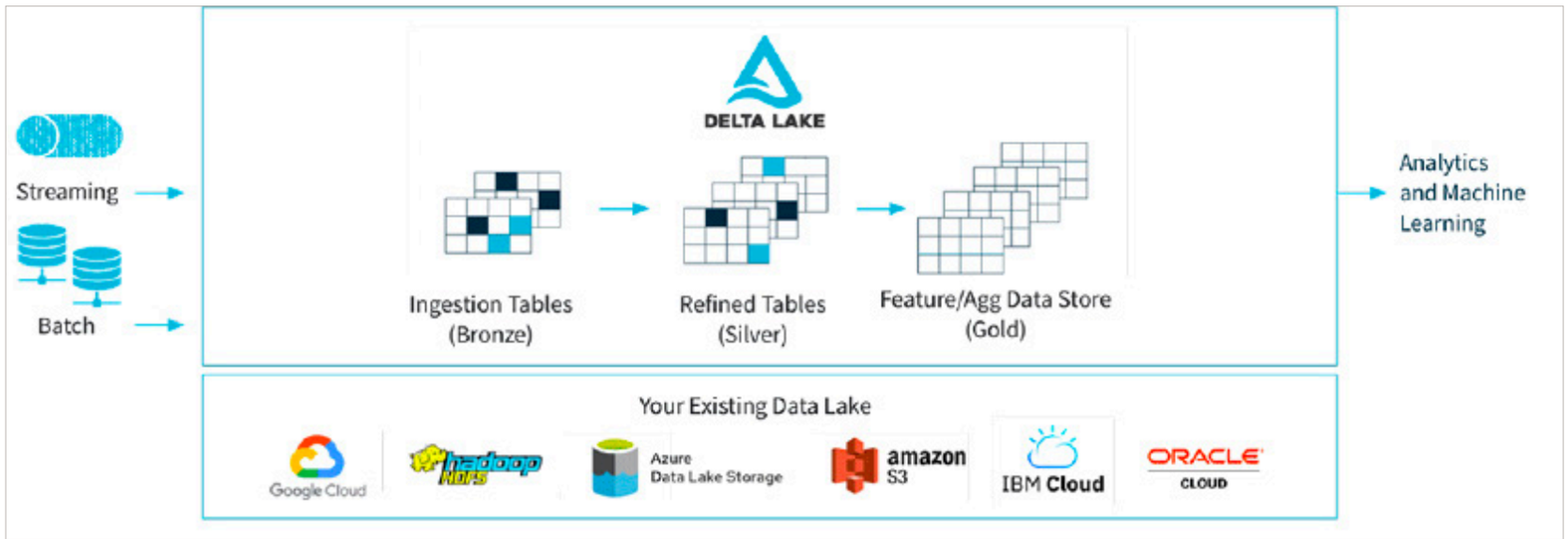
Em contraste com a análise qualitativa discutida anteriormente do valor do compartilhamento de dados entre governos, a Comissão Europeia prevê que o valor econômico da economia de dados europeia exceda €800 bilhões até 2027 — aproximadamente o mesmo tamanho da economia holandesa em 2021! Além disso, eles preveem mais de 10 milhões de profissionais de dados somente na Europa. A tecnologia e a infraestrutura para dar suporte à sociedade de dados precisam ser acessíveis a todos, interoperáveis, extensíveis, flexíveis e abertas. Imagine um mundo em que você precisa de um caminhão diferente para transportar produtos entre diferentes armazéns, pois cada estrada exige um conjunto diferente de pneus — toda a cadeia de suprimentos entraria em colapso. Quando se trata de dados, muitas vezes vivenciamos o paradoxo "um conjunto de pneus para uma única estrada". Foram propostos APIs e protocolos de troca de dados de repouso, mas não conseguiram atender à necessidade de simplicidade, facilidade de uso e custo de expansão com o número de produtos de dados.

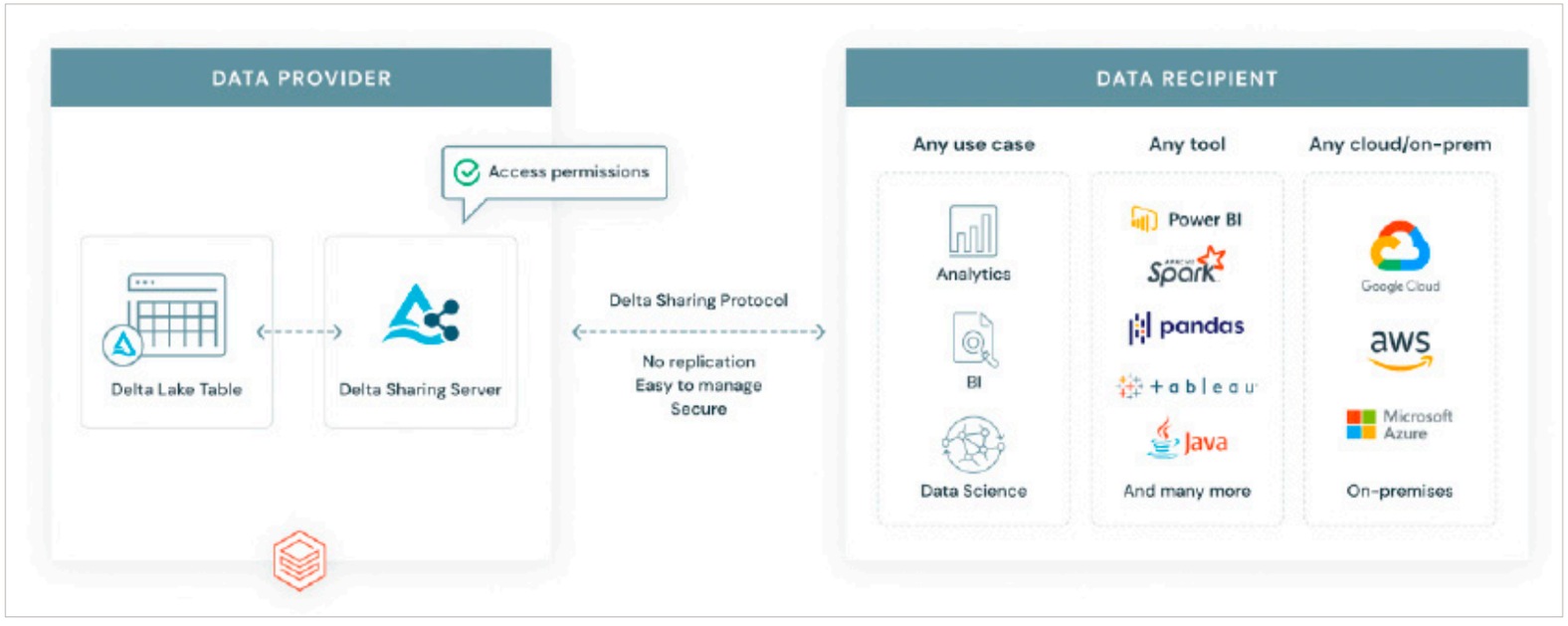
Delta Sharing — a nova estrada de dados

O Delta Sharing oferece um protocolo aberto para o compartilhamento seguro de dados em qualquer plataforma de compute. O protocolo é baseado no formato de dados Delta e não depende da nuvem que você escolher.

Delta é um formato de dados de código aberto que evita a dependência de fornecedores, plataformas e nuvem, aderindo totalmente aos princípios de sustentabilidade de dados, design consciente da Estratégia Federal de Dados dos EUA e missão 4 da estratégia nacional de dados do Reino Unido.

O Delta fornece uma camada de governança sobre o formato de dados do Parquet. Além disso, ele oferece muitas otimizações de desempenho não disponíveis no Parquet, prontas para uso. A abertura do formato de dados é uma consideração crítica. É o principal fator para impulsionar a geração de hábitos e a adoção das práticas e dos padrões recomendados.





databricks

Delta Sharing é um protocolo baseado em um conjunto enxuto de APIs REST para gerenciar o compartilhamento, as permissões e o acesso a qualquer ativo de dados armazenado nos formatos Delta ou Parquet. O protocolo define dois atores principais, o provedor de dados (fornecedor de dados, proprietário de dados) e o destinatário de dados (consumidor de dados). O destinatário, por definição, é independente do formato dos dados na origem. O Delta Sharing fornece as abstrações necessárias para acesso controlado a dados em diversas linguagens e ferramentas.

O Delta Sharing está posicionado exclusivamente para responder a muitos dos desafios do compartilhamento de dados de maneira escalonável dentro do contexto de domínios altamente regulamentados, como o setor público:

- **Preocupações de privacidade e segurança** — Dados pessoais identificáveis ou dados de outra forma confidenciais ou restritos são uma parte importante das necessidades de troca de dados de um governo orientado por dados e modernizado. Devido à natureza sensível desses dados, é fundamental que a governança do compartilhamento de dados seja mantida de forma coerente e unificada. Qualquer processo desnecessário e complexidade tecnológica aumentam o risco de compartilhamento excessivo de dados. Com isso em mente, o Data Sharing foi projetado com práticas recomendadas de segurança desde o início. O protocolo fornece criptografia de ponta a ponta, credenciais de curta duração e recursos de auditoria e governança acessíveis e intuitivos. Todos esses recursos estão disponíveis de forma centralizada em todas as suas tabelas Delta em todas as nuvens.

- **Qualidade e precisão** — Outro desafio do compartilhamento de dados é garantir que os dados compartilhados sejam de alta qualidade e precisão. Como os dados subjacentes são armazenados como tabelas Delta, podemos garantir que a natureza transacional dos dados seja respeitada; o Delta garante as propriedades ACID dos dados. Além disso, o Delta oferece suporte a restrições de dados para garantir os requisitos de qualidade dos dados no armazenamento. Infelizmente, outros formatos, como CSV, CSVW, ORC, Avro, XML etc., não têm essas propriedades sem um esforço adicional significativo. A questão se torna ainda mais relevante pelo fato de que a qualidade dos dados não pode ser garantida da mesma forma no lado do provedor e do destinatário dos dados sem a reimplementação exata dos sistemas de origem. É fundamental incorporar a qualidade e os metadados aos dados para garantir que a qualidade acompanhe os dados. Qualquer abordagem dissociada para gerenciar dados, metadados e qualidade separadamente aumenta o risco de compartilhamento e pode levar a resultados indesejáveis.

- **Falta de padronização** — Outro desafio do compartilhamento de dados é a falta de padronização em como os dados são coletados, organizados e armazenados. Isso é particularmente pronunciado no contexto de atividades governamentais. Embora os governos tenham proposto formatos padrão (por exemplo, o Office for National Statistics promove o uso do CSVW), alinhar todas as empresas do setor público e privado com os padrões propostos por essas iniciativas é um grande desafio. Outros setores podem ter requisitos diferentes de escalabilidade, interoperabilidade, complexidade de formato, falta de estrutura em dados etc. A maioria dos padrões atualmente defendidos não tem vários desses aspectos. O Delta é o candidato mais maduro para assumir a função central na padronização do formato de troca de dados. Ele foi criado como um formato de dados transacional e escalável, aceita dados estruturados, semiestruturados e não estruturados, armazena esquemas de dados e metadados junto com os dados e fornece um protocolo de compartilhamento escalonável de nível empresarial por meio do Delta Sharing. Por fim, o Delta é um dos projetos de código aberto mais populares do ecossistema e, desde maio de 2022, ultrapassou 7 milhões de downloads mensais.

- **Barreiras culturais e organizacionais** — Esses desafios podem ser resumidos em uma palavra: atrito. Infelizmente, é um problema comum para os funcionários públicos lutarem para obter acesso a dados internos e externos devido a processos complicados, políticas e padrões desatualizados. Os princípios que estamos usando para construir nossas plataformas de dados e nossas plataformas de compartilhamento de dados precisam ser autopromocionais, impulsionar a adoção e gerar hábitos que sigam as práticas recomendadas.

  Se houver atrito com a adoção de padrões, a única maneira de garantir que os padrões sejam respeitados é por meio da aplicação, o que, por si só, é mais uma barreira para alcançar a sustentabilidade dos dados. As organizações já adotaram o Delta Sharing nos setores público e privado. Por exemplo, o U.S. Citizenship and Immigration Services (USCIS) usa o Delta Sharing para atender a vários requisitos de compartilhamento de dados entre agências. Da mesma forma, a Nasdaq descreve o Delta Sharing como o "futuro do compartilhamento de dados financeiros", e esse futuro é aberto e governado.

- **Desafios técnicos** — A Federação em escala governamental ou ainda mais em vários setores e regiões geográficas representa desafios técnicos. Cada organização dentro dessa federação tem sua plataforma e impulsiona escolhas de tecnologia, arquitetura, plataforma e ferramentas.

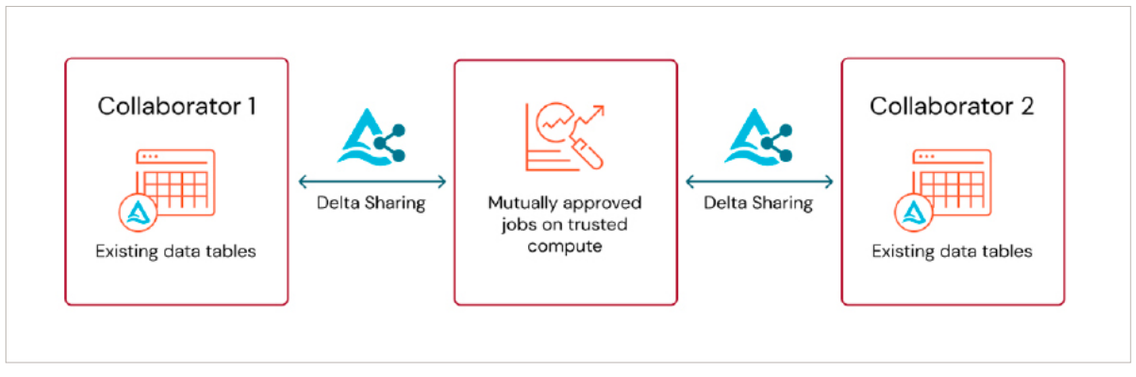
  Como podemos promover a interoperabilidade e troca de dados neste vasto e diversificado ecossistema tecnológico? Os dados são o único veículo de integração viável. Desde que os formatos de dados que usamos sejam escaláveis, abertos e governados, podemos usá-los para abstrair de plataformas individuais e de suas complexidades intrínsecas.

O formato Delta e o Delta Sharing resolvem essa grande variedade de requisitos e desafios de maneira escalável, robusta e aberta. Isso posiciona o Delta Sharing como a escolha mais sólida para unificação e simplificação do protocolo e mecanismo por meio do qual compartilhamos dados entre os setores público e privado.

Data Sharing por meio de clean rooms de dados

Levando um passo adiante as complexidades do compartilhamento de dados em um espaço altamente regulamentado e no setor público, o que fazer se precisarmos compartilhar o conhecimento contido nos dados sem nunca conceder acesso direto aos dados de origem a partes externas? Esses requisitos podem acabar sendo viáveis e desejáveis quando o risco do compartilhamento de dados é muito baixo.

Em muitos contextos do setor público, há uma preocupação de que a combinação dos dados que descrevem os cidadãos possa levar a um cenário de big brother, onde muitos dados sobre um indivíduo estão concentrados em um único ativo de dados. Se caísse nas mãos erradas, tal ativo de dados hipotéticos poderia levar a consequências imensuráveis para os indivíduos e a confiança nos serviços do setor público poderia desmoronar. Por outro lado, o valor de uma visão abrangente do cidadão poderia acelerar a tomada de decisões importantes. Poderia melhorar imensamente a qualidade das políticas e dos serviços prestados aos cidadãos.



As clean rooms de dados atendem a essa necessidade específica. Com clean rooms de dados, você pode compartilhar dados com terceiros em um ambiente com privacidade segura. Com o Unity Catalog, você pode ativar controles de acesso refinados aos dados e atender aos seus requisitos de privacidade. Nessa arquitetura, os participantes dos dados nunca têm acesso aos dados brutos. As únicas saídas das clean rooms são os ativos de dados gerados de forma pré-acordada, governada e totalmente controlada que garante a conformidade com os requisitos de todas as partes envolvidas.

Por fim, as clean rooms de dados e o Delta Sharing podem atender a implantações híbridas on-premises e externamente, em que os dados com acesso mais restrito permanecem no local. Por outro lado, dados menos restritos são gratuitos para aproveitar o poder das ofertas de nuvem. Nesse cenário, pode haver a necessidade de combinar o poder da nuvem com os dados restritos para resolver casos de uso avançados em que os recursos não estão disponíveis nas plataformas de dados locais. As clean rooms de dados podem garantir que nenhuma cópia física dos dados brutos restritos seja criada, que os resultados sejam produzidos dentro do ambiente controlado da clean room e que os resultados sejam compartilhados com o ambiente local (se os resultados mantiverem o acesso restrito dentro das políticas definidas) ou sejam encaminhados para qualquer outro sistema de destino compatível e predeterminado.

Valor do compartilhamento de dados do cidadão

Cada decisão tomada pelo governo é uma decisão que afeta seus cidadãos. Seja uma mudança de política, concessão de um benefício ou prevenção do crime, a decisão pode influenciar significativamente a qualidade da nossa sociedade. Os dados são um fator crucial para tomar as decisões certas e justificar as decisões tomadas. Simplificando, não podemos esperar decisões de alta qualidade sem a alta qualidade dos dados e uma visão completa dos dados (dentro do contexto permitido). Sem o compartilhamento de dados, permaneceremos em uma posição altamente fragmentada, na qual nossa capacidade de tomar essas decisões é severamente limitada ou mesmo completamente comprometida. Neste blog, abordamos várias soluções tecnológicas disponíveis no lakehouse que podem reduzir os riscos e acelerar a forma como o governo aproveita o ecossistema de dados de forma sustentável e escalável.

Para mais detalhes sobre os casos de uso do setor que o Delta Sharing está abordando, consulte o e-book Uma nova abordagem ao compartilhamento de dados.

Comece a experimentar com estes notebooks gratuitos da Databricks.

SEÇÃO

03

Notebooks e conjuntos de dados prontos para uso

databricks

Esta seção inclui vários aceleradores de soluções — exemplos gratuitos e prontos para uso de soluções de dados de diferentes setores, desde varejo até manufatura e saúde. Cada um dos seguintes cenários inclui notebooks com código e instruções passo a passo para ajudar você a começar. Obtenha experiência prática com a Plataforma Databricks Lakehouse testando o seguinte:

Eficácia geral do equipamento

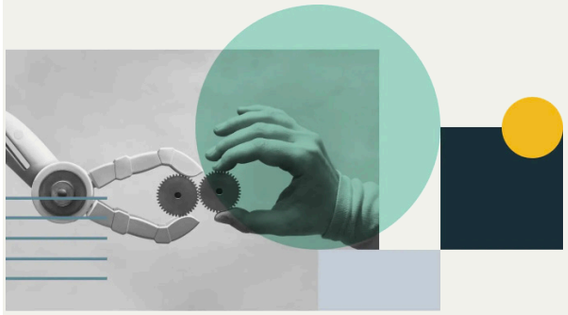
Ingira dados de sensores de equipamentos para geração de métricas e tomada de decisão baseada em dados

Explore a solução

Análise do ponto de venda em tempo real

Calcule os estoques atuais de vários produtos em diversas lojas com o Delta Live Tables

Explore a solução



Gêmeos digitais

Aproveite os gêmeos digitais — representações virtuais de dispositivos e objetos — para otimizar as operações e obter insights

Explore a solução

Mecanismos de recomendação para personalização

Melhore a experiência do usuário e a conversão dos clientes com recomendações personalizadas

Explore a solução

Compreender os dados de transparência de preços

Ingira com eficiência grandes conjuntos de dados de saúde para criar transparência de preços e entender melhor os custos de assistência médica

Explore a solução

Aceleradores de soluções adicionais com notebooks prontos para uso podem ser encontrados aqui:

Aceleradores de soluções da Databricks

SEÇÃO

04

Estudos de caso

SETOR
Tecnologia e software

SOLUÇÃO
Detecção de ameaças

CASO DE USO DA PLATAFORMA
Delta lake, Streaming de dados, Photon, Databricks SQL

NUVEM
Azure

SEÇÃO 4.1

Akamai oferece análise de segurança em tempo real usando o Delta Lake

<1
Tempo mínimo de ingestão, reduzido de 15 minutos

<85%
Das queries têm tempo de resposta de 7 segundos ou menos

A Akamai administra uma rede de distribuição de conteúdo (CDN) difundida e altamente distribuída. Sua CDN usa aproximadamente 345.000 servidores em mais de 135 países e mais de 1.300 redes em todo o mundo para rotear o tráfego da internet para algumas das maiores empresas de mídia, comércio, finanças, varejo e muitos outros setores. Cerca de 30% do tráfego da internet flui pelos servidores da Akamai. A Akamai também oferece soluções de segurança na nuvem.

Em 2018, a empresa lançou uma ferramenta de análise de segurança na web que oferece aos clientes da Akamai uma interface única e unificada para avaliar uma grande variedade de eventos de segurança de streaming e analisar esses eventos. A ferramenta de análise da web ajuda os clientes da Akamai a tomar medidas informadas em relação a eventos de segurança em tempo real. A Akamai consegue transmitir grandes quantidades de dados e atender aos SLAs rigorosos que fornece aos clientes, aproveitando o Delta Lake e a Plataforma Databricks Lakehouse para a ferramenta de análise da web.

databricks

Ingestão e streaming de enormes quantidades de dados

A ferramenta de análise de segurança na web da Akamai ingere aproximadamente 10 GB de dados relacionados a eventos de segurança por segundo. O volume de dados pode aumentar significativamente quando os clientes de varejo realizam um grande número de vendas — ou em grandes dias de compras, como Black Friday ou Cyber Monday. A ferramenta de análise de segurança da web armazena vários petabytes de dados para fins de análise. Essas análises são realizadas para proteger os clientes da Akamai e fornecer a capacidade de explorar e consultar eventos de segurança por conta própria.

A ferramenta de análise de segurança na web inicialmente contou com uma arquitetura local que executa o Apache Spark™ no Hadoop. A Akamai oferece contratos rigorosos de nível de serviço (SLAs) a seus clientes de 5 a 7 minutos a partir do momento em que ocorre um ataque até que ele seja exibido na ferramenta. A empresa buscou melhorar a velocidade de ingestão e consulta para atender a esses SLAs. “Os dados precisam ser o mais em tempo real possível para que os clientes possam ver o que os está atacando”, diz Tomer Patel, gerente de engenharia da Akamai. “Fornecer rapidamente dados consultáveis aos clientes é fundamental. Queríamos nos afastar do on-prem para melhorar o desempenho e nossos SLAs para que a latência fosse de segundos em vez de minutos.”

> **O Delta Lake nos permite não só consultar melhor os dados, mas também adquirir um aumento no volume de dados. Vimos um aumento de 80% no tráfego e nos dados no último ano, portanto, ser capaz de expandir rapidamente é fundamental.**
>
> **Tomer Patel**
> Gerente de engenharia, Akamai

Depois de realizar provas de conceito com várias empresas, a Akamai optou por basear sua arquitetura de análise de streaming no Spark e na Plataforma Databricks Lakehouse. “Devido à nossa escala e às demandas de nosso SLA, determinamos que a Databricks era a solução certa para nós”, diz Patel. “Quando consideramos a otimização do armazenamento e o armazenamento em cache de dados, se fôssemos com outra solução, não conseguiríamos alcançar o mesmo nível de desempenho”.

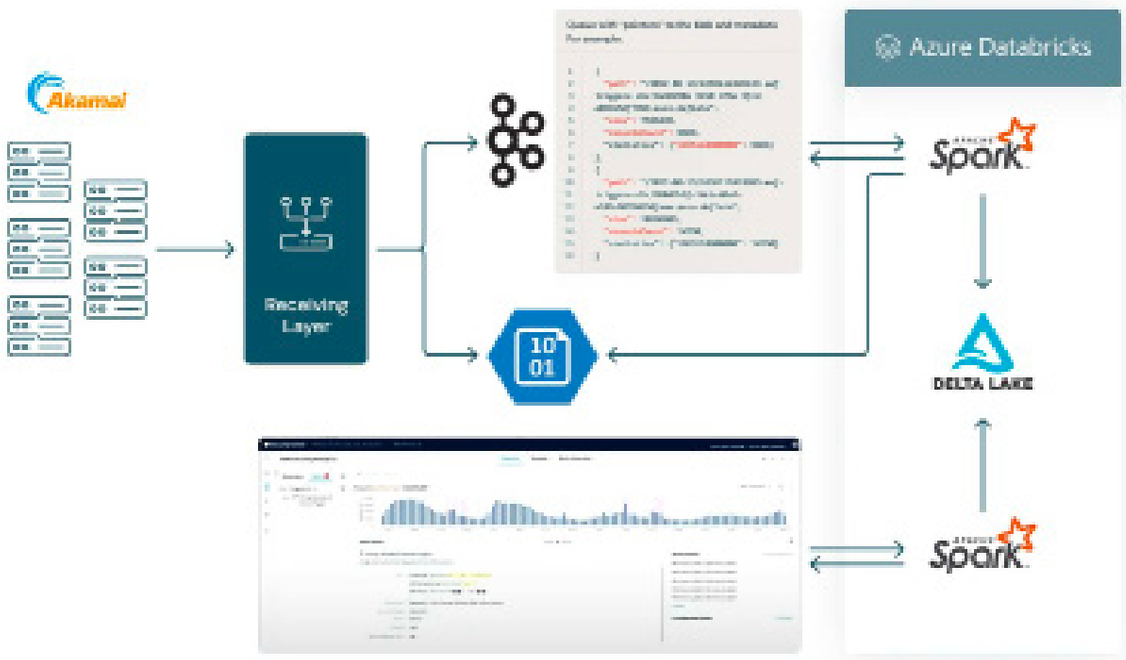
Melhora da velocidade e redução de custos

Hoje, a ferramenta de análise de segurança da web ingere e transforma dados, armazena-os no armazenamento em nuvem e envia a localização do arquivo via Kafka. Em seguida, ela usa um job do Databricks como o aplicativo de ingestão. O Delta Lake, que é o formato de armazenamento de código aberto na base da Plataforma Databricks Lakehouse, aceita queries em tempo real sobre os dados de análise de segurança da web. O Delta Lake também permite que a Akamai cresça rapidamente. “O Delta Lake nos permite não só consultar melhor os dados, mas também adquirir um aumento no volume de dados”, diz Patel. “Vimos um aumento de 80% no tráfego e nos dados no último ano, portanto, ser capaz de expandir rapidamente é fundamental.”

A Akamai também usa o Databricks SQL (DBSQL) e Photon, que fornecem desempenho de query extremamente rápido. Patel acrescentou que a Photon proporcionou um impulso significativo para o desempenho das queries. Em geral, a arquitetura de streaming da Databricks combinada com DBSQL e Photon permite à Akamai alcançar análises em tempo real, o que se traduz em benefícios de negócios em tempo real.

databricks

Patel gosta que o Delta Lake seja de código aberto, pois a empresa se beneficiou de uma comunidade de usuários trabalhando para melhorar o produto. “O fato de que o Delta Lake é de código aberto e há uma grande comunidade por trás dele significa que não precisamos implementar tudo por conta própria”, diz Patel. “Nós nos beneficiamos de bugs corrigidos que outros encontraram e de otimizações que são contribuídas para o projeto”. A Akamai trabalhou em conjunto com a Databricks para garantir que o Delta Lake pudesse atender aos requisitos de escala e desempenho definidos pela empresa. Essas melhorias voltaram para o projeto (muitas das quais foram disponibilizadas como parte do Delta Lake 2.0). Assim, qualquer usuário que executar o Delta Lake agora se beneficia da tecnologia sendo testada em grande escala em um cenário de produção do mundo real.

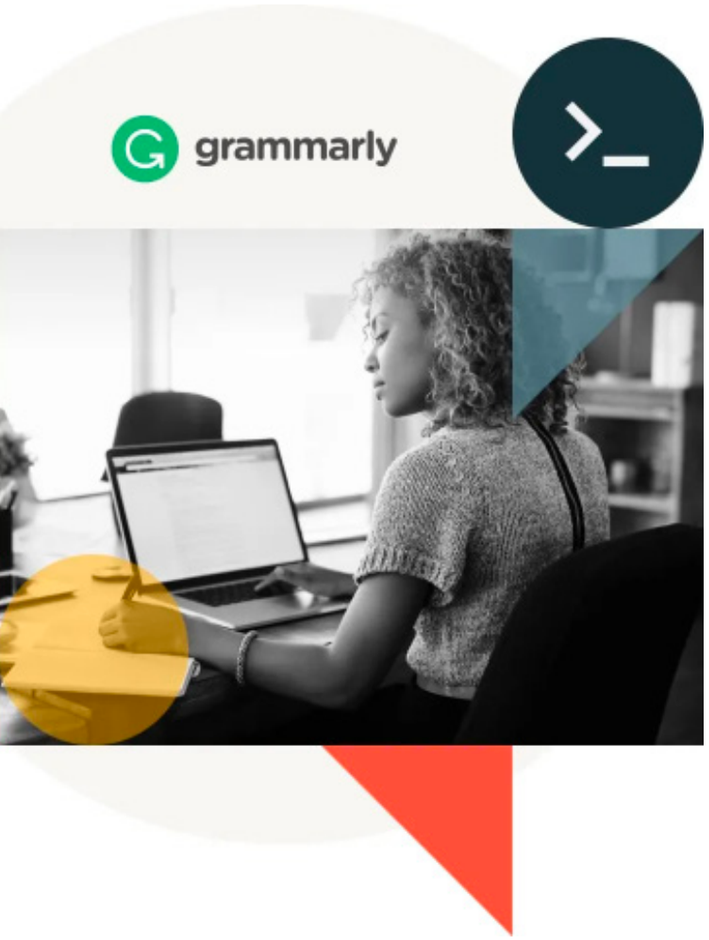


Atender a requisitos agressivos de escala, confiabilidade e desempenho

O uso do Spark Structured Streaming na Plataforma Databricks Lakehouse permite que a ferramenta de análise de segurança na web transmita grandes volumes de dados e forneça análises em tempo real e de baixa latência como serviço aos clientes da Akamai. Dessa forma, a Akamai consegue disponibilizar dados de eventos de segurança aos clientes dentro do SLA de 5 a 7 minutos a partir do momento em que ocorre um ataque. “Nosso foco é desempenho, desempenho e desempenho”, reforça Patel. “O desempenho e a escalabilidade da plataforma são o que nos motiva.”

Usando a Plataforma Databricks Lakehouse, agora leva menos de 1 minuto para ingerir os dados do evento de segurança. “Reduzir o tempo de ingestão de 15 minutos para menos de 1 minuto é uma enorme melhoria”, diz Patel. “Isso beneficia nossos clientes porque eles podem ver os dados do evento de segurança mais rapidamente e têm uma visão precisa do que está acontecendo, bem como a capacidade de filtrar tudo isso.”

A maior prioridade da Akamai é proporcionar aos clientes uma boa experiência e tempos de resposta rápidos. Até o momento, a Akamai transferiu cerca de 70% dos dados de eventos de segurança de sua arquitetura local para o Databricks, e o SLA para o tempo de query e resposta do cliente melhorou significativamente como resultado. “Agora, com a mudança para o Databricks, nossos clientes têm um tempo de resposta muito melhor, e mais de 85% das queries são concluídas em menos de 7 segundos.” Fornecer esse tipo de dados em tempo real significa que a Akamai pode ajudar seus clientes a se manterem vigilantes e manter uma configuração de segurança ideal.



SETOR
Tecnologia e software

SOLUÇÃO
Mecanismos de recomendação, eficácia da publicidade, valor da vida útil do cliente

CASO DE USO DA PLATAFORMA
Lakehouse, Delta Lake, Unity Catalog, machine learning, ETL

NUVEM
AWS

databricks

SEÇÃO 4.2

Grammarly usa o Databricks Lakehouse para melhorar a experiência do usuário

110%

Queries mais rápidas do que um data warehouse a 10% do custo de ingestão

5 bilhões

Eventos diários disponíveis para análise em menos de 15 minutos

A missão da Grammarly é melhorar a vida melhorando a comunicação. A assistência de comunicação confiável baseada em IA da empresa fornece sugestões em tempo real para ajudar indivíduos e equipes a escrever com mais confiança e atingir melhores resultados. Suas ofertas abrangentes — Grammarly Premium, Grammarly Business, Grammarly for Education e Grammarly for Developers — fornecem suporte de comunicação líder onde quer que a escrita aconteça. À medida que a empresa cresceu ao longo dos anos, seu sistema de análise legado e desenvolvido internamente dificultou a avaliação de grandes conjuntos de dados de forma rápida e econômica.

Ao migrar para a Plataforma Databricks Lakehouse, a Grammarly agora pode manter uma plataforma de análise flexível, dimensionável e altamente segura que ajuda 30 milhões de pessoas e 50.000 equipes em todo o mundo a escrever com mais eficiência todos os dias.

Aproveitar os dados para melhorar as comunicações para milhões de usuários e milhares de equipes

Quando as pessoas usam a assistência de comunicação de IA da Grammarly, elas recebem sugestões que ajudam a melhorar vários aspectos de comunicação, incluindo ortografia e correção de gramática, clareza e concisão, escolha de palavras, estilo e tom. A Grammarly recebe feedback quando os usuários aceitam, rejeitam ou ignoram suas sugestões por meio de eventos criados pelo aplicativo, que totalizam cerca de 5 bilhões de eventos por dia.

Historicamente, a Grammarly dependia de uma plataforma de análise legada interna e usava uma linguagem interna semelhante a SQL que consumia muito tempo para aprender e dificultava integrar novos contratados. À medida que a empresa crescia, os analistas de dados da Grammarly descobriram que a plataforma não atendia suficientemente às necessidades de suas funções essenciais de negócios, principalmente marketing, vendas e sucesso do cliente. Os analistas tinham que copiar e colar dados de planilhas porque o sistema existente não conseguia ingerir efetivamente os dados externos necessários para responder a perguntas como “Qual canal de marketing oferece o maior ROI?”. Era um desafio gerar relatórios porque o sistema existente não oferecia suporte aos painéis do Tableau e os líderes e analistas da empresa precisavam garantir que pudessem tomar decisões com rapidez e confiança.

> **O Databricks Lakehouse nos deu a flexibilidade de liberar nossos dados sem comprometer. Essa flexibilidade nos permitiu acelerar a análise a um ritmo que nunca alcançamos antes.**
>
> **Chris Locklin**
> gerente de engenharia de plataformas de dados da Grammarly

A Grammarly também buscou unificar seus data warehouses para dimensionar e melhorar os recursos de armazenamento e consulta de dados. Do jeito que estava, grandes clusters do Amazon EMR funcionavam 24 horas por dia e aumentavam os custos. Com as várias fontes de dados, a equipe também precisava manter o controle de acesso. “O controle de acesso em um sistema de arquivos distribuído é difícil, e só fica mais complicado à medida que você ingere mais fontes de dados”, diz Chris Locklin, gerente de engenharia de plataformas de dados da Grammarly. Entretanto, a confiança em um único fluxo de trabalho de streaming dificultou a colaboração entre as equipes. Os silos de dados surgiram à medida que diferentes áreas de negócios implementaram ferramentas de análise individualmente. “Cada equipe decidiu resolver suas necessidades de análise da maneira que achava melhor”, diz Locklin. “Isso criou desafios de consistência e para saber qual conjunto de dados estava correto.”

À medida que a estratégia de dados evoluía, a prioridade da Grammarly era aproveitar ao máximo os dados analíticos e, ao mesmo tempo, mantê-los seguros. Isso foi crucial porque a segurança é a prioridade número um e o recurso mais importante da Grammarly, tanto na forma como protege os dados de seus usuários quanto na forma como garante que os dados da própria empresa permaneçam seguros. Para isso, a equipe de plataforma de dados da Grammarly buscou consolidar dados e unificar a empresa em uma única plataforma. Isso significava manter uma infraestrutura altamente segura que pudesse acompanhar o crescimento da empresa, melhorando a flexibilidade de ingestão, reduzindo os custos e estimulando a colaboração.

Melhorar a análise, a visualização e a tomada de decisões com o lakehouse

Depois de realizar várias provas de conceito para aprimorar a infraestrutura, a Grammarly migrou para a Plataforma Databricks Lakehouse. Colocar todos os dados analíticos no lakehouse criou um hub central para todos os produtores de dados e consumidores em todo o Grammarly, com o Delta Lake no núcleo.

Usando a arquitetura de lakehouse, os analistas de dados da Grammarly agora têm uma interface consolidada para análises, o que leva a uma única fonte de verdade e confiança na precisão e disponibilidade de todos os dados gerenciados pela equipe da plataforma de dados. Em toda a organização, as equipes estão usando o Databricks SQL para realizar queries dentro da plataforma em dados de produtos gerados internamente e dados externos de parceiros de plataformas de publicidade digital. Agora, eles podem se conectar facilmente ao Tableau e criar dashboards e visualizações para apresentar aos executivos e às principais partes interessadas.

"A segurança é de extrema importância na Grammarly, e o objetivo número um da equipe é possuir e proteger nossos dados analíticos", diz Locklin. "Outras empresas pedem seus dados, armazenam-nos e permitem que você faça análises neles. Da mesma forma que a Grammarly garante que os dados dos usuários permaneçam sempre deles, queríamos garantir que os dados de nossa empresa permanecessem nossos. Os dados da Grammarly permanecem dentro da Grammarly".

Com os dados consolidados no lakehouse, diferentes áreas de negócios da Grammarly agora podem analisar dados de forma mais completa e eficaz. Por exemplo, a equipe de marketing da Grammarly usa publicidade para atrair novos negócios. Usando o Databricks, a equipe pode consolidar dados de várias fontes para extrapolar o valor da vida útil de um usuário, compará-lo com os custos de aquisição de clientes e obter feedback rápido sobre as campanhas. Em outros lugares, os dados capturados das interações do usuário fluem para um conjunto de tabelas usadas por analistas para análise ad hoc para informar e melhorar a experiência do usuário.

Ao consolidar dados em uma plataforma unificada, a Grammarly eliminou silos de dados. "A capacidade de trazer todos esses recursos, processamento de dados e análise sob a mesma plataforma usando o Databricks é extremamente valiosa", diz Sergey Blanket, diretor de inteligência de negócios da Grammarly. "Fazer tudo, desde ETL e engenharia até análise e ML sob o mesmo guarda-chuva, remove barreiras e facilita o trabalho de todos com os dados e uns com os outros."

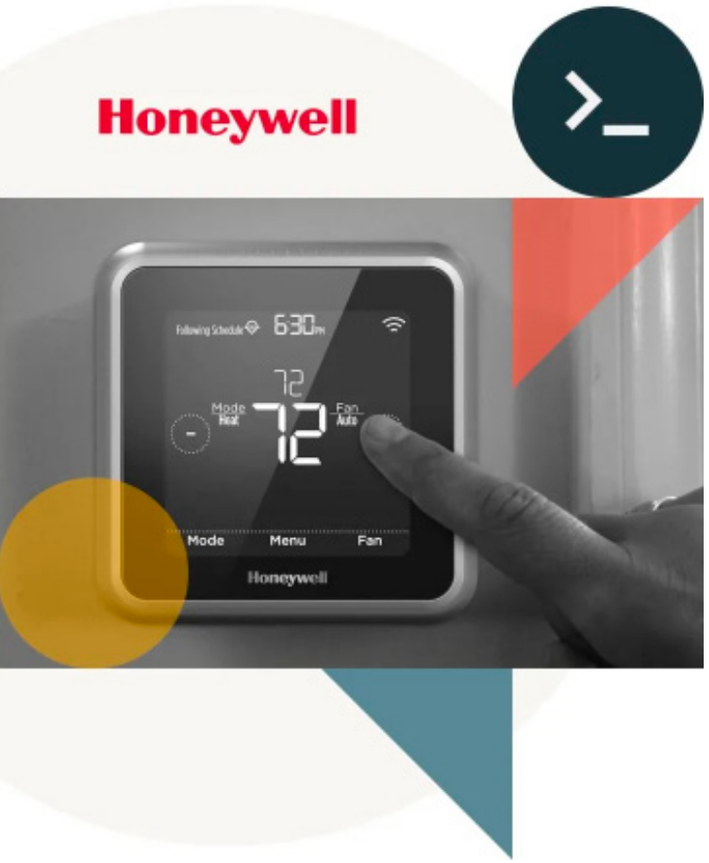
Para gerenciar o controle de acesso, permitir a observabilidade de ponta a ponta e monitorar a qualidade dos dados, a Grammarly conta com os recursos de linhagem de dados no Catálogo Unity. “A linhagem de dados nos permite monitorar efetivamente o uso de nossos dados e garantir que eles mantenham os padrões definidos como uma equipe de plataforma de dados”, diz Locklin. “A linhagem é a última peça crucial para o controle de acesso. Ela permite que os analistas aproveitem os dados para realizar suas tarefas e, ao mesmo tempo, aderir a todos os padrões de uso e controles de acesso, mesmo ao recriar tabelas e conjuntos de dados em outro ambiente.”

O tempo mais rápido para obter insights gera decisões de negócios mais inteligentes

Usando a Plataforma Databricks Lakehouse, as equipes de engenharia da Grammarly agora têm uma plataforma centralizada e personalizada e uma fonte de dados consistente em toda a empresa, resultando em maior velocidade e eficiência e custos reduzidos. A arquitetura de lakehouse levou a queries 110% mais rápidas, a 10% do custo para ingerir em relação a um data warehouse. Agora, a Grammarly pode disponibilizar 5 bilhões de eventos diários para análise em menos de 15 minutos, em vez de 4 horas, permitindo agregar dados de baixa latência e otimização de queries. Isso permite que a equipe receba rapidamente feedback sobre os novos recursos que estão sendo implantados e entenda se estão sendo adotados conforme o esperado. Em última análise, isso os ajuda a entender como os grupos de usuários interagem com a UX, melhorando a experiência e garantindo que os recursos e as versões de produtos agreguem mais valor aos usuários. “Tudo o que minha equipe faz é se concentrar em criar uma experiência rica e personalizada que capacite as pessoas a se comunicarem de forma mais eficaz e alcançarem seu potencial”, diz Locklin.

Mudar para a arquitetura de lakehouse também solucionou o desafio do controle de acesso sobre os sistemas de arquivos distribuídos, enquanto o Unity Catalog habilitou controles de acesso refinados baseados em função e linhagem de dados em tempo real. “O Unity Catalog nos permite gerenciar permissões de arquivos com mais flexibilidade do que seria possível com um banco de dados”, afirma Locklin. “Ele solucionou um problema que minha equipe não poderia resolver em escala. Enquanto o uso do Databricks nos permite manter dados analíticos internos, o Unity Catalog nos ajuda a continuar a manter os mais altos padrões de proteção de dados, controlando paradigmas de acesso dentro dos nossos dados. Isso abre um mundo totalmente novo de coisas que podemos fazer.”

Em última análise, migrar para a Plataforma Databricks Lakehouse ajudou a Grammarly a promover uma cultura orientada por dados, em que os funcionários obtêm acesso rápido a análises sem a necessidade de escrever queries complexas, sempre mantendo as práticas de segurança de nível empresarial da Grammarly. “A missão da nossa equipe é ajudar a Grammarly a tomar decisões de negócios melhores e mais rápidas”, acrescenta Blanket. “Minha equipe não seria capaz de executar efetivamente essa missão se não tivéssemos uma plataforma como a Databricks disponível para nós.” Talvez o mais importante seja que a migração de uma rígida infraestrutura legada dá à Grammarly a capacidade de adaptação para fazer mais, sabendo que a plataforma evoluirá à medida que suas necessidades evoluem. “O Databricks nos deu a flexibilidade de liberar nossos dados sem comprometer”, diz Locklin. “Essa flexibilidade nos permitiu acelerar a análise a um ritmo que nunca alcançamos antes.”



SETOR

Manufatura

CASO DE USO DA PLATAFORMA

Lakehouse, Delta Lake, Delta Live Tables

NUVEM

azure

SEÇÃO 4.3

Honeywell seleciona o Delta Live Tables para streaming de dados

As empresas estão sob crescente pressão para reduzir o uso de energia e, ao mesmo tempo, querem reduzir os custos e aumentar a eficiência. A Honeywell fornece soluções específicas do setor que incluem produtos e serviços aeroespaciais, tecnologias de controle para edifícios e indústrias e materiais de desempenho em todo o mundo. A divisão de Soluções Ambientais e de Energia da Honeywell usa sensores de IoT e outras tecnologias para ajudar as empresas em todo o mundo a gerenciar a demanda de energia, reduzir o consumo de energia e as emissões de carbono, otimizar a qualidade do ar interno e melhorar o bem-estar dos ocupantes.

Para isso, a Honeywell precisa coletar grandes quantidades de dados. Usando o Delta Live Tables na Plataforma Databricks Lakehouse, a equipe de dados da Honeywell agora pode ingerir bilhões de linhas de dados de sensores no Delta Lake e criar automaticamente endpoints SQL para queries em tempo real e insights de várias camadas sobre os dados em escala. Isso ajuda a Honeywell a melhorar a forma como gerenciam os dados e extraem mais valor deles, tanto para eles próprios quanto para seus clientes.

> **O Databricks nos ajuda a reunir muitas fontes de dados diferentes, fazer agregações e controlar a quantidade significativa de dados que coletamos de nossos edifícios para podermos oferecer valor aos clientes.**
>
> **Dr. Chris Inkpen**
> arquiteto de soluções globais, Soluções Ambientais e de Energia da Honeywell

databricks

Processamento de bilhões de pontos de dados de IoT por dia

As soluções e os serviços da Honeywell são usados em milhões de edifícios em todo o mundo. Ajudar seus clientes a criar edifícios seguros, mais sustentáveis e produtivos pode exigir milhares de sensores por edifício. Esses sensores monitoram os principais fatores, como temperatura, pressão, umidade e qualidade do ar. Além dos dados coletados pelos sensores dentro de um prédio, os dados também são coletados da área externa, como dados climáticos e de poluição. Outro conjunto de dados consiste em informações sobre os próprios edifícios, como tipo de construção, propriedade, planta baixa, metragem quadrada de cada andar e metragem quadrada de cada quarto. Esse conjunto de dados é combinado com os dois fluxos de dados diferentes, somando muitos dados em vários formatos estruturados e não estruturados, incluindo imagens e fluxos de vídeo, dados de telemetria, dados de eventos etc. Em dias de pico, a Honeywell ingere entre 200 a 1.000 eventos por segundo em qualquer edifício, o que equivale a bilhões de pontos de dados por dia. A infraestrutura de dados existente da Honeywell foi desafiada a atender a essa demanda. Isso também dificultou para a equipe de dados da Honeywell consultar e visualizar seus dados díspares para que pudesse fornecer aos clientes informações e análises rápidas e de alta qualidade.

ETL simplificado: pipelines de dados reutilizáveis e de alta qualidade

Com as tabelas do Delta Live Tables (DLT) na Plataforma Databricks Lakehouse, a equipe de dados da Honeywell agora pode ingerir bilhões de linhas de dados de sensores no Delta Lake e criar automaticamente endpoints SQL para queries em tempo real e insights multicamadas de dados em escala. “Não precisamos fazer nada para escalonar o DLT”, diz Chris Inkpen, arquiteto de soluções globais das Soluções Ambientais e de Energia da Honeywell. “Mesmo quando fornecemos mais dados ao sistema, ele dá conta. Desde o início, tivemos a confiança de que o sistema poderia lidar com todos os tipos de dados inseridos.”

A Honeywell credita à Plataforma Databricks Lakehouse por ajudá-la a unificar seus vastos e variados dados — em batch, streaming, estruturados e não estruturados — em uma única plataforma. “Temos muitos tipos de dados diferentes. A Plataforma Databricks Lakehouse nos permite usar programas como Apache Kafka e Auto Loader para carregar e processar vários tipos de dados e tratar tudo como um fluxo de dados, o que é incrível. Depois de obtermos dados estruturados a partir de dados não estruturados, podemos escrever pipelines padronizados.”

Os engenheiros de dados da Honeywell agora podem criar e aproveitar seus próprios pipelines ETL com o Delta Live Tables e obter insights e análises rapidamente. Os pipelines ETL podem ser reutilizados, independentemente do ambiente, e os dados podem ser executados em batches ou streams. Isso também ajudou a equipe de dados da Honeywell a fazer a transição de uma equipe pequena para outra maior. “Quando escrevíamos nossos primeiros pipelines antes do DLT, somente uma pessoa poderia trabalhar em uma parte da funcionalidade. Agora que temos o DLT e a capacidade de ter pastas com funcionalidade comum, temos uma plataforma muito boa onde podemos facilmente girar pipelines diferentes.”

O DLT também ajudou a Honeywell a estabelecer arquivos de log padrão para monitorar e justificar os custos de seus pipelines de produtos. “Utilizando o DLT, podemos analisar quais partes do nosso pipeline precisam de otimização”, diz Inkpen. “Com os pipelines padrão, isso era muito mais caótico.”

Facilidade, simplicidade e escalabilidade em toda a infraestrutura

O Delta Live Tables ajudou a equipe de dados da Honeywell a consultar consistentemente dados complexos, oferecendo simplicidade de escala. Ele também permite visualizar dados de ponta a ponta dos streams de dados da Honeywell à medida que fluem para sua infraestrutura, são transformados e, em seguida, fluem para fora. "Noventa por cento do nosso ETL agora é capturado em diagramas, e isso é uma ajuda considerável e melhora a governança de dados. O DLT incentiva — e quase impõe — um bom design", diz Inkpen.

Usar o lakehouse como um workspace compartilhado ajudou a promover o trabalho em equipe e a colaboração na Honeywell. "A equipe colabora muito bem agora, trabalhando juntos todos os dias para dividir o pipeline em suas próprias histórias e cargas de trabalho", conta Inkpen.

Entretanto, a capacidade de gerenciar dados de streaming com baixa latência e melhor throughput aprimorou a precisão e reduziu os custos. "Uma vez que projetamos algo usando o DLT, estamos muito seguros contra problemas de escalabilidade — certamente cem vezes melhor do que se não tivéssemos escrito no DLT", explica Inkpen. "Podemos então voltar e ver como deixar um job tradicional mais eficiente e menos dispendioso. Estamos em uma posição muito melhor para tentar fazer isso a partir do DLT."

O uso do Databricks e do DLT também ajuda a equipe da Honeywell a atuar com maior agilidade, o que permite inovar mais rapidamente e, ao mesmo tempo, capacitar os desenvolvedores a responder aos requisitos dos usuários quase imediatamente. "Nossa arquitetura anterior tornava impossível saber quais eram nossos gargalos e o que precisávamos para escalar. Agora podemos fazer data science quase em tempo real."

Em última análise, a Honeywell agora pode fornecer aos clientes os dados e a análise necessários para tornar seus edifícios mais eficientes, saudáveis e seguros para os ocupantes. "Estou sempre procurando maneiras de melhorar nossos ciclos de vida, o tempo de lançamento no mercado e a qualidade dos dados", diz Inkpen. "O Databricks nos ajuda a reunir muitas fontes de dados diferentes, fazer agregações e controlar a quantidade significativa de dados que coletamos de nossos edifícios para podermos oferecer valor aos clientes."

Pronto para começar? Saiba mais sobre o Delta Live Tables aqui.

SETOR
Energia e serviços públicos

CASO DE USO DA PLATAFORMA
Lakehouse, Workflows

NUVEM
AWS

SEÇÃO 4.4

Wood Mackenzie ajuda os clientes na transição para um futuro mais sustentável

12 bilhões
Pontos de dados processados a cada semana

80-90%
Redução no tempo de processamento

Economia de custos
Nas operações por meio da automação do fluxo de trabalho

A Wood Mackenzie oferece consultoria e análise personalizadas para uma ampla gama de clientes nos setores de energia e recursos naturais. Fundada em Edimburgo, a primeira empresa cultivou profunda experiência em petróleo e gás upstream e, em seguida, ampliou seu foco para fornecer uma visão detalhada de cada setor interconectado dos setores de energia, produtos químicos, metais e mineração.

Hoje, ela se vê desempenhando um papel importante na transição para um futuro mais sustentável. Usar o Databricks Workflows para automatizar pipelines ETL ajuda a Wood Mackenzie a ingerir e processar grandes quantidades de dados. O uso de um fluxo de trabalho comum proporciona maior visibilidade aos membros da equipe de engenharia, incentivando uma melhor colaboração. Com um fluxo de trabalho automatizado e transparente, a equipe teve um aumento na produtividade e na qualidade dos dados e um caminho mais fácil para corrigir problemas de pipeline.

databricks

Fornecendo insights para o setor de energia

Atendendo à missão da Wood Mackenzie, o produto Lens é uma plataforma de análise de dados criada para fornecer insights nos principais pontos de decisão para os clientes no setor de energia. Alimentando o Lens estão grandes quantidades de dados coletados de várias fontes de dados e sensores usados para monitorar a criação de energia, produção de petróleo e gás e muito mais. Essas fontes de dados atualizam cerca de 12 bilhões de pontos de dados toda semana, que devem ser ingeridos, limpos e processados como parte da entrada para a plataforma Lens. Yanyan Wu, vice-presidente de dados da Wood Mackenzie, gerencia uma equipe de profissionais de big data que constroem e mantêm o pipeline ETL que fornece dados de entrada para o Lens. A equipe está usando a Plataforma Databricks Lakehouse e o Apache Spark™ para processamento paralelo, o que proporciona maiores benefícios de desempenho e escalabilidade em comparação com um sistema de nó único anterior que opera sequencialmente. “Vimos uma redução de 80 a 90% no tempo de processamento de dados, o que resulta em dados mais atualizados, mais completos e mais precisos para os nossos clientes”, diz Wu.

> **Nossa missão é transformar a forma como alimentamos o planeta. Nossos clientes do setor de energia precisam de dados, consultoria e pesquisas para alcançar essa transformação. O Databricks Workflows nos dá a velocidade e a flexibilidade para fornecer os insights de que nossos clientes precisam.**
>
> **Yanyan Wu**
> vice-presidente de dados, Wood Mackenzie

Maior colaboração e transparência com um fluxo de trabalho comum

O pipeline de dados gerenciado pela equipe inclui vários estágios para padronizar e limpar dados brutos, que podem ser estruturados ou não estruturados e podem estar na forma de PDFs ou mesmo de anotações manuscritas.

Diferentes integrantes da equipe de dados são responsáveis por diferentes partes do pipeline, e há uma dependência entre os estágios de processamento de cada integrante. Usando o Databricks Workflows, a equipe definiu um fluxo de trabalho comum usado por todos os integrantes. Cada estágio do pipeline é implementado em um notebook Python, que é executado como um job no fluxo de trabalho principal.

Agora, cada membro da equipe pode ver exatamente qual código está sendo executado em cada estágio, facilitando a identificação da causa do problema. Saber quem é o proprietário da parte do pipeline que originou o problema torna a correção de problemas muito mais rápida. “Sem um fluxo de trabalho comum, diferentes membros da equipe executavam seus notebooks de forma independente, sem saber que a falha na sua execução afetava os estágios posteriores”, explica Meng Zhang, analista de dados principal da Wood Mackenzie. “Ao tentar executar novamente os notebooks, era difícil saber qual versão foi executada inicialmente e a versão mais recente a ser usada.”

Usar os recursos de alerta do Workflows para notificar a equipe quando há falha em uma tarefa de fluxo de trabalho garante que todos saibam que ocorreu a falha e permite que a equipe trabalhe em conjunto para resolver o problema rapidamente. A definição de um fluxo de trabalho comum criou consistência e transparência, o que facilitou a colaboração. "Usar o Databricks Workflows nos permitiu incentivar a colaboração e romper as paredes entre diferentes estágios do processo", explica Wu. "Permitiu que todos nós falássemos a mesma língua."

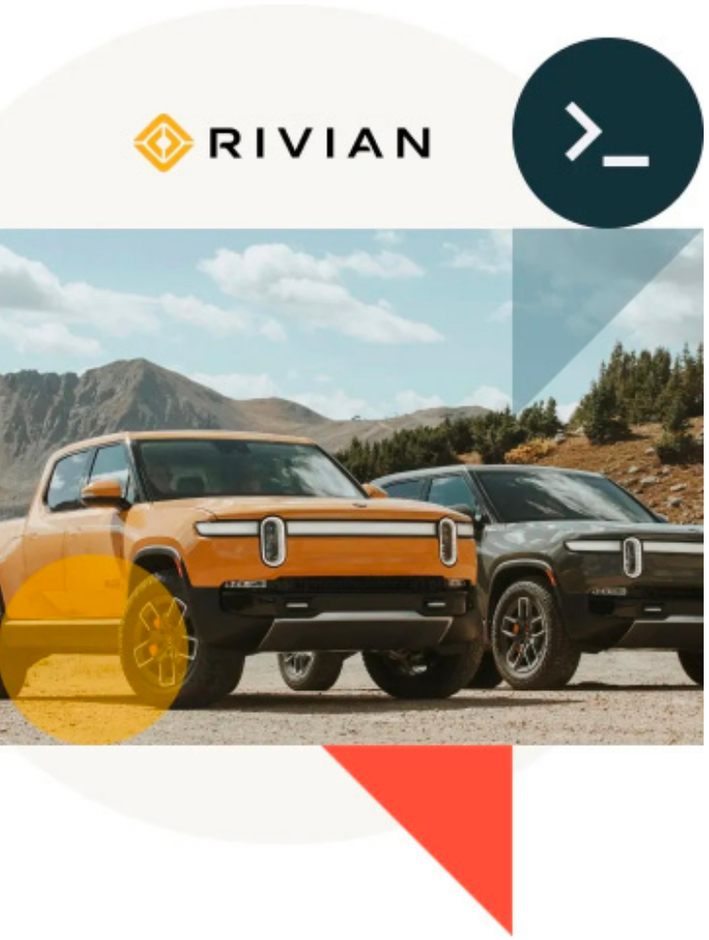
Criar transparência e consistência não foi a única vantagem percebida pela equipe. O uso do Workflows para automatizar a execução de notebooks também levou a uma economia de custos em comparação com a execução manual de notebooks interativos.

Melhor produtividade no desenvolvimento de código

O processo de desenvolvimento do pipeline de ETL da equipe envolve a iteração em notebooks PySpark.A utilização de notebooks interativos na IU do Databricks facilita o desenvolvimento e o teste manual de um notebook para os profissionais de dados da equipe. Como o Databricks Workflows permite a execução de notebooks como tipo de tarefa (juntamente com arquivos Python, arquivos JAR e outros tipos), quando o código está pronto para produção, é fácil e econômico automatizá-lo, adicionando-o a um fluxo de trabalho. O fluxo de trabalho pode ser facilmente revisado, adicionando ou removendo quaisquer etapas do fluxo definido. Essa maneira de trabalhar mantém o benefício do desenvolvimento manual de notebooks com a interface interativa do notebook e, ao mesmo tempo, aproveita o poder da automação, o que reduz os possíveis problemas que podem ocorrer ao executar notebooks manualmente.

A equipe aumentou ainda mais a produtividade desenvolvendo um processo de CI/CD. "Ao conectar nosso repositório de código de controle de origem, sabemos que o fluxo de trabalho sempre executa a versão de código mais recente com a qual é feita o commit no repositório", explica Zhang. "Também é fácil mudar para um branch de desenvolvimento para desenvolver um novo recurso, corrigir um bug e executar um fluxo de trabalho de desenvolvimento. Quando o código passa por todos os testes, é feito o merge de volta ao branch principal e o fluxo de trabalho de produção é atualizado automaticamente com o código mais recente."

No futuro, a Wood Mackenzie planeja otimizar o uso dos fluxos de trabalho da Databricks para automatizar os processos de machine learning, como treinamento de modelos, monitoramento de modelos e manipulação de drift de modelos. A empresa usa ML para melhorar a qualidade dos dados e extrair insights para agregar mais valor aos seus clientes. "Nossa missão é transformar a forma como alimentamos o planeta", diz Wu. "Nossos clientes do setor de energia precisam de dados, consultoria e pesquisas para alcançar essa transformação. O Databricks Workflows nos dá a velocidade e a flexibilidade para fornecer os insights de que nossos clientes precisam."



SETOR
Manufatura

SOLUÇÃO
Manutenção preditiva, Dimensionamento de modelos de ML para IoT, ESG orientada por dados

PLATAFORMA
Lakehouse, Delta Lake, Unity Catalog

NUVEM
AWS

databricks

SEÇÃO 4.5

Rivian redefine a experiência de direção com o Databricks Lakehouse

250 usuários da plataforma

Aumento de 50x em relação ao ano anterior

A Rivian está preservando o mundo natural para futuras gerações com os revolucionários veículos elétricos de aventura (EAVs). Com mais de 25.000 EAVs em trânsito gerando vários terabytes de dados de IoT por dia, a empresa está usando insights de dados e machine learning para melhorar a integridade e o desempenho do veículo. No entanto, com as ferramentas de cloud legadas, a empresa sofreu para escalar pipelines de forma econômica e gastou recursos significativos em manutenção, diminuindo sua capacidade de ser verdadeiramente orientada por dados.

Desde que migrou para a Plataforma Databricks Lakehouse, a Rivian agora pode entender como um veículo se comporta e como isso afeta o motorista que o utiliza. Equipada com esses insights, a Rivian está inovando mais rapidamente, reduzindo custos e, em última análise, oferecendo uma melhor experiência de direção aos clientes.

Sofrendo para democratizar os dados em uma plataforma legada

Construir um mundo que continuará a ser desfrutado pelas gerações futuras exige uma mudança na forma como operamos. Na vanguarda desse movimento está a Rivian, uma fabricante de veículos elétricos focada em mudar os sistemas de energia e transporte do nosso planeta, abandonando por completo o combustível fóssil. Hoje, a frota da Rivian inclui veículos pessoais e envolve uma parceria com a Amazon para entregar 100.000 vans comerciais. Cada veículo usa sensores e câmeras de IoT para capturar petabytes de dados, desde a condução do veículo até o funcionamento de várias peças. Com todos esses dados na ponta dos dedos, a Rivian usa machine learning para melhorar a experiência geral do cliente com a manutenção preditiva, de modo que os possíveis problemas sejam resolvidos antes que afetem o motorista.

Antes mesmo de a Rivian lançar seu primeiro EAV, ela já enfrentava limitações de visibilidade de dados e ferramentas que diminuíam a produção, impediam a colaboração e aumentavam os custos operacionais. Eles tinham de 30 a 50 clusters de compute grandes e operacionalmente complicados a qualquer momento, o que saía caro. Além de ser difícil gerenciar o sistema, a empresa também sofria interrupções frequentes de cluster, forçando as equipes a dedicar mais tempo à solução de problemas do que à análise de dados. Além disso, os silos de dados criados por sistemas desarticulados desaceleraram o compartilhamento de dados, o que contribuiu ainda mais para problemas de produtividade. As linguagens de dados necessárias e a experiência específica dos conjuntos de ferramentas criaram uma barreira à entrada que limitou os desenvolvedores de fazer pleno uso dos dados disponíveis. Jason Shiverick, principal cientista de dados da Rivian, disse que o maior problema era o acesso aos dados. “Eu queria abrir nossos dados para um público mais amplo de usuários menos técnicos para que eles também pudessem aproveitar os dados com mais facilidade.”

A Rivian sabia que, uma vez que seus EAVs chegassem ao mercado, a quantidade de dados ingeridos explodiria. Para oferecer a confiabilidade e o desempenho prometidos, a Rivian precisava de uma arquitetura que não apenas democratizasse o acesso aos dados, mas também fornecesse uma plataforma comum para criar soluções inovadoras que ajudassem a garantir uma experiência de direção confiável e agradável.

> **O Databricks Lakehouse nos permite reduzir a barreira de entrada para o acesso aos dados em toda a nossa organização. Com isso, podemos construir os veículos elétricos mais inovadores e confiáveis do mundo.**
>
> **Wassym Bensaid**
> vice-presidente de desenvolvimento de software, Rivian

Previsão de problemas de manutenção com o Databricks Lakehouse

A Rivian optou por modernizar sua infraestrutura de dados na Plataforma Databricks Lakehouse, oferecendo a capacidade de unificar todos os seus dados em uma visão comum para análise downstream e machine learning. Agora, equipes de dados exclusivas têm diversas ferramentas acessíveis para fornecer insights acionáveis para diferentes casos de uso, desde manutenção preditiva até desenvolvimento de produto mais inteligente. Venkat Sivasubramanian, diretor sênior de Big Data da Rivian, afirma: "Conseguimos construir uma cultura em torno de uma plataforma de dados abertos que fornecesse um sistema para realmente democratizar dados e análises de forma eficiente". O suporte flexível do Databricks a todas as linguagens de programação e a integração perfeita com uma variedade de conjuntos de ferramentas eliminaram os obstáculos de acesso e desbloquearam novas oportunidades. Wassym Bensaid, vice-presidente de desenvolvimento de software da Rivian, explica: "Hoje temos várias equipes, técnicas e de negócios, que usam o Databricks Lakehouse para explorar nossos dados, construir pipelines de dados de desempenho e extrair insights acionáveis de negócios e produtos por meio de dashboards visuais".

A equipe de ADAS (sistemas avançados de assistência ao condutor) da Rivian agora pode preparar facilmente dados de acelerômetro telemétrico para entender todas as movimentações dos EAVs. Esses dados de gravação central incluem informações sobre atividades de inclinação, rotação, velocidade, suspensão e airbag, para ajudar a Rivian a compreender o desempenho do veículo, os padrões de direção e a previsibilidade do sistema do carro conectado. Com base nessas métricas de desempenho importantes, a Rivian pode melhorar a precisão dos recursos inteligentes e o controle dos motoristas sobre eles. Projetados para aliviar o estresse de longas viagens e de dirigir em trânsito intenso, recursos como controle de cruzeiro adaptativo, assistência para mudança de faixa, direção automática de emergência e aviso de colisão frontal podem ser aperfeiçoados com o tempo para otimizar continuamente a experiência de direção para os clientes.

O compartilhamento e a colaboração de dados seguros também foram facilitados com o Unity Catalog da Databricks. Shiverick descreve como a governança unificada para o lakehouse beneficia a produtividade da Rivian. "O Unity Catalog nos oferece um catálogo de dados verdadeiramente centralizado em todas as nossas diferentes equipes", disse ele. "Agora temos gerenciamento e controles de acesso adequados." Venkat acrescenta: "Com o Unity Catalog, estamos centralizando o catálogo de dados e o gerenciamento de acesso em várias equipes e workspaces, o que simplificou a governança". A governança controlada por versão de ponta a ponta e a auditabilidade de fontes de dados confidenciais, como as usadas para sistemas de direção autônoma, produzem uma solução simples e segura para engenharia de recursos. Isso dá à Rivian uma vantagem competitiva na corrida para capturar a rede de direção autônoma.

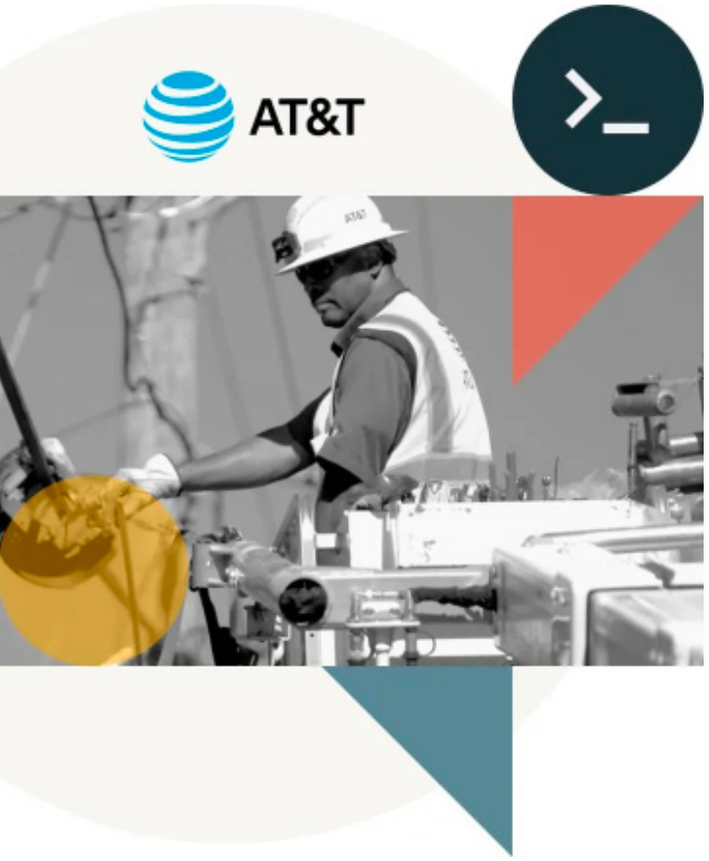
Acelerando em um mundo eletrificado e sustentável

Ao expandir sua capacidade de fornecer insights de dados valiosos com velocidade, eficiência e economia, a Rivian está preparada para usar mais dados a fim de melhorar as operações e o desempenho de seus veículos para aprimorar a experiência do cliente. Venkat diz: "A flexibilidade que o lakehouse oferece nos economiza muito dinheiro do ponto de vista da nuvem, e isso é uma grande vitória para nós." Uma vez que o Databricks Lakehouse fornece uma abordagem unificada e de código aberto para dados e análises, a equipe de confiabilidade do veículo consegue entender melhor como as pessoas usam seus veículos, e isso ajuda a definir o design das futuras gerações de veículos. Ao usar a Plataforma Databricks Lakehouse, eles observaram um aumento de 30% a 50% no desempenho do tempo de execução, o que levou a insights mais rápidos e ao desempenho do modelo.

Shiverick explica: "Do ponto de vista da confiabilidade, podemos garantir que os componentes resistirão aos ciclos de vida apropriados. Pode ser tão simples quanto certificar-se de que as maçanetas das portas são robustas o suficiente para suportar o uso constante ou tão complicado quanto a manutenção preditiva e preventiva para eliminar a chance de falha no campo. De um modo geral, estamos melhorando a qualidade do software com base nas principais métricas do veículo para uma melhor experiência do cliente."

Do ponto de vista da otimização de design, a visão de dados desobstruída da Rivian também está produzindo novos insights de diagnóstico que podem melhorar a saúde, a segurança, a estabilidade e a segurança da frota. Venkat diz: "Podemos realizar diagnósticos remotos para fazer a triagem de um problema rapidamente, chamar um atendimento móvel ou potencialmente enviar uma OTA para corrigir o problema com o software. Tudo isso precisa de muita visibilidade dos dados, e isso tem sido possível com nossa parceria e integração na própria plataforma." Os desenvolvedores constroem ativamente o software do veículo para melhorar os problemas ao longo do caminho.

Seguindo em frente, a Rivian está vendo rápida adoção do Databricks Lakehouse em diferentes equipes, aumentando o número de usuários da plataforma de 5 para 250 em apenas um ano. Isso desbloqueou novos casos de uso, incluindo o uso de machine learning para otimizar a eficiência da bateria em temperaturas mais frias, aumentar a precisão dos sistemas de direção autônoma e atender a depósitos comerciais com dashboards de saúde do veículo para manutenção precoce e contínua. À medida que mais EAVs são enviados e sua frota de vans comerciais se expande, a Rivian continuará a aproveitar os dados gerados por seus EAVs para oferecer novas inovações e experiências de direção que revolucionam o transporte sustentável.



SETOR

Provedores de serviços de comunicação

SOLUÇÃO

Retenção de clientes, Previsão de perda de assinatura, Detecção de ameaças

PLATAFORMA

Lakehouse, Data science, Machine learning, Streaming de dados

NUVEM

Azure

SEÇÃO 4.6

Migração para a nuvem para atender melhor a milhões de clientes

300%

ROI da economia de custos operacionais e redução de custos

3X

Entrega mais rápida de casos de uso de ML/data science

Consistência em inovação é o que mantém os clientes com uma empresa de telecomunicações, e é por isso que a AT&T está entre as melhores. No entanto, o enorme sistema Hadoop legado local da AT&T acabou sendo complexo e dispendioso de gerenciar, impedindo a agilidade operacional e a eficiência e os recursos de engenharia. A necessidade de mudar para a nuvem para oferecer melhor suporte a centenas de milhões de assinantes era óbvia.

Ao migrar do Hadoop para o Databricks na nuvem Azure, a AT&T teve economias significativas nos custos operacionais. Além disso, o novo ambiente baseado na nuvem desbloqueou o acesso a petabytes de dados para análise correlativa e uma oferta de IA como serviço para mais de 2.500 usuários em mais de 60 unidades de negócios. A AT&T agora pode aproveitar todos os seus dados, sem sobrecarregar sua equipe de engenharia ou estourar os custos operacionais, para fornecer novos recursos e inovações aos milhões de usuários finais.

A tecnologia Hadoop adiciona complexidade operacional e custos desnecessários

A AT&T é uma gigante da tecnologia, com centenas de milhões de assinantes e ingere mais de 10 petabytes[a] de dados em toda a plataforma de dados todos os dias. Para aproveitar esses dados, ela tem uma equipe de mais de 2.500 usuários de dados em mais de 60 unidades de negócios para garantir que os negócios sejam baseados em dados — da criação de funções analíticas para garantir que as decisões sejam baseadas na melhor conscientização da situação orientada por dados à criação de modelos de ML que trazem inovações para seus clientes. Para atender a esses requisitos, a AT&T precisava democratizar e estabelecer uma versão única da verdade (SVOT) dos dados e, ao mesmo tempo, simplificar o gerenciamento da infraestrutura para aumentar a agilidade e reduzir os custos gerais.

No entanto, a infraestrutura física consumia muitos recursos. A combinação de uma configuração de hardware altamente complexa (12.500 fontes de dados e mais de 1.500 servidores) com uma arquitetura Hadoop on-premises se mostrou complexa de manter e cara de gerenciar. Não apenas os custos operacionais para dar suporte às cargas de trabalho eram altos, mas também havia custos de capital adicionais relacionados a data centers, licenciamento e outros. Até 70% da plataforma on-prem precisava ser priorizada para garantir que 50 mil jobs de pipeline de dados fossem bem-sucedidos e atendessem aos SLAs e aos objetivos de qualidade de dados. O tempo dos engenheiros era concentrado no gerenciamento de atualizações, na correção de problemas de desempenho ou simplesmente no provisionamento de recursos, e não em tarefas de maior valor. As restrições de recursos da infraestrutura física também levaram à serialização das atividades de data science, retardando a inovação. Outro obstáculo enfrentado na operacionalização de petabytes de dados foi o desafio de criar pipelines de dados de streaming para análise em tempo real, uma área que era fundamental para dar suporte a casos de uso inovadores necessários para atender melhor aos clientes.

Com esses problemas tecnológicos profundamente enraizados, a AT&T não estava na melhor posição para atingir suas metas de aumentar o uso de insights para melhorar a experiência do cliente e operar com mais eficiência. “Para realmente democratizar os dados em toda a empresa, precisávamos migrar para um ambiente de tecnologia nativa em nuvem”, disse Mark Holcomb, arquiteto de soluções da AT&T. “Isso liberou os recursos que estavam focados no gerenciamento da nossa infraestrutura e moveu-os para cima na cadeia de valor, além de ter liberado capital para investir em iniciativas voltadas para o crescimento.”

Uma jornada de migração perfeita para a Databricks

Como parte de sua due diligence, a AT&T realizou uma análise de custos abrangente e concluiu que a Databricks foi a mais rápida e alcançou o melhor preço/desempenho para pipelines de dados e cargas de trabalho de machine learning. A AT&T sabia que a migração seria uma tarefa enorme. Como tal, a equipe fez muito planejamento inicial — eles priorizaram a migração de suas maiores cargas de trabalho primeiro para reduzir imediatamente a área de cobertura de infraestrutura. Também decidiram migrar seus dados antes de migrar os usuários para garantir uma transição e experiência tranquilas para seus milhares de profissionais de dados.

> **A migração do Hadoop para o Databricks nos permite agregar mais valor aos nossos clientes e fazer isso de forma mais econômica e muito mais rápida do que antes.**
>
> **Mark Holcomb**
> arquiteto de soluções, AT&T

Eles passaram um ano desduplicando e sincronizando dados na nuvem antes de migrar qualquer usuário. Essa foi uma etapa fundamental para garantir a migração bem-sucedida de um ambiente multi-tenant tão grande e complexo de mais de 2.500 usuários de mais de 60 unidades de negócios e de suas cargas de trabalho. O processo de migração dos usuários ocorreu ao longo de nove meses e permitiu que a AT&T retirasse o hardware on-premises em paralelo com a migração para acelerar a economia o mais rápido possível. Além disso, devido à natureza horizontal e escalável da Databricks, a AT&T não precisava ter tudo em um ambiente contíguo. Separar dados e compute, e em várias contas e workspaces, garantiu que a análise funcionasse perfeitamente, sem limites de chamadas de API ou problemas de largura de banda e consumo claramente atribuídos às mais de 60 unidades de negócios.

Em suma, a AT&T migrou mais de 1.500 servidores, mais de 50.000 CPUs de produção, 12.500 fontes de dados e 300 esquemas. Todo o processo levou cerca de dois anos e meio. E ainda conseguiu gerenciar toda a migração com o equivalente a 15 recursos internos em tempo integral. “A Databricks foi uma valiosa colaboradora durante todo o processo”, disse Holcomb. “A equipe trabalhou em estreita colaboração conosco para resolver os recursos do produto e as preocupações de segurança para dar suporte ao nosso cronograma de migração.”

Databricks reduz o TCO e abre novos caminhos para a inovação

Um dos benefícios imediatos de migrar para a Databricks foi a enorme economia de custos. A AT&T conseguiu racionalizar cerca de 30% de seus dados identificando e não migrando dados subutilizados e duplicados. E priorizar a migração das maiores cargas de trabalho permitiu que metade do equipamento on-prem fosse racionalizado durante a migração. “Ao priorizar a migração de nossas cargas de trabalho mais intensivas em compute para a Databricks, conseguimos reduzir significativamente os custos, ao mesmo tempo que nos colocamos em posição de escalar com mais eficiência no futuro”, explicou Holcomb. O resultado é um ROI de migração de cinco anos previsto de 300% a partir da economia de despesas operacionais e da redução de custos (por exemplo, não é necessário atualizar o hardware do data center).

Com os dados prontamente disponíveis e os meios para analisar os dados em qualquer escala, as equipes de data scientists e analistas de dados dos cidadãos agora podem passar mais tempo inovando, em vez de serializar os esforços de análise ou esperar que a engenharia forneça os recursos necessários, ou fazer com que os data scientists gastem seu valioso tempo em análises menos complexas ou menos perspicazes. Os data scientists agora são capazes de colaborar de forma mais eficaz e acelerar os fluxos de trabalho de machine learning para que as equipes possam entregar valor mais rapidamente, com um tempo 3 vezes mais rápido para a entrega de novos casos de uso de data science.

“Historicamente, teríamos tido operações em um sistema e análises em um sistema separado”, disse Holcomb. “Agora podemos fazer mais casos de uso, como análises operacionais, em uma plataforma que promove a colaboração entre equipes, reduz custos e melhora a consistência das respostas.” Desde a migração para a Databricks, a AT&T agora tem uma versão única da verdade para criar novas oportunidades orientadas por dados, incluindo uma plataforma de análise self-service de IA como serviço que permitirá novos fluxos de receita e ajudará a continuar oferecendo inovações excepcionais aos milhões de clientes.

Sobre a Databricks

Databricks é a empresa de dados e IA. Mais de 10.000 organizações em todo o mundo — incluindo Comcast, Condé Nast e mais de 50% das empresas Fortune 500 — contam com a Plataforma Databricks Lakehouse para unificar seus dados, análises e IA. A Databricks está sediada em São Francisco, com escritórios em todo o mundo. Fundada pelos criadores originais do Apache Spark™, Delta Lake e MLflow, a Databricks tem a missão de ajudar as equipes de dados a resolver os problemas mais difíceis do mundo. Para saber mais, siga a Databricks no Twitter, LinkedIn e Facebook.

COMECE SUA AVALIAÇÃO GRATUITA

Entre em contato conosco para obter uma demonstração personalizada
databricks.com/contact

databricks