O'REILLY®

Inżynieria danych w praktyce

Kluczowe koncepcje i najlepsze technologie

Helion

Joe Reis, Matt Housley

Tytuł oryginału: Fundamentals of Data Engineering: Plan and Build Robust Data Systems

Tłumaczenie: Radosław Meryk

ISBN: 978-83-8322-154-0

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *https://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
https://helion.pl/user/opinie/indapr
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

ROZDZIAŁ 2.

Cykl życia inżynierii danych

Głównym celem tej książki jest zachęcenie czytelników do wyjścia poza postrzeganie inżynierii danych jako specyficznego zbioru technologii związanych z danymi. W przestrzeni danych można zaobserwować eksplozję nowych technologii i nowych praktyk związanych z danymi. Charakteryzują się one coraz większym poziomem abstrakcji i są coraz łatwiejsze w użyciu. Ze względu na wyższy poziom technicznej abstrakcji inżynierowie danych coraz częściej spełniają funkcję *inżynierów cyklu życia danych*, myślących i działających w kategoriach *zasad* zarządzania cyklem życia danych.

W tym rozdziale zapoznasz się z **cyklem życia inżynierii danych**, który jest głównym tematem tej książki. Cykl życia inżynierii danych to framework opisujący inżynierię danych „od kołyski do grobu". W rozdziale zostaną omówione również zasadnicze nurty cyklu życia inżynierii danych, stanowiące kluczowe podstawy dla wszystkich działań związanych z inżynierią danych.

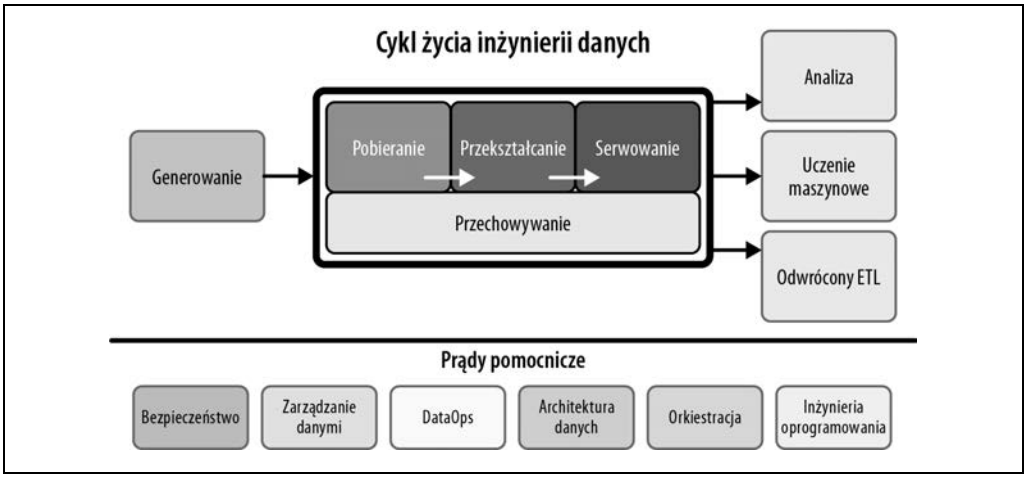
Czym jest cykl życia inżynierii danych?

Cykl życia inżynierii danych składa się z faz, w których składniki nieprzetworzonych danych są przekształcane w użyteczny produkt końcowy, gotowy do wykorzystania przez analityków, badaczy danych (ang. *data scientists*), inżynierów uczenia maszynowego i innych specjalistów. W tym rozdziale wprowadzono główne etapy cyklu życia inżynierii danych, z akcentem na omówienie podstawowych pojęć związanych z każdym etapem, natomiast szczegóły zostaną opisane w rozdziałach dalszych.

Cykl życia inżynierii danych można podzielić na pięć etapów (górna część rysunku 2.1):

- Generowanie.
- Przechowywanie.
- Pozyskiwanie.
- Przekształcanie.
- Serwowanie danych.

Cykl życia inżynierii danych rozpoczyna się od pozyskania danych z systemów źródłowych i ich zapisania w magazynie danych. Następnie dane są przekształcane, po czym przechodzimy do głównego celu — serwowania danych analitykom, badaczom danych, inżynierom uczenia maszynowego



Rysunek 2.1. Komponenty i zasadnicze nurty cyklu życia inżynierii danych

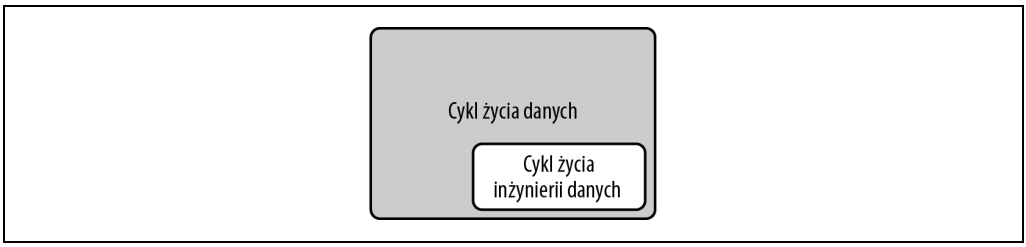
i innym specjalistom. W gruncie rzeczy z przechowywaniem mamy do czynienia przez cały cykl życia. Dane przepływają bowiem od początku do końca — stąd na diagramie „faza" przechowywania stanowi podstawę dla innych faz.

Ogólnie rzecz biorąc, etapy środkowe — przechowywanie, pozyskiwanie, przekształcanie — czasami mieszają się ze sobą. Nie ma w tym niczego złego. Chociaż można wyszczególnić części składowe cyklu życia inżynierii danych, nie zawsze tworzą one schludny, ciągły przepływ. Niektóre etapy cyklu życia inżynierii danych mogą się powtarzać, występować w innej kolejności niż typowa, nakładać się na siebie lub łączyć ze sobą w interesujący i nieoczekiwany sposób.

Podstawą są **główne nurty** (ang. *undercurrents*) — przedstawione w dolnej części rysunku 2.1 — obejmujące wiele faz cyklu życia inżynierii danych: bezpieczeństwo, zarządzanie danymi, DataOps, architektura danych, orkiestracja i inżynieria oprogramowania. Bez tych nurtów nie może właściwie funkcjonować żadna część cyklu życia inżynierii danych.

Cykl życia danych a cykl życia inżynierii danych

Być może zastanawiasz się nad różnicą między ogólnym pojęciem cyklu życia danych a cyklem życia inżynierii danych. Między tymi dwoma pojęciami istnieje subtelna różnica. O ile pełny cykl życia danych obejmuje dane w całym okresie ich istnienia, o tyle cykl życia inżynierii danych koncentruje się na etapach zarządzanych przez inżyniera danych (rysunek 2.2).



Rysunek 2.2. Cykl życia inżynierii danych jest podzbiorem całego cyklu życia danych

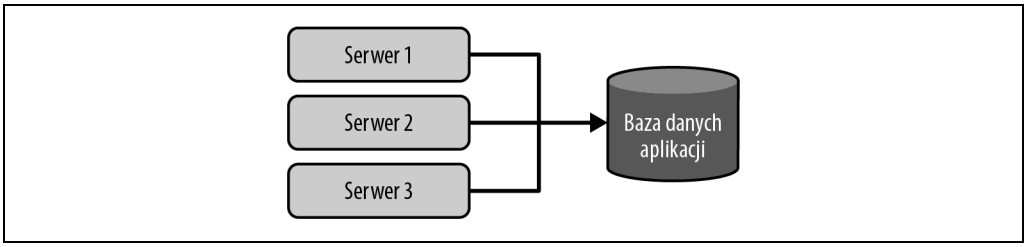
Generowanie — systemy źródłowe

System źródłowy jest źródłem danych wykorzystywanych w cyklu życia inżynierii danych. Na przykład systemem źródłowym może być urządzenie IoT, kolejka komunikatów aplikacji lub transakcyjna baza danych. Inżynierowie danych korzystają z danych pochodzących z systemu źródłowego, ale zazwyczaj nie są właścicielami samego systemu źródłowego ani go nie kontrolują. Inżynierowie danych muszą mieć praktyczną wiedzę na temat sposobu działania systemów źródłowych, sposobu generowania danych, częstotliwości i tempa zmian danych oraz różnorodności generowanych danych.

Inżynierowie muszą również utrzymywać z właścicielami systemów źródłowych otwarty kanał komunikacyjny w celu wymiany informacji na temat zmian, które mogą przerwać potoki i zakłócić analizy. Kod aplikacji może zmienić strukturę danych wykorzystywanych w systemie. Może się również zdarzyć, że zespół aplikacji przeprowadzi migrację backendu do całkowicie nowej technologii bazy danych.

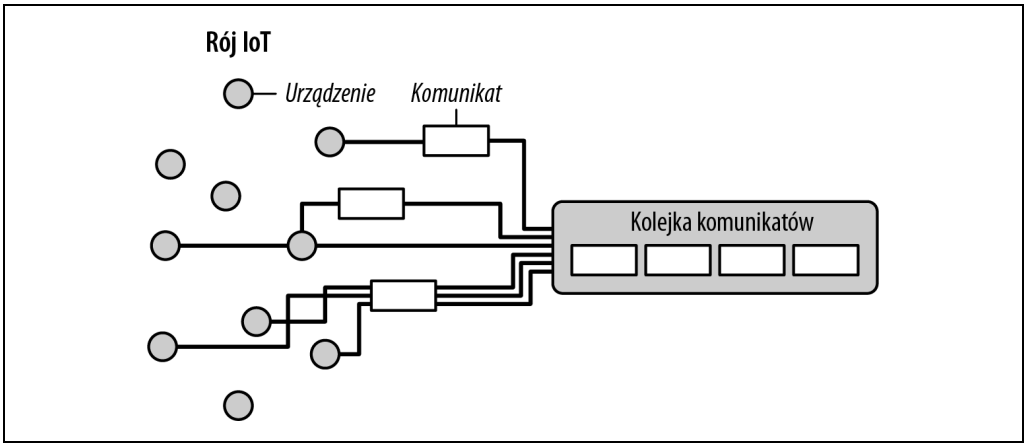
Poważnym wyzwaniem w inżynierii danych jest olbrzymia liczba systemów źródeł danych, z którymi muszą pracować inżynierowie i które muszą rozumieć. W celu zilustrowania tematu przyjrzymy się dwóm powszechnie stosowanym systemom źródłowym — jednemu bardzo tradycyjnemu (baza danych aplikacji) oraz drugiemu, nowszemu (roje urządzeń IoT — ang. *IoT swarms*).

Tradycyjny system źródłowy z kilkoma serwerami aplikacji obsługiwanymi przez bazę danych zilustrowano na rysunku 2 3. Ten wzorzec systemu źródłowego stał się popularny w latach osiemdziesiątych ubiegłego wieku. Zbiegł się z olbrzymim sukcesem systemów zarządzania relacyjnymi bazami danych (ang. *relational database management systems* — RDBMS). Wzorzec aplikacja + baza danych pomimo stosowania wielu nowoczesnych praktyk tworzenia oprogramowania pozostaje popularny także dziś. Na przykład aplikacje często składają się z wielu par usługa i baza danych. Mają więc architekturę mikrousług, a nie pojedynczego monolitu.



Rysunek 2.3. Przykład systemu źródłowego: baza danych aplikacji

Przyjrzyjmy się innemu przykładowi systemu źródłowego. Na rysunku 2.4 pokazano rój IoT: flota urządzeń (oznaczonych okręgami) wysyła komunikaty (oznaczone prostokątami) do centralnego systemu zbierania danych. Wraz z upowszechnianiem się urządzeń IoT, takich jak czujniki, inteligentne urządzenia i wiele innych, systemy źródłowe w postaci rojów urządzeń IoT stają się coraz bardziej powszechne.



Rysunek 2.4. Przykład systemu źródłowego: rój IoT i kolejka komunikatów

Ocena systemów źródłowych — najważniejsze dylematy inżynierskie

Podczas oceny systemów źródłowych należy wziąć pod uwagę wiele kwestii, w tym sposób, w jaki system obsługuje pozyskiwanie danych, ustala ich stan i generuje wyniki. Poniżej znajduje się podstawowy zestaw pytań zmierzających do oceny systemów źródłowych, które inżynierowie danych powinni wziąć pod uwagę:

- Jakie są podstawowe cechy źródła danych? Czy jest to aplikacja? Czy raczej jest to rój urządzeń IoT?
- Jaki jest model trwałości danych w systemie źródłowym? Czy dane są przechowywane przez dłuższy czas, czy też są tymczasowe i po wykorzystaniu są szybko usuwane?
- W jakim tempie są generowane dane? Ile zdarzeń na sekundę zachodzi w systemie źródłowym? Ile gigabajtów danych jest przetwarzanych w ciągu godziny?
- Jakiego poziomu spójności mogą oczekiwać inżynierowie danych od danych wynikowych? Jak często podczas przeprowadzania kontroli jakości danych wynikowych występują niespójności w danych, takie jak wartości `null`, nieprawidłowy format itp.?
- Jak często występują błędy?
- Czy dane zawierają duplikaty?
- Czy niektóre wartości danych docierają późno, być może znacznie później niż inne wiadomości generowane natychmiast?
- Jaki jest schemat pozyskiwanych danych? Czy inżynierowie danych, aby uzyskać pełny obraz danych, są zmuszeni do łączenia danych z kilku tabel lub nawet z kilku systemów?
- W jaki sposób są wprowadzane zmiany schematu (na przykład dodanie nowej kolumny) oraz w jakiś sposób są o tym powiadamiani interesariusze pracujący w dolnej części strumienia przetwarzania?
- Jak często dane powinny być pobierane z systemu źródłowego?

- Czy w przypadku systemów stanowych (np. bazy danych monitorującej informacje o koncie klienta) dane są dostarczane jako okresowe migawki lub zdarzenia aktualizacji z systemów przechwytywania zmian danych (ang. *change data capture* — CDC)? Jaka jest logika wykonywania zmian i jak są one śledzone w źródłowej bazie danych?
- Kto (co) jest dostawcą danych przesyłającym je do dalszego wykorzystania?
- Czy odczyt danych ze źródła wpływa na wydajność źródła danych?
- Czy system źródłowy zależy od danych z górnej części strumienia przetwarzania? Jakie są cechy systemów działających w górnej części strumienia przetwarzania?
- Czy istnieją mechanizmy kontroli jakości danych pozwalające sprawdzić, czy nie ma spóźnionych lub brakujących danych?

Źródła generują dane wykorzystywane przez systemy z dolnej części strumienia przetwarzania, w tym tworzone ręcznie arkusze kalkulacyjne, czujniki IoT oraz aplikacje webowe i mobilne. Każde źródło danych ma swoją unikatową objętość i rytm generowania informacji. Inżynier danych powinien wiedzieć, w jaki sposób źródło generuje dane, oraz znać wszystkie związane z tym zawiłości lub niuanse. Inżynierowie danych muszą również rozumieć ograniczenia systemów źródłowych, z którymi wchodzą w interakcje. Na przykład czy zapytania analityczne dotyczące źródłowej bazy danych aplikacji spowodują rywalizację o zasoby i problemy z wydajnością?

Jednym z najtrudniejszych niuansów związanych ze źródłami danych jest schemat danych. **Schemat** definiuje hierarchiczną organizację danych. Z logicznego punktu widzenia dane możemy rozpatrywać na poziomie całego systemu źródłowego — schodząc do pojedynczych tabel, a nawet do struktury poszczególnych pól. Schemat danych przesyłanych z systemów źródłowych jest obsługiwany na różne sposoby. Dwie popularne opcje to dane bez struktury oraz stały schemat.

Z faktu, że dane *nie mają struktury*, nie wynika, że nie mają one schematu. Oznacza to raczej, że aplikacja definiuje schemat podczas zapisywania danych, niezależnie od tego, czy jest to kolejka komunikatów, płaski plik, obiekt blob, czy baza danych dokumentów, taka jak MongoDB. Bardziej tradycyjny model oparty na relacyjnych bazach danych wykorzystuje *stały schemat* zdefiniowany w bazie danych. Z tym schematem muszą być zgodne zapisy aplikacji.

Każdy z tych modeli stanowi dla inżynierów danych osobne wyzwanie. Schematy zmieniają się w czasie; w gruncie rzeczy w zwinnym podejściu do tworzenia oprogramowania ewolucja schematu jest zalecana. Kluczową częścią pracy inżyniera danych jest wprowadzanie surowych danych do schematu systemu źródłowego i przekształcanie ich w cenne dane wynikowe wykorzystywane do analiz. Gdy schemat źródłowy zmienia się, zadanie to staje się trudniejsze.

Systemy źródłowe zostaną opisane bardziej szczegółowo w rozdziale 5., z kolei schematy danych i modelowanie danych zostaną opisane odpowiednio w rozdziałach 6. i 8.

Przechowywanie

Do przechowywania danych potrzebne jest miejsce. Kluczem do sukcesu w dalszej części cyklu życia danych jest wybór odpowiedniego rozwiązania pamięci masowej. Z różnych powodów jest to również jeden z najbardziej skomplikowanych etapów cyklu życia danych. Po pierwsze architektury

danych w chmurze często wykorzystują kilka rozwiązań pamięci masowej. Po drugie niewiele rozwiązań do przechowywania danych działa wyłącznie jako pamięć masowa. Wiele z nich obsługuje złożone zapytania obejmujące transformacje; do obsługi zaawansowanych zapytań zdolne są nawet rozwiązania obiektowych pamięci masowych — np. Amazon S3 Select (*https://oreil.ly/XzcKh*). Po trzecie, chociaż przechowywanie danych jest etapem cyklu życia inżynierii danych, często dotyczy również innych etapów, takich jak pozyskiwanie, przekształcanie i serwowanie.

Przechowywanie danych jest wykorzystywane w całym cyklu życia inżynierii danych. Często występuje w wielu miejscach potoku danych, a role systemów pamięci masowej krzyżują się z rolami systemów źródłowych oraz mechanizmami pozyskiwania przekształcania i serwowania danych. Pod wieloma względami sposób przechowywania danych wpływa na sposób ich wykorzystania na wszystkich etapach cyklu życia inżynierii danych. Na przykład hurtownie danych w chmurze mogą przechowywać dane, przetwarzać dane w potokach i udostępniać je analitykom. Frameworki przesyłania strumieniowego, takie jak Apache Kafka i Pulsar, mogą działać jednocześnie jako systemy pozyskiwania, przechowywania oraz jako mechanizmy zapytań o wiadomości, przy czym standardową warstwą do transmisji danych jest w nich magazyn obiektowy.

Ocena systemów pamięci masowej — najważniejsze dylematy inżynierskie

Oto kilka kluczowych pytań, które należy zadać przy wyborze systemu pamięci masowej dla hurtowni danych, jeziora danych, bazy danych lub magazynu obiektów:

- Czy wybrane rozwiązanie pamięci masowej spełnia wymagane przez architekturę warunki dotyczące szybkości zapisu i odczytu danych?
- Czy magazyn danych tworzy wąskie gardło dla procesów działających w dolnej części strumienia przetwarzania?
- Czy istnieje jasność co do sposobu działania wybranej technologii pamięci masowej? Czy korzystasz z systemu pamięci masowej w sposób optymalny, czy też wykorzystujesz go w nienaturalny sposób? Na przykład czy w magazynie obiektowym stosujesz częste, losowe aktualizacje (to antywzorzec, którego stosowanie powoduje znaczące obniżenie wydajności)?
- Czy wybrany system pamięci masowej będzie w stanie w przyszłości obsłużyć przewidywaną skalę? Należy wziąć pod uwagę wszystkie ograniczenia pojemności w systemie pamięci masowej: całkowitą dostępną pamięć, szybkość operacji odczytu, wolumin zapisu itp.
- Czy użytkownicy i procesy działające w dolnej części strumienia przetwarzania będą w stanie pobierać dane w zgodzie z warunkami określonymi w umowie dotyczącej poziomu usług (ang. *service level agreement* — SLA)?
- Czy przechwytujesz metadane dotyczące ewolucji schematu, przepływów danych, pochodzenia danych itp.? Metadane mają znaczący wpływ na możliwości wykorzystania danych. Metadane mogą być uznane za inwestycję w przyszłość, ponieważ znacznie zwiększają możliwości wykrywania danych oraz poziom wiedzy w organizacji, co pozwala uprościć przyszłe zmiany w projekcie i architekturze.
- Czy jest to rozwiązanie dotyczące wyłącznie pamięci masowej (obiektowa pamięć masowa), czy też obsługuje ono złożone wzorce zapytań (np. hurtownia danych w chmurze)?

- Czy system pamięci masowej jest niezależny od schematu (obiektowa pamięć masowa)? Czy ma elastyczny schemat (Cassandra)? A może jest to schemat wymuszony (hurtownia danych w chmurze)?
- W jaki sposób w procesie „opieki nad danymi" (ang. *data governance*) monitorujesz dane podstawowe, jakość danych z tzw. złotych rekordów i rodowód danych? (Więcej informacji na ten temat znajdziesz w podrozdziale „Zarządzanie danymi").
- Jak radzisz sobie ze zgodnością z przepisami oraz suwerennością danych? Na przykład czy możesz przechowywać swoje dane w określonych lokalizacjach geograficznych, ale nie możesz ich przechowywać w innych?

Częstotliwość dostępu do danych

Nie do wszystkich danych uzyskuje się dostęp w ten sam sposób. Wzorce pobierania znacznie różnią się pomiędzy sobą w zależności od rodzaju przechowywanych danych oraz konkretnych zapytań. Wiąże się z tym pojęcie „temperatury" danych. Temperaturę danych określa częstotliwość dostępu do nich.

Dane, do których najczęściej sięgamy, są nazywane **gorącymi** (ang. *hot data*). Gorące dane są zwykle pobierane wiele razy dziennie, a czasami nawet kilka razy na sekundę — na przykład w systemach obsługujących żądania użytkowników. Dane te powinny być przechowywane tak, aby możliwe było ich szybkie pobieranie, gdzie znaczenie przymiotnika „szybkie" jest względne w stosunku do przypadku użycia. Dostęp do danych **letnich** (ang. *lukewarm data*) powinien być możliwy co jakiś czas — na przykład co tydzień lub co miesiąc.

Zapytania o **zimne dane** (ang. *cold data*) są rzadkie, dlatego mogą być one przechowywane w archiwach. Zimne dane często są składowane dla celów zgodności z przepisami lub na potrzeby odtwarzania po awarii w innym systemie. W „dawnych czasach" zimne dane były przechowywane na taśmach i wysyłane do odległych archiwów. Dostawcy środowisk chmurowych oferują wyspecjalizowane warstwy pamięci masowej z bardzo niskimi miesięcznymi kosztami składowania, ale ze stosunkowo wysokimi cenami za pobieranie danych.

Wybór systemu pamięci masowej

Z jakiego rodzaju rozwiązania pamięci masowej powinieneś skorzystać? Wszystko zależy od przypadku użycia, ilości danych, częstotliwości ich pozyskiwania, formatu, a także rozmiaru pozyskiwanych danych — są to w gruncie rzeczy kluczowe pytania wymienione na liście zamieszczonej w poprzednim punkcie. Nie istnieje jedno pasujące do wszystkich przypadków rozwiązanie pamięci masowej. Z każdą technologią składowania danych wiążą się pewne kompromisy. Istnieją niezliczone odmiany technologii pamięci masowych. Ich liczba jest przytłaczająca, dlatego wybór najlepszej opcji odpowiadającej określonej architekturze danych jest bardzo trudny.

Więcej informacji dotyczących najlepszych praktyk i podejść do wyboru pamięci masowej, a także implikacje wyboru rozwiązania pamięci masowej w kontekście innych faz cyklu życia danych można znaleźć w rozdziale 6.

Pozyskiwanie

Gdy już wiesz, czym jest źródło danych, znasz charakterystykę używanego systemu źródłowego oraz wiesz, jak są składowane dane, powinieneś poznać sposoby ich pobierania. Kolejnym etapem cyklu życia inżynierii danych jest pozyskiwanie danych z systemów źródłowych.

Z naszych doświadczeń wynika, że systemy źródłowe i pozyskiwanie danych stanowią najważniejsze wąskie gardła w cyklu życia inżynierii danych. Zazwyczaj nie mamy bezpośredniej kontroli nad systemami źródłowymi, zatem musimy być przygotowani, że mogą one losowo przestać odpowiadać lub od czasu do czasu mogą dostarczać dane niskiej jakości. Usługa pozyskiwania danych może również, z wielu powodów, w tajemniczy sposób przestać działać. W rezultacie przepływ danych zatrzymuje się lub dostarczane dane są niewystarczająco kompletne do realizacji zadań związanych z przechowywaniem, przetwarzaniem i serwowaniem.

Zawodne systemy źródłowe i wadliwe systemy pozyskiwania danych wywierają efekt falowania na cały cykl życia inżynierii danych. Ale jeśli odpowiedziałeś twierdząco na większość wymienionych wcześniej pytań dotyczących systemów źródłowych, to Twój system pozyskiwania danych jest prawdopodobnie w dobrej kondycji.

Najważniejsze dylematy inżynierskie dotyczące fazy pozyskiwania danych

Oto lista podstawowych pytań dotyczących fazy pozyskiwania danych, które powinieneś zadać podczas przygotowywania się do tworzenia architektury lub budowy systemu danych:

- Jakie są przypadki użycia danych, które pozyskujesz? Czy istnieje możliwość wielokrotnego wykorzystania danych zamiast tworzenia wielu wersji tego samego zestawu danych?
- Czy systemy generują i pozyskują dane niezawodnie i czy dane są dostępne, gdy ich potrzebujesz?
- Jakie jest miejsce docelowe danych po ich pozyskaniu?
- Jak często musisz uzyskiwać dostęp do danych?
- W jakiej objętości dane zazwyczaj docierają do Twojego systemu?
- W jakim są formacie? Czy Twoje systemy przechowywania i przekształcania danych działające w dolnej części strumienia przetwarzania potrafią obsłużyć ten format?
- Czy dane źródłowe są w dobrym stanie, umożliwiającym ich natychmiastowe dalsze wykorzystanie? Jeśli tak, to przez jaki czas tak będzie i co może spowodować, że dane przestaną nadawać się do użytku?
- Czy dane pochodzące ze źródła strumieniowego muszą zostać przekształcone przed dotarciem do miejsca docelowego? Czy właściwa byłaby transformacja danych „w locie”, polegająca na przekształceniu danych w samym strumieniu?

To tylko kilka czynników, które powinieneś wziąć pod uwagę podczas pobierania danych. Wymienione pytania oraz kwestie dodatkowe z nimi związane zostaną szczegółowo omówione w rozdziale 7. Zanim przejdziemy do dalszej części rozdziału, zwróćmy uwagę na dwa główne pojęcia związane z pozyskiwaniem danych: przetwarzanie wsadowe a przesyłanie strumieniowe oraz systemy typu *push* (dosłownie: wypychaj) kontra systemy typu *pull* (dosłownie: ciągnij).

Pozyskiwanie wsadowe a strumieniowe

Prawie wszystkie dane, z którymi mamy do czynienia, są zwykle przesyłane *strumieniowo*. Dane są prawie zawsze wytwarzane i aktualizowane na poziomie źródła. To po prostu specjalistyczny i wygodny sposób przetwarzania strumienia danych w dużych porcjach — na przykład obsługa w jednej partii danych z całego dnia.

Pozyskiwanie strumieniowe pozwala dostarczać dane do systemów działających w dolnej części strumienia przetwarzania — niezależnie od tego, czy są to aplikacje zewnętrzne, bazy danych, czy systemy analityczne — w sposób ciągły oraz w czasie rzeczywistym. W tym przypadku *czas rzeczywisty* (lub *zbliżony do rzeczywistego*) oznacza, że dane są dostępne dla systemów działających w dolnej części strumienia przetwarzania wkrótce po ich wytworzeniu (na przykład mniej niż sekundę później). Opóźnienie wymagane do zakwalifikowania danych jako danych czasu rzeczywistego różni się w zależności od domeny i konkretnych wymagań.

Dane pozyskiwane wsadowo są pobierane co określony przedział czasu lub wtedy, gdy dane osiągną wstępnie ustalony próg rozmiaru. Pobieranie partiami można porównać do przechodzenia przez jednokierunkowe drzwi: gdy dane zostaną podzielone na partie, występuje naturalne opóźnienie ich dostarczania do dalszych konsumentów. Ze względu na ograniczenia starszych systemów pobieranie partiami było przez długi czas domyślnym sposobem pozyskiwania danych. Przetwarzanie wsadowe pozostaje niezwykle popularnym sposobem pozyskiwania danych do dalszego wykorzystania, zwłaszcza w zadaniach analitycznych oraz uczeniu maszynowym.

Jednak ze względu na oddzielenie w wielu systemach pamięci masowej od obliczeń oraz z powodu powszechnej dostępności platform do przesyłania strumieniowego i przetwarzania zdarzeń ciągłe przetwarzanie strumieni danych stało się znacznie bardziej dostępne i zyskuje coraz większą popularność. Wybór w dużej mierze zależy od przypadku użycia i oczekiwań dotyczących terminowości danych.

Kluczowe kwestie dotyczące wyboru pomiędzy pozyskiwaniem wsadowym a strumieniowym

Czy powinieneś domyślnie wybierać pozyskiwanie strumieniowe? Pomimo atrakcyjności pozyskiwania strumieniowego trzeba wziąć pod uwagę wiele aspektów, decydując się na wybór tej opcji. Poniżej zamieszczono listę pytań, które należy sobie zadać przy ustalaniu, czy uzyskiwanie strumieniowe jest właściwą opcją, czy raczej należy zastosować pozyskiwanie wsadowe:

- Czy systemy pamięci masowej są w stanie obsłużyć tempo danych pozyskiwanych w czasie rzeczywistym?
- Czy w przypadku pozyskiwania danych w czasie rzeczywistym potrzebne jest odmierzanie czasu z dokładnością do milisekund? A może sprawdziłoby się podejście pozyskiwania danych w mikro-partiach, gdzie dane byłyby gromadzone i przekazywane na przykład co minutę?
- Jakie są przypadki użycia dla pozyskiwania strumieniowego? Jakie konkretne korzyści mogę osiągnąć dzięki zastosowaniu przesyłania strumieniowego? Jakie działania wykonywane na danych pozyskiwanych w czasie rzeczywistym wprowadziłyby usprawnienia w porównaniu z pozyskiwaniem danych partiami?

- Czy podejście polegające na domyślnym wyborze pozyskiwania strumieniowego będzie kosztować więcej pod względem czasu, pieniędzy, kosztów utrzymania, przestojów oraz kosztów alternatywnych w porównaniu z pozyskiwaniem danych partiami?
- Czy w przypadku awarii infrastruktury mój potok strumieniowy oraz mój system będą działać niezawodnie?
- Jakie narzędzia są najbardziej odpowiednie dla danego przypadku użycia? Czy powinienem skorzystać z usługi zarządzanej (Amazon Kinesis, Google Cloud Pub/Sub, Google Cloud Dataflow), czy też tworzyć własne egzemplarze systemów Kafka, Flink, Spark, Pulsar itp.? Kto będzie zarządzał moimi egzemplarzami systemów, jeśli zdecyduję się na drugą opcję? Jakie koszty trzeba będzie ponieść i jakie kompromisy zastosować?
- Jakie korzyści uzyskam z prognoz online i ewentualnie z zastosowania technik ciągłego szkolenia, jeśli zdecyduję się na wdrożenie modelu uczenia maszynowego?
- Czy pozyskiwane dane pochodzą z produkcyjnego egzemplarza systemu źródłowego? Jeśli tak, to jaki będzie wpływ mojego procesu pozyskiwania na system źródłowy?

Jak widać, domyślny wybór pozyskiwania strumieniowego może wydawać się dobrym pomysłem, ale jego zastosowanie nie zawsze jest proste — z natury występują dodatkowe koszty i dodatkowa złożoność. Wiele świetnych frameworków pozyskiwania obsługuje oba style: zarówno pozyskiwanie wsadowe, jak i pozyskiwanie w mikropartiach. Uważamy, że pozyskiwanie wsadowe świetnie nadaje się dla wielu typowych przypadków użycia, takich jak szkolenie modelu i tworzenie cotygodniowych raportów. Zastosuj przesyłanie strumieniowe w czasie rzeczywistym tylko po zidentyfikowaniu przypadku użycia biznesowego, który uzasadnia skorzystanie z tej opcji.

Pozyskiwanie typu push kontra pozyskiwanie typu pull

W modelu pozyskiwania danych typu *push* system źródłowy zapisuje dane do miejsca docelowego, niezależnie od tego, czy jest to baza danych, magazyn obiektów, czy system plików. W modelu pozyskiwania danych typu *pull* dane są ściągane z systemu źródłowego. Granica między paradygmatami *push* i *pull* jest dość rozmyta; podczas przechodzenia przez różne etapy potoku danych dane są często zarówno wypychane, jak i ściągane.

Rozważmy dla przykładu proces wyodrębniania, przekształcania i ładowania danych (ang. *extract*, *transform*, *load* — ETL), powszechnie wykorzystywany w przepływach pracy z pozyskiwaniem partiami. Część wyodrębniania (ang. *extract* — E) oznacza, że mamy do czynienia z modelem pozyskiwania typu *pull*. W tradycyjnym przetwarzaniu ETL system pozyskiwania odpytuje zgodnie z ustalonym harmonogramem bieżącą migawkę tabeli źródłowej. Więcej informacji o ETL oraz o fazach wyodrębniania, ładowania i przekształcania znajdziesz w dalszej części tej książki.

W ramach innego przykładu rozważmy ciągłe przechwytywanie zmian danych (ang. *change data capturing* — CDC), które można zrealizować na kilka sposobów. Jedna z typowych metod polega na wyzwalaniu komunikatu za każdym razem, gdy w źródłowej bazie danych zmienia się wiersz. Komunikat ten jest *wypychany* do kolejki, z której jest odbierany przez system pozyskiwania danych. Inna popularna metoda CDC wykorzystuje binarne logi, w których jest rejestrowany każdy commit do bazy danych. Baza danych *wypycha* informacje do swoich logów. System pozyskiwania

danych odczytuje logi, ale poza tym nie wchodzi bezpośrednio w interakcje z bazą danych. Powoduje to niewielkie dodatkowe obciążenie źródłowej bazy danych lub nie wprowadza żadnego dodatkowego obciążenia. W niektórych wersjach wsadowego pozyskiwania CDC wykorzystywany jest wzorzec *pull*. Na przykład w systemie CDC opartym na znacznikach czasu wysyłane są kwerendy do źródłowej bazy danych i pobierane wiersze, które zmieniły się od czasu poprzedniej aktualizacji.

W przypadku pozyskiwania strumieniowego dane pomijają bazę danych w warstwie backend i są wypychane bezpośrednio do punktu końcowego. Dane są zazwyczaj buforowane przez platformę przesyłania strumieniowego. Taki wzorzec jest przydatny do obsługi flot czujników IoT emitujących odczytywane przez czujniki dane. Ten wzorzec nie opiera się na utrzymywaniu bieżącego stanu z bazy danych. Zamiast tego każdy zarejestrowany odczyt jest traktowany jako zdarzenie. Ten wzorzec również zyskuje w aplikacjach na popularności, ponieważ upraszcza przetwarzanie w czasie rzeczywistym, pozwala programistom aplikacji dostosować swoje komunikaty do dalszych analiz i znacznie upraszcza pracę inżynierów danych.

Najlepsze praktyki i techniki pozyskiwania danych zostaną szczegółowo omówione w rozdziale 7. Natomiast teraz przejdźmy do kolejnego etapu w cyklu życia inżynierii danych — przekształcania danych.

Przekształcanie

Gdy dane zostaną pozyskane i zeskładowane, trzeba z nimi coś zrobić. Kolejnym etapem cyklu życia inżynierii danych jest **przekształcanie** (ang. *transformation*). Na tym etapie dane muszą zostać zmienione z oryginalnego formatu do postaci przydatnej do wykorzystania w przypadkach użycia w dolnej części strumienia przetwarzania. Bez odpowiednich przekształceń dane będą bezużyteczne i nie będą nadawały się do wykorzystania w raportach, analizach lub algorytmach uczenia maszynowego. Zazwyczaj w fazie przekształcania zaczyna się tworzenie wartości dla użytkowników z dolnej części strumienia przetwarzania.

Natychmiast po pozyskaniu danych wykonywane są podstawowe przekształcenia, w wyniku których dane są mapowane na poprawne typy (na przykład następuje przekształcenie pozyskanych danych tekstowych na typy liczbowe i daty). W wyniku tych działań dane rekordów uzyskują poprawne formaty, a dane nieprawidłowe są usuwane. Na późniejszych etapach transformacji mogą być wykonywane przekształcenia schematu danych i stosowane normalizacje. Następnie w dolnej części strumienia przetwarzania można zastosować wielkoskalowe agregacje na potrzeby generowania raportów bądź wyodrębnić cechy na potrzeby procesów uczenia maszynowego.

Kluczowe zagadnienia dotyczące fazy przekształcania danych

Podczas przygotowań do przekształcenia danych w cyklu życia inżynierii danych warto wziąć pod uwagę następujące kwestie:

- Jaki jest koszt przekształcenia danych i zwrot z inwestycji (ang. *return of investment* — ROI)? Jaka wartość biznesowa wiąże się z przekształceniem danych?
- Czy przekształcenie danych jest proste i maksymalnie odizolowane od innych operacji?
- Z jakimi regułami biznesowymi są związane transformacje?

Dane mogą być przekształcane partiami lub „w locie", podczas przesyłania strumieniowego. Jak wspomniano w punkcie „Pozyskiwanie", cykl życia praktycznie wszystkich danych rozpoczyna się od ciągłego strumienia; przetwarzanie partiami to tylko wyspecjalizowany sposób przetwarzania strumieniowego. Transformacje partiami są niezwykle popularne, ale biorąc pod uwagę rosnącą popularność rozwiązań przetwarzania strumieniowego i ogólny wzrost ilości danych przesyłanych strumieniowo, spodziewamy się, że popularność transformacji strumieniowych będzie nadal rosła. Być może w niektórych domenach wkrótce całkowicie zastąpi przetwarzanie partiami.

Z logicznej perspektywy przekształcanie można potraktować jako samodzielny obszar cyklu życia inżynierii danych, ale w praktyce realia cyklu życia inżynierii danych mogą być znacznie bardziej złożone. Transformacje często są powiązane także z innymi fazami cyklu życia. Zazwyczaj dane są przekształcane w systemach źródłowych lub „w locie", podczas pozyskiwania. Na przykład system źródłowy przed przekazaniem zdarzenia do procesu pozyskiwania może dodać do rekordu sygnaturę czasową. Zanim rekord w potoku przesyłania strumieniowego zostanie wysłany do magazynu danych, może zostać „wzbogacony" o dodatkowe pola i obliczenia. Transformacje są wszechobecne w różnych fazach cyklu życia. Przygotowywanie danych, ich przetwarzanie i czyszczenie — te zadania transformacyjne stanowią dla końcowych odbiorców danych wartość dodaną.

Głównym czynnikiem napędzającym transformacje danych, często wykonywanym podczas modelowania danych, jest logika biznesowa. Dzięki danym logika biznesowa zmienia się w elementy wielokrotnego użytku (np. sprzedaż oznacza „ktoś kupił ode mnie 12 ramek do zdjęć za 30 PLN za sztukę lub za łączną sumę 360 PLN"). W tym przypadku ktoś kupił 12 ramek do zdjęć w cenie 30 PLN za sztukę. Modelowanie danych ma kluczowe znaczenie dla uzyskania jasnego i aktualnego obrazu procesów biznesowych. Prosty widok surowych transakcji, bez dodania do nich logiki zasad rachunkowości detalicznych, może nie być wystarczający do tego, aby dyrektor finansowy uzyskał czytelny obraz kondycji finansowej. Warto zapewnić standardowe podejście do implementacji logiki biznesowej w transformacjach.

Kolejnym procesem transformacji danych przydatnym z perspektywy uczenia maszynowego jest tzw. **cechowanie danych** (ang. *data featurization*). Proces ten ma na celu wyodrębnienie i ulepszenie cech danych przydatnych do szkolenia modeli uczenia maszynowego. Cechowanie może być postrzegane jako „wiedza tajemna", łącząca wiedzę specjalistyczną (w celu określenia cech, które mogą być ważne z punktu widzenia prognozowania) z bogatym doświadczeniem w nauce o danych. Z perspektywy zagadnień omówionych w tej książce chodzi głównie o to, że gdy badacze danych określą sposób cechowania danych, inżynierowie danych mogą zautomatyzować procesy cechowania na etapie transformacji danych w potoku.

Przekształcanie danych to obszerny temat, którego nie da się dokładnie opisać w tym krótkim wprowadzeniu. Zagadnienia związane z zapytaniami, modelowaniem danych oraz różnymi praktykami związanymi z przekształcaniem danych zostaną szczegółowo omówione w rozdziale 8.

Serwowanie danych

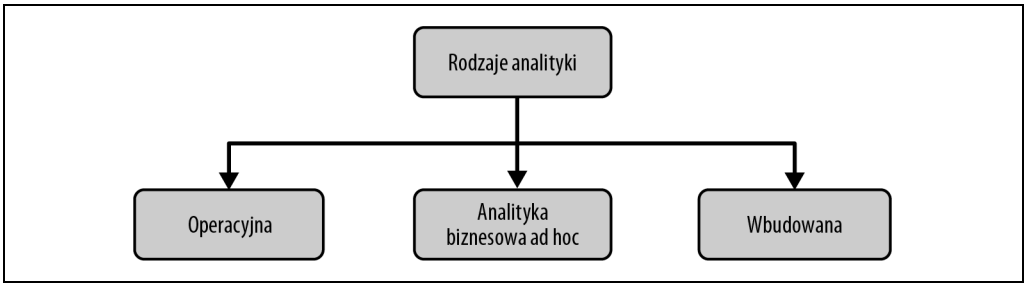
Właśnie dotarłeś do ostatniej fazy cyklu życia inżynierii danych. Teraz gdy dane zostały pozyskane, zapisane i przekształcone w spójne i użyteczne struktury, nadszedł czas, aby uzyskać z danych wartość. „Uzyskiwanie wartości" z danych dla różnych użytkowników oznacza różne rzeczy.

Dane mają *wartość*, gdy są wykorzystywane do celów praktycznych. Dane, które nie są wykorzystywane, lub które nie występują w zapytaniach, są po prostu obojętne. Projekty „próżnych danych" (ang. *data vanity projects*) stanowią dla firm poważne zagrożenie. Wiele firm realizowało projekty próżnych danych w erze big data, gromadząc w jeziorach danych ogromne zbiory, które nigdy nie zostały wykorzystane w żaden użyteczny sposób. Era chmury wyzwoliła nową falę projektów próżnych danych zbudowanych w oparciu o najnowsze hurtownie danych, systemy obiektowej pamięci masowej oraz technologie przesyłania strumieniowego. Projekty danych muszą mieć określony cel w całym cyklu życia inżynierii danych. Jaki jest ostateczny cel biznesowy danych tak starannie gromadzonych, oczyszczanych i przechowywanych?

Serwowanie danych jest prawdopodobnie najbardziej ekscytującą częścią cyklu życia inżynierii danych. To tutaj dzieje się magia. W tej fazie inżynierowie ML mogą zastosować najbardziej zaawansowane techniki. Przyjrzyjmy się niektórym popularnym zastosowaniom danych: analityce, uczeniu maszynowemu i odwrotnemu procesowi ETL.

Analityka

Analityka jest podstawą większości przedsięwzięć związanych z danymi. Po zapisaniu i przekształceniu danych możesz generować raporty lub tworzyć pulpity nawigacyjne i przeprowadzać analizy danych ad hoc. Podczas gdy dawniej większa część zadań analityki obejmowała zagadnienia związane z analityką biznesową (ang. *business intelligence* — BI), obecnie analityka obejmuje inne aspekty, takie jak analityka operacyjna i analityka wbudowana (rysunek 2.5).



Rysunek 2.5. Rodzaje analityki

Analityka biznesowa (ang. *business intelligence*). Specjaliści w dziedzinie BI zbierali dane, aby opisać przeszły i obecny stan biznesu. Analityka biznesowa do przetwarzania surowych danych wymaga użycia logiki biznesowej. Należy pamiętać, że serwowanie danych na potrzeby analizy to kolejny obszar, w którym różne etapy cyklu życia inżynierii danych mogą się ze sobą splatać. Jak wspomnieliśmy wcześniej, logika biznesowa jest często stosowana do danych na etapie ich przekształcania w cyklu życia inżynierii danych, choć coraz bardziej popularne staje się podejście polegające na stosowaniu logiki biznesowej przy odczytywaniu danych. Dane są przechowywane w czystej, ale dość surowej formie, a logika biznesowa w minimalnej formie jest stosowana dopiero w końcowej fazie przetwarzania. Repozytorium logiki biznesowej i związane z nią definicje są utrzymywane przez system BI. Ta logika biznesowa jest wykorzystywana do tworzenia zapytań do magazynu danych. Dzięki temu raporty i pulpity nawigacyjne są zgodne z definicjami biznesowymi.

W miarę jak firma uzyskuje większą dojrzałość w kontekście danych, przechodzi od doraźnej analizy danych do analizy samoobsługowej, co umożliwia użytkownikom biznesowym swobodny dostęp do danych, bez angażowania pracowników działu IT. Możliwość przeprowadzania samoobsługowych analiz wynika z założenia, że dane są na tyle dobrej jakości, że osoby z całej organizacji mogą uzyskać do nich dostęp samodzielnie, podzielić je w dowolny sposób i uzyskać natychmiastowe wnioski. Chociaż analityka samoobsługowa jest teoretycznie prosta, trudno zrealizować ją w praktyce. Głównymi przeszkodami w powszechnym stosowaniu analityki samoobsługowej jest niska jakość danych, silosy organizacyjne oraz brak odpowiedniej wiedzy w zakresie danych.

Analityka operacyjna. Analityka operacyjna koncentruje się na szczegółach operacji z danymi. Promuje dane, na podstawie których użytkownik raportów może natychmiast zareagować. Analityka operacyjna może być podglądem na żywo zapasów albo generowanym w czasie rzeczywistym pulpitem nawigacyjnym kondycji witryny lub aplikacji. W takim przypadku dane są wykorzystywane w czasie rzeczywistym, bezpośrednio z systemu źródłowego lub z potoku przesyłanych strumieniowo danych. Rodzaje wniosków w analityce operacyjnej różnią się od tradycyjnych analiz biznesowych, ponieważ analityka operacyjna koncentruje się na teraźniejszości i niekoniecznie dotyczy trendów historycznych.

Analityka wbudowana. Możesz się zastanawiać, dlaczego oddzieliliśmy analitykę wbudowaną (analitykę skierowaną do klienta) od analityki biznesowej. W praktyce dostarczanie klientom danych analitycznych na platformie SaaS wiąże się z osobnym zestawem wymagań i komplikacji. Wewnętrzne systemy analizy biznesowej mają ograniczone grono odbiorców i na ogół prezentują ograniczoną liczbę ujednoliconych widoków. Mechanizmy kontroli dostępu są ważne, ale nie są szczególnie skomplikowane. Do zarządzania dostępem wykorzystywanych jest kilka ról oraz kilka warstw.

W przypadku zastosowania mechanizmów analityki wbudowanej liczba żądań raportów i związane z tym obciążenia systemów analitycznych znacząco wzrastają. W związku z tym kontrola dostępu jest znacznie bardziej skomplikowana i ma kluczowe znaczenie. Firmy mogą dostarczać oddzielne analizy i dane tysiącom lub większej liczbie klientów. Każdy klient powinien widzieć swoje i tylko swoje dane. Wewnętrzny błąd dostępu do danych firmy powinien doprowadzić do przeglądu obowiązujących procedur. Wyciek danych między klientami zostałby uznany za masowe naruszenie zaufania, co zwróciłoby uwagę mediów i spowodowało znaczącą utratę klientów. Należy zadbać o zminimalizowanie „promienia wybuchu" związanego z wyciekami danych i lukami w zabezpieczeniach. Należy stosować zabezpieczenia na poziomie dzierżawcy lub zabezpieczenia danych w pamięci masowej oraz wszędzie tam, gdzie istnieje możliwość wycieku danych.

Obsługa wielu dzierżawców (ang. multitenancy)

Wiele współczesnych systemów pamięci masowej i systemów analitycznych na różne sposoby obsługuje wielu dzierżawców (ang. *tenants*). Aby zapewnić ujednolicony widok na potrzeby wewnętrznej analizy i zadań uczenia maszynowego, inżynierowie danych mogą zdecydować się na umieszczenie danych wielu klientów we wspólnych tabelach. Dane te są prezentowane na zewnątrz poszczególnym klientom za pośrednictwem logicznych widoków z odpowiednio zdefiniowanymi kontrolkami i filtrami. W celu zapewnienia maksymalnego bezpieczeństwa i izolacji danych na inżynierach danych spoczywa obowiązek zrozumienia wielodostępności w systemach, które wdrażają.

Uczenie maszynowe

Pojawienie się i sukces technik uczenia maszynowego to jedna z najbardziej ekscytujących rewolucji technologicznych. Gdy organizacje osiągną wysoki poziom dojrzałości w kontekście danych, mogą pokusić się o zidentyfikowanie problemów nadających się do rozwiązania za pomocą mechanizmów uczenia maszynowego i zacząć stosować odpowiednie praktyki w tym zakresie.

Obowiązki inżynierów danych w znacznym stopniu pokrywają się z obowiązkami analityków danych i inżynierów uczenia maszynowego, a granice między inżynierią danych, inżynierią ML i inżynierią analityczną są rozmyte. Na przykład inżynier danych może potrzebować obsługi klastrów Spark, ułatwiających obsługę potoków analitycznych i szkolenie modeli uczenia maszynowego. Mogą również potrzebować systemu, który organizuje zadania dla wielu zespołów i obsługuje metadane oraz systemy katalogowania zdolny do śledzenia historii danych i ich rodowodu. Ustalenie tych domen odpowiedzialności i odpowiednich struktur raportowania to jedna z kluczowych decyzji organizacyjnych.

Magazyn funkcji (ang. *feature store*) to niedawno opracowane narzędzie, które łączy inżynierię danych i inżynierię uczenia maszynowego. Magazyny funkcji mają na celu zmniejszenie obciążeń operacyjnych inżynierów uczenia maszynowego dzięki utrzymywaniu historii i wersji funkcji, udostępnianiu współdzielenia funkcji między zespołami oraz zapewnieniu podstawowych funkcji operacyjnych i organizacyjnych, takich jak tzw. **wypełnianie wsteczne** (ang. *backfiling*). W praktyce inżynierowie danych są częścią podstawowego zespołu pomocy technicznej dla magazynów funkcji w celu obsługi inżynierii uczenia maszynowego.

Czy inżynier danych powinien znać techniki uczenia maszynowego? Z pewnością wiedza na ten temat jest przydatna. Niezależnie od granic operacyjnych między inżynierią danych, inżynierią uczenia maszynowego, analizą biznesową i tak dalej inżynierowie danych powinni mieć podstawową wiedzę operacyjną ze wszystkich tych obszarów. Dobry inżynier danych powinien znać podstawowe techniki uczenia maszynowego i związane z nimi wymagania dotyczące przetwarzania danych, przypadki użycia modeli w swojej firmie oraz obowiązki różnych zespołów analitycznych pracujących w organizacji. Pomaga to utrzymać efektywną komunikację i ułatwia współpracę między zespołami. Byłoby idealnie, gdyby inżynierowie danych tworzyli narzędzia we współpracy z innymi zespołami oraz żaden z zespołów nie mógł tworzyć tych narzędzi niezależnie od innych.

Szczegółowe omówienie zagadnień związanych z uczeniem maszynowym wykracza poza zakres tej książki. Dla osób, które chcą dowiedzieć się więcej, dostępny jest obszerny ekosystem książek, filmów, artykułów i społeczności. Kilka przykładów wartościowych zasobów zamieściliśmy w punkcie „Zasoby dodatkowe” na końcu rozdziału.

Poniżej przedstawiono kilka zagadnień dotyczących fazy serwowania danych specyficznych dla uczenia maszynowego:

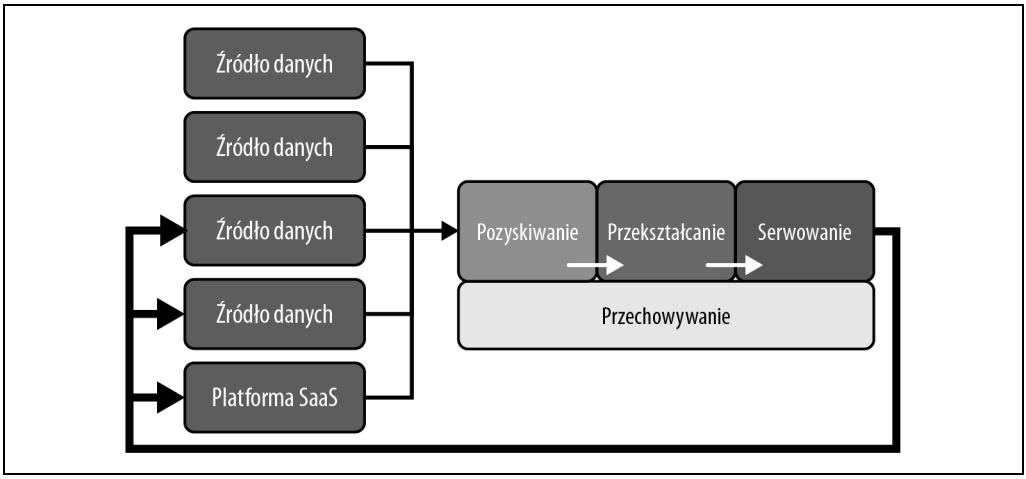
- Czy dane są wystarczającej jakości, aby zrealizować wiarygodną inżynierię funkcji? Wymagania jakościowe i oceny powinny być opracowywane w ścisłej współpracy z zespołami korzystającymi z danych.
- Czy dane są wykrywalne? Czy analitycy danych i inżynierowie uczenia maszynowego mają możliwość łatwego znalezienia wartościowych danych?

- Gdzie są techniczne i organizacyjne granice między inżynierią danych a inżynierią ML? To pytanie, choć dotyczy organizacji, ma znaczące implikacje architektoniczne.
- Czy zbiór danych prawidłowo reprezentuje źródło prawdy? Czy nie jest niesprawiedliwie stronniczy?

Chociaż dziedzina uczenia maszynowego jest ekscytująca, z naszych doświadczeń wynika, że firmy często zaczynają stosować ją zbyt wcześnie. Zanim zainwestujesz w uczenie maszynowe mnóstwo zasobów, poświęć trochę czasu na zbudowanie solidnego fundamentu danych. Oznacza to konfigurowanie najlepszych systemów i architektur w całym cyklu życia inżynierii danych i uczenia maszynowego. Ogólnie rzecz biorąc, przed rozpoczęciem stosowania technik uczenia maszynowego warto rozwinąć kompetencje w zakresie analityki. Wiele firm musiało pogodzić się z porażką w stosowaniu uczenia maszynowego ze względu na to, że podejmowały inicjatywy bez odpowiednich podstaw.

Odwrócony ETL

Odwrócony proces ETL od dawna jest praktyczną rzeczywistością w przetwarzaniu danych, postrzeganą jako antywzorzec, o którym nie chcieliśmy rozmawiać ani nie chcieliśmy jawnie go nazywać. **Odwrócony ETL** polega na pobieraniu przetworzonych danych ze strony wynikowej cyklu życia inżynierii danych i przekazywaniu ich z powrotem do systemów źródłowych (patrz rysunek 2.6). W gruncie rzeczy przepływ ten jest korzystny i często konieczny; odwrócony ETL pozwala nam pobierać analizy, oceniane modele itp. i przekazywać je do systemów produkcyjnych lub platform SaaS.



Rysunek 2.6. Odwrócony ETL

Analitycy marketingowi na podstawie danych z hurtowni danych mogą obliczać stawki w programie Microsoft Excel, a następnie przesyłać je do systemu Google Ads. Proces ten często był realizowany całkowicie ręcznie w bardzo prymitywny sposób.

Kiedy pisaliśmy tę książkę, kilku dostawców usług przyjęło koncepcję odwróconego ETL i zbudowało wokół niej takie produkty jak Hightouch i Census. Odwrócony ETL to praktyka, która dopiero się rodzi, ale podejrzewamy, że zagości w przetwarzaniu danych na stałe.

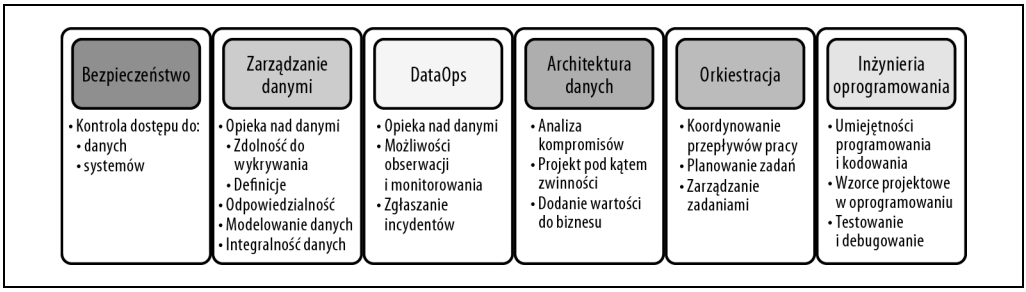
Odwrócony ETL stał się szczególnie ważny ze względu na to, że firmy w coraz większym stopniu polegają na usługach SaaS i platformach zewnętrznych. Firmy mogą na przykład wypychać określone metryki ze swojej hurtowni danych do platform danych klientów lub do systemu CRM. Kolejnym powszechnym przypadkiem użycia odwróconego ETL są platformy reklamowe, na przykład Google Ads. W przyszłości należy się spodziewać większej aktywności w stosowaniu odwróconego ETL, z nakładającymi się na siebie zadaniami inżynierii danych oraz inżynierii uczenia maszynowego.

Pozostaje wątpliwość, czy termin *odwrócony ETL* się przyjmie. Stosowane praktyki mogą ewoluować. Niektórzy inżynierowie twierdzą, że można wyeliminować odwrócony ETL, a transformacje danych obsłużyć w strumieniu zdarzeń i w razie potrzeby wysyłać te zdarzenia z powrotem do systemów źródłowych. Powszechne przyjęcie tego wzorca w różnych firmach to inna sprawa. Istotne jest to, że przekształcone dane będą musiały zostać w jakiś sposób zwrócone do systemów źródłowych, najlepiej z oznaczeniem ich prawidłowego pochodzenia oraz procesów biznesowych związanych z systemem źródłowym.

Główne nurty w cyklu życia inżynierii danych

Inżynieria danych to dziedzina, która szybko dojrzewa. Podczas gdy wcześniejsze cykle inżynierii danych koncentrowały się głównie na warstwie technologicznej, ciągła abstrakcja i upraszczanie narzędzi oraz praktyk przesunęło to skupienie. Inżynieria danych obejmuje dziś znacznie więcej niż narzędzia i technologię. Dziedzina przesuwa się obecnie w górę łańcucha wartości, wykorzystując tradycyjne praktyki korporacyjne, takie jak zarządzanie danymi i optymalizacja kosztów, oraz nowsze praktyki, na przykład DataOps.

Nazwaliśmy te praktyki **nurtami** (ang. *undercurrents*); zaliczamy do nich bezpieczeństwo, zarządzanie danymi, DataOps, architekturę danych, orkiestrację oraz inżynierią oprogramowania. Są to dziedziny, które wspierają każdy aspekt cyklu życia inżynierii danych (rysunek 2.7).



Rysunek 2.7. Główne nurty inżynierii danych

Bezpieczeństwo

Bezpieczeństwo musi mieć dla inżynierów danych najwyższy priorytet, a ci, którzy je ignorują, robią to na własne ryzyko. Dlatego bezpieczeństwo jest wymieniane jako pierwszy nurt inżynierii danych. Inżynierowie danych muszą rozumieć zarówno zagadnienia związane z bezpieczeństwem danych,

jak i z kontrolą dostępu do danych. Powinni oni stosować zasadę najmniejszych uprawnień. Zasada najmniejszych uprawnień (*https://oreil.ly/6RGAq*) to reguła, która mówi o zapewnieniu użytkownikowi lub systemowi dostępu tylko do tych danych i zasobów, które są niezbędne do wykonania zamierzonej funkcji. Powszechnym antywzorcem, często stosowanym przez inżynierów danych z niewielkim doświadczeniem w zakresie bezpieczeństwa, jest zapewnienie wszystkim użytkownikom uprawnień administratora. Takie postępowanie to proszenie się o katastrofę!

Należy udzielić użytkownikom tylko takich uprawnień dostępu, których potrzebują w tym momencie do wykonania swojej pracy, nic poza tym. Nie pracuj z poziomu powłoki użytkownika root, gdy chcesz poszukać plików widocznych z uprawnieniami standardowego użytkownika. Podczas wykonywania zapytań do bazy danych dotyczących tabel o mniejszej roli nie należy używać roli administratora bazy danych. Przestrzeganie zasady najmniejszych uprawnień może zapobiec przypadkowym uszkodzeniom danych oraz utrwala nawyki wykonywania działań z myślą o bezpieczeństwie.

W każdej firmie największymi lukami w zabezpieczeniach są ludzie i struktura organizacyjna. Kiedy słyszymy o poważnych naruszeniach bezpieczeństwa w mediach, często okazuje się, że ktoś w firmie zignorował podstawowe środki ostrożności, padł ofiarą ataku phishingowego lub postąpił nieodpowiedzialnie w inny sposób. Pierwszą linią obrony bezpieczeństwa danych jest stworzenie kultury bezpieczeństwa, która przenika całą organizację. Wszystkie osoby, które mają dostęp do danych, muszą mieć świadomość swojej odpowiedzialności za ochronę wrażliwych danych firmy i jej klientów.

Bezpieczeństwo danych jest związane również z czasem — należy zadbać o zapewnienie dostępu do danych dokładnie tym osobom i systemom, które potrzebują do nich dostępu i *tylko przez czas niezbędny do wykonania zaplanowanej pracy*. Dane powinny być chronione przed niepożądaną widocznością zarówno podczas ich transmisji, jak i w spoczynku. Do tego celu należy wykorzystać mechanizmy szyfrowania, tokenizacji, maskowania danych, zaciemniania oraz proste, solidne mechanizmy kontroli dostępu.

Inżynierowie danych powinni być kompetentnymi administratorami zabezpieczeń, ponieważ zabezpieczenia należą do ich domeny. Inżynierowie danych powinni znać najlepsze praktyki bezpieczeństwa zarówno dla systemów działających w chmurze, jak i „w siedzibie". Warto rozpocząć od zapoznania się z rolami, zasadami, grupami, zabezpieczeniami sieci, zasadami zarządzania hasłami oraz technikami szyfrowania, czyli komponentami mechanizmów zarządzania użytkownikami i tożsamością (ang. *identity access management* — IAM).

W całej książce wymieniamy te obszary cyklu życia inżynierii danych, w których bezpieczeństwo powinno mieć najwyższy priorytet. Bardziej szczegółowe informacje na temat bezpieczeństwa możesz uzyskać w rozdziale 10.

Zarządzanie danymi

Prawdopodobnie myślisz, że zarządzanie danymi brzmi bardzo… korporacyjnie. „Oldschoolowe" praktyki zarządzania należą do obszarów inżynierii danych i uczenia maszynowego. To, co było stare, dziś znów jest nowe. Zarządzanie danymi istnieje od dziesięcioleci, ale do niedawna

nie miało dużego znaczenia w kontekście inżynierii danych. Narzędzia do obsługi danych stają się coraz prostsze, a inżynierowie danych muszą zmagać się z mniejszą złożonością. W rezultacie inżynierowie danych przesuwają się w górę łańcucha wartości w kierunku następnego szczebla najlepszych praktyk. Najlepsze praktyki obsługi danych, niegdyś zarezerwowane dla dużych firm — opieka nad danymi, zarządzanie podstawowymi danymi, zarządzanie jakością danych, zarządzanie metadanymi — są teraz typowe dla firm każdej wielkości i na wszystkich poziomach dojrzałości. Jak lubimy mówić, inżynieria danych staje się „coraz bardziej przyjazna przedsiębiorstwom". To w końcu świetna sprawa!

Książka *Data Management Body of Knowledge* (*DMBOK*) wydana przez stowarzyszenie DAMA (*Data Management Association International*), którą uważamy za ostateczne źródło informacji na temat zarządzania danymi w przedsiębiorstwie, oferuje następującą definicję:

> Zarządzanie danymi to opracowywanie, wykonywanie i nadzorowanie planów, zasad, programów i praktyk, które dostarczają, kontrolują, chronią i zwiększają wartość danych i zasobów informacyjnych w całym cyklu ich życia.

Ta definicja jest dość długa, spójrzmy zatem, jak wiąże się ona z inżynierią danych. Inżynierowie danych zarządzają cyklem życia danych, a zarządzanie danymi obejmuje zestaw najlepszych praktyk, które inżynierowie danych wykorzystają do wykonania tego zadania, zarówno z punktu widzenia technicznego, jak i z perspektywy strategicznej. Bez frameworka zarządzania danymi inżynierowie danych są po prostu pracownikami technicznymi działającymi w próżni. Inżynierowie danych potrzebują szerszej perspektywy w kwestii przydatności danych w całej organizacji, od systemów źródłowych, po wyższe kierownictwo (C-suite) i we wszystkich warstwach pośrednich.

Dlaczego zarządzanie danymi jest ważne? Zarządzanie danymi pokazuje, że dane są niezbędne do codziennej działalności firm. W podobny sposób firmy postrzegają jako swoje aktywa zasoby finansowe, wyroby gotowe lub nieruchomości. Praktyki zarządzania danymi tworzą spójny framework, który może wykorzystać każdy, aby zapewnić organizacji otrzymywanie wartości z danych i odpowiednią ich obsługę.

Zarządzanie danymi ma kilka aspektów, w tym następujące elementy:

- Opieka nad danymi (ang. *data governance*), w tym możliwości wykrywania danych i przypisania odpowiedzialności za dane.
- Modelowanie i projektowanie danych.
- Rodowód danych.
- Przechowywanie danych i wykonywanie z nimi operacji.
- Integracja i interoperacyjność danych.
- Zarządzanie cyklem życia danych.
- Systemy danych do zaawansowanej analityki i zadań uczenia maszynowego.
- Etyka i prywatność.

Chociaż ta książka w żadnym razie nie jest wyczerpującym źródłem informacji na temat zarządzania danymi, krótko omówimy kilka istotnych punktów z każdego obszaru, ponieważ odnoszą się one do inżynierii danych.

Opieka nad danymi

Według *Data Governance: The Definitive Guide* „opieka nad danymi jest przede wszystkim funkcją zarządzania danymi mającą na celu zapewnienie jakości, integralności, bezpieczeństwa i użyteczności danych gromadzonych przez organizację”[1].

Możemy rozwinąć tę definicję i powiedzieć, że zarządzanie danymi angażuje ludzi, procesy i technologie, aby zmaksymalizować wartość danych w całej organizacji, a jednocześnie chroni dane dzięki stosowaniu odpowiednich mechanizmów kontroli bezpieczeństwa. Skuteczna opieka nad danymi jest opracowywana w sposób celowy i wspierana przez organizację. Gdy opieka nad danymi jest realizowana przypadkowo i bez planu, mogą wystąpić skutki uboczne, począwszy od niezaufanych danych, a skończywszy na naruszeniach bezpieczeństwa. Celowe podejście do opieki nad danymi maksymalizuje możliwości organizacji w zakresie danych i generowaną z danych wartość. Dzięki temu również (miejmy nadzieję) firma nie trafi na pierwsze strony gazet z powodu wątpliwych lub wręcz lekkomyślnych praktyk związanych z danymi.

Pomyśl o typowym przykładzie złej opieki nad danymi. Analityk biznesowy otrzymuje prośbę o raport, ale nie wie, jakich danych użyć, aby spełnić tę prośbę. Przekopywanie się przez dziesiątki tabel w transakcyjnej bazie danych w celu odgadnięcia, które pola mogą być przydatne, może zająć mu wiele godzin. Analityk kompiluje raport „kierunkowo poprawny”, ale nie ma pewności, czy bazowe dane wykorzystane do raportu są dokładne lub solidne. Poprawność danych kwestionuje również odbiorca raportu. Kwestionowana jest integralność analizy i wszystkich danych w systemach firmy. Firma jest zdezorientowana co do swoich wyników, co uniemożliwia planowanie biznesowe.

Opieka nad danymi jest podstawą praktyk biznesowych opartych na danych i kluczową częścią cyklu życia inżynierii danych. Gdy opieka nad danymi jest realizowana właściwie, ludzie, procesy i technologie dostosowują się. Zaczynają traktować dane jako kluczowy czynnik biznesowy; jeśli wystąpią problemy z danymi, są one natychmiast obsługiwane.

Podstawowe kategorie opieki nad danymi to możliwość odkrywania, bezpieczeństwo i zdolność do przypisania odpowiedzialności za dane[2]. Wymienione kategorie podstawowe składają się z podkategorii, takich jak jakość danych, metadane i prywatność. Przyjrzyjmy się kolejno każdej z kategorii podstawowych.

Zdolność do wykrywania. W firmie, której działalność jest oparta na danych, dane muszą być dostępne i wykrywalne. Docelowi użytkownicy powinni mieć szybki i niezawodny dostęp do danych potrzebnych do wykonywania ich pracy. Powinni wiedzieć, skąd pochodzą dane, jaki mają związek z innymi danymi i co oznaczają.

[1] Evren Eryurek et al., *Data Governance: The Definitive Guide*, s. 1, O'Reilly, Sebastopol 2021, *https://oreil.ly/LFT4d*.

[2] Eryurek, *Data Governance*, s. 5.

Niektóre kluczowe obszary wykrywalności danych obejmują zarządzanie metadanymi i zarządzanie danymi podstawowymi. Spróbujmy pokrótce opisać te obszary.

Metadane. *Metadane* to „dane o danych". Stanowią podstawę każdej fazy cyklu życia inżynierii danych. Metadane to te dane, które są potrzebne do tego, aby dane były wykrywalne i zarządzalne.

Metadane można podzielić na dwie główne kategorie: generowane automatycznie i generowane przez człowieka. Współczesna inżynieria danych obraca się wokół automatyzacji, ale zbieranie metadanych często jest wykonywane ręcznie i jest to proces podatny na błędy.

W realizacji tego procesu może pomóc technologia, dzięki której można usunąć wiele podatnych na błędy działań związanych z ręcznym zbieraniem metadanych. W ostatnim czasie można zaobserwować rozpowszechnianie się wykazów danych, systemów śledzenia rodowodu danych oraz narzędzi do zarządzania metadanymi. Narzędzia potrafią przeszukiwać bazy danych w celu wyszukiwania relacji oraz monitorować potoki danych w celu śledzenia, skąd pochodzą dane i dokąd trafiają. Podejście ręczne ma niską wiarygodność. Opiera się na pracach wykonywanych wewnątrz organizacji, a polegających na gromadzeniu danych przez różnych interesariuszy. Narzędzia do zarządzania danymi zostały szczegółowo omówione w całej książce, ponieważ dotyczą znacznej części cyklu życia inżynierii danych.

Metadane stają się produktem ubocznym danych i związanych z nimi procesów. Nadal jednak pozostają kluczowe wyzwania. W szczególności nadal brakuje interoperacyjności i odpowiednich standardów. Narzędzia do zbierania metadanych są tak dobre, jak ich łączniki z systemami danych oraz ich zdolność do współdzielenia metadanych. Ponadto zautomatyzowane narzędzia do obsługi metadanych nie powinny całkowicie eliminować udziału ludzi.

Dane mają aspekt społeczny; każda organizacja gromadzi społeczny kapitał i wiedzę na temat procesów, zestawów danych i potoków. Systemy metadanych zorientowane na ludzi koncentrują się na społecznym aspekcie metadanych. Na te kwestie zwrócił uwagę Airbnb w różnych postach na swoim blogu poświęconym narzędziom danych. W szczególności opisał koncepcję systemu, który nazwał **portalem danych**[3]. Takie narzędzia powinny umożliwiać ujawnianie właścicieli danych, konsumentów danych i ekspertów dziedzinowych. Podstawę zarządzania metadanymi stanowią dokumentacja oraz wewnętrzne narzędzia typu wiki, ale narzędzia te powinny również integrować się ze zautomatyzowanymi systemami katalogowania danych. Na przykład narzędzia do skanowania danych mogą generować strony typu wiki zawierające łącza do odpowiednich obiektów danych.

Po utworzeniu systemów i procesów metadanych inżynierowie danych mogą korzystać z metadanych w użyteczny sposób. Metadane stają się podstawą do projektowania potoków i zarządzania danymi w całym cyklu życia danych.

W książce *DMBOK* zidentyfikowano cztery główne kategorie metadanych przydatne dla inżynierów danych:

- Metadane biznesowe.
- Metadane techniczne.

[3] Chris Williams et al., *Democratizing Data at Airbnb*, blog techniczny The Airbnb, 12 maja 2017, *https://oreil.ly/dM332*.

- Metadane operacyjne.
- Metadane referencyjne.

Poniżej pokrótce opiszemy każdą kategorię metadanych.

Metadane biznesowe odnoszą się do sposobu, w jaki w firmie są wykorzystywane dane, zawierają definicje biznesowe i definicje danych, reguły i logikę związaną z danymi, sposoby i miejsca użycia danych oraz właściciela (właścicieli) danych.

Inżynierowie danych używają metadanych biznesowych, aby odpowiedzieć na nietechniczne pytania dotyczące tego kto, co, gdzie i jak. Na przykład inżynier danych może otrzymać zadanie utworzenia potoku danych w celu analizy sprzedaży klientów. Ale kim jest klient? Czy jest to ktoś, kto coś od nas kupił w ciągu ostatnich 90 dni? A może jest to ktoś, kto kiedykolwiek dokonał jakiegoś zakupu w naszej firmie? Inżynier danych użyje poprawnych danych, aby odwołać się do metadanych biznesowych (słownika danych lub katalogu danych) i tam sprawdzić, jak zdefiniowany jest „klient". Metadane biznesowe zapewniają inżynierom danych odpowiedni kontekst i definicje umożliwiające prawidłowe korzystanie z danych.

Metadane techniczne opisują dane tworzone i używane przez systemy w całym cyklu życia inżynierii danych. Obejmują model danych i schemat, rodowód danych, mapowanie pól i przepływy pracy potoków. Inżynierowie danych używają metadanych technicznych do tworzenia, łączenia i monitorowania różnych systemów w całym cyklu życia inżynierii danych.

Oto kilka podstawowych typów metadanych technicznych, z których będą korzystać inżynierowie danych:

- Metadane potoków (często wytwarzane w systemach orkiestracji).
- Rodowód danych.
- Schemat.

Orkiestracja to centralny koncentrator koordynujący przepływ pracy w różnych systemach. *Metadane potoków* przechwycone w systemach orkiestracji zawierają szczegółowe informacje na temat między innymi harmonogramu przepływu pracy, zależności między systemem a danymi, konfiguracji oraz połączeń.

Metadane rodowodu danych pozwalają na śledzenie pochodzenia i zmian danych oraz ich zależności w czasie. Gdy dane przepływają przez fazy cyklu życia inżynierii danych, ewoluują poprzez transformacje oraz ich łączenie z innymi danymi. Rodowód danych to ścieżka audytu ewolucji danych podczas ich przemieszczania się przez różne systemy i przepływy pracy.

Metadane schematu opisują strukturę danych przechowywanych w systemie, takim jak baza danych, hurtownia danych, jezioro danych lub system plików; jest to jeden z kluczowych wyróżników w różnych systemach pamięci masowej. Na przykład magazyny obiektów nie zarządzają metadanymi schematu; należy nimi zarządzać w magazynie metastore. Z drugiej strony hurtownie danych w chmurze są wyposażone w wewnętrzne mechanizmy zarządzania metadanymi schematu.

To tylko kilka przykładów metadanych technicznych, o których powinni wiedzieć inżynierowie danych. Nie jest to pełna lista, a w całej książce omawiamy dodatkowe aspekty związane z metadanymi technicznymi.

Metadane operacyjne opisują wyniki operacyjne różnych systemów i obejmują statystyki dotyczące procesów, identyfikatory zadań, logi z działania aplikacji, dane wykorzystywane przez procesy oraz logi błędów. Inżynierowie danych używają metadanych operacyjnych do stwierdzenia, czy proces zakończył się powodzeniem, czy nie, oraz do określania danych wykorzystanych w procesie.

Ograniczony obraz metadanych operacyjnych dostarczają systemy orkiestracji, ale w gruncie rzeczy metadane te są rozproszone w wielu systemach. Potrzeba lepszej jakości metadanych operacyjnych i lepszego zarządzania metadanymi jest główną motywacją dla systemów orkiestracji i zarządzania metadanymi nowej generacji.

Metadane referencyjne to dane używane do klasyfikowania innych danych. Są one nazywane również danymi odnośników. Standardowymi przykładami metadanych referencyjnych są wewnętrzne kody, kody geograficzne, jednostki miary oraz wewnętrzne standardy kalendarza. Należy zauważyć, że większość metadanych referencyjnych jest w pełni zarządzana wewnętrznie, choć takie elementy jak kody geograficzne mogą pochodzić ze standardowych systemów zewnętrznych. Metadane referencyjne są w gruncie rzeczy standardem interpretacji innych danych, więc jeśli się zmieniają, zmiana ta rozciąga się w czasie.

Odpowiedzialność za dane. *Odpowiedzialność za dane* (ang. *data accountability*) oznacza przypisanie osoby do zarządzania częścią danych. Następnie osoba odpowiedzialna koordynuje działania w zakresie opieki nad danymi wykonywane przez innych interesariuszy. Jeśli nikt nie jest odpowiedzialny za dane, zarządzanie ich jakością jest trudne.

Należy pamiętać, że osoby odpowiedzialne za dane nie muszą być inżynierami danych. Osoba odpowiedzialna za dane może być inżynierem oprogramowania, menedżerem produktu lub pełnić zupełnie inną rolę. Ponadto osoba odpowiedzialna za dane na ogół nie ma wszystkich zasobów niezbędnych do utrzymania jakości danych. Jej zadaniem jest raczej koordynowanie działań dotyczących danych ze wszystkimi osobami, które z nich korzystają, w tym z inżynierami danych.

Odpowiedzialność za dane może być realizowana na różnych poziomach: na poziomie tabeli lub strumienia logów, ale może także być bardzo szczegółowa — tzn. może dotyczyć pojedynczej encji pola występującego w wielu tabelach. Osoba fizyczna może być odpowiedzialna za zarządzanie identyfikatorem klienta w wielu systemach. W przypadku zarządzania danymi przedsiębiorstwa domena danych to zestaw wszystkich możliwych wartości, które mogą wystąpić dla danego typu pola, na przykład wspomnianego identyfikatora. Może się to wydawać nadmiernie biurokratyczne i skrupulatne, ale może znacząco wpłynąć na jakość danych.

Jakość danych

> Czy mogę zaufać tym danym?
>
> — Wszyscy interesariusze biznesowi

Jakość danych to optymalizacja danych w kierunku ich pożądanego stanu. Definicję jakości danych można streścić pytaniem: „Co otrzymujesz w porównaniu z tym, czego oczekujesz?”. Dane powinny być zgodne z oczekiwaniami zawartymi w metadanych biznesowych. Czy dane są zgodne z definicją uzgodnioną przez interesariuszy biznesowych?

Inżynierowie danych zapewniają jakość danych w całym cyklu życia inżynierii danych. Działania w zakresie zapewnienia jakości danych obejmują przeprowadzanie testów jakości danych oraz zapewnianie zgodności danych z oczekiwaniami schematu, kompletności danych i precyzji.

W książce *Data Governance: The Definitive Guide* stwierdzono, że jakość danych jest zdefiniowana przez trzy główne cechy[4]:

Dokładność (ang. *accuracy*)

Czy zebrane dane są zgodne ze stanem faktycznym? Czy istnieją duplikaty wartości? Czy wartości liczbowe są dokładne?

Kompletność

Czy rekordy danych są kompletne? Czy wszystkie wymagane pola zawierają prawidłowe wartości?

Terminowość

Czy rekordy danych są dostępne w odpowiednim czasie?

Z każdą z tych cech wiąże się sporo niuansów. Na przykład jak traktujemy boty oraz programy typu *web scraper*, gdy obsługujemy dane o zdarzeniach internetowych? Jeśli zamierzamy analizować ruch klientów, powinniśmy znaleźć proces, który pozwoli oddzielić ruch ludzi od ruchu generowanego przez komputery. Wszelkie zdarzenia wygenerowane przez boty i błędnie sklasyfikowane jako działania *człowieka* (i odwrotnie działania ludzi błędnie sklasyfikowane jako działania botów) stwarzają problemy z dokładnością danych.

Pojawia się także wiele interesujących problemów dotyczących kompletności i terminowości. Autorzy opublikowanego przez Google artykułu wprowadzającego model Dataflow podali przykład platformy wideo offline, która wyświetlała reklamy[5]. Platforma w czasie trwania połączenia pobierała filmy i reklamy, pozwalała użytkownikowi oglądać je w trybie offline, a następnie, gdy połączenie zostało nawiązane ponownie, przesyłała dane widoku reklamy. Dane te mogły pojawić się z opóźnieniem, długo po obejrzeniu reklam. W jaki sposób platforma obsługiwała rozliczanie reklam?

Ogólnie rzecz biorąc, problemu tego nie można rozwiązać za pomocą czysto technicznych środków. Inżynierowie będą musieli określić zasady postępowania z późno przybywającymi danymi i konsekwentnie je egzekwować, być może za pomocą różnych narzędzi technicznych.

Jakość danych przekracza granice zarówno działań ludzkich, jak i technologicznych. Inżynierowie danych potrzebują solidnych procesów, aby zbierać od ludzi przydatne informacje zwrotne dotyczące jakości danych i korzystać z narzędzi technologicznych do prewencyjnego wykrywania problemów z jakością, zanim dotrą one do użytkowników działających w dolnej części strumienia przetwarzania. Wspomniane procesy zbierania zostały omówione w dalszych rozdziałach tej książki.

[4] Eryurek, *Data Governance*, s. 113.

[5] Tyler Akidau i współpracownicy, *The Dataflow Model: A Practical Approach to Balancing Correctness, Latency, and Cost in Massive-Scale, Unbounded, Out-of-Order Data Processing*, materiały z konferencji *VLDB Endowment 8* (2015): 1792 – 1803, *https://oreil.ly/Z6XYy*.

Zarządzanie danymi podstawowymi

Dane podstawowe (ang. *master data*) to dane o podmiotach gospodarczych, takich jak pracownicy, klienci, produkty i lokalizacje. W miarę jak dzięki organicznemu wzrostowi i przejęciom oraz współpracy z innymi firmami organizacje stają się coraz większe i bardziej złożone, utrzymanie spójnego obrazu podmiotów i ich tożsamości staje się coraz trudniejsze.

Zarządzanie danymi podstawowymi (ang. *master data management* — MDM) to praktyka budowania spójnych definicji encji znanych jako tzw. **złote rekordy**. Złote rekordy harmonizują dane encji w całej organizacji, wraz z jej partnerami. MDM to proces operacji biznesowych wspierany przez budowanie i wdrażanie narzędzi technologicznych. Na przykład zespół MDM może określić standardowy format adresów, a następnie współpracować z inżynierami danych w celu utworzenia interfejsu API zwracającego spójne adresy oraz systemu, który używa danych adresowych w celu dopasowywania rekordów klientów w różnych działach firmy.

Zespół MDM sięga do operacyjnych baz danych przez cały cykl życia danych. Zadania te mogą wchodzić bezpośrednio w zakres kompetencji inżynierii danych, ale często są one przypisane do dedykowanego zespołu, który pracuje w całej organizacji. Nawet jeśli inżynierowie danych nie odpowiadają za MDM, powinni zawsze być świadomi istnienia tego zespołu, ponieważ są zobowiązani do współpracy z nim w zakresie MDM.

Modelowanie i projektowanie danych

Aby za pośrednictwem narzędzi analityki biznesowej oraz nauki o danych można było wyciągać z danych wnioski biznesowe, dane muszą być w użytecznym formacie. Proces konwertowania danych na odpowiedni format jest określany jako **modelowanie i projektowanie danych**. Podczas gdy tradycyjnie uważamy modelowanie danych za problem rozwiązywany przez administratorów baz danych (DBA) i programistów ETL, modelowanie danych może być realizowane niemal w każdym miejscu w organizacji. Inżynierowie oprogramowania firmware opracowują format danych rekordu dla urządzenia IoT, a programiści aplikacji webowych projektują odpowiedź JSON na wywołanie interfejsu API lub tworzą schemat tabeli MySQL — wszystko to są przykłady modelowania i projektowania danych.

Ze względu na różnorodność nowych źródeł danych i przypadków użycia modelowanie danych stało się trudniejsze. Na przykład ścisła normalizacja nie sprawdza się dobrze w przypadku danych dotyczących zdarzeń. Na szczęście nowa generacja narzędzi danych zwiększa elastyczność modeli danych, a jednocześnie pozwala zachować logiczne rozdzielenie miar, rozmiarów, atrybutów i hierarchii. Hurtownie danych w chmurze obsługują pozyskiwanie ogromnych ilości danych zdenormalizowanych i częściowo pozbawionych struktury, a jednocześnie obsługują typowe wzorce modelowania danych, takie jak Kimball, Inmon i Data Vault. Frameworki przetwarzania danych, takie jak Spark, umożliwiają pozyskiwanie całego spektrum danych, począwszy od płaskich rekordów relacyjnych, po surowy, nieposiadający żadnej struktury tekst. Te wzorce modelowania i transformacji danych omawiamy bardziej szczegółowo w rozdziale 8.

Biorąc pod uwagę szeroką gamę danych, które inżynierowie danych muszą przetwarzać, istnieje pokusa, aby się poddać i zrezygnować z modelowania danych. To bardzo zły pomysł, a jego konsekwencje

mogą być bardzo dotkliwe. Można się o tym przekonać, czytając o wzorcu WORN (ang. *write once, read never* — dosłownie: zapisz raz, nie odczytasz nigdy) lub o tzw. **bagnach danych** (ang. *data swamp*). Inżynierowie danych muszą znać najlepsze praktyki modelowania, a także być na tyle elastyczni, aby umieć zastosować odpowiedni poziom i typ modelowania do źródła danych i przypadku użycia.

Rodowód danych

Skąd można się dowiedzieć, jaki system wpłynął na dane lub z czego składają się dane, gdy w całym cyklu swojego życia są one przekazywane i przekształcane? **Rodowód danych** (ang. *data lineage*) opisuje rejestr ścieżki audytu danych w całym cyklu ich życia, co pozwala śledzić zarówno systemy przetwarzające dane, jak i dane z górnej części strumienia przetwarzania, od których te dane zależą.

Rodowód danych pomaga w śledzeniu błędów, ustalaniu odpowiedzialności za dane oraz debugowaniu danych i systemów, które je przetwarzają. Ma to oczywistą zaletę w postaci udostępnienia ścieżki audytu dla całego cyklu życia danych oraz pomaga w zapewnieniu zgodności z przepisami. Na przykład jeśli użytkownik chciałby, aby jego dane zostały usunięte z systemów, to dzięki znajomości rodowodu tych danych może się dowiedzieć, gdzie te dane są przechowywane i jakie są ich zależności.

W większych firmach, o surowych standardach zgodności z przepisami, pojęcie rodowodu danych istnieje od dawna. Jednak obecnie, ze względu na to, że zarządzanie danymi staje się głównym nurtem, jest ono szerzej stosowane także w mniejszych firmach. Warto zwrócić uwagę, że koncepcja Andy'ego Petrelli DODD (*Data Observability Driven Development* — dosłownie: programowanie sterowane obserwowalnością danych; *https://oreil.ly/3f4WS*) jest ściśle związana z rodowodem danych. Technika DODD polega na obserwowaniu rodowodu danych w całym cyklu ich życia. Proces ten jest stosowany podczas opracowywania, testowania i ostatecznie produkcji w celu zapewnienia rozwiązania wysokiej jakości, zgodnego z oczekiwaniami.

Integracja danych i interoperacyjność

Integracja danych i interoperacyjność to proces wykorzystywania danych za pomocą różnych narzędzi oraz w różnych procesach. W miarę odchodzenia od podejścia do analizy opartego na jednym stosie technologicznym i dążenia do utworzenia heterogenicznego środowiska chmury, w którym różne narzędzia przetwarzają dane na żądanie, coraz większy zakres prac inżyniera danych dotyczy integracji danych i interoperacyjności.

Coraz częściej integracja odbywa się za pośrednictwem interfejsów API ogólnego przeznaczenia, a nie niestandardowych połączeń z bazami danych. Na przykład potok danych może pobierać dane z interfejsu API Salesforce, składować je w usłudze Amazon S3, wywoływać interfejs API Snowflake w celu załadowania ich do tabeli, ponownie wywołać interfejs API w celu uruchomienia zapytania, a następnie eksportować wyniki do usługi S3, skąd mogą być pobrane przez system Spark.

Cała ta aktywność może być zarządzana za pomocą stosunkowo prostego kodu w Pythonie, który nie obsługuje bezpośrednio danych, lecz komunikuje się z odpowiednimi systemami danych. Podczas gdy złożoność interakcji z systemami danych zmniejszyła się, liczba systemów i złożoność potoków znacząco wzrosła. Inżynierowie zaczynający tworzenie systemu od podstaw szybko dochodzą

do przekonania, że wymagania dotyczące danych przekraczają możliwości tworzonych przez nich skryptów, i uświadamiają sobie potrzebę *orkiestracji*. Orkiestracja jest jednym z głównych nurtów inżynierii danych. Omawiamy ją szczegółowo w podrozdziale „Orkiestracja".

Zarządzanie cyklem życia danych

Pojawienie się jezior danych zachęciło organizacje do ignorowania zagadnień archiwizacji i niszczenia danych. Po co usuwać dane, skoro można w prosty sposób w nieskończoność dodawać coraz więcej pamięci masowej? Do zwrócenia większej uwagi na to, co dzieje się pod koniec cyklu życia inżynierii danych, zachęciły inżynierów dwie zmiany.

Po pierwsze dane są coraz częściej przechowywane w chmurze. Oznacza to konieczność ponoszenia kosztów jedynie za faktycznie wykorzystaną pamięć masową zamiast ponoszenia początkowo wysokich nakładów inwestycyjnych na utworzenie lokalnego jeziora danych. Kiedy na miesięcznym wyciągu AWS pojawia się każdy wykorzystany bajt, dyrektorzy finansowi starają się znaleźć oszczędności. Archiwizacja danych w środowiskach chmurowych jest dość prostym procesem. Główni dostawcy chmury obliczeniowej oferują klasy obiektowej pamięci masowej specjalnie przeznaczone dla archiwów, które umożliwiają długoterminowe przechowywanie danych przy wyjątkowo niskich kosztach, przy założeniu bardzo rzadkiego dostępu (należy zwrócić uwagę, że pobieranie danych z archiwów nie jest już tak tanie, ale to osobny temat). Wspomniane klasy pamięci masowej obsługują również dodatkowe mechanizmy kontroli przestrzegania zasad, co ma zapobiec przypadkowemu lub celowemu usunięciu archiwów o kluczowym znaczeniu.

Po drugie przepisy dotyczące prywatności i przechowywania danych, takie jak RODO i CCPA, wymagają od inżynierów danych aktywnego zarządzania niszczeniem danych w celu poszanowania „prawa użytkowników do bycia zapomnianym". Inżynierowie danych muszą wiedzieć, jakie dane konsumentów przechowują, oraz muszą dysponować procedurami niszczenia danych w odpowiedzi na żądania i wymagania dotyczące zgodności z przepisami.

Niszczenie danych w hurtowni danych w chmurze jest bardzo proste. Semantyka SQL umożliwia usuwanie wierszy spełniających warunki określone w klauzuli `where`. Niszczenie danych było trudniejsze w jeziorach danych, gdzie domyślnym sposobem składowania danych w pamięci masowej był wzorzec WORM (ang. *write once, read many* — dosłownie: zapisz raz, czytaj wiele razy). Łatwe zarządzanie transakcjami usuwania na dużą skalę umożliwiają takie narzędzia jak Hive ACID i Delta Lake. Końcową fazę cyklu życia inżynierii danych usprawniają także nowe generacje narzędzi do zarządzania metadanymi, rodowodem danych i katalogowaniem.

Etyka i prywatność

Z ostatnich kilku lat doświadczeń z naruszeń bezpieczeństwa danych, dezinformacji i niewłaściwego obchodzenia się z danymi jasno wynika jeden wniosek: dane mają wpływ na ludzi. W przeszłości dane „żyły na Dzikim Zachodzie" — były swobodnie gromadzone i sprzedawane jak karty baseballowe. Te czasy już dawno minęły. Podczas gdy obsługa zagadnień etyki i prywatności danych, podobnie jak zabezpieczenia danych, były kiedyś uważane za nieobowiązkowe dodatki, dziś mają one kluczowe znaczenie w całym cyklu życia danych. Inżynierowie danych muszą postępować

właściwie nawet wtedy, gdy nikt inny na nich nie patrzy, ponieważ nadejdą czasy, gdy będą ich obserwować wszyscy[6]. Mamy nadzieję, że coraz więcej organizacji będzie dążyć do stworzenia kultury właściwej obsługi etyki danych i dbania o ich prywatność.

W jaki sposób etyka i prywatność wpływają na cykl życia inżynierii danych? Inżynierowie danych muszą zapewnić maskowanie przechowywanych w zbiorach danych osobowych (ang. *personally identifiable information* — PII) oraz innych poufnych informacji; podczas przekształcania zbiorów danych można łatwo zidentyfikować i śledzić tzw. stronniczość (ang. *bias*). Przepisy prawa w tym zakresie są coraz ostrzejsze, a kary coraz wyższe. Zadbaj o to, aby Twoje zasoby danych były zgodne ze wszystkimi przepisami dotyczącymi danych, takimi jak RODO i CCPA. Warto potraktować to poważnie. Wskazówki pomocne w kwestii wprowadzania zagadnień związanych z etyką i prywatnością danych do całego cyklu życia inżynierii danych zostały zamieszczone w całej treści książki.

DataOps

Podstawową ideą DataOps jest zastosowanie do danych najlepszych praktyk metodologii Agile, DevOps i statystycznej kontroli procesów (ang. *statistical proces control* — SPC). Podczas gdy DevOps ma na celu poprawę procesu wydawania i jakości oprogramowania, DataOps ma taki sam cel w odniesieniu do produktów danych.

Produkty danych różnią się od produktów oprogramowania sposobami wykorzystania danych. Oprogramowanie zapewnia docelowym użytkownikom określone funkcjonalności i możliwości techniczne. Z kolei produkt danych jest zbudowany wokół solidnej logiki biznesowej i metryk, na podstawie których użytkownicy podejmują decyzje lub budują modele umożliwiające automatyzację działań. Inżynierowie danych powinni rozumieć zarówno techniczne aspekty tworzenia oprogramowania, jak i logikę biznesową, jakość i metryki, które tworzą doskonałe produkty danych.

Podobnie jak DevOps, DataOps czerpie wiele zasad z metodyki produkcji Lean oraz teorii zarządzania łańcuchem dostaw, zajmuje się rozwiązywaniem problemów związanych z ludźmi, procesami i technologiami w celu skrócenia czasu do uzyskania wartości. Oto definicja DataOps społeczności skupionych wokół witryny Data Kitchen (ekspertów w dziedzinie DataOps)[7]:

> DataOps to zbiór praktyk technicznych, przepływów pracy, norm kulturowych i wzorców architektonicznych, które umożliwiają:
>
> - wprowadzanie innowacji i przeprowadzanie eksperymentów pozwalających dostarczać klientom nowych informacji z coraz większą szybkością;
> - wyjątkowo wysoką jakość danych i bardzo niski poziom błędów;
> - współpracę między złożonymi grupami ludzi, technologii i środowisk;
> - czytelny pomiar, monitorowanie i przejrzystość wyników.

[6] Opowiadamy się za poglądem, że zachowanie etyczne polega na właściwym postępowaniu wtedy, gdy nikt nie patrzy. Jest to koncepcja, która pojawia się w publikacjach C.S. Lewisa, Charlesa Marshalla i wielu innych autorów.

[7] *What Is DataOps*, strona FAQ witryny DataKitchen, dostęp 5 maja 2022, *https://oreil.ly/Ns06w*.

Praktyki Lean (takie jak redukcja czasu realizacji i minimalizacja defektów) właściwe dla DataOps oraz wynikającą z ich zastosowania poprawę jakości i produktywności można zauważyć zarówno w operacjach związanych z oprogramowaniem, jak i danymi.

DataOps to przede wszystkim zestaw nawyków kulturowych; zespół inżynierii danych musi przyjąć cykl komunikacji i współpracy z przedstawicielami biznesowymi, dążyć do eliminowania silosów, stosować ciągłe uczenie się na sukcesach i błędach oraz wykonywać szybkie iteracje. Tylko wtedy, gdy zostaną wprowadzone te nawyki kulturowe, zespół może skorzystać z technologii i narzędzi w celu uzyskania najlepszych wyników.

W zależności od dojrzałości firmy w kontekście danych inżynier danych ma pewne możliwości włączenia technik DataOps w strukturę całego cyklu życia inżynierii danych. Jeśli firma nie ma wcześniej istniejącej infrastruktury danych lub nie stosuje praktyk związanych z danymi, zastosowanie DataOps jest szansą na zbudowanie kultury postępowania z danymi od podstaw. W przypadku istniejącego projektu lub infrastruktury, w których brakuje DataOps, inżynierowie danych mogą rozpocząć dodawanie DataOps do swoich przepływów pracy. Sugerujemy zacząć od wprowadzenia mechanizmów obserwowalności i monitorowania, aby uzyskać wgląd w wydajność systemu, a następnie dodać elementy automatyzacji i systemów reagowania na incydenty. W dojrzałej firmie inżynierowie danych mogą współpracować z istniejącym zespołem DataOps w celu poprawy cyklu życia inżynierii danych. We wszystkich przypadkach inżynierowie danych muszą być świadomi filozofii i technicznych aspektów DataOps.

DataOps składa się z trzech podstawowych elementów technicznych: automatyzacji, monitorowania i obserwowalności oraz reagowania na incydenty (rysunek 2.8). Przyjrzyjmy się każdemu z tych elementów oraz ich związkom z cyklem życia inżynierii danych.

Rysunek 2.8. Trzy filary DataOps

Automatyzacja

Automatyzacja zapewnia niezawodność i spójność procesu DataOps oraz umożliwia inżynierom danych szybkie wdrażanie nowych funkcji produktu oraz usprawnień w istniejących przepływach pracy. Automatyzacja DataOps ma podobną strukturę i przepływy pracy jak DevOps. Składa się z mechanizmów zarządzania zmianami (środowiska, kodu i wersji danych), potoków ciągłej integracji i ciągłego wdrażania (CI/CD) oraz konfiguracji jako kodu. Podobnie jak w przypadku DevOps, praktyki DataOps pozwalają na monitorowanie i utrzymywanie niezawodności technologii i systemów (potoki danych, orkiestracja itp.). Zapewniają również dodatkowy wymiar pozwalający między innymi na sprawdzanie jakości danych, przenoszenie danych (modelu) oraz sprawdzanie integralności metadanych.

Spróbujmy pokrótce omówić ewolucję automatyzacji DataOps w hipotetycznej organizacji. Organizacja o niskim poziomie dojrzałości DataOps często próbuje zaplanować wiele etapów procesów transformacji danych przy użyciu zadań mechanizmu harmonogramowania cron. Takie rozwiązanie

sprawdza się przez jakiś czas. Wraz ze wzrostem złożoności potoków danych może się wydarzyć kilka rzeczy. Jeśli zadania cron są hostowane w egzemplarzu usługi w chmurze, mogą wystąpić problemy z działaniem tego egzemplarza, co może doprowadzić do nieoczekiwanego zatrzymania określonego zadania. Gdy odstępy między zadaniami są krótkie, działanie jednego z zadań może się przedłużyć, co może spowodować niepowodzenie kolejnego zadania lub wygenerowanie nieaktualnych danych. Inżynierowie mogą nie być świadomi niepowodzeń określonych zadań, dopóki nie usłyszą od analityków, że ich raporty są nieaktualne.

Kiedy dojrzałość organizacji w kontekście danych wzrasta, inżynierowie danych zazwyczaj zaczynają stosować framework orkiestracji, na przykład Airflow lub Dagster. Inżynierowie danych zdają sobie sprawę, że stosowanie frameworka Airflow stanowi obciążenie operacyjne, ale korzyści z orkiestracji ostatecznie przewyższają koszty związane z dodatkową złożonością. Inżynierowie danych stopniowo migrują swoje zadania cron do zadań Airflow. Teraz przed uruchomieniem zadań sprawdzane są zależności. W określonym przedziale czasu można zmieścić więcej zadań transformacji, ponieważ każde zadanie może rozpocząć się natychmiast po otrzymaniu sygnału o dostępności danych z górnej części strumienia przetwarzania, a nie w ustalonym z góry, określonym czasie.

Zespół inżynierów danych wciąż ma możliwość wprowadzenia ulepszeń operacyjnych. W pewnym momencie może dojść do sytuacji, w której badacz danych wdroży uszkodzony DAG (ang. *directed acyclic graph* — dosłownie: acykliczny graf skierowany), co spowoduje wyłączenie serwera WWW Airflow i pozbawienie zespołu inżynierów danych dostępu do informacji. Po wystąpieniu pewnej liczby tego rodzaju incydentów członkowie zespołu inżynierii danych uświadamiają sobie, że nie mogą nadal zezwalać na ręczne wdrażanie grafów DAG. W kolejnej fazie dojrzałości operacyjnej wdrażają mechanizmy automatycznego wdrażania grafów DAG. Przed wdrożeniem nowe grafy DAG są testowane, a procesy monitorowania zapewniają ich prawidłowe działanie. Ponadto do czasu sprawdzenia poprawności instalacji inżynierowie danych blokują wdrażanie nowych zależności języka Python. Po zastosowania mechanizmów automatyzacji zespół ds. danych zaczyna obserwować znacznie mniej problemów.

Jedna z zasad manifestu DataOps (*https://oreil.ly/2LGwL*) brzmi „uwzględniaj zmiany". Nie oznacza to zmiany dla samej zmiany, ale raczej zmiany zorientowane na cel. Możliwości usprawnień operacyjnych istnieją na każdym etapie wdrażania mechanizmów automatyzacji. Nawet na wysokim poziomie dojrzałości firmy, który opisaliśmy powyżej, pozostaje wiele miejsca do usprawnień. Inżynierowie danych mogą wdrożyć framework orkiestracji nowej generacji, który zapewnia lepsze możliwości wykorzystania metadanych. Mogą też spróbować opracować framework, który automatycznie buduje grafy DAG na podstawie specyfikacji rodowodu danych. Najważniejsze, aby uświadomić sobie, że inżynierowie danych powinni stale dążyć do wprowadzenia usprawnień w automatyzacji, które zmniejszą ich obciążenie pracą i zwiększą dostarczaną wartość biznesową.

Obserwowalność i monitorowanie

Jak często mówimy naszym klientom: „dane to cichy zabójca"? Widzieliśmy niezliczone przykłady złych danych utrzymujących się w raportach przez miesiące lub nawet lata. Kierownictwo może podjąć na podstawie tych złych danych kluczowe decyzje i odkryć błąd znacznie później. Efekty takich decyzji dla biznesu są zazwyczaj złe, a czasem katastrofalne. Może dojść do niepowodzenia inicjatyw, konieczności zamykania projektów i marnowania wielu lat pracy. W niektórych, najgorszych przypadkach złe dane mogą doprowadzić firmę do ruiny finansowej.

Fatalne skutki może mieć również nagłe zatrzymanie działania systemów przygotowujących dane do raportów. W efekcie raporty mogą być opóźnione nawet o kilka dni. Zespół ds. danych nie dowie się tego, dopóki interesariusze nie zapytają o powód opóźnienia raportów lub umieszczenia w nich nieaktualnych informacji. W końcu może dojść do sytuacji, w której różni interesariusze stracą zaufanie do możliwości zespołu ds. podstawowych danych i zaczną tworzyć własne zespoły zajmujące się danymi. W rezultacie powstaje wiele różnych, niestabilnych systemów, niespójnych raportów i silosów informacyjnych.

Jeśli nie obserwujesz i nie monitorujesz swoich danych i systemów, które je generują, z pewnością doświadczysz własnej historii horroru dotyczącego danych. Kluczowe znaczenie dla zapobieżenia wszelkim problemom w całym cyklu życia inżynierii danych ma zastosowanie mechanizmów obserwowalności, monitorowania, rejestrowania, ostrzegania i śledzenia. Zalecamy włączenie mechanizmów SPC, pozwalających zrozumieć potencjalne problemy z monitorowanymi zdarzeniami oraz incydenty, na które warto reagować.

Wspomniana wcześniej w tym rozdziale metoda DODD Petrelli zapewnia doskonały framework obsługi obserwowalności danych. DODD to koncepcja przypominająca stosowaną w inżynierii oprogramowania technikę TDD (ang. *test driven development* — dosłownie: rozwój sterowany testami)[8]:

> Celem DODD jest zapewnienie wszystkim zaangażowanym w łańcuch danych wglądu w dane i aplikacje przetwarzające dane, tak aby wszyscy interesariusze należący do łańcucha wartości danych mieli możliwość identyfikowania zmian w danych lub aplikacjach danych na każdym etapie ich życia — od pozyskiwania, przez transformację, aż do analizy — aby rozwiązywać problemy z danymi lub im zapobiegać. Technika DODD koncentruje się na tym, aby obserwowalność danych była pierwszorzędnym czynnikiem w cyklu życia inżynierii danych.

Wiele aspektów monitorowania i obserwowalności w całym cyklu życia inżynierii danych omówimy w dalszych rozdziałach.

Reagowanie na incydenty

Dobrze funkcjonujący zespół danych stosujący techniki DataOps potrafi szybko dostarczać nowe produkty danych. Ale błędy nieuchronnie się zdarzają. System może mieć przestoje, nowy model danych może zakłócić tworzenie raportów w dolnej części strumienia przetwarzania, model uczenia maszynowego może stać się przestarzały i dostarczać złe prognozy — cykl życia inżynierii danych może zostać zakłócony przez niezliczone problemy. **System reagowania na incydenty** to mechanizm polegający na wykorzystaniu wspomnianych wcześniej funkcji automatyzacji i obserwowalności w celu szybkiego zidentyfikowania pierwotnych przyczyn incydentów i rozwiązania ich tak niezawodnie i szybko, jak to możliwe.

Reagowanie na incydenty to nie tylko technologie i narzędzia, choć ich stosowanie może przynieść korzyści; chodzi również o otwartą i pozbawioną szukania winnych komunikację, zarówno w zespole inżynierii danych, jak i w całej organizacji. Jak powiedział Werner Vogels, CTO firmy Amazon Web Services: „Wszystko się psuje przez cały czas”. Inżynierowie danych muszą być przygotowani na awarie i gotowi do jak najszybszej i skutecznej reakcji.

[8] Andy Petrella, *Data Observability Driven Development: The Perfect Analogy for Beginners*, Kensu, dostęp 5 maja 2022, *https://oreil.ly/MxvSX*.

Inżynierowie danych powinni znajdować problemy proaktywnie, zanim zostaną znalezione przez interesariuszy biznesowych. Awarie się zdarzają, a gdy interesariusze lub docelowi użytkownicy końcowi zauważą problemy, z pewnością o nich opowiedzą. Będą z tego niezadowoleni. Odniosą lepsze wrażenie, gdy będą mieli możliwość zgłoszenia napotkanych problemów wyznaczonemu zespołowi i zobaczą, że zgłoszone przez nich problemy są aktywnie rozwiązywane. Zastanów się, kiedy zaufałbyś zespołowi rozwiązującemu Twój problem, gdybyś był docelowym użytkownikiem? Budowanie zaufania zajmuje dużo czasu, a żeby je stracić, wystarczy kilka minut. Reagowanie na incydenty polega zarówno na reagowaniu na incydenty, które już wystąpiły, jak i na proaktywnym wykonywaniu działań zaradczych, zanim problemy się pojawią.

Podsumowanie informacji o DataOps

W czasie kiedy powstaje ta książka, koncepcja DataOps nadal dopiero się rodzi. Zwolennicy DataOps wykonali świetną robotę polegającą na dostosowaniu zasad DevOps do domeny danych i zmapowaniu początkowej wizji w manifeście DataOps i innych zasobach. Inżynierowie danych przyczyniliby się do rozwoju tej koncepcji, gdyby nadali praktykom DataOps wysoki priorytet we wszystkich swoich projektach. Początkowy wysiłek przyniesie znaczące długoterminowe korzyści w postaci szybszego dostarczania produktów, lepszej niezawodności i dokładności danych oraz większej ogólnej wartości dla biznesu.

Stan operacji w inżynierii danych w porównaniu z inżynierią oprogramowania jest wciąż bardzo niedojrzały. Wielu narzędziom do inżynierii danych, zwłaszcza starszym monolitom, brakuje mechanizmów automatyzacji. Niedawno pojawił się ruch mający na celu zastosowanie najlepszych praktyk automatyzacji w całym cyklu życia inżynierii danych. Takie narzędzia jak Airflow utorowały drogę nowej generacji narzędzi do automatyzacji tych danych i zarządzania nimi. Ogólne praktyki DataOps są bardzo obiecujące. Sugerujemy, aby firmy starały się je zastosować w jak najszerszym zakresie, z uwzględnieniem obecnie dostępnych narzędzi i stanu wiedzy.

Architektura danych

Architektura danych odzwierciedla obecny i przyszły stan systemów danych, które wspierają długoterminowe potrzeby organizacji i jej strategię w zakresie danych. Ponieważ wymagania organizacji dotyczące danych prawdopodobnie będą się szybko zmieniać, a nowe narzędzia i praktyki wydają się pojawiać niemal codziennie, inżynierowie danych powinni dobrze rozumieć problematykę architektury danych. Zagadnienia związane z architekturą danych zostały szczegółowo omówione w rozdziale 3., w tym miejscu chcemy podkreślić jednak, że architektura danych jest podstawą cyklu życia inżynierii danych.

Inżynierowie danych powinni najpierw zrozumieć potrzeby firmy, a następnie zebrać wymagania dotyczące nowych przypadków użycia. Następnie inżynier danych powinien przetłumaczyć te wymagania, aby zaprojektować nowe sposoby przechwytywania i serwowania danych, zrównoważone pod kątem kosztów i prostoty operacyjnej. Oznacza to konieczność znajomości kompromisów związanych ze wzorcami projektowymi, technologiami i narzędziami w systemach źródłowych, pozyskiwaniem, przechowywaniem, przekształcaniem i dostarczaniem danych.

Nie oznacza to, że inżynier danych jest architektem danych, ponieważ zazwyczaj są to dwie oddzielne role. Jeśli inżynier danych współpracuje z architektem danych, powinien być w stanie dostarczyć produkty w odpowiedzi na projekty architekta danych i przekazywać architektowi informacje zwrotne dotyczące architektury.

Orkiestracja

> Uważamy, że orkiestracja ma znaczenie, ponieważ w kontekście danych postrzegamy ją jako centralny punkt zarówno platformy danych, jak i cyklu życia danych oraz cyklu życia oprogramowania.
>
> — Nick Schrock, założyciel firmy Elementl[9]

Orkiestracja to nie tylko centralny proces DataOps, ale także kluczowa część przepływu inżynierii i wdrażania dla zadań związanych z danymi. Czym jest więc orkiestracja?

Orkiestracja to proces koordynowania wielu zadań w celu jak najszybszego i jak najwydajniejszego ich uruchomienia w zaplanowanej kadencji. Na przykład narzędzia orkiestracji, takie jak Apache Airflow, często są wykorzystywane jako **narzędzia harmonogramowania**. Nie są one zbyt dokładne. Klasyczne narzędzia do obsługi harmonogramów, takie jak cron, są świadome jedynie czasu, natomiast silnik orkiestracji buduje metadane, zazwyczaj w postaci skierowanego wykresu acyklicznego (DAG), dotyczące zależności pomiędzy zadaniami. Graf DAG może być uruchamiany raz lub można zaplanować jego uruchamianie w ustalonych odstępach czasu: codziennie, co tydzień, co godzinę, co pięć minut itp.

Gdy omawiamy orkiestrację w tej książce, zakładamy, że system orkiestracji jest w trybie online i charakteryzuje się wysoką dostępnością. Dzięki temu system orkiestracji jest zdolny do stałego wykonywania zadań wykrywania i monitorowania bez interwencji człowieka oraz uruchamia nowe zadania za każdym razem, gdy zostaną wdrożone. System orkiestracji monitoruje zadania, którymi zarządza, a po zakończeniu wewnętrznych zależności grafu DAG uruchamia nowe zadania. Potrafi również monitorować zewnętrzne systemy i narzędzia pod kątem odbierania oczekiwanych danych oraz spełniania zadanych kryteriów. Gdy jakieś warunki wykraczają poza ustalone normy, system zgłasza również błędy i wysyła alerty za pośrednictwem poczty e-mail lub innych kanałów komunikacyjnych. Dla conocnych potoków danych można ustawić oczekiwany czas ukończenia na dziesiątą rano. Jeśli do tego czasu zadania nie zostaną wykonane, do inżynierów danych i konsumentów zostaną wysłane odpowiednie alerty.

Systemy orkiestracji budują również funkcje historii zadań, wizualizacji i alertów. Zaawansowane silniki orkiestracji potrafią wypełniać nowe grafy DAG lub dodawać do grafów DAG pojedyncze zadania. Obsługują również zależności w ustalonym przedziale czasu. Na przykład przed uruchomieniem zadania tworzenia raportu miesięcznego można sprawdzić, czy dla pełnego miesiąca zostały ukończone zadania ETL.

Orkiestracja od dawna stanowiła kluczową zdolność do przetwarzania danych, ale często nie była najważniejsza ani dostępna dla nikogo poza największymi firmami. Przedsiębiorstwa wykorzystywały różne narzędzia do zarządzania przepływami pracy, ale były one drogie, niedostępne dla małych

[9] Ternary Data, *An Introduction to Dagster: The Orchestrator for the Full Data Lifecycle — UDEM June 2021*, nagranie wideo w serwisie YouTube, 1:09:40, *https://oreil.ly/HyGMh*.

start-upów i na ogół nierozszerzalne. W drugim dziesięcioleciu XXI wieku niezwykle popularny był system Apache Oozie, ale został on zaprojektowany do pracy w klastrze Hadoop i posługiwanie się nim w bardziej heterogenicznym środowisku było trudne. Pod koniec pierwszego dziesięciolecia XXI wieku firma Facebook opracowała do swojego wewnętrznego użytku system Dataswarm; stało się to inspiracją do powstania popularnych narzędzi. Jednym z nich był Airflow, wprowadzony w 2014 roku przez firmę Airbnb.

Airflow od samego początku był oprogramowaniem typu open source i szybko zyskał dużą popularność. Narzędzie zostało napisane w Pythonie, dzięki czemu może być łatwo rozszerzane na potrzeby prawie każdego przypadku użycia, jaki można sobie wyobrazić. Chociaż istnieje wiele innych interesujących projektów orkiestracji open source, takich jak Luigi i Conductor, Airflow jest na razie zdecydowanym liderem. System Airflow pojawił się w momencie, gdy przetwarzanie danych stało się bardziej abstrakcyjne i znacznie bardziej dostępne, a inżynierowie zaczęli bardziej interesować się koordynacją złożonych przepływów obejmujących wiele procesorów i systemów pamięci masowej, zwłaszcza w środowiskach chmurowych.

W czasie kiedy powstawała ta książka, trwały prace nad kilkoma projektami open source naśladującymi najlepsze elementy podstawowego projektu Airflow, a jednocześnie zawierającymi usprawnienia w jego kluczowych obszarach. Niektóre z najciekawszych przykładów to Prefect i Dagster, mające na celu poprawę przenośności i testowalności wewnętrznych grafów DAG, aby umożliwić inżynierom łatwiejsze przejście od rozwoju lokalnego do produkcji. Argo to silnik orkiestracji zbudowany wokół prymitywów systemu Kubernetes, natomiast Metaflow to projekt open source opracowany przez firmę Netflix, którego celem jest poprawa orkiestracji na potrzeby inżynierii danych.

Warto podkreślić, że orkiestracja jest koncepcją ściśle wsadową. Alternatywą dla grafów DAG orkiestrowanych zadań są strumieniowe grafy DAG. Budowanie strumieniowych grafów DAG i ich utrzymywanie jest dużym wyzwaniem, ale zastosowanie platform przetwarzania strumieniowego nowej generacji, takich jak Pulsar, pozwala znacznie zmniejszyć obciążenia inżynieryjne i operacyjne. Więcej informacji o tych projektach można znaleźć w rozdziale 8.

Inżynieria oprogramowania

Inżynieria oprogramowania zawsze była centralną dziedziną dla inżynierów danych. We wczesnych dniach współczesnej inżynierii danych (2000 – 2010) inżynierowie danych pracowali nad niskopoziomowymi frameworkami i pisali zadania MapReduce w językach C, C++ i Java. W szczytowym momencie ery big data (połowa drugiego dziesięciolecia XXI wieku) inżynierowie zaczęli używać frameworków, które pozwalały abstrahować od tych niskopoziomowych szczegółów.

Taki stan trwa do dziś. Hurtownie danych w chmurze obsługują zaawansowane przekształcanie danych przy użyciu semantyki SQL, a dzięki odejściu od konieczności znajomości szczegółów kodowania niskiego poziomu na rzecz łatwych w życiu ramek danych, narzędzia takie jak Spark stały się bardziej przyjazne dla użytkowników. Pomimo tej abstrakcji inżynieria oprogramowania nadal ma dla inżynierii danych kluczowe znaczenie. Poniżej krótko omówiono kilka typowych obszarów inżynierii oprogramowania, które mają zastosowanie dla cyklu życia inżynierii danych.

Podstawowy kod przetwarzania danych

Chociaż podstawowy kod przetwarzania danych stał się bardziej abstrakcyjny i łatwiejszy w zarządzaniu, nadal musi zostać napisany i występuje w całym cyklu życia inżynierii danych. Niezależnie od tego, czy chodzi o pozyskiwanie, transformację, czy serwowanie danych, inżynierowie danych muszą biegle i wydajnie posługiwać się frameworkami i językami takimi jak Spark, SQL lub Beam (nie podzielamy poglądu, że SQL nie jest kodem).

Inżynierowie danych powinni również dobrze rozumieć odpowiednie metodologie testowania kodu, takie jak testy jednostkowe, testy regresji, testy integracyjne, testy od końca do końca oraz testy dymne.

Rozwój frameworków open source

Wielu inżynierów danych angażuje się w rozwój frameworków open source. Stosują te frameworki do rozwiązywania określonych problemów w cyklu życia inżynierii danych, a następnie kontynuują rozwijanie kodu frameworków w celu usprawnienia narzędzi dla swoich przypadków użycia oraz udostępnienia nowych możliwości społeczności.

W erze big data byliśmy świadkami kambryjskiej eksplozji frameworków przetwarzania danych wewnątrz ekosystemu Hadoop. Narzędzia te koncentrowały się przede wszystkim na fazach przekształcania i serwowania cyklu życia inżynierii danych. Proces powstawania narzędzi inżynierii danych nie ustał ani nie spowolnił, ale nacisk przesunął się w górę drabiny abstrakcji — z dala od bezpośredniego przetwarzania danych. Narzędzia open source nowej generacji pomagają inżynierom w zarządzaniu danymi, ich ulepszaniu, łączeniu, optymalizacji i monitorowaniu.

Na przykład od 2015 roku do początku lat dwudziestych XXI stulecia Airflow zdominował przestrzeń orkiestracji. Obecnie pojawiła się nowa partia konkurencyjnych systemów open source (w tym Prefect, Dagster i Metaflow), których celem jest pokonanie obserwowanych ograniczeń Airflow, zapewnienie lepszej obsługi metadanych, poprawa przenośności oraz zarządzania zależnościami. Nikt się nie domyśla, dokąd zmierza przyszłość orkiestracji.

Zanim inżynierowie danych przystąpią do projektowania nowych wewnętrznych narzędzi, powinni przeanalizować krajobraz narzędzi dostępnych publicznie. Warto obserwować metryki takie jak TCO (ang. *total cost of ownership* — dosłownie: całkowity koszt posiadania) oraz koszty alternatywne związane z wdrożeniem narzędzia. Jest duża szansa, że istnieje już projekt open source rozwiązujący Twój problem. Zamiast wymyślać koło na nowo, lepiej skorzystać z pracy innych.

Przesyłanie strumieniowe

Strumieniowe przetwarzanie danych jest z natury bardziej skomplikowane niż przetwarzanie wsadowe, a narzędzia i paradygmaty przetwarzania strumieniowego są w porównaniu z przetwarzaniem wsadowym mniej dojrzałe. Ponieważ strumieniowe przesyłanie danych staje się coraz bardziej powszechne na każdym etapie cyklu życia inżynierii danych, inżynierowie danych muszą rozwiązywać interesujące problemy inżynierii oprogramowania.

Na przykład zadania przetwarzania danych, które uważamy za oczywiste w świecie przetwarzania wsadowego (takie jak wyznaczanie złączeń), często stają się bardziej skomplikowane, jeśli spróbujemy je wykonywać w czasie rzeczywistym, i wymagają zastosowania bardziej wyrafinowanych technik inżynierii oprogramowania. Inżynierowie muszą również pisać kod, aby zastosować różne metody **wyznaczania okien** (ang. *windowing*). Technika ta umożliwia systemom czasu rzeczywistego obliczanie cennych metryk, takich jak statystyki końcowe. Inżynierowie mają do wyboru wiele frameworków, w tym różne platformy programowania funkcyjnego (OpenFaaS, AWS Lambda, Google Cloud Functions) umożliwiające obsługę pojedynczych zdarzeń lub dedykowane procesory strumieniowe (Spark, Beam, Flink lub Pulsar), za pomocą których mogą wykonywać analizę strumieni w celu tworzenia w czasie rzeczywistym raportów oraz wykonywania innych zadań.

Infrastruktura jako kod

Infrastruktura jako kod (ang. *Infrastructure as a Code* — IaC) polega na zastosowaniu praktyk inżynierii oprogramowania do konfiguracji infrastruktury i zarządzania nią. Obciążenia związane z zarządzaniem infrastrukturą w erze big data zmniejszyły się, ponieważ firmy przeprowadziły migrację danych do zarządzanych systemów big data — takich jak Databricks i Amazon Elastic MapReduce (EMR) — oraz hurtowni danych w chmurze. Kiedy inżynierowie danych muszą zarządzać swoją infrastrukturą w środowisku chmury, coraz częściej nie robią tego ręcznie, poprzez uruchamianie nowych egzemplarzy i instalowanie odpowiedniego oprogramowania, lecz wykorzystują do tego celu frameworki IaC. Dostępnych jest kilka frameworków ogólnego przeznaczenia umożliwiających zautomatyzowane wdrażanie infrastruktury w oparciu o zestaw specyfikacji. Wiele z tych frameworków pozwala zarządzać zarówno usługami w chmurze, jak i infrastrukturą. Dzięki takim narzędziom jak Helm dostępne są również rozwiązania IaC korzystające z kontenerów i systemu Kubernetes.

Praktyki te są istotną częścią metodyki DevOps, ponieważ umożliwiają kontrolę wersji i powtarzalność wdrożeń. Oczywiście możliwości te są niezbędne w całym cyklu życia inżynierii danych, zwłaszcza w przypadku zastosowania praktyk DataOps.

Potoki jako kod

Potoki jako kod (ang. *pipelines as code*) to podstawowa koncepcja współczesnych systemów orkiestracji, stosowanych na każdym etapie cyklu życia inżynierii danych. Do deklarowania zadań związanych z danymi i opisywania występujących pomiędzy nimi zależności inżynierowie danych używają kodu (zazwyczaj Pythona). Silnik orkiestracji interpretuje te instrukcje w celu wykonania działań przy użyciu dostępnych zasobów.

Rozwiązywanie problemów ogólnych

W praktyce, niezależnie od tego, jakie wysokopoziomowe narzędzia zastosują inżynierowie danych w całym cyklu życia inżynierii danych, mogą oni napotkać przypadki brzegowe, które będą wymagały od nich rozwiązywania problemów przekraczających możliwości wybranych narzędzi, co spowoduje konieczność napisania niestandardowego kodu. Podczas korzystania z takich frameworków jak Fivetran, Airbyte lub Matillion inżynierowie danych mogą napotkać źródła danych

bez istniejących konektorów, co zrodzi potrzebę napisania konektora niestandardowego. Biegłość w posługiwaniu się technikami inżynierii oprogramowania jest im potrzebna w celu zrozumienia interfejsów API, pobierania i przekształcania danych, obsługiwania wyjątków i tak dalej.

Podsumowanie

Większość publikacji związanych z inżynierią danych dostępnych w przeszłości dotyczyła technologii, ale nie uwzględniała szerszego obrazu zarządzania cyklem życia danych. Dzięki temu, że technologie stają się coraz bardziej abstrakcyjne, a za ich pomocą są wykonywane coraz bardziej złożone zadania, inżynierowie danych mogą skupić się na myśleniu i działaniu na wyższym poziomie. Cykl życia inżynierii danych, wspierany przez związane z nim nurty, jest niezwykle przydatnym modelem mentalnym umożliwiającym organizację pracy związanej z inżynierią danych.

Cykl życia inżynierii danych podzieliliśmy na następujące fazy:

- Generowanie.
- Przechowywanie.
- Pozyskiwanie.
- Przekształcanie.
- Serwowanie danych.

Istnieje również kilka motywów, które obejmują cały cykl życia inżynierii danych. Są to podstawowe nurty cyklu życia inżynierii danych. Należą do nich:

- Bezpieczeństwo.
- Zarządzanie danymi.
- DataOps.
- Architektura danych.
- Orkiestracja.
- Inżynieria oprogramowania.

W całym cyklu życia danych inżynierowie danych mają do osiągnięcia kilka celów najwyższego poziomu: zapewnienie optymalnego zwrotu z inwestycji i obniżenie kosztów (finansowych i innych), zmniejszenie ryzyka (bezpieczeństwo, jakość danych) oraz maksymalizację wartości i użyteczności danych.

Kolejne dwa rozdziały omawiają, w jaki sposób te elementy wpływają na dobry projekt architektury, oraz zawierają propozycje zastosowania odpowiednich technologii. Jeśli dobrze znasz te dwa tematy, możesz przejść do części II, w której szczegółowo omawiamy każdą z faz cyklu życia inżynierii danych.

Zasoby dodatkowe

- Ludovic Santos, *A Comparison of Data Processing Frameworks* (*https://oreil.ly/tq61F*).
- Witryna internetowa DAMA International (*https://oreil.ly/mu7oI*).
- Tyler Akidau i współautorzy, *The Dataflow Model: A Practical Approach to Balancing Correctness, Latency, and Cost in Massive-Scale, Unbounded, Out-of-Order Data Processing* (*https://oreil.ly/nmPVs*).
- Strona Wikipedii *Data Processing* (*https://oreil.ly/4mllo*).
- Strona Wikipedii *Data Transformation* (*https://oreil.ly/tyF6K*).
- Chris Williams i współautorzy, *Democratizing Data at Airbnb* (*https://oreil.ly/E9CrX*).
- Strona internetowa Lean-Data *Five Steps to Begin Collecting the Value of Your Data* (*https://oreil.ly/F4mOh*).
- Jared Murrell, *Getting Started with DevOps Automation* (*https://oreil.ly/euVJJ*).
- Strona internetowa Atlassian, *Incident Management in the Age of DevOps* (*https://oreil.ly/O8zMT*).
- Nick Schrock, *An Introduction to Dagster: The Orchestrator for the Full Data Lifecycle* (*https://oreil.ly/PQNwK*).
- Carol Jang i Jove Kuang, *Is DevOps Related to DataOps?* (*https://oreil.ly/J8ZnN*).
- Strona internetowa Atlassian *The Seven Stages of Effective Incident Response* (*https://oreil.ly/Lv5XP*).
- Etai Mizrahi, *Staying Ahead of Debt* (*https://oreil.ly/uVz7h*).
- Michelle Knight, *What Is Metadata* (*https://oreil.ly/65cTA*).

Ta książka sygnalizuje kolejny krok w ewolucji i dojrzałości branży.

Bill Inmon, twórca hurtowni danych

Świat IT zachłysnął się inżynierią danych. Powstało mnóstwo technologii, platform i produktów, a nawet języków programowania służących inżynierom danych. Ten dynamiczny rozwój ma jednak dość nieoczekiwane konsekwencje: nawet bardzo zdolni inżynierowie danych mają problem w składaniu tych komponentów w spójną całość, która okazałaby się przydatna w rozwiązywaniu rzeczywistych problemów biznesowych. Dzięki tej książce uzyskasz całościowe spojrzenie na system danych. Zrozumiesz zasady rządzące cyklem życia inżynierii danych, na który składają się generowanie, przechowywanie, pozyskiwanie, przekształcanie i udostępnianie danych.

Dowiesz się, jak zastosować te zasady do wybierania technologii, aby móc tworzyć architektury, które przetrwają próbę czasu. Nauczysz się łączyć ze sobą różnorodne technologie chmurowe i wybierać najlepsze praktyki w zależności od etapu cyklu życia danych. Sporo miejsca poświęcono niezwykle dziś ważnym zagadnieniom bezpieczeństwa danych i ochrony prywatności użytkowników. Książka ułatwia zrozumienie i odpowiednie stosowanie kluczowych koncepcji inżynierii danych niezależnie od używanych technologii.

To świetne wprowadzenie do branży przenoszenia, przetwarzania i obsługi danych!

Jordan Tigani, założyciel i dyrektor generalny MotherDuck

Dowiedz się, jak:

- wygląda środowisko inżynierii danych
- oceniać problemy inżynierii danych i korzystać z frameworków
- wybierać odpowiednie technologie, architekturę danych i procesy
- stosować cykl życia inżynierii danych do projektowania i budowania solidnej architektury
- działają mechanizmy służące do zarządzania danymi i zapewniania ich bezpieczeństwa

Joe Reis zajmuje się inżynierią danych od 20 lat. Jest dyrektorem generalnym i współzałożycielem firmy konsultingowej Ternary Data, a także wykładowcą Uniwersytetu Utah.

Dr Matt Housley jest konsultantem w dziedzinie inżynierii danych i specjalistą do spraw chmury. Wcześniej interesował się analizą danych, obecnie specjalizuje się w inżynierii danych w chmurze.

Autorzy wspólnie prowadzą podcast **The Monday Morning Data Chat**.

Helion

helion.pl

HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej!

ISBN 978-83-8322-154-0

9 788383 221540

Cena: 119,00 zł