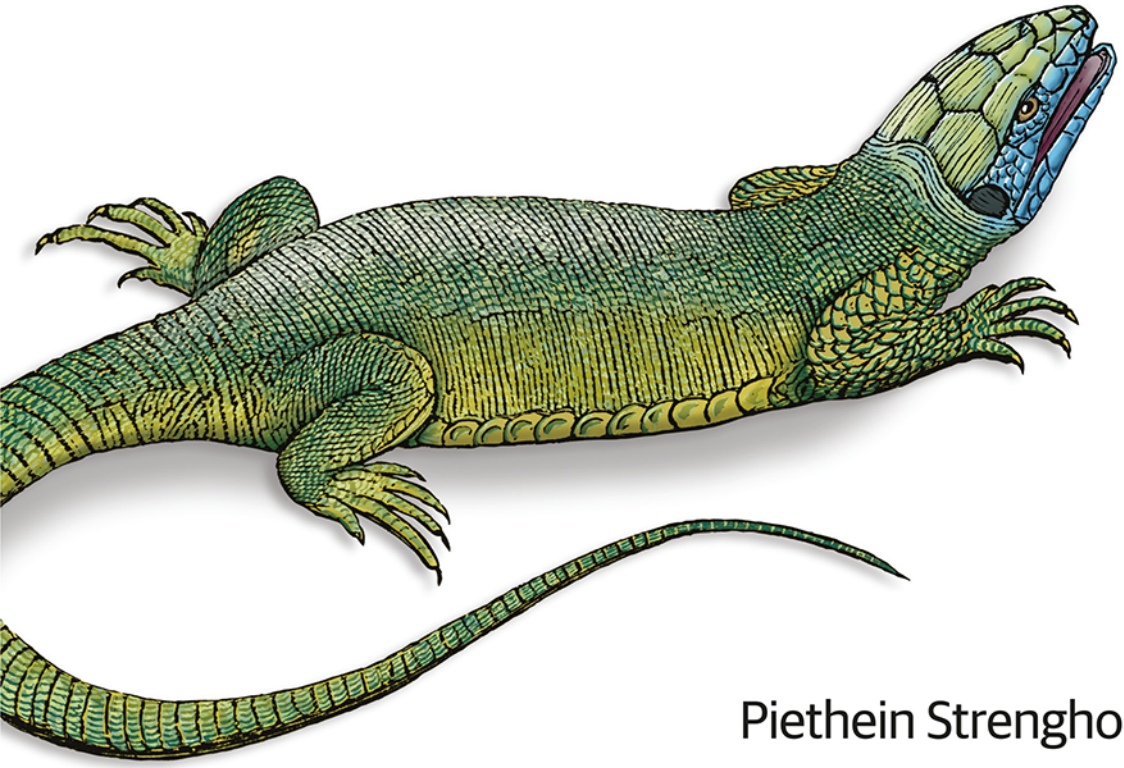
O'REILLY®

Wydanie II

Zarządzanie danymi w zbiorach o dużej skali

Nowoczesna architektura z siatką danych i technologią Data Fabric



Piethein Strengholt

Helion

Tytuł oryginału: Data Management at Scale Modern Data Architecture
with Data Mesh and Data Fabric, 2nd Edition

Tłumaczenie: Piotr Pilch

ISBN: 978-83-289-0546-7

Authorized Polish translation of the English edition of *Data Management at Scale, 2E*
ISBN 9781098138868 © 2023 Strengholt.

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
https://helion.pl/user/opinie/zadaz2
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

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)

Printed in Poland.

Spis treści

ROZDZIAŁ 4.

Zarządzanie produktami z danymi

W przypadku terminu **produkt z danymi** (ang. *data product*) możesz się zastanawiać, czy jest to tylko kolejny popularny termin. W tym rozdziale wyjaśnię, czym tak naprawdę są produkty z danymi. Przedstawię wszystkie zasadnicze informacje niezbędne do tego, aby udostępniać duże ilości danych innym domenom. Zaczniemy od niejednoznaczności terminu, ponieważ osoby posługujące się nim w praktyce kierują się różnymi definicjami i interpretacjami. W dalszej kolejności przeanalizujemy wzorzec CQRS (*Command Query Responsibility Segregation*) i dowiesz się, dlaczego powinien on mieć wpływ na projekt Twoich architektur produktów z danymi. Omówimy różne zasady projektowe związane z produktami z danymi, a ponadto przybliżę to, dlaczego z punktu widzenia Twoich konsumentów kluczowy jest dobrze opracowany i zoptymalizowany pod kątem odczytu model danych umożliwiający integrację. Dalej przyjrzymy się architekturom produktów z danymi. Dowiesz się, czym one są, jak inżynierowie mogą je tworzyć, jakie są zwykle wymagane możliwości, a także jaka jest rola metadanych. Posługując się praktycznym przykładem, postaram się zaprezentować to w jak najbardziej konkretnej postaci. Po przeczytaniu rozdziału będziesz dobrze rozumieć, w jaki sposób architektury produktów z danymi mogą okazać się pomocne w udostępnieniu konsumentom sporych ilości danych.

Czym są produkty z danymi?

Ustanawianie twórców danych odpowiedzialnymi za nie i decentralizowanie metody udostępniania danych to znakomity sposób na osiągnięcie skalowalności. Dehghani posługuje się frazą „dane jako produkt" (*https://martinfowler.com/articles/data-monolith-to-mesh.html*) i wprowadza pojęcie „produktów z danymi". Istnieje jednak spora różnica między tymi dwoma określeniami. Termin **dane jako produkt** (ang. *data as a product*) reprezentuje następujący sposób rozumowania: właściciele danych i zespoły zajmujące się aplikacjami muszą traktować dane bardziej jako w pełni odrębny produkt, za który są odpowiedzialni, niż jako produkt uboczny jakiegoś procesu zarządzanego przez innych. Dotyczy to tego, w jaki sposób dostawcy danych powinni traktować ich konsumentów jako klientów, a także zapewniać doświadczenia wywołujące zachwyt. Odnosi się to do tego, jak dane powinny być definiowane i kształtowane w celu zagwarantowania klientom najlepszego możliwego komfortu pracy.

W przypadku *produktu z danymi* sposób rozumowania różni się od tego powiązanego z pojęciem danych jako produktu, ponieważ odnosi się on do architektury. W społeczności zajmującej się danymi spotkasz się z różnymi oczekiwaniami i interpretacjami tego, jak produkty z danymi są powiązane z architekturą. Część praktyków twierdzi, że produkt z danymi nie jest jedynie zbiorem danych zawierającym odpowiednie dane z konkretnego ograniczonego kontekstu. Produkt ten uwzględnia również wszystkie komponenty niezbędne do gromadzenia i udostępniania danych, a także metadane i kod dokonujący transformacji danych. Taka interpretacja jest zgodna z tym, jak Dehghani opisuje produkt z danymi — jako kwant architekturalny (*https://martinfowler.com/articles/data-mesh-principles.html#LogicalArchitecturedataProductTheArchitecturalQuantum*), czyli „węzeł w siatce, który uwzględnia trzy komponenty strukturalne niezbędne do tego, by mógł pełnić swoją funkcję polegającą na zapewnianiu dostępu do danych analitycznych domeny jako produktu". Zgodnie z tym, co twierdzi Dehghani, te trzy komponenty to kod, dane i metadane oraz infrastruktura. Oznacza to, że Dehghani wyraźnie koncentruje się na architekturze rozwiązań, która może obejmować wszystkie komponenty konieczne do uzyskiwania, transformowania, przechowywania i udostępniania danych oraz zarządzania nimi.

Część praktyków przyjmuje różne punkty widzenia na produkty z danymi. Przykładowo firma Accenture (*https://www.accenture.com/fi-en/insights/technology/data-products*) mianem produktów z danymi określa zbiory danych, modele analityczne i raporty panelu kontrolnego. Taki punkt widzenia różni się znacząco od sposobu, w jaki postrzega to Dehghani, ponieważ koncentruje się on na fizycznej reprezentacji danych, a ponadto niekoniecznie obejmuje jakiekolwiek metadane, kod lub infrastrukturę. A zatem zanim możliwe będzie zaprojektowanie architektury, trzeba najpierw ustalić wspólną terminologię w odniesieniu do produktów z danymi i określić, co musi zostać uwzględnione, a co nie.

Problemy z łączeniem ze sobą kodu, danych, metadanych i infrastruktury

W idealnym świecie dane i metadane są umieszczane w pakiecie i dostarczane razem jako produkty z danymi. Ferd Scheepers, który w firmie ING Tech Group Services jest głównym architektem informacji, miał w ramach konferencji Domain-Driven Design Europe 2022 prezentację poświęconą zarządzaniu metadanymi. W trakcie prezentacji wyjaśnił, że metadane są kluczowe z punktu widzenia zarządzania danymi, ponieważ „metadane to dane zapewniające informacje o innych danych". W celu wzmocnienia znaczenia tej kwestii posłużył się analogią: jak wygląda dystrybucja kwiatów w Holandii.

Aalsmeer Flower Auction to największa na świecie aukcja kwiatów. Od poniedziałku do piątku każdego dnia sprzedaje się około 43 milionów kwiatów oraz 5 milionów roślin. Każdego dnia roboczego w aukcji uczestniczą handlowcy z całego świata. W momencie dostarczenia kwiaty są etykietowane i wysyłane do wyznaczonych magazynów. Z punktu widzenia kupujących cały proces przebiega płynnie: w sposób cyfrowy składają oni zamówienia, podają ilości oraz cenę i miejsce docelowe. Bardzo istotną częścią całej historii jest to, że aukcja całkowicie opiera się na metadanych. Wszystkie pudełka z kwiatami są wyposażone w kody kreskowe uwzględniające zasadnicze informacje o ich zawartości, takie jak kraj i miejscowość pochodzenia, ilość, początkowa cena licytacji, waga, daty wyprodukowania i ważności, nazwa producenta itd. Po sprzedaniu kwiatów dodawany

jest kolejny kod kreskowy z informacjami dotyczącymi wysyłki, czyli szczegóły kupującego, który stał się właścicielem kwiatów, miejsce docelowe, instrukcje związane z wysyłką itd. Bez metadanych aukcja kwiatów nie byłaby w stanie funkcjonować.

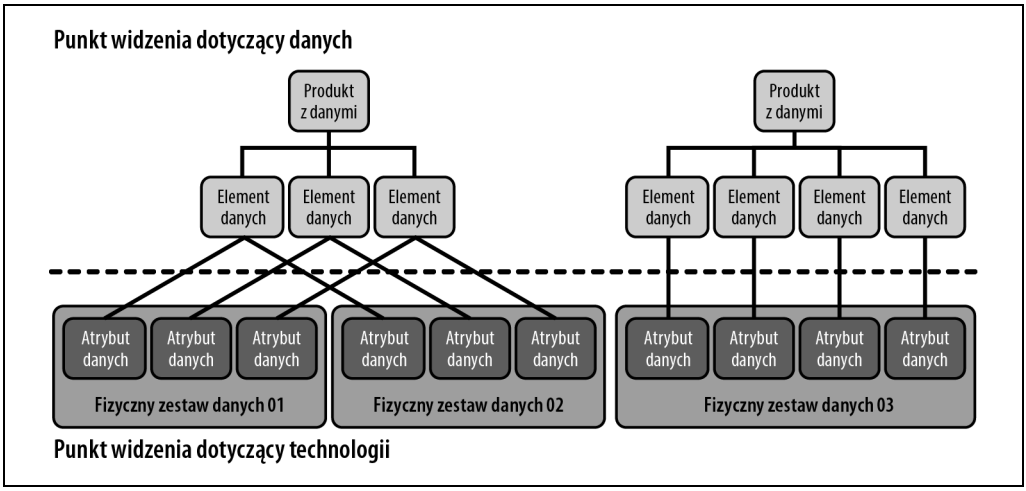
W ramach aukcji kwiaty i metadane są zawsze razem dystrybuowane oraz zarządzane, gdyż obiekty fizyczne są ze sobą połączone. W świecie cyfrowym nie odbywa się to jednak w ten sposób! Warto zaprezentować kilka przykładów, które sprawią, że stanie się to bardziej przejrzyste. Jeśli używasz centralnego katalogu danych do opisywania wszystkich swoich danych i zarządzania nimi, dane i metadane są administrowane niezależnie od siebie. W tym przykładzie całość metadanych znajduje się w Twoim katalogu danych, a wszystkie dane fizyczne są przechowywane gdzie indziej (prawdopodobnie w wielu innych miejscach). Taka sama separacja dotyczy na przykład struktury pozyskiwania zależnej od metadanych, która zarządza procesem wyodrębniania i ładowania danych. Przed rozpoczęciem przetwarzania danych taka struktura nie indeksuje metadanych z setek kontenerów produktów z danymi. Struktura pozyskiwania zależna od metadanych w postaci komponentu architekturalnego generalnie zarządza metadanymi w odrębnej bazie danych, która jest oddzielona od samych danych. Struktura używa tych metadanych do kontrolowania procesu i zależności, za których pośrednictwem dane są wyodrębniane, łączone, przekształcane i ładowane.

To samo dotyczy informacji o pochodzeniu i jakości danych. Jeśli powiązane metadane zostałyby umieszczone w obrębie samego kontenera produktów z danymi, w celu kompleksowego nadzorowania jakości danych lub pochodzenia trzeba byłoby wykonywać złożone zapytania stowarzyszone obejmujące wszystkie produkty z danymi. Alternatywnie mógłbyś replikować wszystkie metadane w innej centralnej lokalizacji, ale spowodowałoby to radykalne i przesadne skomplikowanie ogólnej architektury. Poza tym w jaki sposób radziłbyś sobie z bezpieczeństwem, klasyfikacjami prywatności, etykietami poufności lub informacjami o własności w przypadku tych samych danych semantycznych mających kilka fizycznych reprezentacji? Jeżeli połączyłbyś ze sobą mocno metadane i dane, metadane byłyby duplikowane w momencie kopiowania produktów z danymi, gdyż to i to znajduje się w obrębie tej samej architektury. Wszystkie aktualizacje metadanych wymagałyby wtedy jednoczesnego uwzględnienia dla kilku kontenerów produktów z danymi. Byłoby to złożone przedsięwzięcie wymagające, aby wszystkie te kontenery były zawsze dostępne. W celu przeprowadzania tych aktualizacji w sposób asynchroniczny mógłbyś zaimplementować architekturę nakładkową, ale ponownie — spowodowałoby to radykalne i nadmierne skomplikowanie ogólnej architektury.

Podsumowując, postrzeganie produktu z danymi jako kontenera łączącego ze sobą kod, dane i metadane oraz infrastrukturę to w rzeczywistości zdecydowanie zbyt naiwne podejście. Definicja produktu z danymi wymaga modyfikacji, która lepiej odzwierciedla realia tego, jak obecnie działają platformy danych.

Produkty z danymi jako jednostki logiczne

Jeżeli poprzednia definicja nie jest odpowiednia, jak powinno się zdefiniować produkt z danymi? Jestem mocno przekonany co do tego, że zarządzanie danymi i technologią powinno być realizowane z różnych architekturalnych punktów widzenia: jeden punkt odnosi się do kwestii zarządzania danymi, a drugi do kwestii zarządzania bazową architekturą technologii. Na rysunku 4.1 pokazano, jak to może wyglądać.



Rysunek 4.1. Osobne punkty widzenia dotyczące zarządzania danymi i technologią

Dlaczego miałbyś rozdzielić kwestie zarządzania danymi i technologią? Po pierwsze, dzięki temu możesz zwiększyć opłacalność architektury i ograniczyć dodatkowe obciążenie związane z zarządzaniem złożonością infrastruktury. Stwarzasz możliwość wariantów użycia „architektur produktów z danymi", w przypadku których zarządzanych jest wiele „produktów z danymi".

Po drugie, rozdzielenie danych i architektury ułatwia podjęcie działań w sytuacjach, w których muszą być dystrybuowane te same dane (semantyczne). Dystrybucja danych może być na przykład konieczna w przypadku sprawdzania poprawności danych między domenami lub rozszerzania tych danych. W celu jednoczesnego ułatwienia wielu wariantów zastosowań czasami jest też wymagane duplikowanie i wstępne przetwarzanie danych (np. wtedy, gdy dokładnie te same dane są przechowywane z użyciem formatów plików Parquet i Delta Lake). W takiej sytuacji duplikujesz dane bez modyfikowania podstawowej semantyki. Scenariusz ten jest pokazany na rysunku 4.1: fizyczne zbiory danych 01 i 02 korzystają z tej samej semantyki, a poszczególne atrybuty danych fizycznych są połączone z tymi samymi elementami.

Po trzecie, unikasz silnego sprzężenia między danymi a metadanymi. Wyobraź sobie sytuację, w której dane i metadane są zawsze przechowywane razem w tym samym kontenerze. Jeśli na przykład zmienią się właściciele, jednostki biznesowe lub klasyfikacje, konieczne będzie też zmodyfikowanie wszystkich odpowiednich metadanych w obrębie każdej architektury produktów z danymi. Jak poza tym zapewnisz spójną własność danych? Jeżeli zduplikujesz metadane, istnieje ryzyko, że zmiana będzie także dotyczyć własności danych. Nie powinno być to możliwe w przypadku tych samych danych semantycznych mających różne reprezentacje fizyczne w osobnych lokalizacjach.

Rozdzielenie danych i metadanych zmienia definicję produktu z danymi: *produkt z danymi to autonomiczna jednostka logiczna opisująca dane przewidziane do wykorzystania*. Na poziomie tej jednostki logicznej określasz relacje z bazową architekturą technologii, czyli z fizycznymi lokalizacjami, w których są przechowywane fizyczne zasoby danych przygotowane dla konsumenta i zoptymalizowane pod kątem odczytu. Sam produkt z danymi zawiera nazwę logicznego zbioru danych, opis relacji z domeną źródłową, unikalne elementy danych, terminy biznesowe, nazwę właściciela

zbioru danych oraz odwołania do fizycznych zasobów danych (faktyczne dane). Z biznesowego punktu widzenia jest to spójne semantycznie, ale na poziomie fizycznym może mieć wiele różnych postaci i reprezentacji. Takie zmiany wymagają zdefiniowania produktu z danymi jako niezależnego od technologii. Z myślą o zapewnieniu elastyczności związane z tym metadane utrzymuje się osobno.

Prezentując w rozdziale 9. zrzuty ekranu metamodelu powiązanego z zarządzaniem produktami z danymi, skonkretyzuję logiczny punkt widzenia dotyczący produktu z danymi. Jeśli chcesz od razu się dowiedzieć, jak to działa, zachęcam Cię, żebyś przed kontynuowaniem lektury przeczytał w rozdziale 9. podrozdział „Model metadanych przedsiębiorstwa".

W przypadku zdefiniowania produktów z danymi jako jednostek logicznych masz możliwość opisywania, klasyfikowania, etykietowania lub łączenia danych (np. z domenami, właścicielami danych, metadanymi organizacyjnymi, informacjami o procesach oraz innymi metadanymi) bez zagłębiania się w szczegóły implementacji. Jeden poziom niżej produktu z danymi znajdują się elementy danych, czyli atomowe jednostki informacji odgrywające rolę punktów łączących z danymi fizycznymi, metadanymi interfejsu, metadanymi aplikacji oraz metadanymi modelowania danych. W celu opisania danych fizycznych niezbędne są techniczne metadane uwzględniające informacje o schemacie, typy danych itp.

Gdy już jest dla Ciebie jasne, czym są produkty z danymi, zajmijmy się przeanalizowaniem tego, co wpływa na ich projekt i architekturę. Zaczniemy od wzorca CQRS, a następnie przyjrzymy się modelowaniu optymalizowanemu pod kątem odczytu oraz innym zasadom projektowym. Po tym wszystkim przeniesiemy się na trochę wyższy poziom, aby omówić architekturę produktów z danymi.

Wzorce projektowe produktów z danymi

Zmiana definicji produktu z danymi nie oznacza, że powinno się porzucić pojęcie zarządzania danymi jako produktem. Właściciele aplikacji oraz zajmujące się nimi zespoły muszą traktować dane jako odrębne produkty, za które odpowiadają, a nie jako produkt uboczny jakiegoś procesu zarządzanego przez innych. Produkty z danymi tworzy się specjalnie z myślą o konsumentach danych. Mają one określone postaci, interfejsy oraz cykle utrzymywania i odświeżania. Wszystko to jest udokumentowane. Produkty z danymi zawierają przetworzone dane domeny współużytkowane z niżej umiejscowionymi procesami za pośrednictwem interfejsów wchodzących w skład celu SLO (*Service Level Objective*). Jeśli nie istnieją inne wymagania, zanim dane Twojej aplikacji (technicznej) zostaną udostępnione konsumentom, powinny być przetwarzane, dopasowywane, oczyszczane, agregowane i poddawane normalizacji (denormalizacji), by spełniały ustalone standardy jakości.

W rozdziale 1. zaznajomiłeś się z problemami inżynierskimi związanymi z ograniczaniem transferów danych, projektowaniem systemów transakcyjnych oraz zajmowaniem się znacznie zwiększonym wykorzystaniem danych. Usuwanie dużych wolumenów danych z intensywnie używanego systemu operacyjnego może być ryzykowne, gdyż zbyt obciążone systemy mogą przestać działać albo mogą stać się nieprzewidywalne, niedostępne lub, co gorsza, mogą ulec uszkodzeniu. Z tego

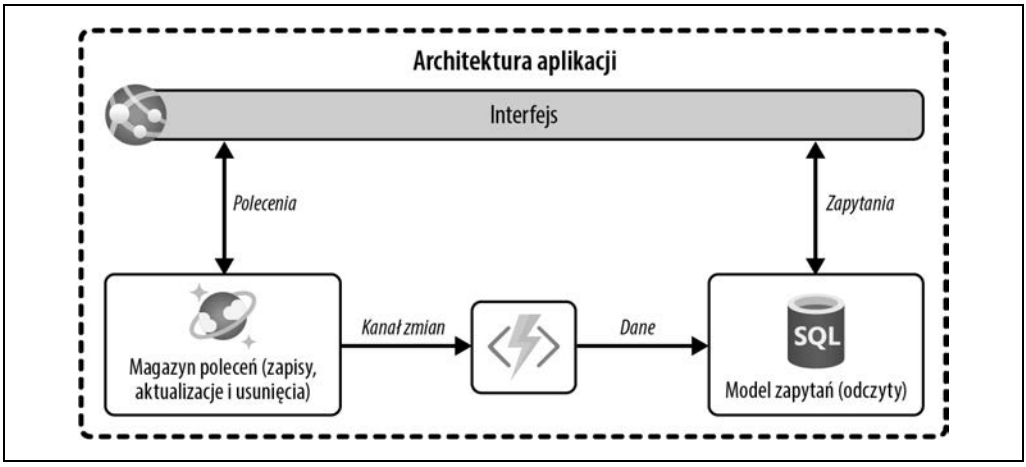
właśnie powodu mądrą decyzją jest przygotowanie wersji danych tylko do odczytu lub zoptymalizowanych pod kątem odczytu, która może następnie zostać udostępniona konsumentom. Przyjrzyjmy się odpowiedniemu w tym przypadku powszechnemu wzorcowi projektowemu CQRS.

Czym jest wzorzec CQRS?

CQRS (*https://learn.microsoft.com/en-us/azure/architecture/patterns/cqrs*) to wzorzec projektowy aplikacji oparty na tworzeniu kopii danych, które mają być intensywnie odczytywane.

Polecenia operacyjne i zapytania analityczne (często określane mianem *zapisów* i *odczytów*) to bardzo odmienne operacje, które we wzorcu CQRS są traktowane jako osobne (jest to pokazane na rysunku 4.2). W trakcie analizowania obciążenia zajętego systemu prawdopodobnie stwierdzisz, że po stronie poleceń wykorzystywana jest większość zasobów obliczeniowych. Jest to logiczne, ponieważ w celu pomyślnego (czyli z trwałym efektem) zrealizowania operacji zapisu, aktualizacji lub usunięcia baza danych musi zwykle wykonać serię następujących kroków:

1. Sprawdzenie ilości dostępnej przestrzeni magazynowej.
2. Przydzielenie dodatkowej przestrzeni magazynowej na potrzeby zapisu.
3. Pobranie metadanych kolumny lub tabeli (typy, ograniczenia, wartości domyślne itp.).
4. Znalezienie rekordów (w przypadku aktualizacji lub usuwania).
5. Zablokowanie tabeli lub rekordów.
6. Zapisanie nowych rekordów.
7. Zweryfikowanie wstawionych wartości (np. pod względem unikalności, poprawności typów itp.).
8. Wykonanie operacji zatwierdzenia.
9. Zwolnienie blokady.
10. Zaktualizowanie indeksów.



Rysunek 4.2. Aplikacja korzystająca z wzorca CQRS oddziela zapytania i polecenia za pomocą dwóch różnych modeli danych: modelu poleceń przeznaczonego do transakcji oraz modelu zapytań przewidzianego do odczytów

W porównaniu z zapisem operacja odczytywania z bazy danych wymaga mniejszej ilości zasobów obliczeniowych, gdyż musi zostać zrealizowana mniejsza liczba wyżej wymienionych zadań. W celu zapewnienia optymalizacji wzorzec CQRS oddziela zapisy (polecenia) od odczytów (zapytania) za pomocą dwóch modeli, co zostało zilustrowane na rysunku 4.2. Po rozdzieleniu operacje te muszą być cały czas synchronizowane, co zwykle odbywa się przez publikowanie zdarzeń ze zmianami. Na poniższym rysunku zilustrowano to z użyciem „strzałki zdarzeń" (ikona błyskawicy).

Zaletą wzorca CQRS jest to, że nie jesteś ograniczony do tego samego typu bazy danych w przypadku operacji zapisu i odczytu[1]. Możesz pozostawić bazę danych zapisów obiektywnie złożoną, a zoptymalizować bazę danych odczytów pod kątem wydajności operacji odczytu. Jeśli masz inne wymagania co do różnych zastosowań, możesz nawet utworzyć więcej niż jedną bazę danych odczytów, z których każda będzie dysponować zoptymalizowanym modelem odczytu na potrzeby konkretnego, implementowanego wariantu zastosowania. Różne wymagania umożliwiają również zastosowanie odmiennych modeli spójności między magazynami odczytów, nawet pomimo tego, że magazyn zapisywanych danych pozostaje ten sam.

Kolejną zaletą wzorca CQRS jest to, że nie ma konieczności jednoczesnego skalowania zarówno operacji odczytu, jak i zapisu. Gdy braknie zasobów, możesz dokonać skalowania jednego lub drugiego typu operacji. Elastyczność technologii to ostatnia znacząca korzyść wynikająca z rezygnacji z pojedynczego modelu obsługującego operacje zapisu i odczytu. Jeśli wymagasz, aby odczyt był wykonywany bardzo szybko, albo potrzebujesz różnych struktur danych, w przypadku magazynu odczytów możesz zdecydować się na inną technologię przy dalszym respektowaniu właściwości ACID (*Atomicity, Consistency, Isolation, and Durability*) bazy danych niezbędnych w odniesieniu do magazynu poleceń.

Wzorzec CQRS jest silnie powiązany z **widokami zmaterializowanymi**[2]. Tego typu widok w obrębie bazy danych jest fizyczną kopią danych, która jest nieustannie aktualizowana przez bazowe tabele. Widoków zmaterializowanych często używa się na potrzeby optymalizacji wydajności. Zamiast uzyskiwać dane z bazowych tabel, zapytanie wykonuje się względem wcześniej obliczonego i zoptymalizowanego podzbioru, który znajduje się wewnątrz widoku zmaterializowanego. Takie rozdzielenie bazowych tabel (używanych do zapisów) oraz podzbioru zmaterializowanego (stosowanego do odczytów) jest tym samym, które ma miejsce we wzorcu CQRS, z tą subtelną różnicą, że obie kolekcje danych znajdują się w tej samej bazie danych, a nie w dwóch różnych.

Choć CQRS to wzorzec projektowy związany z inżynierią oprogramowania, który może pomóc ulepszyć proces projektowania w przypadku specyficznych (i być może większych) systemów, powinien w znacznym stopniu stanowić inspirację dla projektu i umożliwić wyobrażenie sobie swojej architektury produktów z danymi. *Przyszły model architektury musi być taki, że przynajmniej*

[1] Mankamentem wzorca CQRS jest to, że sprawia on, że wszystko jest bardziej złożone. Trzeba będzie utrzymywać synchronizację dwóch modeli z dodatkową warstwą. Może to też spowodować dodatkowe opóźnienie.

[2] W serwisie TechDifferences (*https://techdifferences.com/difference-between-view-and-materialized-view.html*) dokonano przeglądu różnic między widokiem bazy danych a widokiem zmaterializowanym.

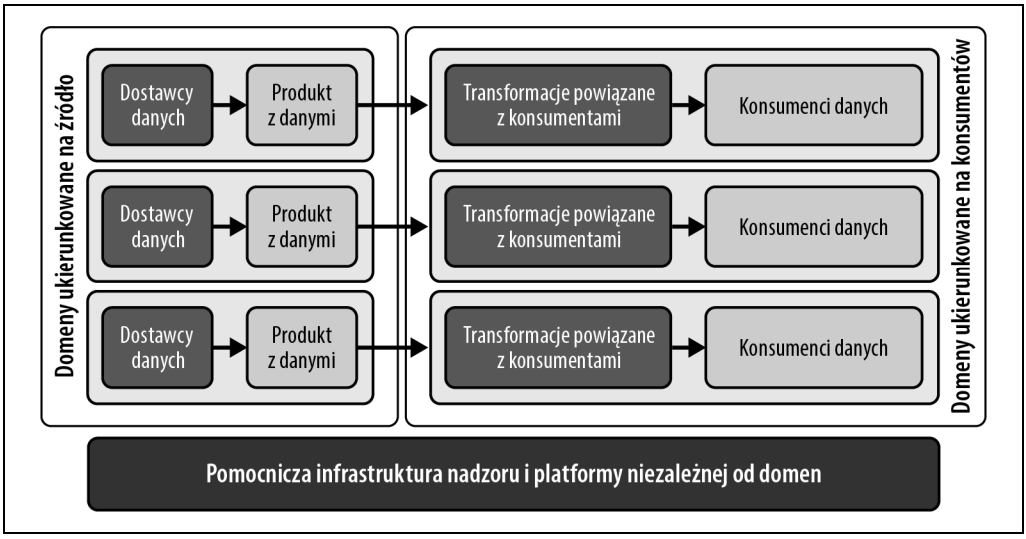
jeden magazyn danych do odczytu *dla każdej aplikacji jest tworzony lub używany każdorazowo, gdy inne aplikacje zamierzają intensywnie wczytywać dane*. Takie rozwiązanie uprości przyszły projekt i implementację architektury, a także zwiększy możliwości skalowania pod kątem intensywnego korzystania z danych.

Repliki do odczytu jako produkty z danymi

Zastosowanie replik jako źródła odczytu danych nie jest czymś nowym. Okazuje się, że każdy odczyt z repliki, kopii, hurtowni danych lub „jeziora" danych można postrzegać jako pewną formę wzorca CQRS. Rozdzielanie poleceń i zapytań między systemami OLTP (*Online Transaction Processing*), które zwykle ułatwiają działanie aplikacji ukierunkowanych na transakcje oraz zarządzają nimi, a operacyjnym magazynem danych (używanym na potrzeby raportowania operacyjnego) przebiega podobnie. W tym przykładzie operacyjny magazyn danych jest replikowaną wersją danych z systemu OLTP. Wszystkie te wzorce są zgodne z filozofią towarzyszącą wzorcowi CQRS, która polega na tworzeniu baz danych do odczytu z wykorzystaniem operacyjnych baz danych.

Martin Kleppmann posługuje się wzorcem „odwracania bazy danych" (*https://martin.kleppmann.com/2015/11/05/database-inside-out-at-oredev.html*), będącym kolejną inkarnacją wzorca CQRS, która kładzie nacisk na gromadzenie faktów za pośrednictwem strumieniowania zdarzeń. Przyjrzymy się temu wzorcowi w rozdziale 6.

Architektury produktów z danymi również powinny być zgodne z filozofią wzorca CQRS. Pełnią one funkcję replikowanej kopii danych i mają na celu umożliwienie konsumentom danych intensywne wykonywanie operacji odczytu. Architektury te mogą być w dużym stopniu nadzorowane oraz dziedziczą swój kontekst z domen i bazowych aplikacji. Na ogólnym poziomie oznacza to, że każdorazowo, gdy dostawcy danych i ich konsumenci zamierzają wymienić dane (tak jak na rysunku 4.3), musi zostać zastosowana architektura produktów z danymi.



Rysunek 4.3. Zdecentralizowana współpraca dostawców danych i ich konsumentów

Zauważ, że produkty z danymi umiejscowiono po lewej stronie w pobliżu dostawców danych, a kroki transformacji znajdują się obok konsumentów danych. Takie rozmieszczenie wynika pierwotnie z jednolitej metody udostępniania danych konsumentom. Zamiast używać pojedynczego, jednolitego modelu danych, projekt zmodyfikowano w celu zapewnienia wszystkim aplikacjom będącym konsumentami oczyszczonych wersji danych domenowych, które mogą być łatwo wykorzystane. Dane te nie mają na celu dostarczania zachowania ani funkcjonalności. W porównaniu z danymi operacyjnymi charakter tych danych jest zatem inny: zoptymalizowano je pod kątem możliwości intensywnych odczytów, dzięki czemu każdy może szybko przekształcić dane w wartość!

Zasady projektowe w przypadku produktów z danymi

Dobrym rozwiązaniem jest tworzenie danych przyjaznych dla użytkownika i zoptymalizowanych pod kątem odczytu. Brzmi to prosto, ale w praktyce często jest trudniejsze, niż można by się spodziewać. Dokładniej rzecz ujmując, musisz się zastanowić, jak dane powinny być przechwytywane, strukturyzowane i modelowane. Trudności pojawiają się w momencie, gdy musisz tworzyć produkty z danymi w powtarzalny sposób i równolegle z wieloma zespołami. Zależy Ci na wykonaniu tego w efektywny sposób z uniknięciem sytuacji, w których każdy nowy konsument danych powoduje pojawienie się nowo przygotowanego produktu z danymi lub długiego cyklu obejmującego analizowanie złożonych struktur danych i rozwiązywanie problemów z jakością danych. Chcesz zmaksymalizować możliwość ponownego wykorzystania danych oraz łatwość pracy z nimi! W tym podrozdziale zaprezentowano zestaw zasad projektowych, których uwzględnienie będzie pomocne w trakcie projektowania produktów z danymi. Bądź świadom tego, że jest to długa lista.

Projekt optymalizowany pod kątem odczytu i ukierunkowany na zasoby

Modele analityczne, które ciągle podlegają przekwalifikowywaniu, nieustannie wczytują duże woluminy danych. Wpływa to na projekty produktów z danymi, ponieważ trzeba optymalizować pod kątem możliwości odczytu danych. Najlepszą praktyką jest przyjrzenie się rozmieszczeniu zasobów w celu zaprojektowania interfejsów API. Interfejs API ukierunkowany na zasoby jest przeważnie modelowany jako hierarchia zasobów, gdzie każdy węzeł jest albo zwykłym zasobem, albo zasobem kolekcji. Dla ułatwienia często określa się je mianem, odpowiednio, zasobów i kolekcji.

W przypadku produktów z danymi możesz skorzystać z tej samej metody ukierunkowanej na zasoby polegającej na logicznym grupowaniu danych i tworzeniu ich klastrów na podstawie obszarów zagadnień, gdzie każdy zbiór danych reprezentuje kolekcję jednorodnych danych. Taki projekt skutkuje tym, że dane w produktach są wstępnie przetwarzane, poddawane denormalizacji i zmaterializowane. Po stronie konsumentów prowadzi to do prostszego i szybszego wykorzystania danych przez niżej umieszczone elementy, ponieważ nie ma potrzeby stosowania złączeń, które są kosztowne z punktu widzenia liczby obliczeń. Poza tym skraca się czas, jaki zajmuje znalezienie właściwych danych, gdyż dane przynależące do siebie są logicznie grupowane.

Gdy względem swoich produktów z danymi zastosuje się projekt ukierunkowany na zasoby, w konsekwencji modele danych fizycznych cechujące się wysokim stopniem normalizacji lub zbyt dużą złożonością techniczną muszą zostać przekształcone do postaci logicznie uporządkowanych zbiorów

danych o większych możliwościach ponownego użycia. Wynikiem są modele danych zoptymalizowane pod kątem odczytu o wyższym poziomie denormalizacji, które przypominają schemat gwiazdy Kimballa lub składnicę danych Inmona. Oznacza to również, że abstrakcji musi zostać poddana złożona logika aplikacji. Aby obsłużyć jak największą możliwą liczbę konsumentów, dane trzeba udostępnić innym domenom na odpowiednim poziomie szczegółowości. W przypadku dowolnych połączonych produktów z danymi wskazuje to, że odnośniki i relacje z kluczem obcym muszą być spójne w ramach całego zestawu produktów z danymi. Przykładowo domeny będące konsumentami nie powinny być zmuszone do modyfikowania kluczy w celu połączenia różnych zestawów danych! W konsekwencji tworzenie produktów z danymi z zastosowaniem takiego podejścia oznacza, że domeny dysponują własną logiką semantyczną i odpowiadają za sposób transformowania danych w celu zwiększenia możliwości odczytu. Na razie jednak zostawmy to i przyjrzyjmy się innym najlepszym praktykom i zasadom projektowym.

Produkty z danymi i „skarbce" danych

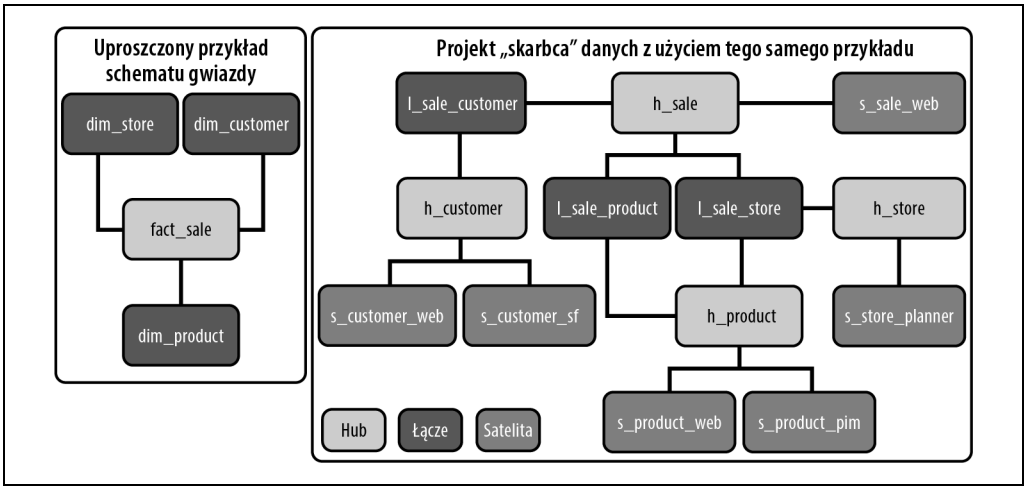
Możesz się zastanawiać, czy masz możliwość projektowania swoich produktów z danymi za pomocą modelu „skarbca" danych (ang. *data vault*). „Skarbiec" danych to wzorzec projektowy związany z modelowaniem danych, który służy do budowania hurtowni danych[3]. Ma on na celu zapewnienie elastyczności, możliwości utrzymania i skalowalności przez eliminowanie sprzężenia między jednostkami z dodatkowymi relacjami. Ze względu na swoją znaczną złożoność „skarbiec" danych sprawia, że dane są trudne do zrozumienia przez „osoby z zewnątrz", które nie zostały przeszkolone w zakresie korzystania z tej techniki. Ponadto sprawia on, że Twoja architektura jest wolniejsza i kosztowniejsza, ponieważ metodyka szczegółowego modelowania danych często kłóci się z bazowymi architekturami rozproszonego magazynu oferowanymi przez dużych dostawców technologii chmury. Metodyka ta wymaga, aby wielu konsumentów przeprowadzało iteracje łączenia i integrowania danych.

Na rysunku 4.4 porównano schemat „skarbca" danych z prostym schematem gwiazdy. Mówiąc wprost, metodyka „skarbca" danych zastępuje jednostki elementami centralnymi (hubami), łączami i satelitami, które umożliwiają spełnianie nieustannie rozwijających się wymagań biznesowych.

Dane produktu są trwałe

Dane produktu są *trwałe*, a zatem *tylko do odczytu*. Dlaczego Twoje magazyny danych tylko do odczytu muszą być trwałe? Właściwość ta gwarantuje, że można bez końca ponownie generować identyczne dane. Jeśli zastosujesz projekt tylko do odczytu, dla tych samych danych domenowych nie będą istnieć żadne inne wersje „prawdy". Architektury produktów z danymi zgodne z tą zasadą nie tworzą nowych danych semantycznych. „Prawda" może być modyfikowana wyłącznie w aplikacji „złotego" źródła.

[3] W artykule Morrisa poświęconym wzorcowi projektowemu „skarbca" danych w wersji 2.0 (*https://www.ben-morris.com/data-vault-2-modelling-the-good-the-bad-and-the-downright-confusing/*) opisano go jako „dobry i zły, a także zdecydowanie niejasny".



Rysunek 4.4. Metodyka „skarbca" danych porównana z prostym schematem gwiazdy (źródło: PhData; https://www.phdata.io/blog/building-modern-data-platform-with-data-vault/)

Zastosowanie języka wszechobecnego

Kluczowe jest zrozumienie kontekstu tego, jak utworzono dane. Przydaje się tutaj język wszechobecny, czyli skonstruowany i sformalizowany język uzgodniony przez udziałowców i projektantów, który ma spełniać wymagania projektu. Język wszechobecny powinien być dopasowany do domeny, czyli do funkcji i celów biznesowych. Kontekst domeny określa projekt produktów z danymi. Typową implementacją jest dysponowanie katalogiem danych, w którym są przechowywane wszystkie informacje wraz z informacjami z domen. Publikowanie kontekstu oraz zapewnianie przejrzystości definicji w domenach w transparentny sposób pozwala każdemu na zaznajomienie się ze źródłem i znaczeniem danych, a także ich zaakceptowanie.

Niektóre firmy wymagają, aby w ramach domen w ich fizycznych zbiorach danych produktów z danymi były używane nazwy kolumn przyjazne dla użytkownika. Nie jest to kluczowe pod warunkiem, że zapewniono odwzorowanie fizycznego modelu danych na koncepcyjny model danych. Takie powiązanie umożliwia dokonanie przez Twoje domeny translacji zagadnień biznesowych z języka wszechobecnego do postaci danych fizycznych.

Przechwytywanie bezpośrednio ze źródła

W ramach poprzedniej pracy dowiedziałem się, że przedsiębiorstwa często korzystają z długich łańcuchów danych, w których dane są przekazywane od systemu do systemu. Z punktu widzenia zarządzania danymi może to być problem, ponieważ ich pochodzenie nie jest oczywiste. Powinno się zatem zawsze przechwytywać dane z systemu źródłowego. Użycie unikalnych danych ze „złotych" źródeł gwarantuje, że istnieje jedno „źródło prawdy" zarówno w przypadku dostępu do danych, jak i zarządzania nimi. Oznacza to, że produkty z danymi są wyraźne i jednoznaczne: należą one do pojedynczej domeny, czyli są izolowane i nie mogą mieć bezpośrednich zależności z aplikacjami

z innych domen. Taka zasada projektowa oznacza również, że domeny nie mogą hermetyzować danych z innych domen z różnymi właścicielami danych, ponieważ zaciemniłoby to kwestię własności danych.

Przejrzyste standardy współdziałania

Ważne jest określenie tego, jak produkty z danymi są udostępniane innym domenom. Produkt z danymi może być zapewniany jako baza danych, standaryzowany format pliku, graf, punkt końcowy interfejsu API, strumień zdarzeń lub jako coś innego. We wszystkich przypadkach powinno się standaryzować specyfikacje interfejsów. Jeśli wszystkie Twoje domeny ustalają własny zestaw standardów, korzystanie z danych staje się wyjątkowo trudne. A zatem standaryzacja ma kluczowe znaczenie. Wymaga to jasnego określenia różnych typów dystrybucji (np. ukierunkowanych na dane wsadowe, interfejsy API i zdarzenia), standardów metadanych itp. Bądź precyzyjny w kwestii tego, jak wygląda architektura produktów z danymi. Przykładowo produkty z danymi wsadowymi są publikowane w formacie Delta, a ponadto muszą być rejestrowane w katalogu danych.

Żadnych nieprzetworzonych danych

Produkt z danymi jest przeciwieństwem danych nieprzetworzonych, ponieważ ujawnienie ich wymaga ponowienia działań przez wszystkie domeny będące konsumentami. A zatem w każdym przypadku powinno się hermetyzować starsze lub złożone systemy oraz ukrywać dane techniczne aplikacji. Jeżeli zależy Ci na udostępnieniu danych nieprzetworzonych (np. na potrzeby analizy wyników badań), zadbaj o to, aby te dane były tymczasowe i oznaczone. Danym nieprzetworzonym lub niezmodyfikowanym nie towarzyszą żadne gwarancje.

Czy systemy mogą tworzyć interfejsy ujawniające dane nieprzetworzone?

Rozważ umożliwienie aplikacjom zapewniania jednocześnie danych nieprzetworzonych i danych zoptymalizowanych pod kątem konsumentów. Z takiego podejścia korzystają danolodzy i konsumenci, którzy oczekują natychmiastowego dostępu. Interfejsy przechowujące dane nieprzetworzone powinny być oznaczane jako prywatne i nie powinny zapewniać żadnych gwarancji. Nigdy nie powinny one być używane jako produkcyjne potoki danych.

Omówiona w rozdziale 2. zasada dotycząca rezygnacji z danych nieprzetworzonych dotyczy również zewnętrznych dostawców danych, którzy działają poza logicznymi granicami ekosystemu Twojego przedsiębiorstwa. Tacy dostawcy funkcjonują zwykle w osobnych, pozbawionych kontroli lokalizacjach sieciowych. Poproś partnerów znajdujących się w Twoim ekosystemie o przestrzeganie Twojego standardu lub zastosuj mediację z wykorzystaniem domeny pośredniczącej, czyli domeny będącej konsumentem, która odgrywa rolę pośrednika dokonującego abstrakcji złożoności oraz zapewniającego stabilne i bezpieczne korzystanie z danych.

Niedostosowywanie się do konsumentów

Prawdopodobne jest to, że te same produkty z danymi będą wielokrotnie używane przez różne zespoły domenowe w przypadku szerokiej gamy zastosowań. A zatem Twoje zespoły nie powinny dostosowywać swoich produktów z danymi do konkretnych wymagań (poszczególnych) konsumentów danych. Podstawową zasadą projektową powinno być maksymalizowanie wydajności domen oraz promowanie wykorzystania i ponownego używania danych.

Z kolei produkty z danymi mogą i rozwijają się w zależności od informacji zwrotnych użytkowników. Powinno się tutaj zachować ostrożność! Zespoły mogą stać przed pokusą, by uwzględniać określone wymagania konsumentów. Jeśli jednak konsumenci umieszczą logikę biznesową w produktach z danymi domeny produkcyjnej, spowoduje to utworzenie ścisłej zależności z dostawcą w zakresie wprowadzania zmian. Może to wywołać intensywną, wzajemną koordynację oraz negocjowanie zakresu prac do zrealizowania przez zespoły domenowe. W tym przypadku zaleca się wprowadzenie jednostki nadzorującej, która będzie pilnować tego, czy produkty z danymi nie są tworzone dla konkretnych konsumentów. Taka jednostka może się sprawdzić, gdy trzeba pokierować pracami zespołów domenowych, zorganizować sesje dzielenia się wiedzą, zapewnić praktyczne opinie oraz rozwiązać problemy zaistniałe między zespołami.

Zagrożenia związane z zewnętrzną obsługą logiki biznesowej

Pozwól, że podzielę się z Tobą osobistą historią. W mojej poprzedniej pracy szef działu księgowości zaproponował ustanowienie reguł księgowości zgodnych ze standardami IFRS (*International Financial Reporting Standards*) integralną częścią wszystkich produktów z danymi. Założył on, że sumowanie obliczonych wyników jest łatwiejsze niż obliczanie wszystkiego samemu. Istnieje kilka dobrych powodów, dla których nigdy nie powinno się przekazywać swojej logiki biznesowej innym domenom. Jednym z nich jest to, że w innych domenach dostępna jest dogłębna wiedza na temat Twojej domeny. Wymagane jest również, aby wszystkie domeny jednocześnie przeprowadzały zmiany, gdy trzeba zaktualizować logikę biznesową. Jak się z tym uporasz w środowisku z dziesiątkami lub setkami domen? Prawdopodobnie niezbędna będzie armia konsultantów odpowiedzialnych za rozplanowanie wszystkich tych działań i zarządzanie nimi. Właściwe podejście polega na utrzymywaniu logiki biznesowej w pobliżu miejsca, do którego ona należy. Z punktu widzenia standardów IFRS oznacza to zażądanie od innych domen dostarczenia do działu księgowości informacji, jakie są niezbędne do poprawnego przeprowadzenia obliczeń. Informacje te powinny zachować oryginalny kontekst domeny udostępniającej. Dzięki temu inne domeny również mogą skorzystać z danych.

Brakujące wartości, wartości domyślne i typy danych

Byłem świadkiem gorących debat o tym, jak powinno się interpretować brakujące dane, a także dane domyślne lub niskiej jakości. Dla przykładu załóżmy, że formularz internetowy systemu operacyjnego wymaga podania wartości reprezentującej datę urodzenia użytkownika, ale użytkownik nie zapewnił tych danych. Jeśli nie udostępniono żadnych wytycznych, pracownicy mogą wprowadzić losową wartość. Takie zachowanie może utrudnić konsumentom stwierdzenie, czy dane są dokładne albo czy zawierają wartości domyślne. Określenie wyraźnych wytycznych powodujących na przykład,

że w przypadku braku dat urodzenia zawsze domyślnie zostaną użyte absurdalne wartości lub przyszłe daty, takie jak 9999-12-31, pozwoli klientom stwierdzić, czy faktycznie brakuje danych.

Może być też wskazane wprowadzenie wytycznych dotyczących typów danych lub danych, które muszą być spójnie formatowane w obrębie całego zbioru danych. Określenie na przykład poprawnej liczby cyfr po przecinku zapewnia, że konsumenci danych nie muszą w celu ich użycia stosować złożonej logiki aplikacji. Mógłbyś uwzględniać takie zasady na poziomie systemu lub domeny, ale spotykam się również z tym, że duże przedsiębiorstwa zapewniają ogólne wytyczne co do tego, jakie formaty i typy danych należy stosować w ramach wszystkich produktów z danymi.

Spójność semantyczna

Produkty z danymi muszą być spójne semantycznie w przypadku wszystkich metod dostarczania, czyli wsadowych, zależnych od zdarzeń i opartych na interfejsie API. Jest to dla mnie oczywiste, ale w dalszym ciągu widzę, że firmy zapewniają osobne wytyczne co do danych ukierunkowanych na metodę wsadową, interfejsy API i zdarzenia. Ponieważ źródło danych jest takie samo dla wszystkich wzorców dystrybucji, zachęcam do tworzenia wytycznych spójnych dla nich wszystkich. Powrócę do tej kwestii w rozdziale 7.

Atomowość

Atrybuty produktów z danymi muszą być atomowe. Reprezentują one najniższy poziom szczegółowości oraz mają dokładne znaczenie lub semantykę. Oznacza to, że ich dekompozycja do postaci innych, konkretnych komponentów nie powinna być możliwa. W stanie idealnym atrybuty danych są powiązane w ramach relacji „jeden do jednego" z elementami biznesowymi w obrębie Twojego katalogu danych. Korzyścią jest tutaj to, że konsumenci danych nie są zmuszeni do dzielenia lub łączenia danych[4]. Kolejną korzyścią wynikającą z używania danych atomowych jest to, że możliwe jest usunięcie sprzężenia danych z jakimikolwiek regułami. Jeśli na przykład przepisy zmuszają do ponownego rozważenia etykiet poufności, nie musisz jeszcze raz używać ich względem wszystkich swoich danych fizycznych. Jeśli dysponujesz dobrym modelem metadanych i danymi atomowymi, wszelkie zmiany powinny być automatycznie dziedziczone z Twoich metadanych biznesowych.

Zgodność

Produkty z danymi powinny pozostać stabilne i oddzielone od aplikacji transakcyjnych lub operacyjnych. Nakazuje to zastosowanie mechanizmu wykrywania dryfu schematu oraz unikanie zakłócających zmian. Wymaga to również wersjonowania, a w niektórych sytuacjach — równoległego działania niezależnych potoków, dzięki którym konsumenci Twoich danych mają czas na dokonanie migracji z jednej wersji do drugiej.

Proces utrzymywania zgodności produktów z danymi nie jest tak prosty, na jaki może wyglądać. Może on obejmować przenoszenie danych historycznych między starymi a nowymi produktami.

[4] Wierz lub nie, ale spotkałem się raz ze starszą aplikacją, w której wiele wartości klientów połączono w ramach pojedynczego pola za pomocą operatorów potoku (np. Imię||Drugie imię||Nazwisko).

Takie działanie może być złożonym przedsięwzięciem, gdyż uwzględnia takie zadania procesu ETL jak odwzorowywanie danych domyślnych, oczyszczanie ich i zapewnianie logiki ich określania.

Uogólnianie zmiennych danych referencyjnych

Może być wskazane zapewnienie wytycznych dotyczących tego, jak złożone wartości referencyjne są odwzorowywane na bardziej abstrakcyjne wartości referencyjne dostosowane do produktów z danymi. Wymaga to uważnego podejścia do niezgodności w metodyce Agile: jeśli tempo zmian po stronie konsumentów jest duże, użyta zasada przewodnia musi być taka, że złożone tabele odwzorowywania są utrzymywane właśnie po tej stronie. W przypadku większego tempa zmian po stronie udostępniającej zasada jest taka, że właściciele produktów z danymi powinni zostać poproszeni o uogólnienie lub przekształcenie szczegółowych, lokalnych wartości referencyjnych do postaci bardziej ogólnych (mniej szczegółowych) wartości referencyjnych niezależnych od konsumenta. Wytyczne te wskazują też, że konsument może realizować dodatkowe działania, takie jak odwzorowywanie bardziej ogólnych wartości referencyjnych na wartości referencyjne powiązane z konsumentem.

Nowe dane oznaczają nową własność

Wszelkie dane tworzone w wyniku transformacji biznesowej (zmiana semantyczna z użyciem logiki biznesowej) i dystrybuowane uważa się za nowe, a ponadto podlegają własności domeny tworzącej. Zasady dystrybucji danych omówione w tym rozdziale powinny być wymuszane dla wszystkich nowo utworzonych danych, które są współużytkowane.

W gruncie rzeczy niepoprawne jest klasyfikowanie jako produktów z danymi (dopasowywanych do konsumentów) integrowanych danych zależnych od wariantu zastosowania. Jeśli zezwolisz na to, aby dane powiązane z wariantem zastosowania były bezpośrednio wykorzystywane przez inne domeny, będą one w dużym stopniu zależne od szczegółów implementacji podstawowego wariantu zastosowania z konsumentem. A zatem zamiast tego zawsze należy tworzyć dodatkową warstwę abstrakcji i oddzielać swój wariant zastosowania od innych konsumentów. Taka metoda przekształcania danych powiązanych z konsumentem w nowy produkt z danymi umożliwia Twoim domenom rozwijanie się niezależnie od innych domen.

Z dystrybuowaniem nowo utworzonych danych wiąże się kwestia możliwości śledzenia, czyli uzyskiwania informacji o tym, co się dzieje z danymi. Aby zminimalizować ryzyka dotyczące transparentności, poproś konsumentów danych o katalogowanie pozyskanych przez siebie danych oraz sekwencji działań i transformacji wykonywanych przez nich względem danych. Takie metadane dotyczące pochodzenia powinny być publikowane centralnie. Powrócę do tego w rozdziałach 10. i 11.

Wzorce bezpieczeństwa danych

Z myślą o bezpieczeństwie danych musisz ustalić wytyczne dotyczące dostawców danych. Powinny to być:

- Wytyczna dotycząca hermetyzowania metadanych na potrzeby filtrowania i dostępu na poziomie wierszy. Dzięki zapewnieniu metadanych razem z danymi możesz tworzyć zasady lub widoki służące do ukrywania lub wyświetlania określonych wierszy danych zależnie od tego, czy konsument ma uprawnienia do ujrzenia tych wierszy.
- Wytyczna co do tagów lub klasyfikacji w przypadku dostępu na poziomie kolumn oraz maskowania danych dynamicznych. Tagi lub klasyfikacje są używane jako dane wejściowe zasad lub widoków.
- Wytyczna dotycząca efektywnego przechowywania danych w osobnych tabelach w celu zapewnienia ogólnego bezpieczeństwa, zwiększenia wydajności i ułatwienia utrzymywania.

Więcej na temat tych zagadnień jest mowa w rozdziale 8.

Ustanowienie metamodelu

Z myślą o właściwym zarządzaniu produktami z danymi oraz ich odpowiednimi metadanymi szczególnie zachęcam do utworzenia spójnego metamodelu, w którym określasz, jak jednostki (np. domeny danych, produkty z danymi, właściciele danych, pojęcia biznesowe, dane fizyczne oraz inne metadane) są ze sobą wzajemnie powiązane[5]. Korzystając ze swojego metamodelu, do wymuszenia przechwytywania wszystkich niezbędnych metadanych najlepiej użyć katalogu danych. Przykładowo każdorazowo podczas tworzenia w Twoim katalogu instancji nowego produktu z danymi może to zainicjować przepływ, który na początku prosi dostawcę danych o dostarczenie opisu, a także połączenie produktu z danymi z właścicielem danych, aplikacją źródłową, domeną danych itd. W dalszej kolejności katalog może poprosić dostawcę o połączenie wszystkich elementów produktów z danymi z pojęciami biznesowymi oraz atrybutami danych technicznych (fizycznych). Katalog może następnie uruchomić zautomatyzowane testy w celu sprawdzenia, czy lokalizacje fizyczne mogą być adresowane, a także poproszenia o klasyfikacje i informacje dotyczące ograniczeń dotyczących użytkowania.

Umożliwienie samoobsługi

Produkty z danymi mają na celu demokratyzację danych. Aby usprawnić tworzenie tych produktów, rozważ zbudowanie platformy danych i zaoferowanie możliwości samoobsługi w zakresie wykrywalności, zarządzania, współużytkowania i obserwowalności. Przykładowo zezwól inżynierom na użycie interfejsu REST katalogu danych, aby można było zautomatyzować rejestrowanie produktów z danymi. Jeśli zostanie to poprawnie przeprowadzone, inżynierowie uzyskają możliwość rejestrowania swoich produktów z danymi jako części swoich potoków procesu integracji ciągłej i dostarczania ciągłego. Więcej wytycznych związanych z tym zagadnieniem zamieszczono w rozdziale 9.

[5] Metamodel to struktura umożliwiająca zrozumienie, jakie metadane muszą zostać przechwycone podczas opisywania danych.

Wzajemne relacje między domenami

W świecie operacji systemy są często powiązane ze sobą lub silnie połączone. Wyobraź sobie system zamówień, za pośrednictwem którego klienci mogą składać zamówienia. Prawdopodobnie taki system będzie przechowywać dane z bazy danych klientów. W przeciwnym razie nie byłby w stanie powiązać zamówień z klientami.

Jeżeli w domenach produkty z danymi są przygotowywane niezależnie od innych domen, w jaki sposób konsumenci stwierdzą, że określone dane przynależą do siebie lub są ze sobą powiązane? W celu zarządzania tymi zależnościami weź pod uwagę zastosowanie katalogu lub grafu wiedzy, który opisuje relacje między zbiorami danych, a ponadto odwołuje się do kluczy złączeń lub kluczy obcych między produktami z danymi.

W zapewnianiu wytycznych dotyczących wzajemnych relacji między domenami ważne jest to, że dane zawsze pozostają ukierunkowane na domeny. Dopasowanie między danymi a ich własnością musi być silne, dlatego przed dostarczeniem jakichkolwiek danych do innych domen nie powinna mieć miejsca żadna integracja między domenami.

Spójność w przedsiębiorstwie

Silna decentralizacja powoduje tworzenie silosów danych w obrębie poszczególnych domen. W efekcie tego pojawiają się kwestie dostępności i użyteczności danych, ponieważ gdy wszystkie domeny zaczynają osobno udostępniać dane, trudniejsze staje się zintegrowanie i połączenie danych po stronie konsumentów. Dlatego może być wskazane wprowadzenie w obrębie przedsiębiorstwa pewnej spójności przez zapewnienie wytycznych dotyczących uwzględniania wartości referencyjnych (kody walut, kody państw, kody produktów, kody segmentacji klientów itp.). W razie potrzeby możesz poprosić właścicieli Twoich produktów z danymi o odwzorowanie ich lokalnych wartości referencyjnych na wartości z list przedsiębiorstwa.

Niektóre osoby zajmujące się danymi będą twierdzić, że zastosowanie przez przedsiębiorstwo wartości referencyjnych nie jest zgodne z prawdziwą implementacją siatki danych. W pewnym stopniu jest to prawda, ale bez jakiejkolwiek integralności referencyjnej będziesz mieć do czynienia z powielonymi działaniami związanymi z harmonizacją i integracją obejmującymi wszystkie domeny. Zauważ, że nie sugeruję zbudowania kanonicznego modelu danych w skali całego przedsiębiorstwa, z czym można się spotkać w przypadku architektur hurtowni danych większości przedsiębiorstw, lecz jedynie zapewnienie pewnych danych referencyjnych jako wytycznej. Uzyskana spójność może być pomocna w organizacji o dużej skali, w której wiele domen jest zależnych od tych samych wartości referencyjnych.

Te same wytyczne dotyczą podstawowych numerów identyfikacyjnych, które łączą ze sobą dane podstawowe z danymi z systemów lokalnych. Te elementy danych są kluczowe w określaniu, jakie dane stały się podstawowymi i co przynależy do czego. W związku z tym możesz skierować do swoich domen lokalnych prośbę o uwzględnienie tych podstawowych identyfikatorów w ich produktach z danymi. Więcej najlepszych praktyk powiązanych z tym obszarem omówiono w rozdziale 10.

Historyzacja, ponowne dostarczenia i nadpisania

W celu spełnienia wymagań konsumentów dostawcy danych często muszą utrzymywać dane historyczne w ich pierwotnym kontekście. Przykładowo konsumenci mogą wymagać tych danych do przeprowadzania retrospektywnych analiz trendu i prognozowania tego, co się stanie w dalszej kolejności. Aby zrobić to dobrze, warto sformułować wytyczne na podstawie projektu systemu i typu danych, a także określić, jak powinien być realizowany proces historyzacji na potrzeby późniejszej analizy. Przykładowo zbiory danych z danymi podstawowymi zawsze powinny być przekształcane do postaci wolno zmieniających się wymiarów. Przetwarzaniu danych historycznych bardziej szczegółowo przyjrzymy się dalej w tym rozdziale, w punkcie „Historyzacja danych".

Możliwości biznesowe z wieloma właścicielami

Gdy możliwości biznesowe są wspólnie używane, zalecam ustalenie metodyki do obsługi danych współużytkowanych. Może to obejmować zastosowanie w obrębie Twoich produktów z danymi nazw zastrzeżonych kolumn w celu określenia własności, fizyczne tworzenie wielu produktów z danymi lub osadzanie metadanych w tych produktach.

Model operacyjny

Warunkiem sukcesu w ramach procesu opracowywania produktów z danymi jest stosowanie modelu operacyjnego zapewniającego efektywne zarządzanie danymi, ustanowienie standardów i najlepszych praktyk, monitorowanie wydajności i zagwarantowanie jakości. Typowy zespół zajmujący się produktami z danymi składa się zwykle z architekta wspomagającego, inżynierów danych, osób modelujących dane oraz danologów. Taki zespół musi być odpowiednio wspierany i finansowany, aby mógł tworzyć produkty z danymi i ciągle je ulepszać. Ponadto musi istnieć jednostka organizacyjna, która ustanawia i nadzoruje standardy oraz najlepsze praktyki procesu budowania produktów z danymi w skali całej organizacji. Uważam, że firmy odnoszą największy sukces, gdy przygotowują instrukcje działań w postaci dokumentów opisujących wszystkie cele firmy, procesy, przepływy, kompromisy, listy kontrolne, role oraz zakresy odpowiedzialności. Takie instrukcje są często udostępniane w otwartych repozytoriach, dzięki czemu każdy może wziąć udział w ich tworzeniu.

Ustanawianie dobrych standardów i najlepszych praktyk uwzględnia określanie, jak zespoły mogą mierzyć wydajność i oceniać jakość, a także w jaki sposób niezbędne usługi muszą być do siebie dopasowane w przypadku każdego wzorca wykorzystania, tak aby można ich było ponownie użyć w obrębie wszystkich produktów z danymi. By zapewnić, że produkty te spełniają wymagania konsumentów i są cały czas ulepszane, zespoły powinny mierzyć wartość i skalę powodzenia swoich działań. Odpowiednie wskaźniki pomiarowe mogą obejmować liczbę konsumentów danego produktu z danymi, zaległe i rozpatrywane problemy dotyczące jakości, wyniki dotyczące poziomu zadowolenia, zwrot z inwestycji lub uwzględnione warianty zastosowań.

Po zaznajomieniu się z podstawami pozwalającymi dokonać niezbędnej zmiany sposobu myślenia w kwestii zarządzania danymi jako produktami zajmijmy się pytaniami, które architekci słyszą

najczęściej. Jaka jest dobra strategia w przypadku tworzenia produktów z danymi? Czy powinno się to ułatwić z użyciem centralnie udostępnionej platformy, czy domeny muszą spełniać swoje własne wymagania?

Architektura produktów z danymi

Aby znacząco zmniejszyć stopień złożoności swojej architektury, musisz ustalić właściwy stan równowagi między centralizacją a decentralizacją. Na jednym końcu spektrum znajduje się metoda pełnego stowarzyszania i autonomii, dzięki której każda domena ma możliwość udostępniania własnej architektury i wyboru własnego stosu technologii. Może to jednak prowadzić do szybkiego wzrostu liczby produktów z danymi, protokołów interfejsu, informacji w postaci metadanych itp. Znacznie utrudnia to też nadzór nad danymi i ich kontrolowanie, ponieważ każda domena musi precyzyjnie implementować centralne zagadnienia, takie jak archiwizowanie danych oraz ich pochodzenie, jakość i bezpieczeństwo. Zajęcie się tymi wszystkimi kwestiami w obrębie samych domen byłoby wyzwaniem i spowodowałoby złożoność z fragmentaryzacją i powstaniem silosów.

Na drugim końcu spektrum jest metoda scentralizowanego nadzoru ze stowarzyszaniem, lecz nie z pojedynczym silosem, ale z wieloma instancjami platformy lub planami stref docelowych korzystającymi z tych samych standardowych usług infrastruktury. Standaryzowane komponenty i wspólna infrastruktura zmniejszają koszty, obniżają liczbę interfejsów i zwiększają poziom kontroli. W idealnej sytuacji wiele architektur produktów z danymi byłoby dostępnych w ramach tej samej infrastruktury platformy, lecz odizolowanych od siebie w logiczny sposób.

Ogólny projekt platformy

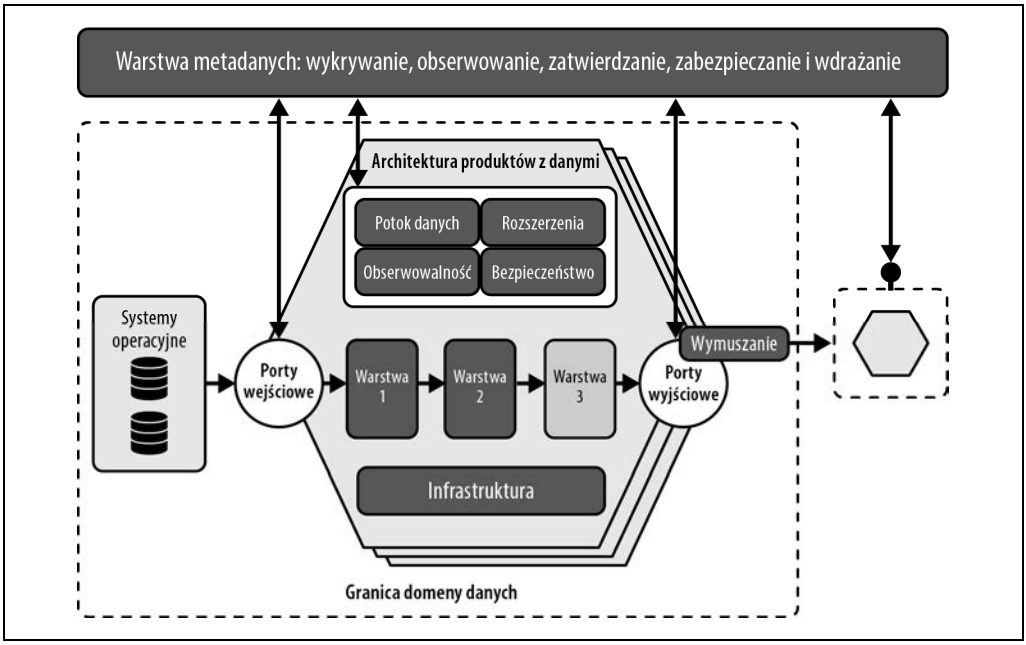
Zaprezentujmy szerzej architekturę i komponenty niezbędne do budowania produktów z danymi. Udostępnienie danych jako produktu umożliwia konsumentom danych łatwe wykrywanie, znajdowanie i analizowanie danych znajdujących się w wielu domenach oraz bezpieczne korzystanie z nich. Niemniej jednak możliwości wspólnej platformy służącej do dystrybucji produktów z danymi powinny być ukierunkowane głównie na przechwytywanie danych, przekształcanie ich do postaci wersji zoptymalizowanych do odczytu oraz bezpieczne dostarczanie danych innym domenom. Wszelkie możliwości analityczne lub raportowania umożliwiające opracowywanie wariantów zastosowań na potrzeby podejmowania decyzji zależnych od danych zostaną później dodane do architektury. Omówiono je w rozdziale 11.

Ujmując to trochę obszerniej, Twoja architektura produktów z danymi w co najmniej minimalnym stopniu powinna uwzględniać:

- możliwości przechwytywania, pozyskiwania i wprowadzania danych za pośrednictwem portów wejściowych,
- możliwości udostępniania lub zapewniania danych różnym domenom za pośrednictwem portów wyjściowych,
- możliwości metadanych związane z dokumentowaniem, wykrywaniem, monitorowaniem, zabezpieczaniem i nadzorowaniem danych,

- możliwości z zakresu inżynierii danych pozwalające na tworzenie, wdrażanie i uruchamianie kodu produktów z danymi,
- możliwości zarządzania danymi podstawowymi i ich jakością,
- możliwości związane z „rynkiem" danych w zakresie zapewniania jednolitego sposobu dokonywania zakupu danych.

Rysunek 4.5 zawiera logiczny projekt architektury produktów z danymi demonstrujący sposób przechwytywania danych, przekształcania ich i udostępniania innym domenom. Zanim bardziej szczegółowo omówimy poszczególne komponenty, dokonam ich ogólnego przeglądu. U samej góry znajduje się warstwa metadanych zapewniająca wgląd w dostępne dane, ich właściwości i relacje z innymi obszarami zarządzania danymi. W obrębie tej warstwy umiejscowiono repozytoria metadanych wykorzystywane na potrzeby orkiestracji, obserwowalności, zarządzania kontraktami itp. Są to punkty łączące istniejące między „rynkiem" danych a architekturą produktów z danymi.



Rysunek 4.5. Przykładowa architektura produktów z danymi

Gdy korzystasz ze wspólnej infrastruktury, różne architektury produktów z danymi są izolowane, a ponadto nie są współużytkowane bezpośrednio między domenami. Każda taka architektura oraz kwestia odpowiedzialności pozostają powiązane z dostawcą danych. Dostawca danych, czyli domena, dysponuje swoją własną instancją architektury produktów z danymi oraz własnym potokiem, a także przejmuje własność w zakresie jakości i integralności danych.

Kluczem do tego, by inżynierowie mogli obsługiwać wspólne platformy i architektury produktów z danymi, jest zapewnienie zestawu centralnych i fundamentalnych komponentów niezależnych od domeny, a także podejmowanie różnych decyzji projektowych w celu wydzielenia

z domen powtarzających się problemów oraz uproszczenia zarządzania danymi. Opierając się na obserwacjach dokonanych w tym obszarze, w dalszej części rozdziału zwrócę uwagę na najważniejsze kwestie.

Możliwości przechwytywania i wprowadzania danych

Wprowadzanie danych to pierwszy kluczowy krok do pomyślnego zbudowania architektury danych. Proces wyodrębniania danych z systemu źródłowego pozostaje też jednak jednym z najtrudniejszych problemów w przypadku korzystania z danych na dużą skalę. Wymaga to uwzględnienia kilku kwestii.

Metoda pozyskiwania

Najpierw powinno się określić szybkość, z jaką dane muszą być dostarczane lub synchronizowane. Ogólnie rzecz ujmując, do wyboru są następujące dwa scenariusze:

Przetwarzanie wsadowe

Obsługę konsumentów wymagających danych tylko sporadycznie można ułatwić za pomocą przetwarzania wsadowego, które stosuje się do procesu jednoczesnego transferowania dużego woluminu danych. Dane przetwarzane wsadowo mają zwykle postać zbioru zawierającego miliony rekordów przechowywanych jako plik. Dlaczego nadal wymagane jest przetwarzanie wsadowe? W wielu przypadkach jest to jedyna metoda zgromadzenia naszych danych. Najbardziej znaczące systemy RDBMS (*Relational Database Management System*) są wyposażone w komponenty wsadowe obsługujące przenoszenie danych partiami. Często problemem okazuje się *uzgadnianie danych*, czyli proces weryfikowania danych w trakcie takiej operacji przenoszenia[6]. Zasadnicze znaczenie ma sprawdzenie, czy dysponuje się wszystkimi danymi, zwłaszcza w przypadku procesów finansowych. Ponadto wiele systemów jest złożonych ze względu na ogromną liczbę tabel, a jedynym sposobem właściwego wybierania i wyodrębniania ich danych jest język SQL (*Structured Query Language*).

Pozyskiwanie danych w czasie rzeczywistym

Konsumenci, którzy żądają przyrostowych aktualizacji danych lub odbywających się niemal w czasie rzeczywistym, mogą być obsługiwani w najlepszy sposób z wykorzystaniem procesu *pozyskiwania opartego na interfejsie API* lub *zależnego od zdarzeń*. Takie przetwarzanie umożliwia względnie szybkie gromadzenie i analizowanie danych, a ponadto sprawdza się dobrze w zastosowaniach, w których ilości danych są stosunkowo małe, obliczenia przeprowadzane na danych są dość proste, a poza tym wymagane są opóźnienia bliskie tych, które występują przy przetwarzaniu w czasie rzeczywistym. Wiele wariantów zastosowań może być obsługiwanych z zastosowaniem pozyskiwania zależnego od zdarzeń. Jeśli jednak główną kwestią jest kompletność i wielkość danych, większość zespołów będzie się posiłkować tradycyjnym przetwarzaniem wsadowym. Więcej uwagi poświęcimy temu zagadnieniu w rozdziale 6.

[6] **Uzgadnianie** można zaimplementować na wiele różnych sposobów. Możesz na przykład porównać liczbę wierszy, zastosować sumy kontrolne kolumn (*https://pl.wikipedia.org/wiki/Suma_kontrolna*), podzielić na części duże tabele i ponownie je przetworzyć lub skorzystać z narzędzi do porównywania danych, takich jak Redgate (*https://www.red-gate.com/*).

Choć pozyskiwanie danych w czasie rzeczywistym zaczęło zyskiwać na popularności, architektura oparta wyłącznie na strumieniowaniu (Kappa) nigdy nie zastąpi w pełni przetwarzania wsadowego we wszystkich zastosowaniach. W przypadku większości przedsiębiorstw oczekuję, że równolegle będą stosowane obie metody pozyskiwania.

Złożone pakiety oprogramowania

W przypadku każdej operacji wprowadzania danych ważne jest, by dostrzec różnorodność specjalistycznych pakietów oprogramowania. Interpretowanie lub używanie wielu z nich jest niezwykle trudne. Schematy baz danych są często nadzwyczaj złożone, a integralność referencyjna jest zwykle utrzymywana programowo za pośrednictwem aplikacji, a nie bazy danych. Niektórzy dostawcy nawet chronią swoje dane, dlatego można je wyodrębnić tylko przy wsparciu zewnętrznego rozwiązania.

We wszystkich tych złożonych sytuacjach zalecam, by w przypadku każdego systemu źródłowego ocenić, czy architektura musi zostać uzupełniona dodatkowymi usługami umożliwiającymi wyodrębnienie danych. Takie usługi mogą ukrywać te złożone struktury systemu źródłowego i chronić przed długotrwałymi i kosztownymi działaniami związanymi z odwzorowywaniem. Chronią one również przed ścisłym sprzężeniem, ponieważ przeważnie nie kontrolujesz bezpośrednio schematu danych złożonego produktu dostawcy. Jeżeli dostawca udostępni aktualizację wersji produktu i zmienią się struktury danych, przestaną działać wszystkie Twoje potoki danych. Końcowy wniosek jest taki, że komponent zewnętrznego dostawcy jest wart zainwestowania.

Zewnętrzni dostawcy interfejsu API i modelu SaaS

Zewnętrzni dostawcy interfejsu API lub modelu SaaS zwykle też wymagają szczególnej uwagi. Doświadczyłem sytuacji, w których niezbędny był pełny zbiór danych, ale dostawca modelu SaaS zapewnił za pośrednictwem interfejsu API jedynie stosunkowo niewielką ilość danych. Inni dostawcy mogą stosować ograniczanie, czyli przydział lub ograniczenie liczby żądań. Są też dostawcy stosujący kosztowne plany płatności za każdy kontakt telefoniczny z nimi. We wszystkich tych sytuacjach zalecam albo uzyskanie narzędzia obsługującego wyodrębnianie z interfejsów API, albo zbudowanie komponentu niestandardowego, który okresowo pobiera dane.

Poprawnie zaprojektowana implementacja niestandardowa może najpierw kierować wszystkie wywołania interfejsu API do Twojej architektury produktów z danymi. Jeżeli ostatnio zostało utworzone to samo żądanie, wyniki są bezpośrednio udostępniane z tej architektury. W przeciwnym razie inicjowany jest interfejs API dostawcy modelu SaaS, a wyniki są zwracane i również zapisywane w Twojej architekturze produktów z danymi na potrzeby wszystkich kolejnych wywołań. W przypadku takiego wzorca cała kolekcja danych może ostatecznie zostać zapewniona dla obsługiwanych konsumentów.

Pochodzenie i metadane

W przypadku wszystkich używanych komponentów służących do wprowadzania i wyodrębniania danych ważne jest, by zwrócić szczególną uwagę na pochodzenie danych. Firmy uważają za istotne stwierdzenie, jakie dane zostały wyodrębnione, a także jakie transformacje zastosowano względem

danych. Ważnym działaniem jest standaryzacja i określanie, jak konektory będą się integrować z Twoim repozytorium pochodzenia danych. Należy to zrobić na początku, tak aby wiadomo było, co będzie integrowane, a co nie. Biorąc pod uwagę silną zależność informacji o pochodzeniu od struktur transformacji i narzędzi procesu ETL, może być trudno dodać te informacje w sposób retrospektywny. Najlepszą opcją jest wybieranie danych, testowanie, sprawdzanie poprawności i standaryzowanie przed rozpoczęciem faktycznej implementacji.

Jakość danych

Jakość danych to kolejna ważna kwestia projektowa. W momencie pozyskania zbiorów danych do architektury produktów z danymi powinno się sprawdzić ich jakość. Wiążą się z tym dwa aspekty.

Po pierwsze, powinno się sprawdzić poprawność integralności posiadanych danych przez porównanie ich z opublikowanymi schematami. Tak się postępuje, ponieważ wskazane jest zapewnienie, że nowe dane są wysokiej jakości oraz zgodne z przyjętymi standardami i oczekiwaniami. W przeciwnym razie podejmuje się działanie. Za tę pierwszą linię obrony odpowiadają właściciele danych, zarządcy danych oraz inżynierowie danych. Są to członkowie zespołu, którzy tworzą dane i określają ich źródło. Do nich należy zapewnienie, że dane spełniają wymagania dotyczące jakości danych, które są ustalane przez organy nadzorujące lub konsumentów danych.

Po drugie, mają miejsce sprawdzenia jakości, które z założenia są bardziej stowarzyszone. Takie sprawdzanie poprawności jest w mniejszym stopniu zależne od poszczególnych zespołów produktów z danymi, gdyż inne zespoły określają to, co stanowi o dobrej jakości danych. Tego rodzaju sprawdzenia jakości danych są zwykle realizowane asynchronicznie, a ponadto powodują oznaczanie i/lub powiadamianie odpowiednich uczestników. Stowarzyszone elementy kontroli są zazwyczaj sprawdzeniami funkcjonalnymi kompletności, dokładności, spójności, wiarygodności itp. Integralność danych oraz jakość danych stowarzyszonych to dwa różne zagadnienia, do których czasem przypisane jest osobne repozytorium metadanych, gdzie są przechowywane reguły sprawdzania poprawności funkcjonalnej lub metadane schematu. W tym przypadku z każdym zagadnieniem są powiązane jego własne standardowe usługi platformy.

W przypadku własnego projektu platformy zaletą wynikającą z użycia wspólnej infrastruktury dla architektur produktów z danymi jest to, że w ramach analizowania jakości danych możesz skorzystać z siły technologii Big Data. Przykładowo platforma Apache Spark (*https://spark.apache.org/*) zapewnia możliwości przetwarzania rozproszonego setek milionów wierszy danych. Aby efektywnie z tego skorzystać, możesz posłużyć się strukturą przeznaczoną do dokumentowania, testowania i profilowania danych pod kątem ich jakości. Na przykład Great Expectations (*https://greatexpectations.io/*) i Soda (*https://www.soda.io/*) to otwarte standardy jakości danych, które są używane przez wiele dużych firm. Kolejną zaletą zarządzania jakością danych w obrębie wspólnej infrastruktury jest to, że możesz przeprowadzać sprawdzenia integralności referencyjnej obejmujące wiele domen. Dzięki wzajemnemu sprawdzaniu integralności odwołań oraz porównywaniu i zestawianiu różnych zbiorów danych znajdziesz błędy i korelacje, o których istnieniu nie wiedziałeś.

Gdy poprawnie zaimplementuje się monitorowanie jakości danych, nie tylko umożliwi ono wykrywanie i obserwowanie problemów z jakością danych, ale też stanie się częścią ogólnej struktury

kontroli jakości danych, która sprawdza nowe dane i dodaje je do istniejących danych[7]. Jeśli z jakiegokolwiek powodu jakość obniży się poniżej określonego progu, struktura może „zaparkować" dane i zwrócić się z prośbą do ich właścicieli o sprawdzenie ich, a następnie albo poręczenie za nie, albo ich usunięcie lub ponowne dostarczenie. Dysponowanie strukturą kontroli jakości danych oznacza, że istnieje zamknięta pętla informacji zwrotnej, która cały czas usuwa problemy z jakością i zapobiega ich ponownemu wystąpieniu. Jakość danych jest nieustannie monitorowana. W przypadku zmian poziomu jakości natychmiast są podejmowania działania. Miej na uwadze to, że problemy z jakością danych muszą być rozwiązywane w systemach źródłowych, a nie w obrębie Twojej architektury produktów z danymi. Usuwanie tych problemów u źródła zapewnia, że nie wystąpią one w innych miejscach.

Wpływu jakości danych nie należy lekceważyć. Jeżeli jakość danych jest zła, konsumenci będą mieć do czynienia z powtarzającym się oczyszczaniem i poprawianiem danych. Chcę również podkreślić wagę standaryzacji. Jeśli poszczególne zespoły zdecydują się na własne struktury jakości danych, nie będzie możliwe porównywanie wskaźników pomiarowych i wyników między domenami.

Historyzacja danych

Poprawne zarządzanie danymi historycznymi jest kluczowe z biznesowego punktu widzenia, ponieważ bez tego nie jest możliwe obserwowanie trendów i opracowywanie prognoz na przyszłość. Standardy powinno się uwzględnić również w przypadku zarządzania danymi historycznymi. W tym punkcie przedstawiam trochę podstawowych informacji, a także objaśniam kilka różnych reprezentacji i metod zarządzania tymi danymi.

Porządkowanie danych historycznych to kluczowy aspekt związany z hurtowniami danych. W tym kontekście często słyszy się terminy **niezmienny** (ang. *nonvolatile*) i **zmienny w czasie** (ang. *time-variant*). Niezmienność oznacza, że dotychczasowe dane nie są usuwane w momencie dodawania do nich nowych danych. Zmienność w czasie wskazuje, że dane są spójne w konkretnym okresie. Przykładowo dane są ładowane codziennie lub z jakąś inną częstotliwością i nie zmieniają się w tym okresie.

Produkty z danymi odgrywają ważną rolę w przypadku przechowywania dużych ilości danych historycznych oraz zarządzania nimi. Jak jednak poradziłbyś sobie z tym w architekturze zdecentralizowanej? W porównaniu z hurtownią danych różnica polega na tym, że produkt z danymi zachowuje je w ich oryginalnym kontekście. Nie oczekuje się żadnej transformacji do modelu danych przedsiębiorstwa, ponieważ produkty z danymi są dopasowywane do domen. Oryginalny (domenowy) kontekst nie jest tracony. To znacząca korzyść: oznacza to, że na potrzeby swoich własnych wariantów zastosowań operacyjnych, takich jak uczenie maszynowe, domeny mogą korzystać z własnych produktów z danymi i jednocześnie oferować dane innym domenom. W konsekwencji tego domeny

[7] By zaznajomić się z omówieniem tego, jak można zintegrować ze sobą narzędzia Azure Synapse Analytics i Microsoft Purview w celu przetwarzania i sprawdzania poprawności danych z wykorzystaniem standardu Great Expectations, przeczytaj post *Modern Data Pipelines with Azure Synapse Analytics and Azure Purview* na moim blogu (*https://piethein.medium.com/modern-data-pipelines-with-azure-synapse-analytics-and-azure-purview-fe752d874c67*).

mają nieodłączną potrzebę dbania o należące do nich dane. Ostatecznie taka praktyka stosowania przez domenę własnych produktów z danymi spowoduje zwiększenie zarówno ich jakości, jak i poziomu zadowolenia klientów.

Choć architektura produktów z danymi to zagadnienie niezależne od technologii, prawdopodobne jest to, że inżynierowie będą opracowywali wiele takich architektur z użyciem rozproszonych systemów plików o mniejszym koszcie, takich jak usługi „jeziora" danych. Oznacza to, że niezbędne są różne metody przetwarzania danych do aktualizowania i nadpisywania danych podstawowych, transakcyjnych i referencyjnych. A zatem zalecam rozważenie użycia jednej z dalej omówionych metod lub ich kombinacji, z których każda uwzględnia kompromis obejmujący inwestowanie i zarządzanie danymi wejściowymi oraz łatwość korzystania z danych. Każda metoda ma zalety i wady.

Punkt w czasie

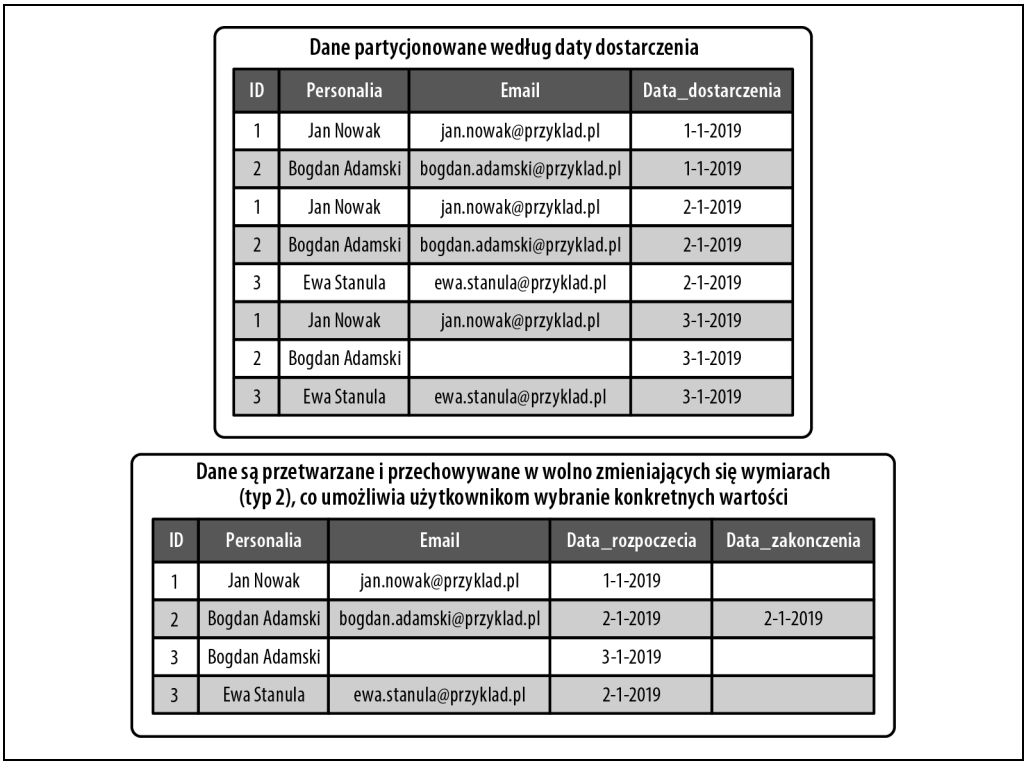
Pierwsza metoda zarządzania danymi historycznymi polega na przechowywaniu oryginalnych danych jako odwołania historycznego, które zwykle jest niezmienne. Pomyśl o rezerwuarze, do którego dane są pozyskiwane przez kopiowanie. W tym wariancie wyłącznie z wstawianiem danych zalecane jest partycjonowanie danych w logiczny sposób, na przykład za pomocą logicznego rozdzielenia daty i godziny (rysunek 4.6)[8].

Zarządzanie tylko pełnymi migawkami jest łatwe do zaimplementowania. W każdym cyklu procesu ETL całe dane wyjściowe są zapisywane jako niezmienna migawka. Później schematy tabel można zastąpić za pomocą lokalizacji, w której są przechowywane nowe dane. Kontener jest kolekcją wszystkich takich migawek reprezentujących punkt w czasie. Część praktyków może twierdzić, że takie podejście powoduje duplikowanie danych. Nie uważam tego za problem, ponieważ obecnie magazynowanie w chmurze jest stosunkowo tanie. Pełne migawki ułatwiają też ponowne dostarczanie lub przebudowywanie zintegrowanego zbioru danych, który przechowuje wszystkie zmiany dokonane z upływem czasu. Jeśli system źródłowy wykryje, że dostarczono niepoprawne dane, mogą one zostać ponownie wysłane, co spowoduje nadpisanie partycji.

Użycie wyłącznie metody z reprezentacją punktu w czasie może być właściwe, gdy zapewniane dane zawierają już wszystkie niezbędne dane historyczne (jeśli na przykład dane dostarczono w taki sposób, że uwzględniają daty rozpoczęcia i zakończenia dla wszystkich rekordów).

Mankamentem tworzenia migawek pełnych zbiorów danych jest to, że trudniejsza jest (retrospektywna) analiza danych. Problematyczne są porównania konkretnych okresów, gdy niedostępne są daty rozpoczęcia i zakończenia, ponieważ konsumenci będą zmuszeni do przetwarzania i przechowywania wszystkich danych historycznych. Przetwarzanie danych historycznych na przykład z okresu trzech lat może wymagać uwzględnienia w tym procesie kolejno tysiąca plików.

[8] Partycjonowanie to powszechna technika organizowania plików lub tabel w ramach odrębnych grup (partycji) w celu zwiększenia możliwości zarządzania, wydajności lub dostępności. Odbywa się to zwykle względem atrybutów danych, takich jak atrybuty geograficzne (miasto, państwo) albo atrybuty oparte na wartości (unikalne identyfikatory, kody segmentów) lub czasie (data dostarczenia, godzina).



Dane partycjonowane według daty dostarczenia

ID	Personalia	Email	Data_dostarczenia
1	Jan Nowak	jan.nowak@przyklad.pl	1-1-2019
2	Bogdan Adamski	bogdan.adamski@przyklad.pl	1-1-2019
1	Jan Nowak	jan.nowak@przyklad.pl	2-1-2019
2	Bogdan Adamski	bogdan.adamski@przyklad.pl	2-1-2019
3	Ewa Stanula	ewa.stanula@przyklad.pl	2-1-2019
1	Jan Nowak	jan.nowak@przyklad.pl	3-1-2019
2	Bogdan Adamski		3-1-2019
3	Ewa Stanula	ewa.stanula@przyklad.pl	3-1-2019

Dane są przetwarzane i przechowywane w wolno zmieniających się wymiarach (typ 2), co umożliwia użytkownikom wybranie konkretnych wartości

ID	Personalia	Email	Data_rozpoczecia	Data_zakonczenia
1	Jan Nowak	jan.nowak@przyklad.pl	1-1-2019	
2	Bogdan Adamski	bogdan.adamski@przyklad.pl	2-1-2019	2-1-2019
3	Bogdan Adamski		3-1-2019	
3	Ewa Stanula	ewa.stanula@przyklad.pl	2-1-2019	

Rysunek 4.6. Przykłady tego, jak wyglądają dane poddane partycjonowaniu z użyciem w pełni wymiarowych migawek lub wolno zmieniających się wymiarów (typ 2)

Przetwarzanie danych może zająć mnóstwo czasu i mocy obliczeniowej, zależnie od wielkości danych. W związku z tym zaleca się, by rozważyć też użycie metody interwałowej.

Interwał

Kolejną metodą prezentowania danych historycznych i zarządzania nimi jest budowanie historycznych zbiorów danych, w których wszystkie dane są porównywane i przetwarzane. Przykładowo możesz tworzyć zbiory danych z użyciem *wolno zmieniających się wymiarów*[9], które prezentują wszystkie wprowadzone z czasem zmiany. Proces ten wymaga zrealizowania procesu ETL, ponieważ musisz porównać dane z bieżącej dostawy z tymi z dowolnych wcześniejszych dostaw. Po dokonaniu porównania rekordy są otwierane i/lub zamykane. W ramach tego procesu data zakończenia

[9] Wolno zmieniający się wymiar to wymiar przechowujący w hurtowni danych zarówno dane bieżące, jak i historyczne, a także zarządzający nimi. W przypadku zarządzania danymi oraz stosowania hurtowni danych tabele wymiarowe można utrzymywać za pomocą kilku metodyk reprezentowanych przez typy numerowane od 0 do 6. Różne metodyki opisano w witrynie Kimball Group (*https://www.kimballgroup.com/2013/02/design-tip-152-slowly-changing-dimension-types-0-4-5-6-7/*).

jest zwykle przypisywana aktualizowanym wartościom, a data rozpoczęcia nowym wartościom. Na potrzeby porównania zaleca się wykluczenie wszystkich danych niefunkcjonalnych[10].

Czy mogę projektować swoje produkty z danymi z użyciem wirtualizacji danych?

Zagadnienia wirtualizacji danych celowo nie poruszałem przed omówieniem historyzacji. Wirtualizacja danych i tworzenie produktów z danymi nie są najlepszym połączeniem z kilku powodów:

- Wirtualizacja danych nie dopełnia zarządzania cyklem istnienia danych. Nie umożliwia ona przenoszenia nieodpowiednich danych poza obręb podstawowych systemów. Wirtualizacja wymaga, aby wszystkie dane historyczne pozostały w systemach OLTP, które w dłuższej perspektywie sprawiają, że wirtualizacja danych jest kosztownym rozwiązaniem.
- Warstwa wirtualizacji danych prowadzi do silniejszego sprzężenia[11]. Jeśli zmieniają się systemy źródłowe, natychmiast trzeba też wprowadzić zmiany w warstwie wirtualizacji danych. Widoki lub dodatkowe warstwy mogą być pomocne, ale każda zmiana nadal wymaga koordynacji między właścicielami systemów źródłowych a inżynierami utrzymującymi warstwę wirtualizacji danych.
- Wirtualizacja danych jest zależna od sieci. W środowisku rozproszonym z mnóstwem sieci i przeskoków (dotyczą one przekazywania danych za pośrednictwem dodatkowych urządzeń sieciowych) oczekuje się zwiększenia opóźnienia. Ponadto pojawia się sprzężenie, dlatego w przypadku niedostępności sieci przestaje działać warstwa wirtualizacji.
- Wirtualizacja danych jest ograniczona przez podstawową technologię wspomagającą, która ma zwykle postać systemu RDBMS. Choć wirtualizacja danych pozwala na odczyt z wielu systemów baz danych, generalnie nie umożliwia ona tworzenia punktów końcowych baz danych dokumentów, baz z parami klucz-wartość, baz kolumnowych i baz grafowych.
- W przypadku intensywnych i powtarzalnych zapytań wirtualizacja danych korzysta z większej mocy obliczeniowej, ponieważ transformacje są przeprowadzane w czasie rzeczywistym w chwili uzyskiwania danych za pomocą zapytania. Techniki buforowania mogą ograniczyć ten efekt, ale ilość niezbędnej mocy obliczeniowej zawsze będzie większa, gdy dane są stosowane po wstępnym przetworzeniu ich na potrzeby konsumentów.

Oto wniosek końcowy: dostęp do produktów z danymi można objąć wirtualizacją, ale nie powinno się projektować i przygotowywać swoich produktów wyłącznie z użyciem mechanizmu wirtualizacji danych.

Metoda polegająca na porównywaniu interwałów i tworzeniu danych historycznych zapewnia korzyść w postaci efektywniejszego przechowywania danych, gdyż są one przetwarzane, poddawane

[10] Przykładowo w obrębie swojej struktury przetwarzającej możesz użyć parametru `exclude_from_delta_processing` na poziomie atrybutu, aby wykluczyć dane z operacji porównywania.

[11] Niektórzy dostawcy baz danych zapewniają warstwę zapytań (wirtualną) kierowanych do bazy, którą określa się też mianem warstwy wirtualizacji danych. Warstwa ta dokonuje abstrakcji bazy danych i optymalizuje dane pod kątem wydajności operacji odczytu. Innym powodem stosowania abstrakcji jest przechwytywanie zapytań w celu zwiększenia bezpieczeństwa.

deduplikacji i łączone. Jak widać na rysunku 4.6, znajdujący się po prawej stronie wolno zmieniający się wymiar obejmuje połowę liczby wierszy. W konsekwencji szybciej i łatwiej uzyskać dane w ramach zapytania na przykład wtedy, gdy korzysta się z relacyjnej bazy danych. Oczyszczanie danych lub usuwanie poszczególnych rekordów, co może być konieczne do zapewnienia zgodności z ogólnym rozporządzeniem o ochronie danych, również będzie prostsze, ponieważ nie będzie wymagane przetwarzanie wszystkich wcześniej dostarczonych zbiorów danych.

Mankamentem jest jednak to, że tworzenie wolno zmieniających się wymiarów wymaga zarządzania w większym zakresie. Przetwarzane muszą być wszystkie dane. Zmiany w podstawowych danych źródłowych muszą zostać wykryte od razu po ich wystąpieniu, a następnie przetworzone. Proces wykrywania wymaga dodatkowego kodu i mocy obliczeniowej. Inną kwestią jest to, że konsumenci danych mają zwykle różne wymagania dotyczące historyzacji. Jeśli na przykład dane są przetwarzane codziennie, ale konsument wymaga ich reprezentacji z całego miesiąca, w dalszym ciągu będzie niezbędny dodatkowy krok przetwarzania.

Poza tym proces ponownego dostarczania może się okazać kłopotliwy i trudny w zarządzaniu, gdyż mogą być przetwarzane niepoprawne dane, które mogą stać się częścią wymiarów. Można to wyeliminować za pomocą logiki ponownego przetwarzania, dodatkowych wersji lub dat ważności, ale nadal niezbędne jest dodatkowe zarządzanie. Pojawia się tutaj pułapka polegająca na tym, że zadanie poprawnego zarządzania danymi może przypaść centralnemu zespołowi, dlatego taka skalowalność wymaga użycia w szerokim zakresie inteligentnej i samoobsługowej funkcjonalności.

Kolejnym mankamentem towarzyszącym tworzeniu danych historycznych na potrzeby ogólnego korzystania z nich przez wszystkich konsumentów jest to, że w dalszym ciągu mogą być oni zmuszeni do pewnego przetwarzania każdorazowo, gdy pomijają kolumny w swoich wybranych danych. Jeżeli na przykład konsument bardziej zawęzi swój wybór, ostatecznie mogą się pojawić zduplikowane rekordy, a zatem konieczne będzie ponowne przetwarzanie danych w celu wyeliminowania z nich duplikatów. Z myślą o skalowalności i zapewnieniu pomocy konsumentom możesz też rozważyć połączenie ze sobą dwóch metod, zachowując w pełni wymiarowe migawki i zapewniając dane historyczne jako usługę. W tym wariancie wszystkim domenom konsumentów oferuje się niewielką strukturę obliczeniową, za pomocą której mogą oni zaplanować utworzenie danych historycznych na podstawie częstotliwości (codziennie, co tydzień lub co miesiąc), zakresu czasu, atrybutów i niezbędnych zbiorów danych. W przypadku instancji krótkotrwałego przetwarzania w chmurze publicznej możesz nawet sprawić, że będzie to efektywne kosztowo. Dużą korzyścią wynikającą z tego podejścia jest to, że możesz utrzymać elastyczność ponownych dostarczeń, lecz nie wymagaj, aby zespół inżynierów przygotowywał zbiory danych oraz zarządzał nimi. Inną korzyścią jest możliwość dostosowywania zakresów czasu, zasięgów i atrybutów do wymagań poszczególnych osób.

Tylko dołączanie

Niektóre style dostarczania danych najlepiej obsłużyć za pomocą metody tylko z dołączaniem. W ramach tej metody z bazy danych aplikacji ładujesz jedynie nowe lub zmodyfikowane rekordy i dołączasz je do istniejącego zestawu rekordów. Sprawdza się to w przypadku danych transakcyjnych, a także danych opartych na zdarzeniach, o czym będzie mowa w rozdziale 6. Metody tylko

z dołączaniem często się używa w połączeniu z przechwytywaniem danych zmian. W tym wariancie konfigurujesz strumień zmian przez identyfikowanie i przechwytywanie zmian dokonanych w standardowych tabelach, tabelach katalogu i widokach.

Historyzacja i pozyskiwanie danych strumieniowanych

Pozyskiwanie danych czasu rzeczywistego jest zwykle obsługiwane na dwa różne sposoby. W pierwszym scenariuszu transformacje odbywają się dynamicznie w momencie pojawienia się danych, a wynik jest bezpośrednio dołączany na przykład w bazie danych NoSQL. Ten cały proces trwa zazwyczaj kilka sekund lub minut. Głównym celem jest udostępnienie danych od razu po ich uzyskaniu. W drugim scenariuszu dane czasu rzeczywistego są przetwarzane okresowo (np. jako mikropartie). Podstawowym celem jest tutaj scalenie i poddanie historyzacji wszystkich danych, tak aby mogły zostać wykorzystane niżej w strukturze w krótszym odstępie czasu. Podczas scalania danych pod uwagę należy wziąć opóźnienie przetwarzania, ponieważ zajmuje ono czas.

Tworzenie własnej strategii historyzacji

Właściwa metoda historyzacji w przypadku produktu z danymi zależy od typów danych, wymagań konsumentów i regulacji. Wszystkie metody historyzacji i tworzenia produktów z danymi można łączyć, a ponadto używać różnych metod na potrzeby odmiennych rodzajów danych. Przykładowo dane podstawowe można dostarczać z aplikacji i rozbudowywać z wykorzystaniem wolno zmieniających się wymiarów. Choć dane referencyjne można z łatwością przetwarzać za pomocą migawek, może być wskazane obsługiwanie danych transakcyjnych dla tego samego systemu za pośrednictwem metody dostarczania tylko z dołączaniem. Możesz utrzymywać pełne migawki wszystkich dostaw danych, a jednocześnie rozbudowywać je z zastosowaniem wolno zmieniających się wymiarów.

W skutecznym przygotowywaniu i projektowaniu produktów z danymi kluczowa jest rozbudowana strategia. Zgodnie z najlepszymi praktykami powinieneś się skupić na sposobie tworzenia warstw danych i budowania historii w obrębie każdej swojej domeny. Taka strategia tworzenia warstw może uwzględniać przygotowywanie i utrzymywanie skryptów oraz standardowych usług, takich jak usługi przechwytywania danych zmian[12].

Zaprezentowane tutaj najlepsze praktyki mają na celu zapewnienie wniosków, ale omawiam je także po to, aby pokazać, że zarządzanie produktami z danymi jest trudnym przedsięwzięciem. W następnym podrozdziale zagłębimy się bardziej w pewne aspekty projektowania rozwiązań w procesie opracowywania architektury. W tym celu użyję rzeczywistego przykładu, uwzględniając to, co zostało dotąd omówione.

[12] Jeden z moich wpisów na blogu (*https://piethein.medium.com/scd2-delta-tables-using-synapse-spark-pools-↪c1c3a6115f5d*) zawiera prosty przykładowy skrypt służący do przetwarzania za pomocą pul platformy Spark tabel wolno zmieniających się wymiarów w formacie Delta.

Projekt rozwiązań

Po tym, jak już się zaznajomiłeś z niektórymi najlepszymi praktykami i zasadami projektowymi, pora na coś bardziej praktycznego, czyli na tworzenie z wykorzystaniem rozwiązań i usług zoptymalizowanych pod kątem odczytu abstrakcji danych Twojej złożonej aplikacji. Firmy stosują do projektowania produktów z danymi bardzo różne kombinacje technologii i metodyk. Jest to też zagadnienie, z którym trudno się uporać, gdyż nie ma dostępnych żadnych platform ani produktów służących do zarządzania pełnym cyklem istnienia produktów z danymi. Opisanie wszystkich różnych metod implementowania kompleksowej architektury produktów z danymi wymagałoby osobnej książki, dlatego zamiast tego przed omówieniem konkretnego przykładu skoncentruję się na kluczowych kwestiach. Oto one:

- Chmura stała się domyślną infrastrukturą przeznaczoną do przetwarzania danych na dużą skalę, ponieważ w porównaniu ze środowiskami lokalnymi oferuje znaczne korzyści. Wszystkie popularne platformy chmurowe zapewniają zestaw samoobsługowych usług danych, które są wystarczające do zainicjowania każdej implementacji.
- Usługi „jeziora" danych to popularna opcja w przypadku wszystkich typów firm. Biorąc pod uwagę zwiększające się codziennie wolumeny danych, umiejscowienie danych na przykład w obrębie chmurowego magazynu obiektów zgodnego z systemem plików HDFS to znakomity sposób na zapewnienie efektywności kosztowej Twojej architektury. Korzyści oferowane przez tego typu usługi obejmują rozdzielenie zasobów magazynowych i obliczeniowych oraz ograniczenie duplikowania danych dzięki zastosowaniu uproszczonych usług zapytań[13].
- Popularny stos służący do przetwarzania danych składa się z platformy Spark, języka Python i notatników. Spośród atutów tego wzorca należy wymienić dużą i aktywną społeczność, korzyści związane z oprogramowaniem *open source* i dużymi możliwościami współdziałania oraz wspieranie w bardzo szerokim zakresie różnorodnych wariantów zastosowań, począwszy od inżynierii danych, a skończywszy na danologii. Choć platforma Spark to znakomity wybór w przypadku procesu ETL i uczenia maszynowego, nie jest ona zoptymalizowana pod kątem wykonywania interaktywnych zapytań o niewielkim opóźnieniu. Korzystając z notatników, można zwracać uwagę na powtarzające się działania, takie jak historyzacja danych, transformacje techniczne danych oraz sprawdzanie poprawności schematu. Gdy stosuje się notatniki, powszechną najlepszą praktyką jest opracowywanie wspólnych procedur i struktury opartej na metadanych z użyciem konfigurowalnych procedur dotyczących miejsc docelowych, sprawdzania poprawności, bezpieczeństwa itp.[14]
- Z myślą o transformowaniu i modelowaniu danych wiele firm uzupełnia swoją architekturę o dodatkowe usługi, takie jak dbt (*https://www.getdbt.com/*). Choć platforma Spark z kodem niestandardowym może posłużyć do transformacji danych, firmy często uznają, że większe

[13] Oddzielenie zasobów magazynowych od obliczeniowych pozwala na automatyczne skalowanie infrastruktury w celu dopasowania do elastycznych obciążeń.

[14] Więcej szczegółów związanych z tym zagadnieniem znajdziesz w moim wpisie na blogu pt. *Designing a metadata-driven processing framework for Azure Synapse and Azure Purview* (*https://piethein.medium.com/designing-a-metadata-driven-processing-framework-for-azure-synapse-↪and-azure-purview-33121a63ebc0*).

projekty można bardziej usprawnić dzięki tworzeniu szablonów i konfiguracji. Zarówno narzędzia do przygotowywania szablonów, jak i notatniki to realne opcje, które mogą się wzajemnie uzupełniać. Dostrzegam mnóstwo firm, które najpierw zajmują się sprawdzaniem poprawności danych, techniczną transformacją i historyzacją z użyciem notatników, a następnie integrują swoje dane z takimi narzędziami jak dbt.

- Orkiestracja i automatyzacja przepływów są niezbędne do zarządzania całym procesem od pozyskania danych do ich udostępnienia. Choć standaryzacja jest kluczowa z punktu widzenia obserwowalności, najlepszą praktyką jest zapewnienie zespołowi swobody w zakresie opracowania własnego, lokalnego zasobu wiedzy oraz trzymania się lokalnych priorytetów. Inną najlepszą praktyką jest integrowanie swoich procesów przepływów oraz integracji ciągłej i dostarczania ciągłego, lecz z utrzymaniem niezależności potoków dla każdej aplikacji lub produktu z danymi.
- Zalecam, by podczas wybierania jakichkolwiek narzędzi wpisać w wyszukiwarce `modern data stack` (nowoczesny stos danych). Dostępnych jest wiele popularnych opcji, wśród których znajdują się zarówno narzędzia *open source*, jak i rozwiązania z niedostępnym kodem źródłowym.
- Usługi zarządzania metadanymi, takie jak narzędzia katalogów danych i określania pochodzenia danych, często znajdują się poza obrębem architektur produktów z danymi. Usługi te są udostępniane przez centralną jednostkę nadzorującą jako ogólne usługi. Kwestie te są dogłębnie omówione w rozdziale 9.

Typowe pytanie o projektowanie produktów z danymi dotyczy sposobu powiązania zagadnień „jeziora" danych i siatki danych. Choć na pierwszy rzut oka pojęcia te mogą wydać się sprzeczne, podstawowa technologia „jezior" danych dopełnia wizję projektowania produktów z danymi. W związku z tym nie ma nic złego w stosowaniu w siatce danych technologii „jeziora" danych.

Inne często zadawane pytanie dotyczy tego, czy każda domena cieszy się autonomią w wyborze własnego nowoczesnego stosu danych. Entuzjaści szybko posłużą się porównaniem architektur mikrousług i siatki danych, a ponadto będą twierdzić, że pełna autonomia umożliwia w obrębie każdej domeny dokonywanie własnych, optymalnych wyborów dotyczących technologii. Wiele firm ponosi jednak porażkę, ponieważ zespoły domenowe podejmują sprzeczne decyzje. Wyobraź sobie, że dysponujesz 25 „autonomicznymi" zespołami, z których każdy implementuje swoje architektury produktów z danymi z wykorzystaniem własnych narzędzi i metod postępowania. Jeśli wszystkie narzędzia są różne, w jaki sposób będziesz gromadzić w ramach procesu zarządzania danymi podstawowymi informacje o pochodzeniu danych, wskaźniki pomiarowe jakości danych, statystyki monitorowania oraz wyniki? Jak możesz zapewnić możliwość współdziałania platform, gdy każdy zespół korzysta z własnego standardu? Na pytania te odpowiem na końcu tego rozdziału, ale godne uwagi jest to, że tożsamość, nadzór i możliwość współdziałania to fundamenty każdego wzorca stowarzyszonego. Jeśli tylko poświęcisz dowolny z tych filarów, szybko pozostaniesz z wieloma kompleksowymi rozwiązaniami, które z punktu widzenia architektury staną się kosztowne i trudne do nadzorowania. Autonomia zaczyna się od architektury przedsiębiorstwa na poziomie centralnym i wymaga standaryzacji. Elastyczność odnosi się do luźnego sprzężenia i eliminowania zależności. Autonomia nie powinna zakończyć się chaosem organizacyjnym, niekontrolowanym wzrostem liczby technologii, nagłym zwiększeniem kosztów ani kombinacją wszystkich tych elementów.

Rzeczywisty przykład

W tym podrozdziale przedstawię rzeczywisty przykład tworzenia architektury produktów z danymi. Przykład nie będzie zbyt złożony, choć zawiera zasadnicze składniki na potrzeby dalszego omówienia i oceny. Przyjmijmy, że architektura produktów z danymi jest projektowana dla firmy, w której w przybliżeniu połowa domen to z założenia domeny operacyjne i transakcyjne. Systemy te, które stanowią punkt początkowy operacji pozyskiwania danych, oznaczono jako „złote źródła". Pozostałe domeny są ukierunkowane na konsumentów. Dane muszą być im udostępniane.

Aby zaznajomić się z rozszerzonym omówieniem tego przykładu z uwzględnieniem zrzutów ekranu i instrukcji przedstawianych kroku po kroku, przeczytaj mój wpis na blogu *Using DBT for building a Medallion Lakehouse architecture* (*https://piethein.medium.com/using-dbt-for-building-a-medallion-lakehouse-architecture-azure-↪databricks-delta-dbt-31a31fc9bd0*).

Jeśli zlecono by mi zadanie polegające na zaprojektowaniu dla tej firmy architektury produktów z danymi dla platformy Azure, być może zaproponowałbym rozwiązanie pokazane na rysunku 4.7. Moja początkowa uwaga dotyczyłaby dostarczenia strefy docelowej danych (*https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/scenarios/cloud-scale-analytics/architectures/data-↪landing-zone*) oraz kilku grup zasobów ze standaryzowanym zestawem usług dla mojej pierwszej domeny, takich jak usługa ADF (*Azure Data Factory*), usługa ALDS (*Azure Data Lake Storage*) i usługa Azure Databricks. Dalej zgromadziłbym dane za pomocą takiej usługi jak ADF (*https://learn.microsoft.com/en-us/azure/data-factory/*). W miarę możliwości skorzystałbym z wcześniej przygotowanych konektorów. Alternatywnie poprosiłbym właścicieli aplikacji o wyeksportowanie i tymczasowe umieszczenie ich danych w pośrednich lokalizacjach. Mogłyby to być na przykład konta usługi Blob Storage (*https://learn.microsoft.com/en-us/azure/storage/blobs/*), z których dane będą pobierane w celu późniejszego przetwarzania.

Do przechowywania danych w swojej architekturze produktów z danymi użyłbym usługi ADLS (*https://learn.microsoft.com/en-us/azure/storage/blobs/data-lake-storage-introduction*). W typowym wzorcu projektowym służącym do grupowania danych stosuje się następujące trzy kontenery[15]:

Bronze

Kontener nieprzetworzonych danych, które mogą mieć wiele formatów i struktur.

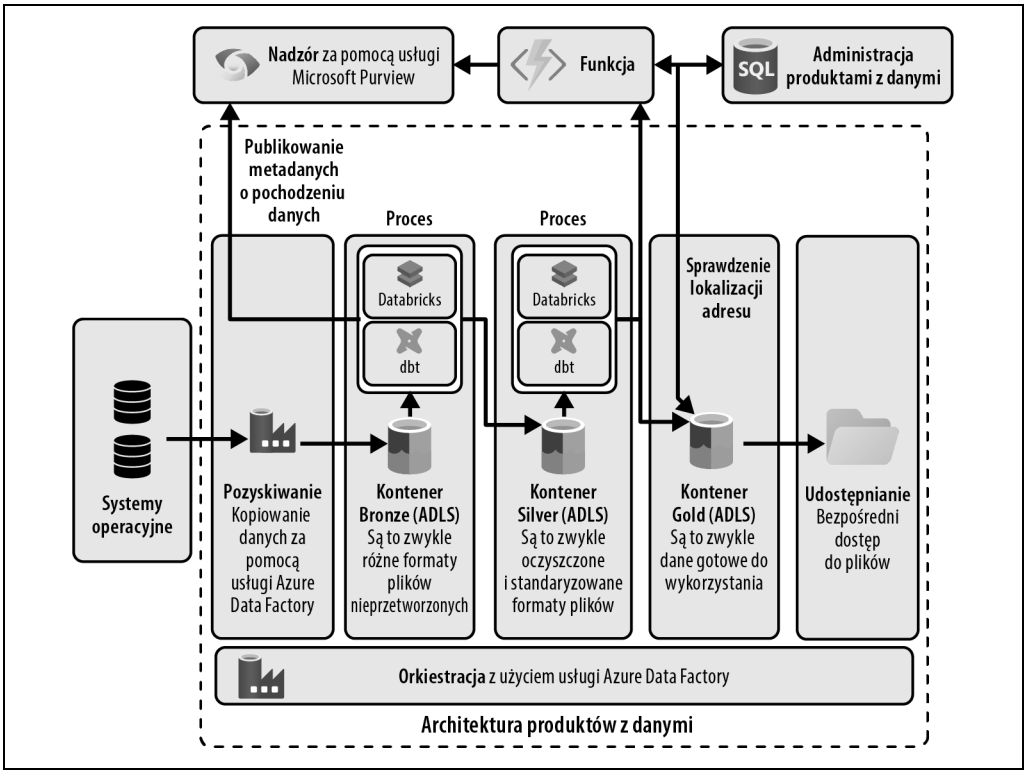
Silver

Kontener filtrowanych i oczyszczonych danych przechowywanych za pomocą standaryzowanego formatu plików z danymi ukierunkowanymi na kolumny.

Gold

Kontener danych zoptymalizowanych pod kątem odczytu, które są gotowe do wykorzystania przez inne domeny. Niektóre firmy określają te zoptymalizowane z myślą o odczycie lub przyjazne dla konsumentów zbiory danych mianem produktów z danymi.

[15] Jeśli planujesz skonfigurować „jezioro" danych, strona internetowa *Hitchhiker's Guide to the Datalake* (*https://github.com/rukmanigopalan/adlsguidancedoc*) stanowi znakomite źródło informacji z wytycznymi i najlepszymi praktykami.



Rysunek 4.7. Prosta architektura produktów z danymi korzystająca z projektu technologii lakehouse. Zawiera ona usługi służące do pozyskiwania danych, tworzenia produktów z danymi i nadzoru nad danymi

Po skonfigurowaniu swojego „jeziora" danych kontynuowałbym działania, tworząc swój pierwszy potok danych. Najpierw użyłbym operacji kopiowania (*https://learn.microsoft.com/en-us/azure/data-factory/copy-activity-overview*) w celu skopiowania danych do warstwy Bronze. Dane w obrębie tej warstwy są przechowywane w formacie Parquet, ponieważ nie jest wymagane żadne wersjonowanie. Na potrzeby historyzacji danych używany jest schemat partycjonowania *RRRRMMDD*. Starsze wersje danych utrzymuje się do celów związanych z debugowaniem lub analizą ad hoc.

Następnie dodałbym kolejne działanie w ramach potoku, aby zostały aktywowane notatniki, a usłudze Databricks[16] zostały przekazane parametry (*https://learn.microsoft.com/en-us/azure/data-factory/how-to-expression-language-functions*). Usługa ta użyje dynamicznego notatnika do sprawdzania poprawności danych, a następnie usunie wszystkie istniejące schematy i utworzy nowe za pomocą lokalizacji partycji *RRRRMMDD*.

[16] W ramach projektu OpenLineage firma Microsoft opracowała rozwiązanie w postaci akceleratora. Ten konektor *open source* przenosi metadane informacji o pochodzeniu do narzędzia Microsoft Purview z operacji platformy Spark realizowanych w usłudze Azure Databricks. Umożliwia to wyświetlenie grafu informacji o pochodzeniu na poziomie tabel. Projekt jest udostępniany w serwisie GitHub (*https://github.com/microsoft/Purview-ADB-Lineage-Solution-Accelerator*).

W przypadku warstwy Silver preferowany przeze mnie format plików to Delta, który zapewnia dużą wydajność i ochronę przed uszkodzeniem. Retencja (*https://docs.databricks.com/en/delta/history.html*) jest możliwa w sytuacjach, gdy musisz cofnąć zmiany i przywrócić poprzednią wersję danych. Dla każdej operacji modyfikującej dane usługa Databricks tworzy nową wersję tabeli, przechowując domyślnie historię z okresu 30 dni. W przypadku wybierania danych z warstwy Bronze i przekształcania ich za pomocą projektu tabel wolno zmieniających się wymiarów (typ 2) preferuję usługę dbt. Przy przenoszeniu danych do warstwy Silver stosuję ograniczone transformacje, wybierając tylko dane funkcjonalne. Przed wybraniem jakichkolwiek danych wykonuję zestaw testów, by się upewnić, że dane są wysokiej jakości i spełniają standardy integralności. Oto ostatnia uwaga dotycząca tej warstwy: jeśli dysponuję różnymi źródłami, traktuję je jako osobne kanały danych. Integrowaniem i łączeniem wszystkich danych zajmują się następne warstwy.

Aby przenieść dane do warstwy Gold, gdzie są one integrowane i przygotowywane dla konsumentów, muszę utworzyć logikę biznesową odwzorowującą wszystkie obiekty na zbiory danych przyjazne dla użytkowników i możliwe do wykorzystania. Ponownie używam w tym celu szablonów usługi dbt, tworząc szablon i model YML (*https://docs.getdbt.com/reference/model-properties*) dla każdej logicznej jednostki danych. Każdy szablon określa operacje wybierania, łączenia i transformacji, które trzeba zastosować, a ponadto dokonuje zapisu w unikalnym folderze (np. z danymi klientów, danymi zamówień, danymi sprzedaży itp.). W ramach każdego wyniku dane są w logiczny sposób klastrowane na podstawie danej tematyki. Dane są materializowane, a formatem plików ponownie jest Delta.

Na koniec zapisałbym i zaplanowałbym swój potok w obrębie usługi ADF. W przypadku wszystkich poszczególnych kroków dodałbym dodatkowe kroki umożliwiające przechwytywanie daty i godziny ładowania, identyfikatora procesu, liczby przetworzonych rekordów, komunikatów o błędach lub powodzeniu itp.

Dla każdej dodatkowej domeny powtórzyłbym wszystkie te kroki. A zatem każda domena będzie przechowywać swoje produkty z danymi wewnątrz własnej architektury produktów z danymi, a także będzie zarządzać własnym potokiem danych i używać tych samych warstw do budowania produktów z danymi. Po uzyskaniu możliwości zrealizowania tego w kontrolowany sposób na dużą skalę zacząłbym dodawać do tego projektu warianty. Przykładowo mogę dysponować kilkoma portami wyjściowymi korzystającymi z tych samych danych semantycznych na potrzeby różnych reprezentacji przeznaczonych dla różnych klientów.

Z myślą o komponentach architektury powiązanych z nadzorem, samoobsługą i bezpieczeństwem zaproponowałbym zastosowanie kombinacji bazy danych Azure SQL (*https://azure.microsoft.com/en-us/products/azure-sql/database/*), usługi Azure Functions (*https://azure.microsoft.com/en-us/products/functions/*) i jednolitej usługi nadzoru nad danymi Microsoft Purview (*https://azure.microsoft.com/en-us/products/functions/*). Komponenty te mogą być wdrażane w strefie docelowej zarządzania danymi oraz używane do katalogowania i kontrolowania wszystkich produktów z danymi.

W celu opublikowania produktów z danymi skierowałbym do wszystkich domen prośbę o opublikowanie ich produktów za pomocą funkcji deklaratywnej jako części ich potoków procesu integracji ciągłej i dostarczania ciągłego. Funkcja ta mogłaby na przykład uzyskać z warstwy Gold

specyfikacje modelu YML usługi dbt jako dane wejściowe i przechowywać je w bazie danych SQL. W dalszej kolejności funkcja weryfikowałaby to, czy zapewnione informacje o schemacie i lokalizacji reprezentują identyfikator URI (*Universal Resource Identifier*) oraz strukturę produktu z danymi. Jeśli tak, funkcja utworzy kolejne wywołanie, aby opublikować wszystkie informacje w katalogu. To ostatnie wywołanie sprawi, że dane staną się publicznie dostępne dla innych konsumentów.

Aby zapewnić swoim konsumentom dostęp do danych, zaproponowałbym zastosowanie samoobsługowego wykrywania danych i uzyskiwania do nich dostępu (*https://learn.microsoft.com/en-us/purview/concept-self-service-data-access-policy*). Spowoduje to utworzenie w tle list kontroli dostępu ACL (*Access Control List*; *https://learn.microsoft.com/en-us/azure/storage/blobs/data-lake-storage-↪access-control-model*) dla poszczególnych produktów z danymi. Listy ACL nie są najlepsze w przypadku dokładnego filtrowania na poziomie kolumn lub wierszy, ale zupełnie wystarczające na potrzeby tego rzeczywistego przykładu. Bardziej złożone scenariusze omówiono w rozdziale 8.

Do celów edukacyjnych prezentowany przykład uprościłem. W rzeczywistości zarządzanie produktami z danymi jest bardziej złożone, gdyż pod uwagę trzeba brać takie czynniki jak bezpieczeństwo, profilowanie, złożone wyszukiwania, pozyskiwanie danych i udostępnianie wariantów oraz różne scenariusze rozszerzające. Przykładowo mogą istnieć konkretne ograniczenia dotyczące archiwizowania danych, tokenizacji lub uzyskiwania dostępu do danych w określonym przedziale czasu.

Podsumujmy szybko kluczowe kwestie zawarte w tym przykładzie. Po pierwsze, w skrócie omówiłem podstawy tworzenia produktów z danymi. Domena stała się odpowiedzialna za cykl istnienia danych podlegających jej kontroli, a także za transformowanie danych z postaci nieprzetworzonej, uzyskanej przez aplikacje, do postaci odpowiedniej do wykorzystania przez zewnętrznych uczestników. Po drugie, architekturę można z łatwością skalować przez dodanie kolejnych domen z dodatkowymi architekturami produktów z danymi. Po trzecie, uwzględniłem kilka zasad siatki danych, takich jak własność danych ukierunkowana na domenę, postrzeganie danych jako produktu, używanie samoobsługowych platform danych oraz zautomatyzowany nadzór stowarzyszony. W następnym punkcie swoją uwagę zwrócimy na kilka kwestii, które trzeba uwzględnić przy skalowaniu w górę.

Dopasowywanie do kont magazynu

Ważne jest, by zwrócić uwagę na kwestię dopasowania danych produktów i kont magazynu. W przykładowym projekcie z poprzedniego punktu zdecydowałem się udostępnić architekturę produktów z danymi wielu zbiorom danych tych produktów z pojedynczej domeny. Alternatywną opcją byłoby dopasowanie na potrzeby przetwarzania danych każdego zbioru danych do wyznaczonego konta magazynu i puli platformy Spark. Taka konfiguracja spowoduje powiązanie ze sobą danych, kodu, metadanych oraz niezbędnej infrastruktury, a ponadto będzie bliższa koncepcji produktu z danymi w przypadku siatki danych. Upraszcza to zarządzanie dostępem i umożliwia automatyzację, dlatego możesz dynamicznie zapewniać klastry przetwarzające i konta magazynu. Jeśli na przykład Twoja warstwa Silver jest tymczasowa, możesz ją usunąć, gdy nie będzie już potrzebna. Z kolei silne połączenie produktów z danymi z infrastrukturą powoduje zwiększenie w znacznym stopniu jej obciążenia oraz liczby potoków, którymi musisz zarządzać. Może to na przykład

skutkować dużą liczbą uprawnień oraz punktów końcowych usługi i sieci, którymi będzie trzeba zarządzać i które będzie trzeba zabezpieczać, skanować i monitorować. Może to też spowodować wygenerowanie złożonej sieci kont magazynu, gdyż produkty z danymi są często uzyskiwane z użyciem tych samych danych podstawowych. Z tego powodu będzie trzeba rozważyć, jak bardzo Twoje produkty z danymi mają zostać dopasowane do infrastruktury.

Co można powiedzieć o sytuacji, gdy wiele domen korzysta z tego samego konta magazynu? To również jest możliwe, ale do tego, żeby wiele domen współużytkowało podstawową infrastrukturę magazynów, wymagana jest inna struktura kontenerów i folderów. Jest to wzorzec, z którym często spotykam się w mniejszych firmach, ponieważ centralnie zarządzany magazyn zmniejsza związane z tym obciążenie oraz ogranicza inne problemy powszechnie występujące w przypadku przenoszenia i zabezpieczania danych. Z drugiej strony wzorzec ten wymaga, aby wszystkie domeny korzystały z tych samych limitów i elementów konfiguracji, takich jak listy ACL i reguły adresów IP. Prawdopodobnie nie jest to też najlepsze rozwiązanie, jeśli wymagasz funkcji nadmiarowości geograficznej oraz elastycznej wydajności[17]. Ponownie będzie trzeba rozważyć kompromisy.

Alternatywnie możesz ustalić stan równowagi między lokalnie i centralnie dopasowywanymi danymi przez zaimplementowanie metody hybrydowej, polegającej na użyciu zarówno lokalnych kont domenowych, jak i kont centralnego magazynu. W tym przypadku każda domena używa wewnętrznego, wydzielonego konta magazynu, w obrębie którego realizowana jest całość przetwarzania dotyczącego domeny (np. w warstwach Bronze i Silver). Takie konto przewidziano wyłącznie do użytku wewnętrznego i nie będzie ono udostępniane żadnym innym domenom. Obok tych lokalnych kont domenowych magazynu znajduje się centralnie współużytkowane konto magazynu, które ma służyć do dystrybucji ostatecznych danych produktów (utworzonych w warstwie Gold) do innych domen. Kontem tym często zarządza centralny dział. Powodem zastosowania takiego rozwiązania jest chęć zapewnienia domenom elastyczności przy jednoczesnym utrzymaniu nadzoru nad wszystkimi operacjami wymiany danych produktów z danymi.

Dopasowywanie do potoków danych

Kolejną ważną kwestią jest dopasowywanie potoków danych oraz produktów z danymi. Jak już wcześniej się dowiedziałeś, potok danych jest zestawem kroków związanych z przenoszeniem i przekształcaniem danych. Tak jak struktura pozyskiwania jest zależna od metadanych, tak potok korzysta przeważnie z metadanych i kodu, aby określić, jakie operacje mają zostać wykonane względem jakich danych. W poprzednim przykładzie zastosowano ten sam potok dla wszystkich produktów z danymi, ale część osób zajmujących się danymi woli używać dla każdego produktu wydzielonego potoku danych. A zatem pytanie brzmi następująco: Jaka metoda jest lepsza w Twoim przypadku?

Aby udzielić odpowiedzi na to pytanie, weź pod uwagę szczegółowość Twoich produktów z danymi i odpowiadające im potoki danych. Przyjmijmy, że w ramach procesu wsadowego przetwarzasz dane z jednej aplikacji i na ich podstawie tworzysz kilka bardziej ogólnych produktów z danymi

[17] Nadmiarowość geograficzną osiąga się dzięki dystrybucji kluczowych komponentów lub infrastruktur (np. serwerów) między wieloma centrami danych znajdującymi się w różnych lokalizacjach geograficznych. Przykładowo w przypadku magazynu w chmurze możesz wybierać między tradycyjnymi dyskami twardymi a szybszymi dyskami SSD (*Solid State Drive*).

(zbiory danych), z których każdy korzysta z własnego potoku danych. Prawdopodobnie wiele potoków będzie przetwarzać ten sam podstawowy system źródłowy, ponieważ dane wejściowe przeznaczone dla produktów z danymi często się pokrywają. Oznacza to, że niektóre kroki przetwarzania mogą być powtarzane, co utrudnia obsługę. Ponadto przenoszenie określonych składników danych za pośrednictwem różnych potoków w innych odstępach czasu powoduje problemy z integralnością, co sprawia, że później konsumentom trudniej będzie połączyć dane.

Utrudnieniem jest również przeprowadzenie zbyt szczegółowej konfiguracji swoich produktów z danymi i potoków danych. Jeśli podstawowy system źródłowy generuje setki niewielkich produktów z danym, z których każdy dysponuje własnym potokiem danych, trudno będzie diagnozować problemy i kontrolować zależności: można zaobserwować złożoną sieć powtarzających się kroków przetwarzania. Poza tym utrudnione będzie korzystanie z danych przez konsumentów, gdyż będą musieli integrować wiele małych zbiorów danych. Dużym problemem jest tutaj również integralność referencyjna, ponieważ oczekuje się, że wiele potoków będzie aktywowanych w trochę innych odstępach czasu.

Podsumowując, zastosowanie wydzielonego potoku dla każdego produktu z danymi zwykle nie jest idealnym rozwiązaniem. Zalecaną najlepszą praktyką jest utworzenie z użyciem odpowiedniego zestawu kroków jednego potoku, który będzie pobierał stan całej aplikacji, kopiował go, a następnie przekształcał dane do postaci przyjaznych dla użytkownika zbiorów danych.

Możliwości udostępniania danych

Zajmijmy się stroną udostępniającą architektury produktów z danymi i sprawdźmy, jakie musimy tam zapewnić komponenty i usługi. Aby udostępniać dane z portów wyjściowych[18], trzeba uwzględnić różne wymiary przetwarzania danych, które mają wpływ na to, jak inżynierowie opracowują architektury produktów z danymi. Projekt takiej architektury w dużej mierze zależy od wariantów zastosowań i wymagań domen będących konsumentami. W wielu sytuacjach przetwarzasz dane, w przypadku których czas nie jest kluczową kwestią. Niekiedy na uzyskanie odpowiedzi na pytanie lub zrealizowanie wymagania biznesowego można czekać kilka godzin lub nawet kilka dni. Zdarzają się jednak też zastosowania, w przypadku których dane muszą być dostarczone w ciągu kilku sekund.

Zarządzania produktami z danymi nie należy mylić z tworzeniem wartości danych (np. za pomocą aplikacji służących do analityki biznesowej i uczenia maszynowego). Pomimo że oba te pojęcia nakładają się na siebie, konsumenci przeważnie używają danych w bardzo różny sposób. A zatem procesem tworzenia produktów z danymi trzeba zarządzać niezależnie od procesu tworzenia wartości danych.

W ostatnich latach bardzo zwiększyła się liczba różnych dostępnych usług baz danych i magazynu danych. Tradycyjnie systemy transakcyjne i operacyjne projektuje się z myślą o dużej integralności, stąd też zastosowanie relacyjnej bazy danych zgodnej z modelem ACID. Rozwinął się

[18] W przypadku siatki danych zaleceniem jest współużytkowanie produktów danych za pomocą interfejsów o wysokim stopniu standaryzacji, które zapewniają dostęp do danych w trybie tylko do odczytu i są zoptymalizowane pod kątem tej operacji. Interfejsy te są też nazywane *portami wyjściowymi*.

jednak także trend opartych na tym modelu baz danych pozbawionych schematu (np. bazy dokumentów lub magazyny z parami klucz-wartość), ponieważ ograniczają one kontrolę nad integralnością na rzecz większej wydajności, a ponadto oferują bardziej elastyczne typy danych (np. typy danych pochodzących z urządzeń przenośnych, gier, kanałów społecznościowych, czujników itp.). Wielkość danych może mieć znaczenie: niektóre aplikacje przechowują względnie małe ilości danych, natomiast inne mogą korzystać z terabajtów danych.

Dlaczego istnieje tak wiele różnych baz danych?

Przy wyborze bazy danych pod uwagę należy wziąć wiele czynników i kompromisów. Oprócz struktury danych konieczna będzie ocena Twoich wymagań związanych ze spójnością, dostępnością, odpornością na partycjonowanie oraz buforowaniem i indeksowaniem pod kątem lepszej wydajności. Wyróżnić można szereg metod przechowywania danych i uzyskiwania ich: niewielkie segmenty, duże segmenty, sortowane segmenty itd. Pojawiają się kompromisy związane z możliwością dystrybucji i spójnością, które mogą mieć wpływ na wydajność, jak również takie opcje jak ciągłe monitorowanie i analityka. I wreszcie — należy rozważyć wymagania o charakterze niefunkcjonalnym, takie jak uzależnienie od konkretnego dostawcy, pomoc techniczną, zgodność i języki zapytań. Wniosek z tego taki, że żadna baza danych nie może jednocześnie górować we wszystkich wymiarach. Wybierz rozwiązanie, które najlepiej pasuje do Twoich wymagań.

Korzyścią wynikającą z użycia architektury produktów z danymi jest to, że możesz dodać wiele projektów dla konkretnych wzorców operacji odczytu powiązanych z różnymi zastosowaniami[19]. Masz możliwość duplikowania danych w celu obsługi różnych struktur danych, szybkości działań i ilości danych albo też obsługiwania różnych reprezentacji tych samych danych. Na potrzeby zastosowań analitycznych, przygotowawczych lub doraźnych mógłbyś zaoferować punkt końcowy oparty na magazynie plików. Punkt końcowy oparty na relacyjnej bazie danych może ułatwić bardziej złożone i nieprzewidywalne zapytania. Punkt końcowy kolumnowej bazy danych może być odpowiedni w przypadku konsumentów danych, którzy wymagają intensywnego agregowania. Z kolei punkt końcowy zależny od bazy danych dokumentów będzie właściwy dla konsumentów danych wymagających większego tempa działań i formatu danych z częściową strukturą (JSON). Mógłbyś nawet utworzyć za pomocą grafowej bazy danych model swoich produktów z danymi, który będzie zawierał węzły i relacje.

Jeśli zapewniasz różne punkty końcowe lub porty wyjściowe, wszystkie muszą być zarządzane w ramach tego samego „parasola" nadzoru nad danymi. Odpowiedzialność tkwi po stronie tej samej domeny udostępniającej. Zasady dotyczące danych zoptymalizowanych pod kątem odczytu, które mogą być ponownie używane, obowiązują również dla wszystkich wariantów produktów z danymi. Dodatkowo oczekuje się, że kontekst i semantyka będą identyczne. Przykładowo pojęcia „klient" i „imię klienta" powinny być spójne dla wszystkich udostępnianych danych. Identyczny język

[19] Gdy do dystrybucji danych używa się hurtowni danych dla przedsiębiorstw, często trudno uprościć różne postaci i wymiary danych. Tego typu hurtownie są przeważnie projektuje się z wykorzystaniem pojedynczego typu technologii (mechanizm bazy danych systemu RDBMS), dlatego nie pozostawiają one zbyt wiele miejsca na warianty.

wszechobecny trzeba stosować do każdego produktu z danymi, który należy do tego samego dostawcy danych. W rozdziale 7. przyjrzymy się temu, jak zapewnia się spójność semantyczną w obrębie wszystkich punktów końcowych oraz innych wzorców integracji.

Usługi udostępniające dane

Gdy bliżej przyjrzysz się interakcjom użytkowników i aplikacji z architekturami produktów z danymi, zwykle stwierdzisz, że można zapewnić dużą różnorodność wzorców dostępu do danych. Mogą one obejmować tworzenie zapytań doraźnych, bezpośrednie raportowanie, tworzenie warstw lub kostek semantycznych, udostępnianie interfejsu ODBC (*Open Database Connectivity*) innym aplikacjom, przetwarzanie z użyciem narzędzi procesu ETL, oczyszczanie i ujednolicanie danych lub wykonywanie operacji przez danologów. Aby zapewnić obsługę tego wszystkiego, przede wszystkim musisz zadbać o to, żeby Twoja architektura produktów z danymi miała wystarczającą wydajność i odpowiednie funkcje zabezpieczeń.

Wskazane jest, by w przypadku wszystkich usług będących konsumentami danych zachować spójność w zakresie dostępu do danych i zarządzania cyklem istnienia. Właśnie to umożliwiają kontrakty danych, omówione w rozdziale 8.

Rozproszone systemy plików, takie jak magazyn obiektów w chmurze, okazują się efektywne kosztowo, gdy trzeba przechowywać duże ilości danych. Generalnie nie są one jednak wystarczająco szybkie z punktu widzenia zapytań doraźnych i interaktywnych. Wielu dostawców zdało sobie sprawę z tego problemu i żeby go rozwiązać, zaoferowało mechanizmy zapytań SQL lub uproszczone warstwy wizualizacji. Korzyścią wynikającą z możliwości bezpośredniego wykonywania zapytań względem produktów z danymi jest to, że dane nie muszą być duplikowane, co obniża koszt architektury. Narzędzia te zapewniają także szczegółowy dostęp do danych i sprawiają, że magazyn danych operacyjnych staje się przestarzały. Dane znajdujące się w architekturze produktów w przypadku raportów operacyjnych i analizy ad hoc uzyskują status kandydata. Opcje wirtualizacji danych to dobre rozwiązania, ale pasujące wyłącznie do prostych wariantów raportowania. Jeśli jest to niewystarczające, rozważ takie alternatywne opcje jak duplikowanie danych.

Usługa modyfikowania plików

Choć mowa jest o powielaniu danych i korzystaniu z nich, w części domen pojawiają się zastosowania lub aplikacje, które jako dane wejściowe akceptują tylko „płaskie” pliki (np. pliki CSV). W takich sytuacjach rozważ zbudowanie usługi modyfikującej pliki automatycznie maskującej lub pomijającej dane poufne, których konsumenci nie mogą zobaczyć. Podobnie do wszystkich innych punktów końcowych korzystających z danych taka usługa musi zostać podłączona do centralnego modelu bezpieczeństwa, który wymaga, by wszystkie operacje używające danych przestrzegały dotyczących ich kontraktów (zajrzyj do punktu „Kontrakty danych” w rozdziale 8.). Model ten określa również, że filtry muszą być uwzględniane automatycznie, a ponadto że zawsze opierają się na klasyfikacjach i atrybutach metadanych. Bardziej szczegółowo zajmiemy się tym w rozdziale 8.

Usługa dezidentyfikacji

Gdy produktów z danymi używa się bezpośrednio do eksploracji danych, a także na potrzeby danologii, uczenia maszynowego i współużytkowania danych z zewnętrznymi dostawcami, ważna jest ochrona danych poufnych przed konsumentami. Dzięki takim metodom zabezpieczeń jak tokenizacja, użycie funkcji mieszającej, kodowanie i szyfrowanie masz możliwość wykorzystania reguł i klasyfikacji do ochrony swoich danych. Zależnie od tego, w jaki sposób przygotowujesz architekturę produktów z danymi, możesz zastosować następujące metody:

- Ochrona danych w środowisku wykonawczym w trakcie korzystania z nich. Technika ta chroni dane poufne bez dokonywania żadnych zmian w przechowywanych danych, lecz tylko wtedy, gdy dane są uzyskiwane za pomocą zapytań. Przykładowo platforma Databricks korzysta z opcji określanej mianem **funkcji widoku dynamicznego** (*https://docs.databricks.com/en/data-governance/table-acls/object-privileges.html#dynamic-view-functions*), aby ukrywać lub maskować dane przy próbie uzyskania do nich dostępu.
- Zaciemnianie lub maskowanie danych poufnych (np. za pomocą tokenizacji), zanim w ogóle trafią do architektury produktów z danymi. Dopasowywanie po dokonanej dezidentyfikacji ma następnie miejsce w fazie korzystania z danych.
- Duplikowanie i przetwarzanie danych dla każdego projektu. Zakres ochrony możesz dostosować przez określenie, z jakich korzystasz klasyfikacji oraz reguł maskowania i filtrowania.

Bezpieczeństwo danych zależy od nadzoru nad nimi. Te dwa zagadnienia szerzej omówiono w rozdziale 8.

Orkiestracja rozproszona

Ostatni aspekt wymagający uwzględnienia kwestii projektowych i standaryzacji dotyczy wspierania zespołów w trakcie implementowania, testowania i orkiestracji ich potoków danych. Budowanie i automatyzowanie tych potoków to proces iteracyjny obejmujący zestaw takich kroków jak wyodrębnianie, przygotowywanie i przekształcanie danych. Powinno się umożliwić zespołom testowanie oraz monitorowanie jakości wszystkich ich potoków danych i artefaktów, a także należy wspierać je we wprowadzaniu zmian w kodzie podczas trwania procesu wdrażania. Taka kompleksowa metoda dostarczania w dużym stopniu przypomina metodykę DataOps w tym, że towarzyszy jej mnóstwo automatyzacji, technicznych działań, przepływów, wzorców architekturalnych oraz narzędzi.

Zalecam, by przy standaryzacji narzędzi i najlepszych praktyk firmy powoływały centralny zespół lub zespół platformy wspierający inne zespoły przez umożliwianie realizacji takich zadań jak planowanie, przechowywanie metadanych, rejestrowanie wskaźników pomiarowych itp. Taki zespół powinien być też odpowiedzialny za nadzorowanie kompleksowego monitorowania, a ponadto powinien zainterweniować, gdy potoki podzielono na zbyt długie. W celu uogólnienia złożoności zależności oraz różnic interwałów czasowych stosowanych przez różnych dostawców danych zespół ten może również przygotować dodatkową strukturę przetwarzającą, która może na przykład informować konsumentów, gdy możliwe jest połączenie wielu źródeł z relacjami zależności.

Inteligentne usługi konsumentów

Korzystając z tych wszystkich różnych możliwości, mógłbyś też rozszerzyć architekturę produktów z danymi o warstwę umożliwiającą zastosowanie w momencie użycia danych takich inteligentnych transformacji danych jak automatyczne profilowanie, filtrowanie, dopasowywanie schematów, łączenie itp. Pozwoliłoby to konsumentom określić dane, których potrzebują, a także otrzymać je przygotowane w postaci, jakiej oczekują. Ten typ inteligentnej usługi zwykle stosuje się też do wdrażania warstwy semantycznej, która ukrywa złożoną logikę, a następnie udostępnia użytkownikom biznesowym przyjazne dla nich reprezentacje. Usługa w dużym stopniu korzysta z metadanych należących do wielu innych usług, które są ściśle powiązane z nadzorem nad danymi.

Wizjonerskim podejściem byłoby zasilenie posiadanej struktury nadchodzącymi technologiami, takimi jak uczenie maszynowe i grafy wiedzy semantycznej. Choć ten trend jest względnie nowy, w rozdziale 9. prezentuję kilka zaleceń dotyczących przygotowania do implementacji tego rozwiązania.

Kwestie dotyczące bezpośredniego korzystania z danych

Trudną kwestią projektową jest to, czy produkty z danymi powinny mieć możliwość odgrywania roli bezpośrednich źródeł dla domen będących konsumentami, czy domeny te muszą wyodrębniać i duplikować dane. Mowa jest tutaj o trwałych transformacjach danych związanych z procesem, w ramach którego dane są zapisywane i przechowywane w nowej lokalizacji („jezioro” danych lub baza danych) poza granicami architektury produktów z danymi. Oto przykład:

- Jeśli produkt z danymi służy do tworzenia nowych danych, z logicznego punktu widzenia muszą one być przechowywane w nowym miejscu i wymagać innego właściciela.
- Jeżeli celem jest zmniejszenie ogólnego opóźnienia, lepszym rozwiązaniem może być zduplikowanie danych tak, aby były dostępne lokalnie bliżej miejsca docelowego aplikacji będącej konsumentem. Przyjrzymy się temu dokładniej w rozdziale 6., w którym dowiesz się, jak tworzyć w pełni kontrolowane, rozproszone i zmaterializowane widoki.
- Bezpośrednie transformacje danych w czasie rzeczywistym mogą być tak intensywne, że pogarszają komfort pracy użytkowników. Przykładowo zapytania dotyczące produktów z danymi mogą trwać tak długo, że powoduje to ich frustrację. Można to ograniczyć dzięki wyodrębnianiu, restrukturyzowaniu i wstępnemu przetwarzaniu danych lub korzystaniu z szybszego mechanizmu baz danych.
- Gdy aplikacja będąca konsumentem jest tak istotna, że w celu uniknięcia niepowodzenia architektury produktów z danymi niezbędne jest eliminowanie sprzężenia, sensowne może się okazać duplikowanie danych.
- We wszystkich tych przypadkach zalecam implementowanie pewnej formy kontroli nad architekturą dla przedsiębiorstw, która dyktuje to, jaką metodę stosuje się do każdego produktu z danymi. Bądź świadom tego, że zduplikowane dane nie tak łatwo oczyścić, a ponadto może być trudno uzyskać wnioski dotyczące sposobu używania danych. Jeśli dane można skopiować bez ograniczeń lub konieczności zarządzania nimi, istnieje możliwość dalszego ich dystrybuowania. Kolejnym potencjalnym problemem jest zgodność z takimi regulacjami jak ogólne rozporządzenie o ochronie danych lub ustawa CCPA (*California Consumer Privacy Act*). Bardziej szczegółowo jest to omówione w rozdziale 11.

Początek działań

Dla wielu zespołów przejście na proces projektowania produktów z danymi uwzględniający kwestię własności danych może się okazać wyzwaniem kulturowym i technicznym. Przykładowo nie wszystkie zespoły dysponują niezbędną wiedzą, dostrzegają wartość danych lub są skłonne do inwestowania w nie. Jeśli chcesz podążać ścieżką naturalnego rozwoju związanego z projektowaniem produktów z danymi, weź pod uwagę następujące najlepsze praktyki:

- Zacznij od czegoś małego. Wywołaj pewne podekscytowanie i zacznij od jednej lub kilku domen, których zespoły dały się przekonać do koncepcji projektowania produktów z danymi. Zanim zaczniesz skalowanie w górę, pochwal się najpierw sukcesem.
- Utwórz spójną definicję tego, czym jest produkt z danymi dla Twojej firmy. Dopasuj definicję do metamodelu używanego do zarządzania metadanymi.
- Nadzór jest ważny, ale nie spiesz się z tym. Utrzymuj subtelną równowagę, aby motywować do wydajności. Skoncentruj się na zasadniczych elementach, takich jak własność, metadane i standardy współdziałania. Gdy tylko jest to możliwe, dopasowuj do siebie standardy branżowe i najlepsze praktyki.
- Twoje pierwsze produkty z danymi powinny stanowić przykład dla reszty posiadanych domen. Wprowadź w życie kryteria projektowe i akceptowalnej jakości, a ponadto skup się na projekcie zbioru danych. Projekt interfejsu API ukierunkowanego na zasoby to znakomity początek w procesie tworzenia wysokiej jakości produktów z danymi.
- Nie próbuj budować platformy mającej jednocześnie obsługiwać wszystkie zastosowania. Na początek Twoja platforma będzie zwykle obsługiwać tylko jeden typ operacji pozyskiwania i wykorzystywania danych, czyli na przykład przenoszenie danych w trybie wsadowym oraz pliki formatu Parquet używanego na potrzeby konsumentów danych.
- Zezwól na centralizację towarzyszącą decentralizacji. Wdrażanie Twojego początkowego produktu z danymi nie musi od razu zaburzać istniejącej struktury organizacyjnej. Utrzymuj scentralizowanie różnych elementów do czasu, aż inżynierowie nie zostaną dopasowani do domen, albo skorzystaj z wariantu „miękkiego” dopasowywania, aby umożliwić inżynierom rozpoczęcie współpracy z ekspertami z danych dziedzin. Na początek zbiory danych poszczególnych domen mogą być tworzone przy wsparciu centralnego zespołu.

Po rozpoczęciu działań nie zwiększaj ich tempa. Zanim przejdziesz do następnego etapu, ucz się i powtarzaj działania. Zestawienie centralizacji z decentralizacją nie oznacza „wszystko albo nic”. Zamiast tego wiąże się to z ciągłą kalibracją obejmującą niuanse i okresowe oceny.

Podsumowanie

Produkty z danymi to wielka zmiana w sposobie zarządzania zależnościami między domenami a aplikacjami. Każdorazowo w momencie przekraczania granic zagadnienia funkcjonalnego lub ograniczonego kontekstu zespoły powinny postępować zgodnie z jednolitą metodą dystrybucji

danych, czyli udostępniać dane innym zespołom jako produkty z danymi. Taki sposób działania jest ważny, ponieważ mniejsza liczba wspólnych zależności oznacza mniej niezbędnej koordynacji między zespołami.

Na początku tego rozdziału nakreśliłem różnicę między terminami „dane jako produkt" i „produkty z danymi". W ramach podsumowania — dane jako produkt reprezentują sposób myślenia, jaki obowiązuje w przypadku zarządzania danymi na dużą skalę. Dotyczy to budowania zaufania, ustanawiania nadzoru i wprowadzania w życie myślenia kategoriami produktów. Dotyczy to dbania o dane, które uważasz za najbardziej wartościowe, a także wykorzystania zasad w praktyce. Jest to sposób myślenia, jakim muszą się posługiwać Twoje zespoły domenowe. Muszą one być właścicielami swoich danych, eliminować u źródła problemy z jakością danych oraz udostępniać dane w sposób korzystny z punktu widzenia dużej grupy odbiorców. Zamiast zostawiać wszystkie zmiany i integrowanie danych centralnemu zespołowi, Twoje zespoły domenowe powinny umieścić fizyczny model danych w obrębie granic swoich domen. To przeciwieństwo „przerzucania" nieprzetworzonych danych „przez ogrodzenie" i zezwalania konsumentom na to, by próbowali się uporać z problemami dotyczącymi jakości danych i modelowania. Twoje zespoły domenowe powinny modelować i udostępniać dane w sposób zapewniający użytkownikom najwyższy komfort pracy.

Produkt z danymi to zagadnienie, które musisz sobie objaśnić sam. Zalecam na początek, żebyś ustandaryzował sposób, w jaki w swojej firmie rozmawiasz na temat zarządzania produktami z danymi, gdyż mogą się pojawić warianty w ramach używanego języka i definicji. Rozważ zdefiniowanie produktów z danymi jako konstrukcji logicznych, zamiast przyjmować punkt widzenia, w którym koncentrujesz się wyłącznie na fizycznych reprezentacjach lub podstawowej architekturze technologii. W pewnym momencie możesz wprowadzić w firmie model semantyczny, tak aby Twoje produkty z danymi stały się graficznymi reprezentacjami pozwalającymi ustalić relacje z domenami, właścicielami danych, źródłowymi aplikacjami oraz fizycznymi reprezentacjami. Może to brzmieć trochę zbyt pojęciowo, ale korzyści wynikające z zarządzania swoimi produktami z danymi staną się bardziej oczywiste w trakcie lektury rozdziału 9.

W tym miejscu jedno zastrzeżenie dotyczące produktów z danymi. Prawdopodobnie będą one narażone na problemy podobne jak w przypadku architektur mikrousług, w których dane są zbyt szczegółowe i rozdzielone, a ponadto trzeba je uzyskiwać z setek różnych lokalizacji. Rozwiązaniem tych problemów jest ustanowienie dobrych standardów i centralnej jednostki nadzorującej, która sprawuje kontrolę nad wszystkimi działaniami związanymi z tworzeniem produktów z danymi. Konieczne jest określenie standardów współdziałania na potrzeby wymiany danych. Niezbędne jest koordynowanie zmian, a także sposobu, w jaki dane są modelowane i udostępniane. Niestety, działania te nie są proste. Jak się dowiedziałeś, proces projektowania produktów z danymi jest silnie powiązany z hurtownią danych, z którą wiąże się wiele zagadnień, takich jak modelowanie danych (zapewnianie spójności danych, jak również tego, aby mogły być odczytywane, wykrywane, adresowane, godne zaufania i bezpieczne). Niektóre z tych działań mogą być złożone, tak jak omówione w niniejszym rozdziale aspekty historyczne.

Sposób myślenia związany z produktami z danymi oraz z danymi jako produktem odnosi się do *dostarczania danych*. Następny etap to *korzystanie z danych*. Choć określone działania i zagadnienia mogą się pokrywać, to jak się przekonasz w rozdziale 11., kwestie dostarczania danych i korzystania z nich nie są tożsame. Dzięki zarządzaniu danymi jako produktami i zapewnianiu

produktów z danymi w postaci interfejsów domeny eliminuje się zależności i lepiej izoluje aplikacje, w przypadku których istnieją kontrastujące ze sobą zagadnienia. Jest to ważne, gdyż mniejsza liczba współużytkowanych zależności oznacza mniej niezbędnej koordynacji między zespołami.

Architektura produktów z danymi powinna być projektowana w ramach współpracy zespołowej oraz z uwzględnieniem wymagań firmy. Przedstaw zespołom domenowym ogólne plany. Pozwól innym zespołom uczestniczyć w pracach zespołów zajmujących się platformą lub tymczasowo do nich dołączyć. Jeśli na przykład niezbędny jest nowy wzorzec wykorzystania danych, dostarczaj go w sposób zrównoważony w postaci wzorca lub komponentu umożliwiającego wielokrotnie użycie. To samo dotyczy przechwytywania i wprowadzania danych. Podczas zwiększania skali i budowania nowoczesnej platformy danych koniecznych będzie wiele dodatkowych, samoobsługowych komponentów aplikacji. Będą one prawdopodobnie obejmować centralnie zarządzane struktury procesu ETL, usługi przechwytujące dane zmian oraz narzędzia do pozyskiwania danych w czasie rzeczywistym.

I ostatnia, ważna uwaga: jak wspomniano w niniejszym rozdziale, przeznaczeniem produktów z danymi jest zapewnienie dostępności danych. Nie było jeszcze mowy o tym, jak sprawić, żeby dane były użyteczne, a także jak uzyskać z nich prawdziwą wartość. Aspekt ten cechuje się inną dynamiką niż proces tworzenia produktów z danymi. Z omówieniem tego zagadnienia poczekamy do rozdziału 11.

W następnym rozdziale przyjrzymy się architekturze interfejsów API, która umożliwia komunikację między usługami, a ponadto dystrybuowanie mniejszych ilości danych w przypadku zastosowań czasu rzeczywistego z niewielkim opóźnieniem.

Ta książka zapewnia bardzo szczegółowe i solidne podstawy z zakresu zarządzania danymi – obecnie i w przyszłości!

Joe Reis, współautor książki *Inżynieria danych w praktyce. Kluczowe koncepcje i najlepsze technologie*

Datafikacja trwa – i zmienia nasze życie z zawrotną prędkością. Danych jest coraz więcej i są coraz bardziej złożone, a poza kwestiami technicznymi trzeba rozstrzygać mnóstwo dylematów etycznych lub prawnych związanych z prywatnością i bezpieczeństwem. Bez wątpienia w zarządzaniu danymi potrzeba nowej, wyrazistej wizji.

W książce w praktyczny sposób ujęto wiele złożonych zagadnień, różnych technologii, metod biznesowych struktur i wzorców architektury. Przeanalizowano abstrakcyjny poziom strategii danych, kwestie zarządcze i architekturę danych, a następnie wyjaśniono, czym są domeny danych i strefy docelowe. Zaprezentowano kwestie zarządzania systemami źródłowymi, a także aplikacji, opisano praktyczne szczegóły z zakresu data science. Nie zabrakło wartościowych informacji o aspektach istotnych dla konsumentów danych. Autor nie skupia się wyłącznie na teorii. Istotnym atutem książki są jasne wskazówki, w jaki sposób zastosować omawianą wiedzę w praktyce.

Ta książka mówi o skalowaniu i pozostaniu konkurencyjnym. Nie ma na rynku drugiej takiej pozycji!

Ole Olesen-Bagneux, autor książki *The Enterprise Data Catalog*

Zagadnienia:

- trendy w zarządzaniu danymi a aktualne wymagania
- nowe technologie projektowe, w tym siatka danych i Data Fabric
- strefy docelowe danych w chmurze, DDD, projektowanie produktów z danymi
- bezpieczeństwo danych
- zarządzanie samoobsługowymi platformami danych
- rola metadanych

Piethein Strengholt jest głównym inspektorem danych w Microsoft Netherlands. Aktywny bloger regularnie komentujący najnowsze trendy w zarządzaniu danymi. Specjalizuje się w takich zagadnieniach jak siatka danych, nadzorowanie danych i strategie działania na dużą skalę.

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 978-83-289-0546-7
9 788328 905467
Cena: 99,00 zł