O'REILLY®

Helion

Nowoczesne architektury danych

Przewodnik po hurtowni danych, siatce danych oraz Data Fabric i Data Lakehouse

James Serra

Tytuł oryginału: Deciphering Data Architectures: Choosing Between a Modern Data Warehouse, Data Fabric, Data Lakehouse, and Data Mesh

Tłumaczenie: Piotr Pilch

ISBN: 978-83-289-1669-2

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
https://helion.pl/user/opinie/noarda
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 2.

Typy architektur danych

Bezwzględnie ważne jest poświęcenie na samym początku czasu na zaprojektowanie i zbudowanie odpowiedniej architektury danych. Przekonałem się o tym na własnej skórze na początku mojej kariery. Byłem tak bardzo podekscytowany faktem rozpoczęcia procesu tworzenia mojego rozwiązania, że beztrosko podszedłem do kwestii istotnych decyzji dotyczących projektu architektury i tego, jakie produkty mają zostać zastosowane. Po trzech miesiącach realizowania projektu uświadomiłem sobie to, że architektura nie będzie obsługiwać niektórych z wymaganych źródeł danych. W zasadzie konieczne było rozpoczęcie projektu od nowa i przygotowanie kolejnej architektury oraz innych produktów, co oznaczało mnóstwo straconego czasu i pieniędzy. Bez właściwego planowania Twoje rozwiązanie nie zapewni użytkownikom wartości. Poza tym będą oni niezadowoleni z powodu niedotrzymanych terminów, a Twoja firma w coraz większym stopniu ryzykuje pozostawaniem w tyle względem swojej konkurencji.

W trakcie budowania rozwiązania do obsługi danych musisz postępować zgodnie z dobrze przemyślanym planem. Właśnie tutaj do gry wkracza **architektura danych**. Definiuje ona ogólną metodę architekturalną i pojęcia, które są w niej stosowane, a ponadto opisuje zestaw używanych technologii oraz określa przepływ danych, jaki posłuży do zbudowania rozwiązania do przechwytywania danych o pokaźnej wielkości. Decydowanie w kwestii architektury danych może być dużym wyzwaniem, gdyż nie istnieje żadna uniwersalna architektura. Nie masz możliwości przejrzenia książki w celu znalezienia wykazu metod architekturalnych powiązanych z odpowiednimi produktami do użycia. Nie istnieje prosty diagram możliwy do wykorzystania w przypadku drzew decyzyjnych, który doprowadzi Cię do idealnej architektury. Twoja metoda architekturalna oraz używane technologie będą się w znacznym stopniu różnić dla poszczególnych klientów oraz wariantów zastosowania.

Podstawowe typy ogólnych metod architekturalnych oraz zagadnienia są dokładnie tym, co przedstawiłem w tej książce. Choć nie ma w niej żadnego wykazu, będziesz mieć możliwość zorientowania się, o co w tym wszystkim chodzi. Wprawdzie przydatne jest podzielenie architektur danych na typy zależnie od ich właściwości (tak też postąpiłem w książce), jednak nie jest to tożsame z dokonywaniem wyboru spośród zestawu predefiniowanych i uniwersalnych szablonów. Każda architektura danych jest unikalna, a ponadto wymaga spersonalizowanego podejścia w celu spełnienia konkretnych wymagań biznesowych.

Architektura danych odnosi się do ogólnego projektu i organizacji danych w obrębie systemu informatycznego. Predefiniowane szablony architektur danych mogą sprawiać wrażenie prostej metody pozwalającej szybko przygotować nowy system. Tego rodzaju szablony często jednak

nie uwzględniają konkretnych wymagań i ograniczeń systemu, względem którego są stosowane. Może to prowadzić do problemów z jakością danych, wydajnością systemu oraz serwisowaniem. Ponadto wymagania organizacji i systemy danych będą prawdopodobnie zmieniać się z upływem czasu, co będzie wymagać aktualizowania architektury danych i dokonywania w niej korekt. Standaryzowany szablon może nie być wystarczająco elastyczny do tego, aby uwzględniać takie zmiany, a to może powodować pojawianie się w systemie nieefektywności i ograniczeń.

W niniejszym rozdziale znajdziesz skrócony przegląd podstawowych typów, które omówię w książce. Jest to relacyjna hurtownia danych, jezioro danych, nowoczesna hurtownia danych, Data Fabric, Data Lakehouse oraz siatka danych. W dalszej części książki każdemu z nich poświęciłem osobny rozdział zawierający mnóstwo szczegółów.

Ewolucja architektur danych

W **relacyjnej bazie danych** dane są przechowywane w sposób ustrukturyzowany. Za pomocą kluczy między elementami danych definiuje się relacje. Dane są zwykle porządkowane w obrębie tabel, z których każda składa się z wierszy i kolumn. Każdy wiersz reprezentuje pojedynczą instancję danych, natomiast poszczególne kolumny reprezentują konkretne atrybuty danych.

Relacyjne bazy danych zaprojektowano do obsługi danych ustrukturyzowanych. Bazy te zapewniają strukturę służącą do tworzenia, modyfikowania i uzyskiwania danych za pomocą standaryzowanego języka SQL (*Structured Query Language*). Model relacyjny po raz pierwszy został zaproponowany w 1970 r. przez Edgara F. Codda[1]. Od połowy lat 70. model ten stał się dominujący w systemach zarządzania bazami danych. Większość aplikacji operacyjnych musi trwale przechowywać dane, a relacyjna baza danych to narzędzie wybierane w przypadku ich zdecydowanej większości.

W relacyjnych bazach danych, gdzie podstawowe znaczenie odgrywa spójność i integralność danych, są one zwykle porządkowane z użyciem metody *Schema-on-Write*. *Schemat* odnosi się do formalnej struktury definiującej organizację tabel, pól, typów danych i ograniczeń oraz relacji między nimi. Schemat pełni rolę planu w zakresie przechowywania danych oraz zarządzania nimi, a ponadto zapewnia spójność, integralność i efektywną organizację wewnątrz bazy danych. Relacyjne bazy danych (oraz relacyjne hurtownie danych) wymagają niewielkiej ilości wstępnych działań, zanim będą mogły zostać umieszczone w nich dane. Musisz utworzyć bazę danych oraz jej tabele, pola i schemat, a następnie kod służący do przeniesienia danych do bazy. W przypadku metody *Schema-on-Write* schemat danych jest definiowany i wymuszany w momencie zapisywania danych w bazie lub pozyskiwania ich. Zanim dane będą mogły zostać zapisane, muszą być zgodne z predefiniowanym schematem obejmującym typy danych, ograniczenia i relacje.

Dla porównania w przypadku metody *Schema-on-Read* schemat stosuje się w chwili odczytu danych lub uzyskiwania do nich dostępu, a nie wtedy, gdy są one zapisywane. Dane mogą być pozyskiwane do systemu magazynowania bez utrzymywania zgodności z konkretnym schematem, a struktura jest

[1] Dokument *Derivability, Redundancy, and Consistency of Relations Stored in Large Data Banks*, „Communications of the ACM", 13, nr 6 (1970): str. 377 – 387. Dokument autorstwa E. F. Codda jest dostępny na stronie internetowej *A Relational Model of Data for Large Shared Data Banks*.

definiowana tylko wtedy, gdy dane są używane lub są celem zapytania. Takie podejście oferuje większą elastyczność w przypadku danych bez struktury lub mających częściową strukturę, a ponadto jest powszechnie stosowane w jeziorach danych, które omówiłem w dalszej części rozdziału.

Na ogólnym poziomie architektury danych zapewniają strukturę służącą do porządkowania danych i zarządzania nimi w sposób spełniający wymagania organizacji. Obejmuje to definiowanie tego, jak dane są gromadzone, przechowywane, przetwarzane i używane, a także utrzymywanie jakości danych, bezpieczeństwa i prywatności. Choć architektury danych mogą przyjmować wiele różnych form, wśród ich wspólnych elementów należy wymienić następujące:

Magazyn danych
: Wszystkie architektury danych muszą określać metodę przechowywania danych z uwzględnieniem fizycznego nośnika danych (np. dyski twarde lub magazyn w chmurze) oraz struktur danych używanych do ich organizacji.

Przetwarzanie danych
: W przypadku architektur danych niezbędne jest podanie, w jaki sposób dane są przetwarzane, co obejmuje wszelkie transformacje lub obliczenia wykonywane względem danych przed ich zapisaniem lub poddaniu analizie.

Dostęp do danych
: Obowiązkową funkcją architektury danych jest zapewnienie mechanizmów uzyskiwania dostępu do danych z uwzględnieniem interfejsów użytkownika oraz interfejsów API (*Application Program Interface*), które umożliwiają analizowanie danych lub tworzenie dotyczących ich zapytań.

Bezpieczeństwo i prywatność danych
: Wymagane jest, aby architektury danych zawierały mechanizmy zapewniające bezpieczeństwo i prywatność danych, takie jak kontrola dostępu, szyfrowanie i maskowanie danych.

Nadzór nad danymi
: Kolejnym elementem, jaki muszą oferować architektury danych, są struktury przeznaczone do zarządzania danymi, w tym standardy jakości, monitorowanie pochodzenia oraz zasady dotyczące retencji.

Generalnie głównym celem architektury danych jest umożliwienie organizacji efektywnego zarządzania swoimi zasobami danych oraz korzystania z nich, tak aby mogła ona realizować własne cele biznesowe oraz procesy podejmowania decyzji.

Tabela 2.1, w której znajduje się ogólne porównanie właściwości architektur danych omawianych w książce, powinna być na początek odpowiednia do określenia, jaka architektura może być najbardziej odpowiednia w konkretnym wariancie zastosowania.

Tabela 2.1. Porównanie architektur danych

Właściwość	Relacyjna hurtownia danych	Jezioro danych	Nowoczesna hurtownia danych	Data Fabric	Data Lakehouse	Siatka danych
Rok wprowadzenia	1984	2010	2011	2016	2020	2019
Scentralizowana/zdecentralizowana	Scentralizowana	Scentralizowana	Scentralizowana	Scentralizowana	Scentralizowana	Zdecentralizowana
Typ magazynu	Relacyjny	Obiektowy	Relacyjny i obiektowy	Relacyjny i obiektowy	Obiektowy	Domenowy
Typ schematu	Schema-on-Write	Schema-on-Read	Schema-on-Write i Schema-on-Read	Schema-on-Write i Schema-on-Read	Schema-on-Read	Domenowy
Bezpieczeństwo danych	Duże	Od małego do średniego	Od średniego do dużego	Duże	Średnie	Powiązane z domeną
Opóźnienie dostępu do danych	Małe	Duże	Od małego do dużego	Od małego do dużego	Od średniego do dużego	Zależne od domeny
Czas uzyskania efektu	Średni	Krótki	Krótki	Krótki	Krótki	Długi
Całkowity koszt rozwiązania	Duży	Mały	Średni	Od średniego do dużego	Od małego do średniego	Duży
Wspierane warianty zastosowania	Mała ilość	Od małej do średniej ilości	Średnia ilość	Od średniej do dużej ilości	Duża ilość	Duża ilość
Trudność projektowa	Mała	Średnia	Średnia	Średnia	Od średniej do dużej	Duża
Dojrzałość technologii	Duża	Średnia	Od średniej do dużej	Od średniej do dużej	Od średniej do dużej	Mała
Niezbędny zestaw umiejętności pracowników firmy	Mały	Od małego do średniego	Średni	Od średniego do dużego	Od średniego do dużego	Duży

Relacyjna hurtownia danych

Przez kilka dekad relacyjne bazy danych były ostoją magazynowania danych. System Teradata zaprojektowany na uniwersytecie Stanford przez dr. Jacka E. Shemera (który w 1979 r. założył firmę Teradata Corporation) był pierwszą relacyjną hurtownią danych używaną w środowiskach produkcyjnych. W 1983 r. bank Wells Fargo *zainstalował pierwszy system Teradata* i używał go do analizowania danych finansowych.

Gdy firmy zaczęły generować coraz większe, pokaźne ilości danych, coraz trudniejsze stawało się ich przetwarzanie i analizowanie bez znacznego opóźnienia. Ograniczenia relacyjnych baz danych

przyczyniły się do opracowania relacyjnych hurtowni danych[2], które znacznie zyskały na popularności pod koniec lat 80.[3], czyli około 15 lat po tym, jak pojawiły się relacyjne bazy danych. **Relacyjna hurtownia danych** (ang. *relational data warehouse*) to konkretny typ relacyjnej bazy danych zaprojektowany pod kątem przechowywania danych w hurtowni oraz zastosowań z zakresu analityki biznesowej, w przypadku którego zoptymalizowano wydajność zapytań i obsługę analizy danych o dużej skali. Choć relacyjne hurtownie danych i przetwarzanie transakcyjne w celu zapewnienia organizacji danych bazują na modelu relacyjnym, to relacyjna hurtownia danych ma zwykle większą skalę, a ponadto jest zoptymalizowana pod kątem zapytań analitycznych.

Relacyjne hurtownie danych są wyposażone zarówno w mechanizm realizujący obliczenia, jak i magazyn. Mechanizm ten zapewnia moc obliczeniową służącą do wykonywania zapytań dotyczących danych. Magazyn ma postać *magazynu relacyjnego* przechowującego dane ze strukturą uzyskiwaną dzięki zastosowaniu tabel, wierszy i kolumn. Moc obliczeniowa relacyjnej hurtowni danych może być używana wyłącznie względem jej magazynu relacyjnego (są one ze sobą sprzężone).

Wśród najważniejszych opcji relacyjnych hurtowni danych należy wymienić obsługę transakcji (zapewnienie, że dane są przetwarzane w niezawodny i spójny sposób), ścieżki audytu (utrzymywanie zapisu całej aktywności związanej z danymi w systemie) oraz wymuszanie schematu (zadbanie o to, aby dane zostały uporządkowane i uzyskały strukturę w predefiniowany sposób).

W latach 70. i 80. firmy używały relacyjnych baz danych na potrzeby aplikacji operacyjnych, takich jak obsługujące wprowadzanie zamówień i zarządzające stanem magazynowym. W odniesieniu do tego rodzaju aplikacji używa się terminu systemów OLTP (*Online Transaction Processing*). Systemy te mogą wprowadzać, odczytywać, aktualizować i usuwać zmiany (są to *operacje CRUD — Create, Read, Update, Delete*) dotyczące danych w bazie danych. Operacje te tworzą fundament modyfikowania danych oraz zarządzania w architekturach danych. Mają one kluczowe znaczenie z punktu widzenia projektowania i implementowania systemów do magazynowania danych oraz interfejsów użytkownika, które prowadzą interakcję z danymi.

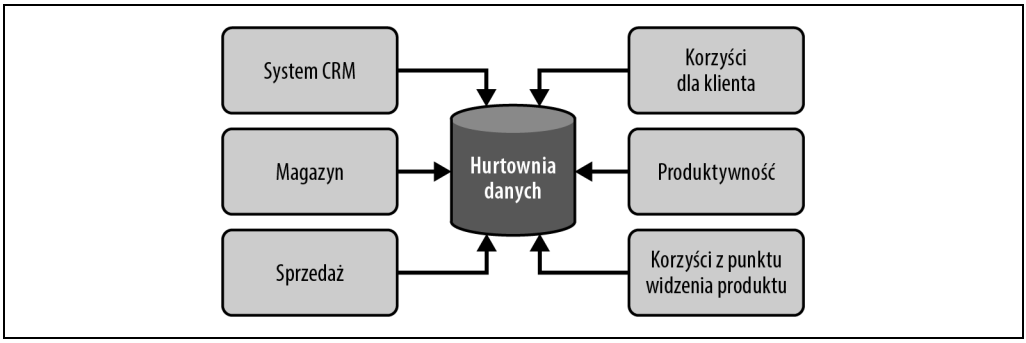
Operacje CRUD wymagają krótkich czasów odpowiedzi aplikacji, tak aby użytkownicy nie irytowali się tym, ile czasu zajmuje aktualizowanie danych. Możliwe jest uruchamianie zapytań i generowanie raportów z wykorzystaniem relacyjnej bazy danych używanej przez aplikację operacyjną, ale wiąże się to z zużyciem wielu zasobów, a ponadto może wywoływać konflikt z innymi operacjami CRUD wykonywanymi w tym samym czasie. W efekcie wszystko może zostać spowolnione.

Po części relacyjne hurtownie danych zostały opracowane w celu rozwiązania tego problemu. Dane z relacyjnej bazy danych są kopiowane do hurtowni danych, a użytkownicy mogą uruchamiać zapytania i raporty względem hurtowni danych, a nie relacyjnej bazy danych. Dzięki temu zapytania i raporty nie obciążają systemu udostępniającego relacyjną bazę danych i nie spowalniają aplikacji, z której korzystają użytkownicy. Relacyjne hurtownie danych centralizują również dane z wielu aplikacji, aby usprawnić raportowanie (zilustrowano to na rysunku 2.1).

---

[2] Uwaga dotycząca języka: możesz spotkać osoby, które w odniesieniu do architektury relacyjnej hurtowni danych używają terminu *tradycyjna hurtownia danych*. W książce przeważnie posługuję się określeniem *relacyjna hurtownia danych*, który czasami skracam do postaci *hurtownia danych*. Wszystkie te terminy odwołują się do tej samej rzeczy.

[3] Popularność relacyjnej hurtowni danych zwiększyła się w dużej mierze dzięki Barry'emu Devlinowi i Billowi Inmonowi, o czym będzie mowa w rozdziale 8.



Rysunek 2.1. Zastosowanie hurtowni danych

Bardziej szczegółowo relacyjne hurtownie danych omówię w rozdziale 4.

Jezioro danych

Jezioro danych (ang. *data lake*) to nowsze pojęcie, które po raz pierwszy pojawiło się około roku 2010. Jezioro danych możesz traktować jako wychwalany system plików, który nie różni się zbytnio od systemu plików obecnego na Twoim laptopie. **Jezioro danych** to po prostu magazyn, z którym — w przeciwieństwie do relacyjnej hurtowni danych — nie jest skojarzony żaden mechanizm obliczeniowy. Na szczęście istnieje wiele takich mechanizmów, które mogą współpracować z jeziorami danych, dlatego w jego przypadku moc obliczeniowa jest zwykle tańsza niż przy korzystaniu z relacyjnej hurtowni danych. Kolejną różnicą jest to, że relacyjne hurtownie danych używają magazynu relacyjnego, natomiast jeziora danych stosują *magazyn obiektowy*, który nie wymaga tworzenia struktury danych w ramach wierszy i kolumn.

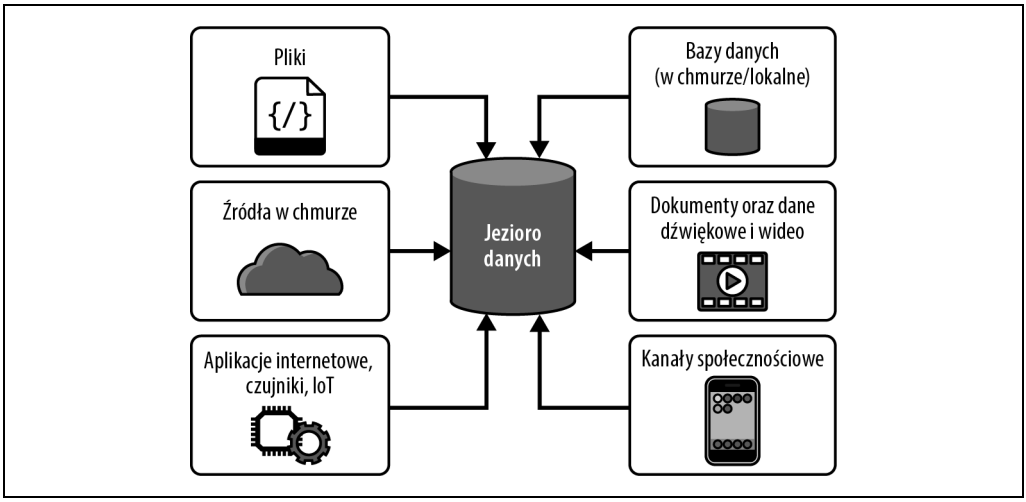
Technologia magazynów jeziora danych została zapoczątkowana systemem Apache HDFS (*Hadoop Distributed File System*), czyli darmową technologią *open source*[4] udostępnianą prawie wyłącznie lokalnie, która cieszyła się bardzo dużą popularnością w pierwszych latach po roku 2010. HDFS to skalowalny i odporny na błędy rozproszony system magazynowania, który zaprojektowano pod kątem działania na powszechnie dostępnym sprzęcie. System plików stanowi główny komponent ekosystemu Apache Hadoop, który szerzej omawiam w rozdziale 16. Wraz z dalszym zwiększaniem się znaczenia obliczeń w chmurze w jej obrębie były tworzone jeziora danych z użyciem różnego typu magazynu, a obecnie większość jezior istnieje w technologii chmury.

W porównaniu z relacyjną hurtownią danych jezioro danych bazuje na metodzie *Schema-on-Read*. Oznacza to, że do umieszczenia danych w jeziorze nie są wymagane żadne wcześniejsze działania: operacja skopiowania plików do jeziora danych może być tak prosta jak w przypadku folderów obecnych na Twoim laptopie. Dane w jeziorze są przechowywane w ich naturalnym (lub nieprzetworzonym) formacie. Oznacza to, że dane mogą trafić z ich systemu źródłowego do jeziora danych bez przekształcania ich do innego formatu. Jeśli na przykład eksportujesz dane z relacyjnej bazy

[4] Termin *open source* odnosi się do oprogramowania, którego kod źródłowy udostępniono za darmo do publicznego wykorzystania, modyfikowania i dystrybuowania.

danych do pliku w nieprzetworzonym formacie CSV, dane mógłbyś przechowywać w jeziorze danych w niezmienionej postaci. Jeżeli jednak zdecydowałbyś się na zapisanie danych w relacyjnej hurtowni danych, konieczne byłoby przekształcenie ich tak, aby mogły zostać umiejscowione w wierszach i kolumnach tabeli.

Podczas kopiowania do jeziora danych pliku z danymi jego schemat może nie zostać uwzględniony w ramach tej operacji albo może znajdować się w innym pliku. W związku z tym musisz zdefiniować schemat przez utworzenie go lub uzyskanie z osobnego pliku (stąd też termin *Schema-on-Read*). Jak widać na rysunku 2.2, dane z systemów źródłowych — takich jak bazy danych aplikacji operacyjnych, dane z czujników oraz dane z mediów społecznościowych — w całości mogą trafić do jeziora danych. W takich plikach mogą być przechowywane dane ze strukturą (np. dane z relacyjnych baz danych), z częściową strukturą (np. pliki CSV, dzienniki oraz pliki XML lub JSON) lub bez struktury (np. dane z wiadomości pocztowych, dokumentów i plików PDF). W plikach mogą nawet znajdować się dane binarne (np. obrazy, dane dźwiękowe i wideo).



Rysunek 2.2. Jezioro danych

Początkowo jeziora danych traktowano jako rozwiązanie wszystkich problemów towarzyszących relacyjnym hurtowniom danych, w tym: dużego kosztu, ograniczonej skalowalności, kiepskiej wydajności, dodatkowego nakładu pracy związanego z przygotowywaniem danych oraz ograniczonego wsparcia złożonych typów danych. Firmy zajmujące się sprzedażą systemu Hadoop i architektury jezior danych, takie jak Cloudera, Hortonworks i MapR, robiły wokół nich taki szum, jakby były one wypełnione jednorożcami i tęczami, które będą kopiować i oczyszczać dane, a następnie udostępniać je użytkownikom z magiczną łatwością. Firmy twierdziły, że jeziora danych mogłyby całkowicie zastąpić relacyjne hurtownie danych w ramach rozwiązania typu „jedna technologia służąca do wszystkiego". Więcej niż kilka firm zdecydowało się na oszczędności dzięki zastosowaniu darmowych narzędzi *open source* na potrzeby całości swojej technologii.

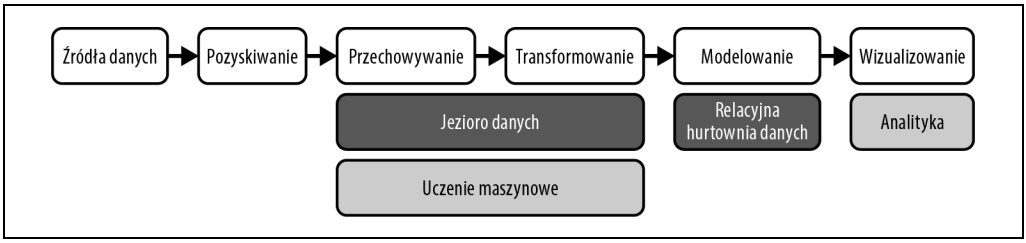
Problem polegał na tym, że w rzeczywistości kierowanie zapytań do jezior danych nie jest takie proste: wymaga to pewnych dość zaawansowanych umiejętności. Informatycy mogliby przekazać użytkownikom następującą informację: „Skopiowaliśmy do tego jeziora danych wszystkie dane,

których potrzebujecie. Po prostu uruchomcie notatnik Jupyter i użyjcie systemu Hive oraz języka Python do utworzenia swoich raportów z wykorzystaniem plików znajdujących się w tych folderach". Skończyło się to żałośnie, gdyż większość użytkowników w żadnym razie nie dysponowała umiejętnościami niezbędnymi do zrealizowania wszystkich tych działań. Firmy przekonały się w dotkliwy sposób, że te złożone i trudne w użyciu rozwiązania ostatecznie okażą się droższe z powodu kosztów sprzętu i obsługi, opóźnień produkcyjnych oraz utraconej produktywności. Ponadto jeziora danych były pozbawione opcji, jakie spodobały się użytkownikom w przypadku hurtowni danych, takich jak obsługa transakcji, wymuszanie schematu oraz ścieżki audytu. Poskutkowało to tym, że swoją działalność biznesową zakończyło dwóch z trzech głównych dostawców architektury jeziora danych, czyli firmy Hortonworks i MapR.

Jednak architektura jeziora danych nie zniknęła. Zamiast tego jej pierwotne przeznaczenie przekształcono do postaci innego, lecz bardzo przydatnego, czyli organizowania i przygotowywania danych. Jeziora danych omówię szczegółowo w rozdziale 5.

Nowoczesna hurtownia danych

Same w sobie relacyjne hurtownie danych i jeziora danych to bardzo proste architektury. Do centralizacji danych używają one tylko jednej technologii, co jest wspierane przez kilka dodatkowych produktów albo nawet żadne. Gdy zastosujesz więcej technologii i produktów do obsługi relacyjnych hurtowni danych lub jezior danych, przyjmą one postać architektur omawianych w tym i kolejnych rozdziałach. Jeziora danych nie zastąpiły relacyjnych hurtowni danych, ale nadal oferowały korzyści z punktu widzenia organizowania i przygotowywania danych. Dlaczego nie można by skorzystać z zalet obu rozwiązań? W okolicach roku 2011 wiele firm zaczęło tworzyć architektury, które umiejscawiają jeziora danych obok relacyjnych hurtowni danych w celu zapewnienia architektury, która obecnie określana jest mianem **nowoczesnej hurtowni danych** (ang. *modern data warehouse*) (rysunek 2.3). Słowo *nowoczesna* użyte w tym terminie odnosi się do zastosowania w przypadku hurtowni danych nowszych technologii i metod.

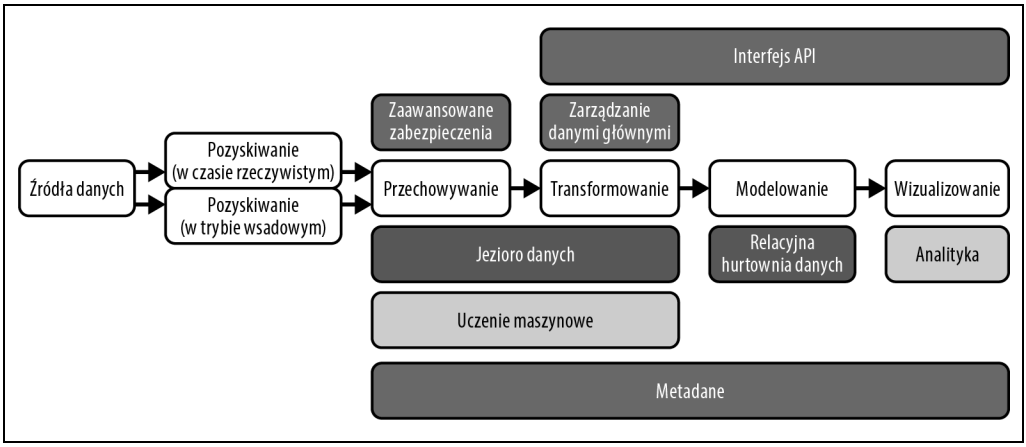


Rysunek 2.3. Nowoczesna hurtownia danych

Oto rozwiązanie reprezentujące „to, co najlepsze z obu światów": jezioro danych służy do organizowania i przygotowywania danych, a danolodzy używają tej architektury do budowania modeli uczenia maszynowego. Z kolei hurtownia danych ma na celu zapewnienie obsługi, bezpieczeństwa i zgodności. Korzystając z niej, użytkownicy tworzą swoje zapytania i generują raporty. W rozdziale 10. znacznie dokładniej objaśniam architektury nowoczesnej hurtowni danych.

Architektura Data Fabric

Architektura **Data Fabric** zaczęła się pojawiać około roku 2016. Możesz ją traktować jako kolejny etap rozwoju architektury nowoczesnej hurtowni danych poszerzonej o dodatkową technologię służącą do pozyskiwania większej ilości danych, zabezpieczania ich oraz udostępniania. Ponadto wprowadzono ulepszenia dotyczące tego, jak system pozyskuje, przekształca, pobiera i wyszukuje dane oraz zapewnia do nich dostęp (rysunek 2.4). Dzięki tym wszystkim dodatkom system staje się „fabryką", czyli dużą strukturą, która umożliwia pozyskiwanie dowolnego rodzaju danych. Zajmiemy się tym bardziej szczegółowo w rozdziale 11.

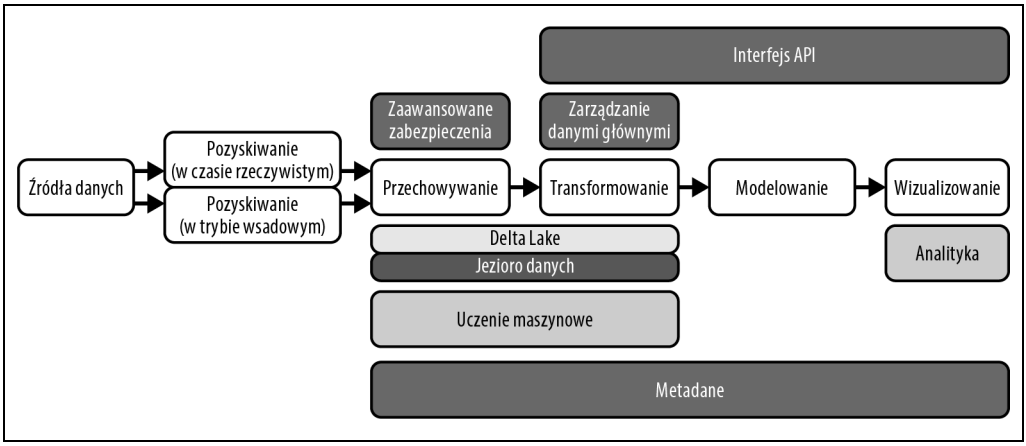


Rysunek 2.4. Architektura Data Fabric

Architektura Data Lakehouse

Termin **Data Lakehouse** powstał w wyniku połączenia terminów *jezioro danych* (ang. *data lake*) i *hurtownia danych* (ang. *data warehouse*). Architektury Data Lakehouse stały się popularne około roku 2020, gdy terminem tym zaczęła się posługiwać firma Databricks. W przypadku zagadnienia reprezentowanego przez termin Lakehouse chodzi o wyeliminowanie relacyjnej hurtowni danych i zastosowanie w posiadanej architekturze danych tylko jednego repozytorium w postaci jeziora danych. Dane każdego typu (ze strukturą, z częściową strukturą i bez struktury) są pozyskiwane do jeziora danych, a wszystkie zapytania i raporty są tworzone z jego poziomu.

Wiem, że możesz stwierdzić następująco: „Jedna chwila. Wspomniałeś, że w jeziorach danych zastosowano takie samo rozwiązanie, gdy się dopiero pojawiły, ale skończyło się to żałośnie! Co się zmieniło?". Jak widać na rysunku 2.5, odpowiedzią na to pytanie jest warstwa oprogramowania magazynu transakcyjnego, która działa ponad istniejącym jeziorem danych i sprawia, że funkcjonuje ono bardziej na podobieństwo relacyjnej bazy danych. Wśród opcji *open source* będących konkurencją dla tej warstwy należy wymienić narzędzia Delta Lake, Apache Iceberg i Apache Hudi. Wszystkie omawiam dokładniej w rozdziale 12.



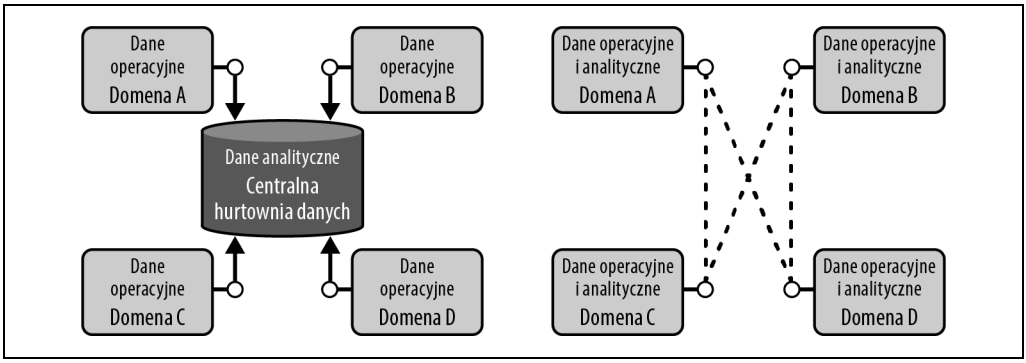
Rysunek 2.5. Architektura Data Lakehouse

Siatka danych

Termin **siatka danych** (ang. *data mesh*) po raz pierwszy został wprowadzony w maju 2019 r. przez Zhamak Dehghani (założycielka i prezes firmy Nextdata oraz autorka książki *Siatka danych. Nowoczesna koncepcja samoobsługowej infrastruktury danych*; Helion, 2023 r.) w jej wpisie blogowym. W grudniu 2020 r. Dehghani *dokładniej objaśniła*, czym jest siatka danych, a ponadto przedstawiła jej cztery fundamentalne zasady. Od tamtego czasu architektura siatki danych stała się wyjątkowo gorącym tematem. Dyskutowano o niej w ramach ogromnej liczby blogów, prezentacji, konferencji i materiałów prasowych. Temat został nawet ujęty w raporcie badawczym na stronie *Gartner Hype Cycle for data management*. W przypadku architektury siatki danych jest wiele rzeczy, które można polubić, ale — pomimo towarzyszącego jej rozgłosowi — jest ona odpowiednia tylko w odniesieniu do niewielkiej liczby wariantów zastosowań.

Nowoczesna hurtownia danych oraz architektury Data Fabric i Data Lakehouse obejmują swoim zakresem *centralizowanie* danych, czyli kopiowanie danych operacyjnych do centralnej lokalizacji będącej w posiadaniu działu informatycznego w ramach architektury kontrolowanej przez jego pracowników, w obrębie której tworzą oni dane analityczne (widoczne po lewej stronie na rysunku 2.6). Takie scentralizowane podejście stawia przed nami następujące trzy główne wyzwania: własność danych, jakość danych oraz skalowanie o charakterze organizacyjnym lub technicznym. Celem siatki danych jest stawienie czoła tym wyzwaniom.

W siatce danych dane są utrzymywane w obrębie kilku domen należących do firmy, takich jak produkcja, sprzedaż i dostawcy (widoczne po prawej stronie na rysunku 2.6). Z każdą domeną jest powiązany jej własny, niewielki zespół informatyków, którzy są właścicielem danych domeny i oczyszczają je, a ponadto tworzą i udostępniają dane analityczne. Każda domena dysponuje również własną infrastrukturą służącą do magazynowania i wykonywania obliczeń. Skutkuje to architekturą *zdecentralizowaną*, w przypadku której skalowane są dane, zasoby ludzkie oraz infrastruktura. Im większą liczbą domen dysponujesz, tym do większej liczby osób i zasobów infrastruktury uzyskujesz dostęp. System może obsługiwać więcej danych, a dział informatyczny nie stanowi już wąskiego gardła.



Rysunek 2.6. Siatka danych

Ważne jest uświadomienie sobie tego, że siatka danych to *pojęcie*, a nie technologia. Nie istnieje żadna „pudełkowa wersja siatki danych", którą możesz kupić. Implementowanie siatki danych wiąże się z dokonaniem bardzo dużej zmiany organizacyjnej i kulturowej, na którą jest gotowa bardzo niewielka liczba firm (tak naprawdę większość firm nie jest nawet na tyle duża, aby rozważać zastosowanie architektury siatki danych: jest to rozwiązanie całkowicie przewidziane dla przedsiębiorstw). Zbudowanie siatki danych wymaga określenia, jakie elementy istniejącej technologii możesz przystosować na jej potrzeby, a jakie elementy będą musiały zostać utworzone. W przypadku każdej domeny konieczne jest ustalenie, jakich technologii użyje ona do zbudowania powiązanej z nią części siatki danych. Może to obejmować utworzenie nowoczesnej hurtowni danych albo architektury Data Fabric lub Data Lakehouse. Architekturom siatki danych można poświęcić sporo miejsca, na co też zdecydowałem się w rozdziałach 13. i 14.

Podsumowanie

Gdy już masz ogólną orientację w zakresie typów architektur danych, w kolejnym rozdziale będzie mowa o tym, jak określić najlepszą architekturę, która może zostać użyta: jest to proces określany mianem **sesji projektowania architektury**.

Niezwykłość tej książki polega na przekształcaniu złożonych zagadnień technicznych w jasne i zrozumiałe objaśnienia.

Annie Xu, starszy inżynier danych, Google

Architektury Data Fabric i Data Lakehouse, a także siatka danych pojawiły się niedawno jako alternatywy hurtowni danych. Te nowe architektury mają swoje mocne strony, ale podczas projektowania rzeczywistych rozwiązań musisz pamiętać o odróżnianiu faktów od przesadnych pochwał i niejasności. Nie zawsze jest to proste i oczywiste zadanie.

Dzięki temu praktycznemu przewodnikowi profesjonaliści zajmujący się danymi dobrze zrozumieją wady i zalety poszczególnych rozwiązań. Omówiono tu typowe zagadnienia dotyczące architektur danych, w tym ich rozwój i możliwości. Żadna architektura nie jest na tyle uniwersalna, by być odpowiednia w każdej sytuacji, dlatego w książce znajdziesz rzetelne porównanie cech poszczególnych architektur. Dowiesz się, jakie kompromisy towarzyszą każdej z nich, niezależnie od popularności. W ten sposób o wiele łatwiej przyjdzie Ci wybór rozwiązania, które najlepiej odpowiada Twoim potrzebom.

Połóż tę książkę na biurku. Będziesz często po nią sięgać!

Sawyer Nyquist, autor, właściciel The Data Shop

Najciekawsze zagadnienia:

- praktyczne działanie architektur danych, ich mocne i słabe strony
- wybór najlepszej architektury pod kątem konkretnego zastosowania
- różnice między hurtowniami i jeziorami danych
- wspólne koncepcje architektur danych i ich historyczny rozwój
- sesje projektowania architektury, organizacja zespołów i najważniejsze uwarunkowania

James Serra — doświadczony architekt rozwiązań w Microsofcie. Uznany lider w dziedzinie stosowania technologii big data i zaawansowanej analityki, w tym takich architektur danych jak nowoczesna hurtownia danych, siatka danych, a także Data Fabric i Data Lakehouse.

Helion
helion.pl
HELION S.A.
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl
KOD KORZYŚCI
Sięgnij po więcej!
ISBN 978-83-289-1669-2
9 788328 916692
Cena: 79,00 zł