NoSQL, NewSQL i BigData

Bazy danych następnej generacji

—

Guy Harrison

Helion

Apress

Tytuł oryginału: Next Generation Databases: NoSQL and Big Data

Tłumaczenie: Piotr Pilch

ISBN: 978-83-283-4751-9

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

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

Printed in Poland.

Spis treści

ROZDZIAŁ 1

Trzy rewolucje związane z bazami danych

Urojenie. Szaleństwo.

Dopóki się nie wydarzą, wszystkie rewolucje tym są, a potem stają się historycznymi nieuchronnościami.

— David Mitchell, *Atlas chmur*

Nadal tkwimy w pierwszych minutach pierwszego dnia rewolucji internetowej.

— Scott Cook

Książka poświęcona jest trzeciej rewolucji związanej z technologiami baz danych. Pierwsza rewolucja została zainicjowana przez pojawienie się komputera elektronicznego, natomiast druga rewolucja była wynikiem powstania relacyjnej bazy danych. Trzecią rewolucję spowodowała eksplozja alternatyw w postaci nierelacyjnych baz danych, które były odpowiedzią na zapotrzebowanie nowoczesnych aplikacji wymagających globalnego zasięgu i ciągłej dostępności. W rozdziale tym dokonamy przeglądu tych trzech fal technologii baz danych, a także omówimy czynniki rynkowe i technologiczne, które doprowadziły do pojawienia się obecnie używanych baz danych następnej generacji.

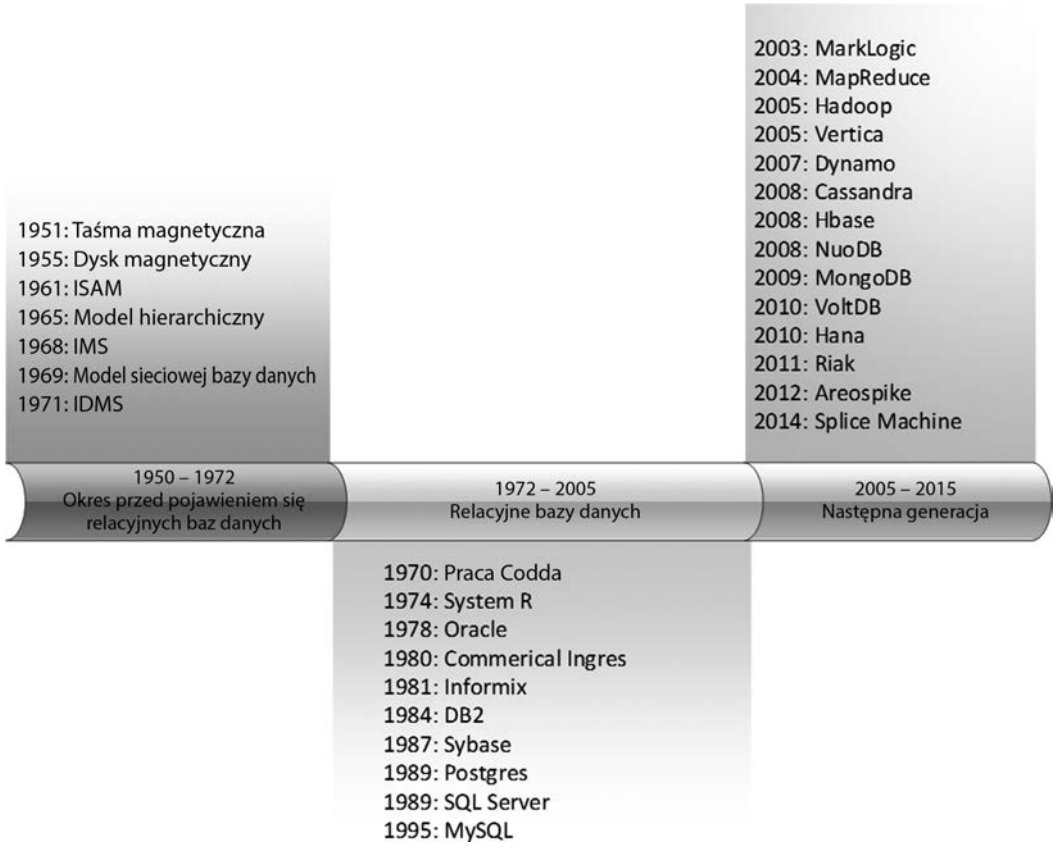
Na rysunku 1.1 pokazano prostą oś czasu z głównymi systemami baz danych.

Na rysunku 1.1 zilustrowano trzy podstawowe okresy związane z technologią baz danych. W ciągu 20 lat następujących od momentu rozpoczęcia powszechnego wykorzystywania komputerów elektronicznych pojawiła się grupa coraz bardziej zaawansowanych systemów baz danych. Krótko po zdefiniowaniu modelu relacyjnego w 1970 r. niemal każdy znaczący system baz danych korzystał ze wspólnej architektury. Trzema filarami tej architektury były: model relacyjny, transakcje ACID (*Atomicity, Consistency, Isolation, Durability*) i język SQL.

Począwszy od około 2008 r. miała jednak miejsce eksplozja nowych systemów baz danych, z których żaden nie opierał się na tradycyjnych implementacjach relacyjnych. Tym nowym systemom baz danych poświęcona jest książka. W niniejszym rozdziale zostanie zaprezentowane, w jaki sposób wcześniejsze fale technologii baz danych doprowadziły do pojawienia się następnej generacji systemów baz danych.

Wczesne systemy baz danych

W serwisie Wikipedia termin **baza danych** zdefiniowano jako „zbiór danych zapisanych zgodnie z określonymi regułami". Choć termin ten pojawił się w używanym słownictwie dopiero pod koniec lat 60., gromadzenie i organizowanie danych stanowiło integralny element rozwoju ludzkiej cywilizacji i technologii.



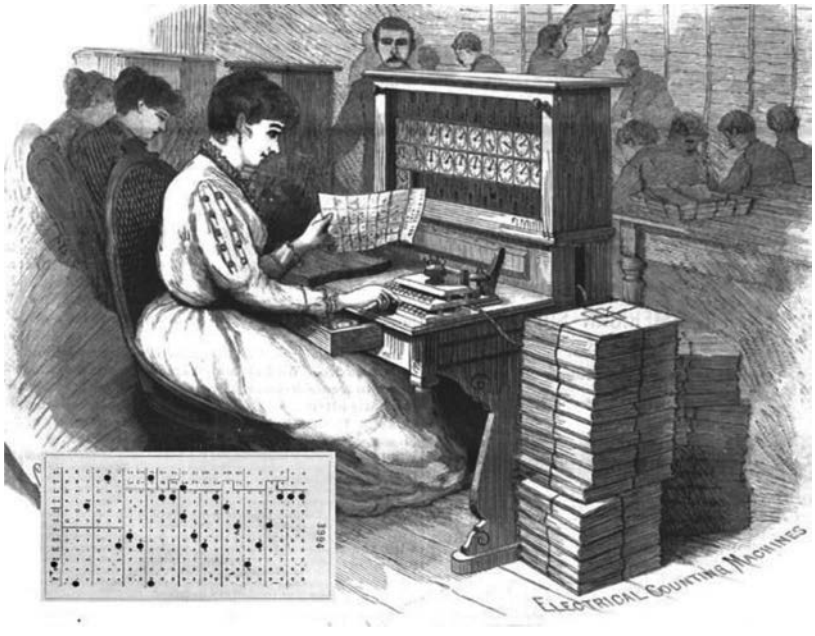
Rysunek 1.1. *Oś czasu z głównymi systemami baz danych i ich innowacjami*

Książki, a zwłaszcza te z wyraźnie narzuconą strukturą, takie jak słowniki i encyklopedie, są zbiorami danych w postaci fizycznej. Biblioteki oraz inne indeksowane archiwa informacji są odpowiednikami nowoczesnych systemów baz danych z czasów przed rozwojem przemysłu.

Możliwe jest też dostrzeżenie genezy cyfrowej bazy danych w zastosowaniu kart dziurkowanych oraz innych nośników fizycznych, które mogły przechowywać informacje w postaci pozwalającej na mechaniczne przetwarzanie. W XIX wieku karty krosien służyły do „programowania" krosien tkackich w celu uzyskania złożonych wzorów na materiale. Z kolei w wypadku maszyn licząco-analitycznych stosowano karty dziurkowane do przygotowywania statystyk dotyczących spisu ludności, a w pianolach używano perforowanych pasków papierowych reprezentujących melodie. Na rysunku 1.2 pokazano maszynę licząco-analityczną Holleritha, używaną w 1890 r. do przetwarzania danych dotyczących spisu ludności w Stanach Zjednoczonych.

Pojawienie się komputerów elektronicznych po drugiej wojnie światowej stanowiło pierwszą rewolucję związaną z bazami danych. Powstało kilka wczesnych komputerów cyfrowych, które wykonywały funkcje czysto matematyczne, takie jak obliczanie tabel balistycznych. Jednakże równie często miały one za zadanie przetwarzać dane, tak jak w wypadku szyfrowanej komunikacji wojskowej Axis.

Początkowo wczesne „bazy danych" korzystały z taśmy papierowej, a ostatecznie do sekwencyjnego przechowywania danych zastosowano taśmę magnetyczną. Choć możliwe było „szybkie przewijanie" i „cofanie" w obrębie takich zbiorów danych, stało się tak dopiero w momencie pojawienia się w połowie lat 50. obrotowego dysku magnetycznego, który umożliwił bezpośredni dostęp z dużą szybkością do poszczególnych rekordów. Bezpośredni dostęp pozwolił na szybki dostęp do dowolnego elementu w obrębie



Rysunek 1.2. *Maszyny licząco-analityczne i karty dziurkowane używane w 1890 r. do przetwarzania danych dotyczących spisu ludności w Stanach Zjednoczonych*

pliku o dowolnej wielkości. Rozwój metod indeksowania, takich jak ISAM (ang. *Index Sequential Access Method*), urzeczywistnił szybki dostęp oparty na rekordach, a w konsekwencji doprowadził do pojawienia się pierwszych systemów komputerowych OLTP (ang. *On-line Transaction Processing*).

Na bazie metody ISAM oraz podobnych struktur indeksowania działały pierwsze elektroniczne bazy danych. Znajdowały się one jednak pod całkowitą kontrolą aplikacji. Były to bazy danych, lecz nie **systemy zarządzania bazami danych DBMS** (ang. *Database Management Systems*).

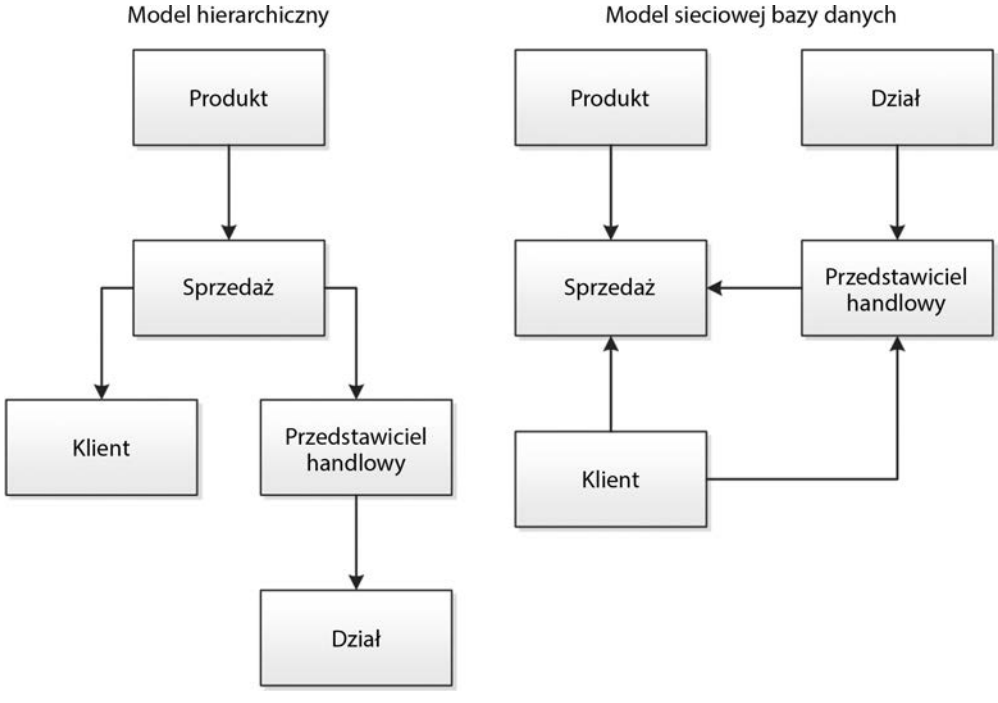
Pierwsza rewolucja związana z bazami danych

Wymaganie tego, aby każda aplikacja tworzyła własny kod obsługujący dane, stanowiło oczywiście problem związany z efektywnością: każda aplikacja musiała w wypadku bazy danych na nowo „odkrywać koło". Co więcej, błędy występujące w kodzie obsługującym dane aplikacji nieuchronnie prowadziły do uszkodzenia danych. Zezwolenie wielu użytkownikom na jednoczesny dostęp do danych lub na ich modyfikowanie bez spowodowania logicznego lub fizycznego uszkodzenia danych wymaga zastosowania zaawansowanego kodu. I wreszcie, optymalizacja dostępu do danych z wykorzystaniem buforowania, wstępnego ładowania danych oraz innych technik wymagała użycia złożonych i specjalizowanych algorytmów, które nie mogły być w prosty sposób duplikowane w każdej aplikacji.

W związku z tym pożądane stało się wyodrębnienie logiki związanej z obsługą bazy danych z aplikacji i przeniesienie jej do osobnej bazy kodu. Taka warstwa, która nazywana jest systemem DBMS, minimalizuje nakład pracy programistów, a ponadto zapewnia wydajność i integralność funkcji dostępu do danych.

We wczesnych systemach baz danych wymuszano zarówno schemat (definicja struktury danych w bazie), jak i ścieżkę dostępu (ustalone metody nawigacji między rekordami). System DBMS mógł na przykład zawierać definicję elementów KLIENT i ZAMÓWIENIE wraz z określoną ścieżką dostępu, która umożliwiała uzyskanie zamówień skojarzonych z konkretnym klientem lub klienta powiązanego z określonym zamówieniem.

Takie bazy danych pierwszej generacji działały wyłącznie w ówczesnych systemach komputerowych typu *mainframe* (głównie produkowanych przez firmę IBM). Z początkiem lat 70. o dominację walczyły ze sobą dwa podstawowe modele systemów DBMS. **Model sieciowej bazy danych** został sformalizowany w ramach standardu CODASYL i zaimplementowany w bazach danych takich jak IDMS. Z kolei **model hierarchiczny** zapewniał trochę prostsze rozwiązanie, które najbardziej było widoczne w systemie IMS (ang. *Information Management System*) firmy IBM. Na rysunku 1.3 porównano strukturalną reprezentację danych tych baz danych.



Rysunek 1.3. *Model sieciowej bazy danych i model hierarchiczny*

Uwaga Te wczesne systemy często opisywano jako z natury „nawigacyjne", ponieważ w ich wypadku niezbędne jest nawigowanie z jednego obiektu do drugiego przy użyciu wskaźników lub łączników. Aby na przykład znaleźć zamówienie w hierarchicznej bazie danych, może być niezbędne zlokalizowanie najpierw klienta, a następnie użycie łącznika do jego zamówień.

Sieciowe i hierarchiczne systemy baz danych dominowały w erze przetwarzania z wykorzystaniem komputerów typu *mainframe*, a ponadto obsługiwały zdecydowaną większość aplikacji komputerowych do końca lat 70. Systemy te miały jednak kilka znaczących wad.

Po pierwsze, nawigacyjne bazy danych cechowały się wyjątkową nieelastycznością w odniesieniu do struktury danych i możliwości tworzenia zapytań. Ogólnie rzecz biorąc, możliwe były tylko zapytania, które mogły zostać przewidziane w początkowej fazie projektowania. Niezmiernie trudne było dodawanie nowych elementów danych do istniejącego systemu.

Po drugie, systemy baz danych opierały się na przetwarzaniu transakcji po jednym rekordzie jednocześnie. Obecnie jest to określane za pomocą terminu CRUD (ang. *Create, Read, Update, Delete*). Operacje zapytań, a zwłaszcza grupa złożonych zapytań analitycznych, które obecnie kojarzone są z analityką biznesową, wymagały tworzenia skomplikowanego kodu. Zapotrzebowanie biznesu związane z raportami

analitycznymi szybko się zwiększało wraz z postępującą integracją systemów komputerowych z procesami biznesowymi. W efekcie większość działów informatycznych miała do czynienia z ogromnymi zaległościami związanymi z żądaniami dotyczącymi raportów, a cała generacja programistów tworzyła w języku COBOL pełen powtórzeń kod obsługujący raportowanie.

Druga rewolucja związana z bazami danych

Bez wątpienia żadna inna osoba nie miała większego wpływu na technologię baz danych niż Edgar Codd. Codd uzyskał tytuł magistra matematyki na uniwersytecie w Oksfordzie niedługo po wybuchu drugiej wojny światowej. Później wyemigrował do Stanów Zjednoczonych, gdzie począwszy od 1949 r. z przerwami pracował dla firmy IBM. Codd pracował na stanowisku „matematyka programującego" (ach, co to były za czasy), a ponadto brał udział w pracach związanych z niektórymi z pierwszych komercyjnych komputerów elektronicznych firmy IBM.

Pod koniec lat 60. Codd pracował w laboratorium firmy IBM znajdującym się w San Jose w stanie Kalifornia. W bardzo dużym stopniu był zaznajomiony z bazami danych z tamtych czasów. Miał spore zastrzeżenia do ich budowy. W szczególności zauważył, że:

- **Korzystanie z istniejących baz danych jest zbyt utrudnione**. Bazy danych z tamtych czasów mogły być używane wyłącznie przez osoby dysponujące specjalnymi umiejętnościami programistycznymi.
- **Istniejące bazy danych są pozbawione podstawy teoretycznej**. Wykształcenie matematyczne Codda sprawiło, że dane traktował jako struktury formalne i operacje logiczne. Postrzegał istniejące bazy danych jako stosowanie przypadkowych reprezentacji, które nie zapewniały logicznej spójności ani możliwości radzenia sobie z brakującymi informacjami.
- **W istniejących bazach danych wymieszano implementacje logiczne i fizyczne**. Reprezentacja danych w istniejących bazach danych była dopasowana do formatu magazynu fizycznego w bazie, zamiast mieć postać reprezentacji logicznej danych, która może być zrozumiała dla użytkownika bez wiedzy specjalistycznej.

Codd opublikował wewnętrzny dokument firmy IBM, w którym opisał swoje pomysły zapewnienia bardziej sformalizowanego modelu systemów baz danych. Doprowadziło to później do zaprezentowania przez niego w 1970 r. pracy zatytułowanej *A Relational Model of Data for Large Shared Data Banks*[1] (model relacyjny danych w wypadku dużych współużytkowanych banków danych). W tej klasycznej pracy zawarto podstawowe idee definiujące **model relacyjnej bazy danych**, który stał się dla całej generacji najbardziej znaczącym — niemal uniwersalnym — modelem systemów baz danych.

Teoria modelu relacyjnego

Zawiłości teorii relacyjnej bazy danych mogą być złożone i wykraczają poza zakres niniejszego wprowadzenia. W zasadzie jednak model relacyjny opisuje to, w jaki sposób konkretny zbiór danych powinien być prezentowany użytkownikowi, a nie to, jak powinien być przechowywany na dysku lub w pamięci. Kluczowe zagadnienia modelu relacyjnego obejmują:

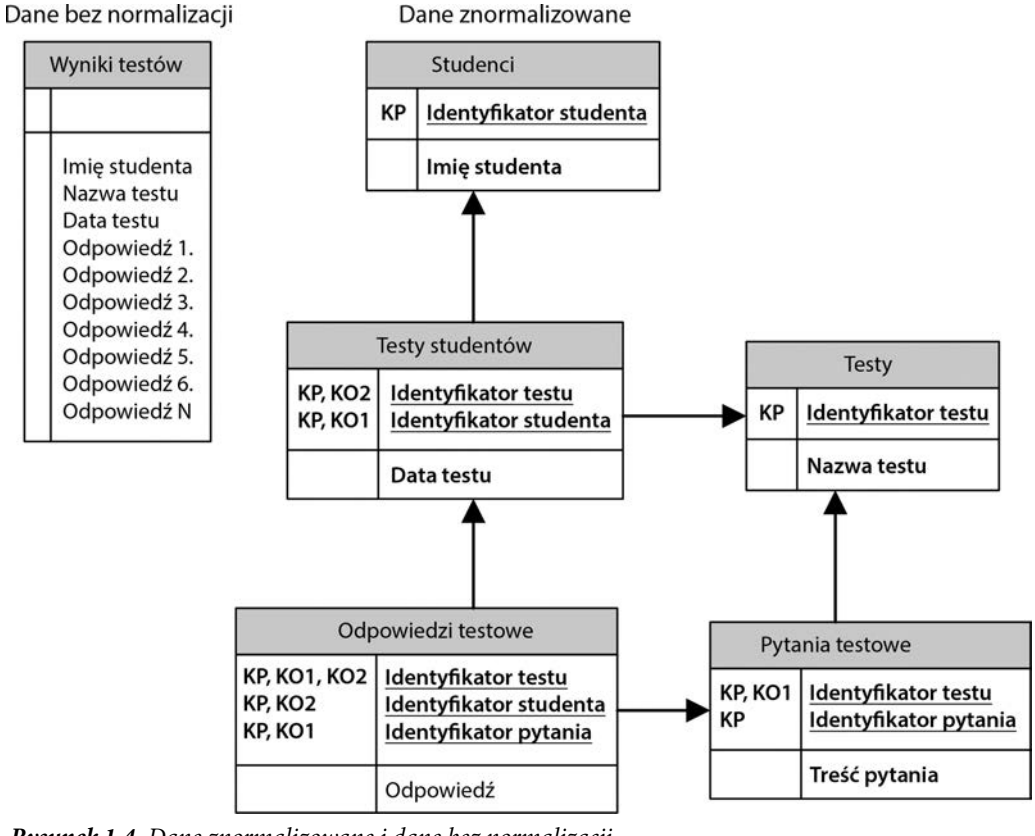
- **Krotki**, czyli nieuporządkowany zbiór wartości **atrybutów**. W samym systemie baz danych krotka odpowiada wierszowi, a atrybut — wartości kolumny.
- **Relację**, czyli kolekcję różnych krotek odpowiadającą tabeli w implementacjach relacyjnych baz danych.
- **Ograniczenia** wymuszające spójność bazy danych. Ograniczenia klucza służą do identyfikowania krotek i relacji między nimi.

- **Operacje** na relacjach, takie jak złączenia, projekcje czy sumy. Operacje te zawsze zwracają relacje. W praktyce oznacza to, że zapytanie dotyczące tabeli zwraca dane w formacie tabelarycznym.

Wiersz w tabeli powinien być możliwy do zidentyfikowania i efektywnie dostępny dla wartości klucza unikatowego. Z kolei każda kolumna w tym wierszu musi być zależna od tej wartości klucza, a nie od żadnego innego identyfikatora. Oznacza to, że tablice oraz inne struktury zawierające zagnieżdżone informacje nie są bezpośrednio obsługiwane.

Poziomy zgodności z modelem relacyjnym opisano w różnych *postaciach normalnych* (ang. *normal forms*). Najpowszechniejszym poziomem jest *trzecia postać normalna*. Praktycy używający baz danych zapamiętują zwykle definicję trzeciej postaci normalnej przez utrwalenie sobie, że wszystkie atrybuty niebędące kluczem muszą być zależne od „klucza, całego klucza i tylko od klucza — tu może pomóc tylko Codd”[2]!

Na rysunku 1.4 zaprezentowano przykład normalizacji: dane po lewej stronie reprezentują dość prosty zbiór danych. Występuje w nim jednak redundancja w imieniu studenta i nazwie testu, a użycie dla odpowiedzi testowych powtarzającego się zestawu atrybutów jest wątpliwe (choć jest to możliwe w obrębie postaci relacyjnej, oznacza, że każdy test zawiera taką samą liczbę pytań i utrudnia określone operacje). Pięć tabel po prawej stronie reprezentuje znormalizowaną postać tych danych.



Rysunek 1.4. *Dane znormalizowane i dane bez normalizacji*

Modele transakcji

Sam model relacyjny nie definiuje sposobu, w jaki baza danych obsługuje współbieżne żądania zmiany danych. Takie zmiany, które ogólnie są określane mianem *transakcji* w bazie danych, wywołują problemy we wszystkich systemach baz danych z powodu konieczności zapewnienia spójności i integralności danych.

Jim Gray zdefiniował pod koniec lat 70. najpowszechniej przyjęty model transakcji. Jak sam to określił, „transakcja jest transformacją stanu, który ma właściwości w postaci niepodzielności (wszystko albo nic), trwałości (efekty są odporne na awarie) i spójności (poprawna transformacja)”[3]. Wkrótce zostało to spopularyzowane jako *transakcje ACID*: *Atomic* (niepodzielność), *Consistent* (spójność), *Independent* (niezależność) i *Durable* (trwałość). Transakcja ACID powinna być:

- **Niepodzielna**. Transakcja jest niepodzielna — albo wszystkie polecenia w transakcji zostaną zastosowane w bazie danych, albo żadne z nich.
- **Spójna**. Baza danych pozostaje w spójnym stanie przed wykonaniem transakcji i po jej wykonaniu.
- **Izolowana**. Choć wiele transakcji może być wykonywanych jednocześnie przez jednego lub wielu użytkowników, jedna transakcja nie powinna mieć dostępu do efektów innych transakcji będących w trakcie realizacji.
- **Trwała**. Po zapisaniu transakcji w bazie danych (w wypadku baz danych SQL odbywa się to za pomocą polecenia `COMMIT`) związane z nią zmiany mają być trwałe nawet wtedy, gdy nastąpi awaria systemu operacyjnego lub sprzętu.

Transakcje ACID stały się standardem we wszystkich poważnych implementacjach baz danych, ale też w największym stopniu zostały powiązane z relacyjnymi bazami danych, które zaczęły się pojawiać mniej więcej w czasie, gdy została opublikowana praca Graya.

Jak się później okaże, narzucane przez model transakcji ACID ograniczenie dotyczące skalowalności nieprzekraczającej pojedynczego centrum danych było głównym czynnikiem motywującym do tworzenia nowych architektur baz danych.

Pierwsze relacyjne bazy danych

Początkowa reakcja na model relacyjny była dość chłodna. Istniejący dostawcy, w tym firma IBM, nie byli skłonni do zaakceptowania podstawowego założenia Codda, zgodnie z którym ówczesne bazy danych oparte były na wadliwym fundamencie. Co więcej, wielu dostawców miało poważne zastrzeżenia co do możliwości zapewnienia przez system odpowiedniej wydajności, jeśli reprezentacja danych nie została dokładnie dopasowana do bazowych mechanizmów dostępu. Czy byłoby możliwe stworzenie systemu baz danych o dużej wydajności, który pozwalałby użytkownikowi na uzyskanie dostępu do danych w dowolny sposób, jaki być może sobie wyobrazi?

Firma IBM zainicjowała jednak w 1974 r. program badawczy w celu opracowania prototypowego systemu relacyjnych baz danych o nazwie System R. Pokazał on, że relacyjne bazy danych mogą zapewnić odpowiednią wydajność. Poza tym po raz pierwszy w systemie tym użyto **języka SQL** (Codd określił, że system relacyjny powinien uwzględniać język zapytań, ale nie definiować konkretnej składni). Również w tym czasie Michael Stonebraker z uczelni w Berkeley rozpoczął pracę nad systemem bazy danych, który ostatecznie przyjął nazwę **INGRES**. Był to też system relacyjny, ale wykorzystywał język zapytań o nazwie **QUEL**, który nie był językiem SQL.

Na tym etapie w prezentowanej historii pojawia się Larry Ellison. Z natury Ellison był bardziej przedsiębiorcą niż naukowcem, choć dysponował wyjątkową wiedzą techniczną, którą zdobył podczas pracy w firmie Amdahl. Zaznajomiony był zarówno z pracą Codda, jak i z systemem System R. Ellison był przekonany, że relacyjne bazy danych są przyszłością technologii baz danych. W 1977 r. założył firmę, która ostatecznie stała się korporacją Oracle Corporation. Może się ona pochwalić udostępnieniem systemu relacyjnych baz danych, który jako pierwszy odniósł sukces komercyjny.

Wojny baz danych!

Miało to miejsce w czasie, gdy na rynku pojawiły się minikomputery, które ostatecznie zakończyły dominację komputerów typu *mainframe*. W porównaniu z obecnym sprzętem komputerowym minikomputery z końca lat 70. i początku lat 80. z trudem można określić mianem „mini". W przeciwieństwie jednak do komputerów typu *mainframe* w niewielkim stopniu wymagały specjalistycznych pomieszczeń albo wcale tego nie wymagały. Dzięki temu średniej wielkości firmy po raz pierwszy mogły sobie pozwolić na posiadanie własnej infrastruktury informatycznej. Na tych nowych platformach sprzętowych działały nowe systemy operacyjne. Spowodowało to pojawienie się zapotrzebowania na nowe bazy danych, które mogłyby zostać uruchomione na tych systemach operacyjnych.

Około 1981 r. firma IBM udostępniła komercyjną relacyjną bazę danych o nazwie SQL/DS. Ponieważ jednak działała ona tylko na systemach operacyjnych komputerów typu *mainframe* firmy IBM, nie miała żadnego wpływu na szybko rozwijający się rynek minikomputerów. W 1979 r. pojawiła się komercyjna wersja systemu baz danych **Oracle** Ellisona, która szybko znalazła zastosowanie w wypadku minikomputerów dostarczanych przez takie firmy jak Digital i Data General. W tym samym czasie w ramach projektu INGRES uczelni w Berkeley powstała komercyjna relacyjna baza danych **Ingres**. Systemy Oracle i Ingres walczyły o dominację na rozwijającym się dopiero rynku relacyjnych baz danych dla minikomputerów.

Około połowy lat 80. powszechnie stały się zrozumiałe, jeśli nie niuanse teorii relacyjnej, to przynajmniej zalety relacyjnej bazy danych. Nabywcy baz danych w szczególności docenili to, że język SQL, który obecnie został zaadaptowany przez wszystkich dostawców, w tym przez system Ingres, zapewniał ogromny wzrost wydajności w zakresie tworzenia raportów i zapytań analitycznych. Ponadto coraz bardziej popularna stawała się następna generacja narzędzi do tworzenia baz danych, znanych wówczas pod nazwą 4GL. Te nowe narzędzia sprawdzały się zwykle najlepiej w wypadku serwerów relacyjnych baz danych. I wreszcie, w porównaniu z komputerami typu *mainframe* minikomputery oferowały wyraźne korzyści pod względem ceny i uzyskiwanej wydajności, co dotyczyło zwłaszcza segmentu średniej wielkości firm. W tej sytuacji relacyjna baza danych była tak naprawdę jedynym słusznym wyborem.

I faktycznie, relacyjne bazy danych w tak dużym stopniu zdominowały świadomość użytkowników, że dostawcy starszych systemów baz danych byli zmuszeni do prezentowania swoich produktów jako również opartych na modelu relacyjnym. Sprowokowało to Codda do sformułowania słynnych 12 postulatów (w rzeczywistości 13 postulatów, gdyż pierwszy z nich ma przypisaną cyfrę 0) jako czegoś w rodzaju sprawdzianu mającego za zadanie odróżnić prawdziwe relacyjne bazy danych od baz, które do tego miana jedynie pretendują.

W ciągu kolejnych dekad pojawiło się wiele nowych systemów baz danych. Są to następujące systemy: **Sybase**, **Microsoft SQL Server**, **Informix**, **MySQL** i **DB2**. Choć każdy z tych systemów próbuje się wyróżniać znakomitą wydajnością, dostępnością, funkcjonalnością lub ceną, wszystkie one są w zasadzie identyczne pod względem bazowania na trzech podstawowych zasadach: modelu relacyjnym Codda, języku SQL i modelu transakcji ACID.

Uwaga Termin RDBMS (ang. *Relational Database Management System*) zasadniczo dotyczy bazy danych implementującej relacyjny model danych, obsługującej transakcje ACID i korzystającej z języka SQL na potrzeby zapytań i przetwarzania danych.

Model przetwarzania klient-serwer

Pod koniec lat 80. model relacyjny osiągnął zdecydowane zwycięstwo w batalii o świadomość użytkowników baz danych. Dominacja ta przełożyła się na dominację na rynku w trakcie przechodzenia do **modelu przetwarzania klient-serwer**.

Pod pewnymi względami minikomputery były małymi komputerami typu *mainframe*: w wypadku aplikacji minikomputera całe przetwarzanie odbywało się w jego obrębie, a użytkownik prowadził

interakcję z aplikacją za pośrednictwem bardzo prostych terminali z zielonym ekranem. Jednakże nawet wtedy, gdy minikomputer stał się podstawą przetwarzania danych biznesowych, nie powstrzymało to nadchodzącej nowej rewolucji związanej z architekturą aplikacji.

Zwiększająca się dominacja platform minikomputerów opartych na standardzie IBM PC, a także pojawienie się graficznych interfejsów użytkownika, takich jak obecny w systemie Microsoft Windows, przyczyniły się do powstania nowego modelu aplikacji, czyli modelu klient-serwer. W modelu tym logika warstwy prezentacji została umiejscowiona na terminalu PC pracującym zwykle pod kontrolą systemu Microsoft Windows. Takie programy klienckie bazujące na komputerze PC komunikowały się z serwerem bazy danych, który zazwyczaj działał na minikomputerze. Logika warstwy aplikacji była często skoncentrowana po stronie klienta, ale mogła też być zlokalizowana w obrębie serwera bazy danych, gdy korzystano z **procedur przechowywanych**, czyli programów działających wewnątrz bazy danych.

Architektura klient-serwer zapewniła bogactwo możliwości użytkowych, które były niespotykane w czasach „zielonego ekranu". Z początkiem lat 90. w zasadzie wszystkie nowe aplikacje aspirowały do tego, by być zgodne z tą architekturą. Praktycznie we wszystkich platformach do tworzenia klientów używano serwera RDBMS. I w rzeczy samej język SQL był zazwyczaj uznawany za „wehikuł" dla wszystkich żądań między klientem i serwerem.

Programowanie obiektowe i system OODBMS

Niedługo po rewolucji w postaci modelu klient-serwer na powszechnie używane języki do tworzenia aplikacji wpłynęło przejście do kolejnego znaczącego modelu. W tradycyjnych, proceduralnych językach programowania dane i logika były w zasadzie rozdzielone. Procedury ładowały i przetwarzały dane w obrębie własnej logiki, ale sama procedura nie uwzględniała danych w żaden istotny sposób. **Programowanie obiektowe** dokonało scalenia w jeden obiekt atrybutów i działań. Oznacza to na przykład, że obiekt pracownika może reprezentować strukturę rekordów pracowników, a także operacje, które mogą być wykonywane na tych rekordach (zmiana pensji, awans, przejście na emeryturę itp.). W niniejszym omówieniu dwiema najistotniejszymi zasadami programowania obiektowego są:

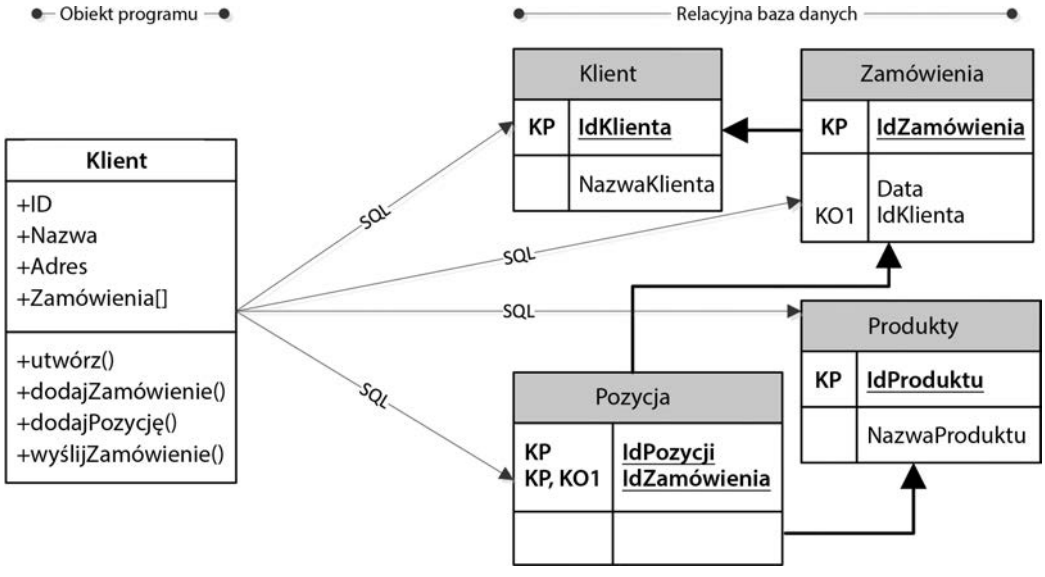
- **Hermetyzacja**. Klasa obiektów hermetyzuje zarówno dane, jak i działania (metody), które mogą być wykonywane na tych danych. Okazuje się, że obiekt może ograniczyć bezpośredni dostęp do bazowych danych, wymagając, aby modyfikacje danych były możliwe tylko za pośrednictwem metod obiektu. Na przykład klasa pracowników może uwzględniać metodę uzyskującą wysokość pensji oraz kolejną metodę modyfikującą tę wartość. Metoda ta może zawierać ograniczenia dotyczące minimalnej i maksymalnej wysokości pensji, a klasa może nie zezwalać na żadne modyfikowanie wysokości pensji poza obrębem tych metod.
- **Dziedziczenie**. Klasy obiektów mogą dziedziczyć właściwości klasy nadrzędnej. Klasa pracowników może dziedziczyć wszystkie właściwości klasy osób (datę urodzenia, imię itp.) w trakcie dodawania właściwości i metod, takich jak wysokość pensji czy data zatrudnienia.

Programowanie obiektowe oznaczało ogromne zwiększenie efektywności pracy programistów, niezawodności aplikacji i wydajności. W latach 80. i na początku lat 90. większość języków programowania została dostosowana do modelu obiektowego, a także pojawiło się wiele znaczących nowych języków, takich jak Java, które z założenia były obiektowe.

Rewolucja związana z programowaniem obiektowym przygotowała grunt pod pierwsze poważne wyzwanie postawione przed relacyjną bazą danych, które pojawiło się w połowie lat 90. Projektanci używający języków obiektowych byli sfrustrowani tym, co uznali za wyjątkowo kłopotliwą niezgodność między reprezentacjami obiektowymi swoich danych wewnątrz tworzonych programów a reprezentacją relacyjną w bazie danych. W programie obiektowym wszystkie szczegóły związane z roboczą jednostką logiczną byłyby przechowywane wewnątrz jednej klasy lub bezpośrednio z nią połączone. Na przykład obiekt klienta zawierałby wszystkie szczegóły dotyczące klienta z łączami prowadzącymi do obiektów zawierających zamówienia klienta. Z kolei te obiekty zawierałyby łącza do pozycji zamówienia. Taka reprezentacja z natury

była nierelacyjna. Tak naprawdę reprezentacja danych w większym stopniu była zgodna z sieciowymi bazami danych z czasów konsorcjum CODASYL.

Gdy obiekt był zapisywany w relacyjnej bazie danych lub z niej pobierany, niezbędnych było wiele operacji SQL do dokonania konwersji reprezentacji obiektowej na reprezentację relacyjną. Było to kłopotliwe dla programisty i mogło spowodować problemy związane z wydajnością lub niezawodnością. Problem ten zilustrowano na rysunku 1.5.



Rysunek 1.5. *Zapisywanie obiektu w systemie RDBMS wymaga wykonania wielu operacji SQL*

Zwolennicy programowania obiektowego zaczęli postrzegać relacyjną bazę danych jako relikt proceduralnej przeszłości. Spowodowało to powstanie raczej niechlubnego powiedzenia: „Relacyjna baza danych jest jak garaż, który zmusza do rozłożenia samochodu na części i przechowywania ich w małych szufladkach".

Szybkie powodzenie programowania obiektowego niemal nieuchronnie doprowadziło do stwierdzenia, że system **OODBMS** (ang. *Object Oriented Database Management System*) jest lepiej dostosowany do spełnienia wymagań nowoczesnych aplikacji. System ten był w stanie przechowywać obiekty programu bezpośrednio bez normalizacji, a ponadto umożliwiał aplikacjom łatwe ładowanie i zapisywanie ich obiektów. Przejście do obiektowej bazy danych spowodowało powstanie manifestu, w którym wyszczególniono kluczowe argumenty popierające system OODBMS oraz jego właściwości[4]. Pod względem implementacji system OODBMS przypomina model nawigacyjny z czasów przed pojawieniem się modelu relacyjnego — wskaźniki w obrębie jednego obiektu (na przykład obiekt klienta) umożliwiały przejście do powiązanych obiektów, takich jak obiekty zamówień.

Wsparcie modelu systemu OODBMS zwiększyło się w połowie lat 90. Dla wielu osób naturalne wydało się, że system ten będzie logicznym sukcesorem systemu RDBMS. Istniejący w tamtym czasie dostawcy relacyjnych baz danych (chodzi głównie o firmy Oracle, Informix, Sybase i IBM) szybko zaczęli stosować elementy systemu OODBMS w swoich systemach RDBMS. Tymczasem zaprojektowanych zostało kilka czystych systemów OODBMS, które zyskały początkowo uznanie.

Jednak pod koniec dekady systemom OODBMS zupełnie nie powiodło się zwiększenie udziału w rynku. Główni dostawcy baz danych, tacy jak Oracle i Informix, z powodzeniem zaimplementowali wiele elementów systemu OODBMS, ale nawet one były rzadko używane. Programiści korzystający z języków obiektowych ponownie sięgali po systemy RDBMS, aby zapewnić trwałość obiektów.

Utrudnienia zostały w pewnym stopniu zniwelowane za pomocą środowisk **ORM** (ang. *Object-Relational Mapping*), które automatyzowały większość uciążliwych aspektów translacji.

Istnieją rywalizujące ze sobą i niekoniecznie sprzeczne wyjaśnienia niepowodzenia obiektowych baz danych. Osobiście uważam, że zwolennicy modelu systemu OODBMS skoncentrowali się wyłącznie na zaletach, jakie oferował on projektantowi aplikacji, ignorując wady nowego modelu mające znaczenie dla osób, które zamierzały wykorzystywać informacje do celów biznesowych. Bazy danych nie mają po prostu zapewniać korzyści jedynie programistom. Reprezentują znaczące zasoby, które muszą być dostępne dla osób, które wymagają informacji w celu podejmowania decyzji i przeprowadzania analiz biznesowych. Implementując model danych, który mógł być używany tylko przez programistę, i pozbawiając użytkownika biznesowego przydatnego interfejsu SQL, system OODBMS nie zyskał uznania poza kręgami programistów.

Jak się jednak okaże w rozdziale 4., motywy związane z systemem OODBMS w dużym stopniu wpłynęły na kilka najpopularniejszych nierelacyjnych baz danych, które są obecnie stosowane.

Okres stabilizacji relacyjnych baz danych

Podekscytowanie związane z obiektowymi bazami danych minęło, a relacyjne bazy danych pozostały niezagrożone aż do kolejnych lat po 2005 r. Okazuje się, że w okresie liczącym w przybliżeniu 10 lat (od 1995 r. do 2005 r.) nie pojawiły się żadne nowe znaczące bazy danych. Dostępna była już wystarczająca liczba systemów RDBMS, aby nasycić rynek. Kontrola, jaką na rynku sprawował model systemu RDBMS, oznaczała, że nie mogły zaistnieć żadne alternatywy w postaci modeli nierelacyjnych. Biorąc pod uwagę, że okres ten w zasadzie jest czasem, w którym Internet przeobraził się z ciekawostki dla informatyków w wszechobecną sieć globalną, zaskakujące jest to, że nie pojawiły się w tym czasie żadne nowe architektury baz danych. Świadczy to o sile modelu relacyjnego.

Trzecia rewolucja związana z bazami danych

Około 2005 r. relacyjne bazy danych wydawały się mieć całkowicie ugruntowaną pozycję. Począwszy od tego roku, choć widoczne były ciągłe i znaczące innowacje istniejących wtedy systemów relacyjnych baz danych, nie było żadnych oznak radykalnych zmian. Okazuje się jednak, że era całkowitej supremacji relacyjnych baz danych dobiegała właśnie końca.

W szczególności różnica między architekturami aplikacji używanymi w czasach modelu klient-serwer i w czasach ogromnych aplikacji internetowych spowodowała obciążenia dotyczące relacyjnej bazy danych, które nie mogły zostać wyeliminowane drogą stopniowej innowacji.

Google i Hadoop

Około 2005 r. witryna internetowa firmy Google była największa na świecie. Taki stan rzeczy utrzymywał się przez kilka lat od powstania tej firmy. Gdy rozpoczynała ona swoją działalność, relacyjna baza danych miała już dobrze ugruntowaną pozycję. Nie była jednak odpowiednia do tego, by poradzić sobie z takimi ilościami danych (a także z ich zmiennością), z jakimi miała do czynienia firma Google. Wyzwania, przed jakimi stoją obecnie przedsiębiorstwa w związku z „dużymi danymi", to problemy, z którymi firma Google borykała się po raz pierwszy niemal 20 lat temu. Bardzo wcześnie firma ta musiała zaprojektować nowe architektury sprzętowe i programowe, aby przechowywać i przetwarzać dane witryn internetowych wymagających zindeksowania, których liczba zwiększała się w tempie wykładniczym.

W 2003 r. firma Google zaprezentowała szczegóły rozproszonego systemu plików **GFS**, który stanowił podwaliny opracowanej przez nią architektury magazynowania[5]. W 2004 r. firma ujawniła szczegóły algorytmu rozproszonego przetwarzania równoległego **MapReduce**, który został wykorzystany do tworzenia indeksów witryn internetowych[6]. W 2006 r. firma przedstawiła szczegóły rozproszonej strukturalnej bazy danych **BigTable**[7].

Rozwiązania te wraz z innymi technologiami, z których wiele również zostało stworzonych przez firmę Google, uformowały bazę projektu **Hadoop**. Dojrzewał on najpierw w firmie Yahoo!, a później, począwszy od 2007 r., doświadczył nagłego przyspieszenia prac. Bardziej niż cokolwiek innego system Hadoop zapewnił możliwości technologiczne na potrzeby systemu Big Data, który bardziej szczegółowo zostanie omówiony w rozdziale 2.

Reszta witryn internetowych

Choć pod względem ogólnej skali działania i ilości danych firma Google znacznie wyprzedziła jakąkolwiek inną firmę internetową, także inne witryny internetowe miały swoje wyzwania. Witryny stworzone na potrzeby handlu internetowego, takie jak Amazon, wymagały możliwości przetwarzania transakcyjnego na masową skalę. Pierwsze witryny społecznościowe, takie jak MySpace i ostatecznie Facebook, stanęły przed podobnymi wyzwaniami związanymi ze skalowaniem swojej infrastruktury z poziomu tysięcy do milionów użytkowników.

Ponownie nawet najdroższy komercyjny system RDBMS, taki jak Oracle, nie był w stanie zapewnić wystarczającej skalowalności spełniającej wymagania tych witryn. W ramach architektury systemu RDBMS firmy Oracle z rozszerzoną skalowalnością (Oracle RAC) podjęto próbę zapewnienia planu związanego z nieograniczoną skalowalnością, ale z ekonomicznego punktu widzenia było to nieatrakcyjne rozwiązanie, które raczej nigdy nie mogło oferować skali niezbędnej największym witrynom internetowym.

W wypadku wielu wczesnych witryn internetowych próbowano skalować bazy danych *open source* z wykorzystaniem różnych technik typu „zrób to sam". Obejmowało to zastosowanie rozproszonych systemów obiektowych, takich jak Memcached, które zmniejszały obciążenie baz danych, replikowanie baz danych w celu rozłożenia obciążenia związanego z operacjami odczytu bazy. Gdy wszystko inne zawiodło, ostatecznie sięgano po technikę ***shardingu***.

Sharding polega na partycjonowaniu danych w wielu bazach danych na podstawie atrybutu klucza, takiego jak identyfikator klienta. Na przykład w wypadku serwisów Twitter i Facebook dane klientów są dzielone między bardzo dużą liczbę baz danych MySQL. Większość danych powiązanych z konkretnym użytkownikiem trafia ostatecznie do jednej bazy danych, dlatego szybkie są operacje dotyczące określonego klienta. Zadaniem aplikacji jest zidentyfikowanie właściwego segmentu (ang. *shard*) i odpowiednie skierowanie żądań.

Witrynom takim jak serwis Facebook *sharding* umożliwił skalowanie systemu opartego na bazie danych MySQL do ogromnych poziomów, ale związane z tym mankamenty są poważne. Traconych jest wiele operacji relacyjnych i transakcji ACID na poziomie bazy danych. Niemożliwe staje się w wypadku wielu segmentów wykonywanie operacji złączeń lub utrzymywanie integralności transakcyjnej. Koszty operacyjne towarzyszące technice *shardingu* w połączeniu z utratą elementów relacyjnych sprawiły, że wiele osób szukało alternatyw dla systemu RDBMS.

Tymczasem podobny dylemat, z jakim miała do czynienia firma Amazon, spowodował zaprojektowanie przez nią alternatywnego modelu dla ścisłego modelu spójności transakcji ACID używanego w magazynie danych firmy. W 2008 r. Amazon zaprezentował szczegóły tego modelu o nazwie Dynamo[8].

Model Dynamo firmy Amazon wraz z innowacjami projektantów aplikacji internetowych szukających bazy danych dostosowanej do skali tych aplikacji doprowadził do powstania tego, co stało się znane jako **bazy danych klucz-wartość**. Bardziej szczegółowo zostaną one omówione w rozdziale 3.

Chmura obliczeniowa

Istnienie aplikacji i baz danych „w chmurze", czyli dostępnych w internecie, było od końca lat 90. stałym elementem krajobrazu tworzonego przez aplikacje. Jednakże około 2008 r. nastąpił gwałtowny wzrost popularności technologii chmury obliczeniowej, co było poważnym zmartwieniem dla dużych organizacji i ogromną możliwością dla zupełnie nowych firm.

We wcześniejszym okresie, wynoszącym od 5 do 10 lat, w wypadku aplikacji komputerowych miało miejsce ogólne przejście z rozbudowanych, tradycyjnych aplikacji opartych na modelu klient-serwer do aplikacji internetowych, których magazyny danych i serwery znajdowały się w miejscu dostępnym za pośrednictwem internetu — „chmury". Stworzyło to poważne wyzwanie dla powstających dopiero firm, które musiały w jakiś sposób zapewnić wystarczającą obsługę swoim pierwszym klientom, a także zagwarantować możliwość szybkiego skalowania w sytuacji, gdy będzie mieć miejsce bardzo pożądany wzrost liczby klientów w tempie wykładniczym.

Między rokiem 2006 a 2008 firma Amazon wdrożyła platformę **EC2** (ang. *Elastic Compute Cloud*). Udostępniała ona obrazy maszyn wirtualnych przechowywane w obrębie infrastruktury sprzętowej Amazona. Obrazy były dostępne za pośrednictwem internetu. Platforma mogła być używana do udostępniania aplikacji internetowych, a jej możliwości obliczeniowe w razie potrzeby mogły być dość szybko zwiększane. Firma Amazon dodała inne usługi, takie jak magazyn (S3, EBS), chmura VPC (ang. *Virtual Private Cloud*) czy usługa MapReduce (EMR). Cała platforma znana jako **AWS** (ang. *Amazon Web Services*) była pierwszą praktyczną implementacją technologii chmury **IaaS** (ang. *Infrastructure as a Service*). Platforma ta stała się inspiracją dla produktów chmury obliczeniowej stworzonych przez Google, Microsoft oraz inne firmy.

Dla aplikacji wymagających wykorzystania elastycznej skalowalności zapewnianej przez platformy chmury obliczeniowej istniejące relacyjne bazy danych były kiepską propozycją. Podejmowane przez firmę Oracle próby zintegrowania technologii siatki obliczeniowej (ang. *grid computing*) z jej architekturą zakończyły się jedynie ograniczonym sukcesem, a ponadto z ekonomicznego i praktycznego punktu widzenia były nieodpowiednie dla takich aplikacji, które wymagały możliwości rozszerzania na żądanie. Takie zapotrzebowanie na elastycznie skalowane bazy danych było wzmacniane przez nowo powstające firmy internetowe, a ponadto spowodowało przyspieszenie wzrostu popularności magazynów klucz-wartość, które często bazowały na projekcie modelu Dynamo firmy Amazon. W rzeczy samej począwszy od bazy danych **SimpleDB**, która ostatecznie została zastąpiona przez bazę danych **DynamoDB**, firma ta oferowała usługi nierelacyjne w ramach swojej chmury obliczeniowej.

Bazy danych dokumentów

Programiści w dalszym ciągu nie byli zadowoleni z wyjątkowo kłopotliwej niezgodności między modelem obiektowym a modelem relacyjnym. Systemy mapowania obiektowo-relacyjnego zmniejszały jedynie w niewielkim zakresie niedogodności pojawiające się, gdy złożony obiekt wymagał przechowywania w relacyjnej bazie danych w postaci znormalizowanej.

Począwszy od około 2004 r. coraz większa liczba witryn internetowych była w stanie oferować znacznie bogatsze możliwości interaktywne, niż miało to miejsce wcześniej. Było to możliwe dzięki stylowi programowania znanemu jako **AJAX** (ang. *Asynchronous JavaScript and XML*), w wypadku którego kod JavaScript w obrębie przeglądarki komunikował się bezpośrednio z serwerem, przesyłając komunikaty XML. Wkrótce XML został zastąpiony przez standard **JSON** (ang. *JavaScript Object Notation*), który jest formatem samoopisującym, podobnym do formatu XML, lecz bardziej zwartym i ściśle zintegrowanym z językiem JavaScript.

JSON stał się *de facto* formatem używanym do zapisywania (serializacji) obiektów na dysku. W niektórych witrynach internetowych rozpoczęto zapisywanie dokumentów JSON bezpośrednio w kolumnach tabel relacyjnych. Kwestią czasu było jedynie to, że ktoś zdecyduje się na wyeliminowanie relacyjnego pośrednika i utworzenie bazy danych, w której dane JSON będą mogły być bezpośrednio zapisywane. Tego rodzaju bazy stały się znane jako **bazy danych dokumentów**.

Couchbase i **MongoDB** to dwie popularne bazy danych oparte na formacie JSON, choć jest on obsługiwany w zasadzie przez wszystkie nierelacyjne bazy danych, a także przez większość relacyjnych baz danych. Programiści lubią bazy danych dokumentów z tych samych powodów, dla których polubili system OODBMS, czyli z racji wyeliminowania żmudnego procesu translacji obiektów do postaci relacyjnej. Bazom danych dokumentów bardziej szczegółowo przyjrzymy się w rozdziale 4.

NewSQL

Ani model relacyjny, ani model transakcji ACID nie narzucał architektury fizycznej w wypadku relacyjnej bazy danych. Po części jednak w wyniku odziedziczonego rodowodu, a po części wskutek realiów ówczesnego sprzętu większość relacyjnych baz danych była implementowana w bardzo podobny sposób. Format danych na dysku, użycie pamięci, natura blokad i tym podobne kwestie zmieniały się tylko nieznacznie w głównych implementacjach systemów RDBMS.

W 2007 r. Michael Stonebraker, pionier systemów baz danych Ingres i Postgres, pokierował zespołem badawczym, który opublikował nowatorską pracę *The End of an Architectural Era (It's Time for a Complete Rewrite)*[9]. W pracy tej wskazano, że nie mają już zastosowania założenia sprzętowe, które stanowią fundament uznanej ogólnie architektury modelu relacyjnego. Ponadto stwierdzono, że różne obciążenia nowoczesnej bazy danych sugerują, że pojedyncza architektura nie może być optymalna dla każdego obciążenia.

Stonebraker i jego zespół zaproponowali kilka wariantów w odniesieniu do istniejącego projektu systemu RDBMS, z których każdy był zoptymalizowany pod kątem konkretnego obciążenia generowanego przez aplikację. Szczególnie znaczące były dwa spośród tych wariantów projektowych (choć prawdę powiedziawszy żaden z nich niekoniecznie był dotąd zupełnie niespotykany). W wariancie **H-Store** opisano całkowicie rozproszoną bazę danych z przetwarzaniem w pamięci, natomiast w wariancie **C-Store** określono projekt przewidziany dla kolumnowej bazy danych. Oba te warianty projektowe wywarły ogromny wpływ w kolejnych latach, a ponadto są pierwszymi przykładami tego, co stało się znane jako systemy baz danych **NewSQL**. W ich wypadku zachowano kluczowe właściwości systemu RDBMS, ale dokonano odejścia od typowej architektury oferowanej przez tradycyjne systemy, takie jak Oracle i SQL Server. Tego typu bazy danych zostaną omówione w rozdziałach 6. i 7.

Eksplozja nierelacyjnych baz danych

Jak widać na rysunku 1.1, ogromna liczba systemów relacyjnych baz danych pojawiła się między rokiem 2000 a 2005. W szczególności w latach 2008 – 2009 nastąpiło coś w rodzaju „eksplozji kambryjskiej": w tym czasie pojawiło się dosłownie dziesiątki nowych systemów baz danych. Wiele z nich wyszło z użycia, ale niektóre, takie jak MongoDB, Cassandra i HBase, mają obecnie znaczny udział w rynku.

Początkowo tej nowej grupie systemów baz danych nie nadano wspólnej nazwy. Zaproponowano termin „systemy zarządzania rozproszonymi, nierelacyjnymi bazami danych" (**DNRDBMS** — ang. *Distributed Non-Relational Database Management System*), ale oczywiście nie miał on uwzględniać wszystkich możliwych opcji. Jednakże pod koniec 2009 r. szybko przyjął się termin **NoSQL** jako skrót odnoszący się do każdego systemu baz danych, który zrywał z tradycyjną bazą danych SQL.

W opinii wielu osób termin NoSQL jest niefortunny: definiuje on, czym baza danych nie jest, a nie, czym jest. Ponadto zwraca uwagę na obecność języka SQL lub jego brak. Choć prawdą jest, że większość systemów nierelacyjnych baz danych nie obsługuje tego języka, w rzeczywistości stanowi on element odróżniający je od ścisłego, transakcyjnego i relacyjnego modelu danych, który był motywacją do powstania większości projektów baz danych NoSQL.

Około 2011 r. zyskał popularność termin **NewSQL**, który służył do opisywania tej nowej grupy baz danych. Choć nie stanowiły one całkowitego odejścia od modelu relacyjnego, poszerzały lub znacznie modyfikowały fundamentalne zasady. Grupa ta obejmuje kolumnowe bazy danych omówione w rozdziale 6., a także niektóre bazy danych z przetwarzaniem w pamięci, które zaprezentowano w rozdziale 7.

I wreszcie, na początku 2012 r. do masowej świadomości trafił termin **Big Data**. Choć odnosi się on głównie do nowych metod wykorzystywania danych w celu uzyskania jakiejś wartości, określenie „rozwiązania Big Data" jest ogólnie rozumiane jako wygodne uproszczenie w wypadku technologii, takich jak Hadoop, które obsługują duże, pozbawione struktury zbiory danych.

■ **Uwaga** Pod wieloma względami terminy NoSQL, NewSQL i Big Data są niejasno zdefiniowane, przesadnie rozreklamowane i zbytnio „przeładowane". Reprezentują one jednak najpowszechniej zrozumiałe frazy odnoszące się do technologii baz danych następnej generacji.
Ogólnie mówiąc, bazy danych NoSQL odrzucają ograniczenia modelu relacyjnego, w tym ścisłą spójność i schematy. W bazach danych NewSQL zachowano wiele elementów modelu relacyjnego, ale w znaczący sposób ulepszono w nich bazową technologię. Systemy Big Data są zasadniczo ukierunkowane na technologie obecne w ekosystemie Hadoop, gdzie w coraz większym stopniu uwzględniana jest platforma Spark.

Podsumowanie: jeden rozmiar nie pasuje wszystkim

Pierwsza rewolucja związana z bazami danych była nieuchronną konsekwencją pojawienia się elektronicznych komputerów cyfrowych. Pod pewnym względem bazy danych pierwszej fali były elektronicznymi odpowiednikami analogowych rozwiązań, takich jak karty dziurkowane i maszyny licząco-analityczne, z czasów przed pojawieniem się technologii komputerowych. Pierwsze próby dodania warstwy struktury i spójności do tych baz danych mogły poprawić efektywność pracy programistów i spójność danych, ale spowodowały pozostawienie danych zamkniętych w systemach, do których kluczami dysponowali wyłącznie programiści.

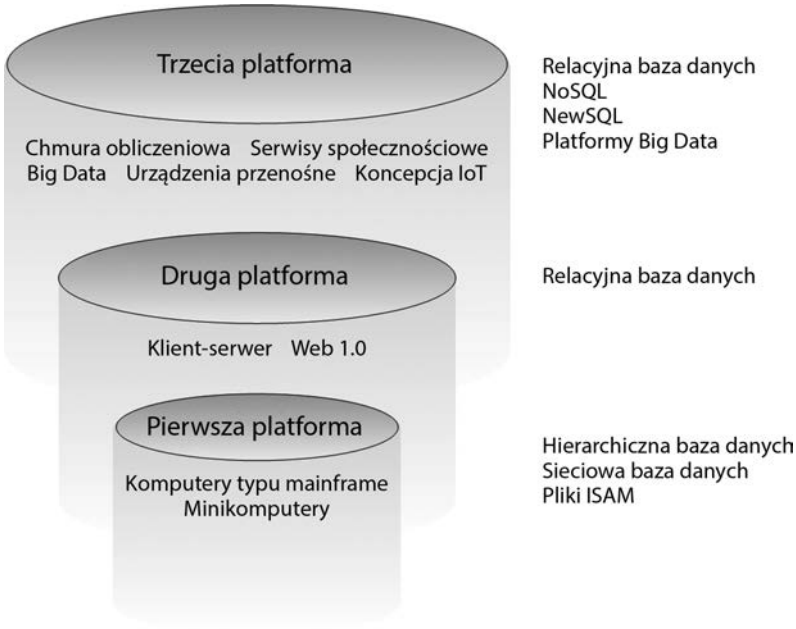
Druga rewolucja związana z bazami danych była wynikiem uświadomienia sobie przez Edgara Codda tego, że systemy baz danych byłyby dobrze obsługiwane, gdyby bazowały na solidnym, formalnym i matematycznym fundamencie, a ponadto tego, że reprezentacja danych powinna być niezależna od implementacji fizycznego magazynu, a bazy danych powinny obsługiwać elastyczne mechanizmy zapytań, które nie wymagają zaawansowanych umiejętności programistycznych.

Pomyślny rozwój nowoczesnej relacyjnej bazy danych przez tak długi czas (ponad 30 lat dominacji na rynku komercyjnym) świadczy o triumfie informatyki i inżynierii oprogramowania. Rzadko się zdarzało, aby teoretyczne zagadnienie związane z oprogramowaniem odniosło taki sukces i było powszechnie stosowane, tak jak miało to miejsce w wypadku relacyjnej bazy danych.

Trzecia rewolucja związana z bazami danych nie opera się na jednym założeniu dotyczącym architektury. Jeśli można tutaj przywołać jakiekolwiek założenie, jest to bazowanie na stwierdzeniu, zgodnie z którym pojedyncza architektura baz danych nie może spełniać wymagań stawianych w obecnym nowoczesnym świecie cyfrowym. Istnienie ogromnych aplikacji społecznościowych z setkami milionów użytkowników oraz pojawienie się aplikacji *IoT* (ang. *Internet of Things*), które potencjalnie będą obsługiwać miliardy wejść sprzętowych, przybliża możliwości relacyjnych baz danych, a zwłaszcza modelu transakcji ACID, do punktu krytycznego. Na drugim końcu skali znajdują się aplikacje, które muszą działać na urządzeniach przenośnych z ograniczoną ilością pamięci i mocą obliczeniową. Mamy do czynienia z zalewem danych, gdzie spora część ma nieprzewidywalną strukturę, w wypadku której przekształcanie w postać relacyjną nie znajduje żadnego uzasadnienia.

Trzecia fala baz danych w przybliżeniu odpowiada trzeciej fali aplikacji komputerowych. IDC oraz inne firmy często posługują się tutaj terminem „trzecia platforma". Pierwszą platformą był komputer typu *mainframe*, który był obsługiwany przez systemy baz danych istniejących przed pojawieniem się modelu relacyjnego. Druga platforma, czyli aplikacje klient-serwer i pierwsze aplikacje internetowe, była obsługiwana przez relacyjne bazy danych. Trzecia platforma charakteryzowana jest przez aplikacje uwzględniające wdrażanie w chmurze obliczeniowej, obsługę urządzeń przenośnych, serwisy społecznościowe i koncepcję IoT. Trzecia platforma wymaga trzeciej fali technologii baz danych obejmujących systemy relacyjne, lecz nie tylko takie. Na rysunku 1.6 podsumowano to, jak trzy platformy odpowiadają trzem przedstawionym falom rewolucji związanej z bazami danych.

Obecnie praca w branży baz danych jest ekscytującym zajęciem. Dla generacji twórców oprogramowania (okres ten obejmuje większość mojego życia zawodowego) innowacja technologii baz danych miała miejsce głównie w ramach relacyjnych baz danych zgodnych z modelem transakcji ACID. Obecnie hegemonia systemów RDBMS dobiegła końca. Możemy bez ograniczeń projektować systemy baz danych,



Rysunek 1.6. *Opracowany przez firmę IDC model trzech platform odpowiada trzem falom technologii baz danych*

w wypadku których jedynym ograniczeniem jest nasza wyobraźnia. Dobrze wiadomo, że porażka napędza innowację. Część tych nowych zagadnień związanych z systemami baz danych nie przetrwa próby czasu. Wydaje się jednak, że są niewielkie szanse na to, że pojedynczy model zdominuje najbliższą przyszłość w tak dużym stopniu, jak wcześniej model relacyjny. Specjaliści od baz danych będą musieli starannie wybrać technologię, która będzie najbardziej odpowiednia do ich potrzeb. W wielu sytuacjach technologia relacyjnych baz danych nadal będzie najlepszą opcją, lecz nie zawsze tak będzie.

W kolejnych rozdziałach przyjrzymy się każdej z podstawowych kategorii systemów baz danych następnej generacji. Dowiemy się dokładnie, jakie są związane z nimi nadzieje, jakie są ich architektury, a także jakie są ich możliwości spełniania wymagań stawianych przez nowoczesne systemy aplikacji.

Źródła

1. *http://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf*
2. William Kent, *A Simple Guide to Five Normal Forms in Relational Database Theory*, 1983.
3. *http://research.microsoft.com/en-us/um/people/gray/papers/theTransactionConcept.pdf*
4. *https://www.cs.cmu.edu/~clamen/OODBMS/Manifesto/*
5. *http://research.google.com/archive/gfs.html*
6. *http://research.google.com/archive/mapreduce.html*
7. *http://research.google.com/archive/bigtable.html*
8. *http://queue.acm.org/detail.cfm?id=1466448*
9. *http://nms.csail.mit.edu/~stavros/pubs/hstore.pdf*

Skorowidz

C

D

E

F

G

H

I

J

K

L

Ł

M

N

O

P

R

S

Ś

T

U

V

W

X

Y

Z

Ż

NoSQL i BigData: potężne bazy danych przyszłości!

Model relacyjnej bazy danych zdecydowanie dominował wśród technologii bazodanowych przez ostatnie 20 lat. Poszczególne rozwiązania były do siebie na tyle podobne, że decyzja o zastosowaniu relacyjnej bazy danych stała się oczywista. Architektura rozwiązań tego typu była zbliżona, a różnice polegały głównie na koszcie wdrożenia, wydajności, niezawodności i łatwości użycia aplikacji. Obecnie sytuacja diametralnie się zmieniła: powstało wiele radykalnie różniących się od siebie technologii bazodanowych, a wybór właściwej bazy danych stał się złożonym zadaniem, wymagającym sporej wiedzy i obarczonym poważnymi konsekwencjami natury ekonomicznej i technologicznej.

Ta książka szczególnie przyda się architektom technologii informatycznych, administratorom baz danych i projektantom, którzy do wykonywania swoich obowiązków potrzebują wiedzy o najświeższych rozwiązaniach z dziedziny technologii baz danych. Omówiono tu najnowsze, wykorzystywane obecnie technologie baz danych. Wyjaśniono, w jakim celu zaprojektowano każdą z nich. Zaprezentowano możliwości poszczególnych baz danych oraz ich potencjał w rozwiązywaniu realnych problemów biznesowych i problemów z aplikacjami. Co najważniejsze, ukazano różnice w architekturze między technologiami, które mają kluczowe znaczenie przy wyborze platformy baz danych dla nowych i planowanych projektów.

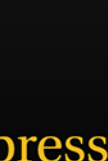
W tej książce między innymi:

- Co zrewolucjonizowało bazy danych
- Google, Hadoop i koncepcja BigData
- Pamięciowe i rozproszone bazy danych
- NoSQL, CQL i nowe odsłony SQL
- Hybrydowe bazy danych Oracle

Guy Harrison projektuje bazy danych od połowy lat 80. zeszłego stulecia i stał się niekwestionowanym autorytetem w tej dziedzinie. Napisał wiele książek poświęconych projektowaniu baz danych i optymalizacji ich wydajności. Obecnie kieruje zespołem rozwijającym rodziny produktów Toad, Spotlight i Shareplex w firmie Dell. Mieszka w Melbourne w Australii ze swoją żoną, zmienną liczbą dorosłych dzieci, kotem, trzema psami i ogromnym królikiem „zabójcą".

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

ISBN 978-83-283-4751-9



Cena: 54,90 zł

Apress®