Model relacyjny – model organizacji danych bazujący na matematycznej teorii mnogości, w szczególności na pojęciu relacji. Na modelu relacyjnym oparta jest **relacyjna baza danych** (ang. *Relational Database*) – baza danych, w której dane są przedstawione w postaci relacyjnej.W najprostszym ujęciu w modelu relacyjnym dane grupowane są w relacje, które reprezentowane są przez tabele. Relacje są pewnym zbiorem rekordów o identycznej strukturze wewnętrznie powiązanych za pomocą związków zachodzących pomiędzy danymi. Relacje zgrupowane są w tzw. schematy bazy danych. Relacją może być tabela zawierająca dane teleadresowe pracowników, zaś schemat może zawierać wszystkie dane dotyczące firmy. Takie podejście w porównaniu do innych modeli danych ułatwia wprowadzanie zmian, zmniejsza możliwość pomyłek, ale dzieje się to kosztem wydajności.

Podejście intuicyjne:

W modelu relacyjnym każda relacja (prezentowana w postaci np. tabeli) posiada unikatową nazwę, *nagłówek* i *zawartość*. Nagłówek relacji to zbiór atrybutów, gdzie atrybut jest parą *nazwa_atrybutu:nazwa_typu*, zawartość natomiast jest zbiorem krotek (reprezentowanych najczęściej w postaci wiersza w tabeli). W związku z tym, że nagłówek jest *zbiorem atrybutów* nie jest ważna ich kolejność. Atrybuty zazwyczaj utożsamiane są z kolumnami tabeli. Każda krotka (wiersz) wyznacza zależność pomiędzy danymi w poszczególnych komórkach (np. osoba o danym numerze PESEL posiada podane nazwisko i imię oraz adres)

Model relacyjny a SQL:

Większość współczesnych relacyjnych baz danych korzysta z jakiejś wersji języka SQL pozwalającego wprowadzać zmiany w strukturze bazy danych, jak również zmiany danych w bazie i wybieranie informacji z bazy danych. Język ten opiera się na silniku bazy danych, który pozwala zadawać w języku SQL pewnego rodzaju pytania (kwerendy) i wyświetlać dane, które spełniają warunki zapytania. Zapytania SQL mogą także wykonywać operacje wstawiania danych, usuwania danych i ich aktualizacji. Język SQL zapewnia również zarządzanie bazą danych. Informacja o samej bazie przechowywana jest w postaci relacji (tabel) wewnątrz bazy danych.

Przykład relacyjnej bazy danych

Oto prosty przykład dwóch tabel, których mała firma może używać do przetwarzania zamówień na swoje produkty. Pierwsza tabela służy do przechowywania informacji na temat klientów. Jej każdy rekord zawiera nazwę, adres, informacje wysyłkowe i rozliczeniowe, numer telefonu oraz inne dane kontaktowe klienta. Każdy fragment informacji (każdy atrybut) znajduje się w oddzielnej kolumnie, a baza danych przypisuje każdemu wierszowi unikatowy identyfikator (klucz). W drugiej tabeli — służącej do przechowywania zamówień klientów — każdy rekord zawiera identyfikator klienta, który złożył zamówienie, zamówiony produkt, ilość, wybrany rozmiar i kolor itd., ale nie zawiera imienia i nazwiska ani danych kontaktowych klienta.

Te dwie tabele mają tylko jedną wspólną cechę: kolumnę identyfikatora (klucz). Dzięki tej wspólnej kolumnie relacyjna baza danych może utworzyć relację między dwiema omawianymi tabelami. Wówczas, kiedy aplikacja przetwarzająca zamówienia firmy przesyła zamówienie do bazy danych, baza danych może przejść do tabeli zamówień klienta, pobrać poprawne informacje o zamówieniu produktu i użyć identyfikatora klienta z tej tabeli, aby

wyszukać informacje rozliczeniowe i wysyłkowe klienta w tabeli informacji o kliencie. Magazyn może następnie pobrać właściwy produkt, klient może otrzymać zamówienie w terminie, a firma może otrzymać zapłatę.

Jak są zorganizowane relacyjne bazy danych

Model relacyjny oznacza, że logiczne struktury danych — tabele danych, widoki i indeksy — są oddzielone od fizycznych struktur pamięci. Dzięki temu administratorzy baz danych mogą zarządzać fizycznym przechowywaniem danych bez wpływu na dostęp do tych danych jako struktury logicznej. Na przykład zmiana nazwy pliku bazy danych nie powoduje zmiany nazw przechowywanych w nim tabel.

Rozróżnienie między strukturą logiczną a fizyczną dotyczy również operacji na bazach danych, które są wyraźnie zdefiniowanymi działaniami umożliwiającymi aplikacjom manipulowanie danymi i strukturami bazy danych. Operacje logiczne umożliwiają aplikacji określenie potrzebnej zawartości, a operacje fizyczne ustalają, w jaki sposób należy uzyskać dostęp do danych, a następnie wykonują to zadanie.
Aby zapewnić zawsze maksymalną dokładność i dostępność danych, relacyjne bazy danych przestrzegają określonych reguł integralności. Reguła integralności może na przykład określać, że w tabeli nie są dozwolone duplikaty wierszy, eliminując w ten sposób możliwość wprowadzenia do bazy danych błędnych informacji.

Model relacyjny

We wczesnych latach rozwoju baz danych każda aplikacja zapisywała dane w swojej własnej unikatowej strukturze. Chcąc tworzyć aplikacje korzystające z tych danych, programiści musieli dobrze znać konkretną strukturę danych, aby znaleźć potrzebne dane. Te struktury danych były nieefektywne i trudne do utrzymania. Trudno je też było zoptymalizować, aby zapewnić wysoką wydajność aplikacji. Relacyjny model bazy danych zaprojektowano w celu rozwiązania problemu istnienia wielu arbitralnych struktur danych.

Relacyjny model danych zapewnił standardowy sposób reprezentowania i wysyłania zapytań dotyczących danych, z którego można było korzystać w każdej aplikacji. Od samego początku programiści uznali, że główną siłą relacyjnego modelu bazy danych było wykorzystywanie tabel, które oferowały intuicyjny, wydajny i elastyczny sposób przechowywania ustrukturyzowanych informacji oraz uzyskiwania do nich dostępu.

Z czasem ujawniła się jeszcze jedna zaleta modelu relacyjnego. Programiści mogli bowiem korzystać z języka SQL (structured query language, strukturalny język zapytań) przeznaczonego do zadawania pytań i wyszukiwania potrzebnych informacji w bazie danych. Przez wiele lat SQL był powszechnie używany jako język do formułowania zapytań do baz danych. Oparty na algebrze relacyjnej język SQL to wewnętrznie spójny język matematyczny, zapewniający wyższą wydajność wszystkich zapytań do baz danych. Dla porównania, inne podejścia wymagają definiowania pojedynczych zapytań.

Korzyści płynące z systemu zarządzania relacyjną bazą danych

Prosty, ale silny model relacyjny jest używany przez różnego rodzaju i różnej wielkości przedsiębiorstwa do zaspokajania szerokiej gamy potrzeb informacyjnych. Relacyjne bazy danych służą do śledzenia zapasów, przetwarzania transakcji w handlu elektronicznym,

zarządzania ogromnymi ilościami kluczowych informacji o klientach itd. Relacyjna baza danych może służyć do zaspokajania dowolnych potrzeb informacyjnych w sytuacjach, w których elementy danych są ze sobą powiązane i muszą być zarządzane w sposób bezpieczny, oparty na regułach i spójny.

Relacyjne bazy danych istnieją od lat 70-tych XX wieku. Zalety modelu relacyjnego sprawiają, że jest to nadal najszerzej akceptowany model baz danych.

Model relacyjny i spójność danych

Model relacyjny najskuteczniej zachowuje spójność danych w aplikacjach i kopiach baz danych (zwanych instancjami). Na przykład, gdy klient wpłaca pieniądze w bankomacie, a następnie sprawdza saldo konta w telefonie komórkowym, oczekuje, że natychmiast zobaczy ten depozyt w zaktualizowanym saldzie konta. Relacyjne bazy danych pozwalają zapewnić taką spójność danych, sprawiając, że wiele instancji bazy danych zawiera przez cały czas te same dane.

W przypadku innych typów baz danych utrzymanie przez cały czas takiego poziomu spójności przy dużej ilości danych jest trudne. Niektóre najnowsze bazy danych, takie jak NoSQL, mogą zapewniać tylko „spójność końcową”. Zgodnie z tą zasadą, gdy baza danych jest skalowana lub gdy wielu użytkowników korzysta z tych samych danych w tym samym czasie, dane potrzebują trochę czasu, aby „dogonić” stan aktualny. Spójność końcowa jest akceptowalna w przypadku niektórych zastosowań, takich jak prowadzenie list w katalogu produktów, ale w przypadku newralgicznych operacji biznesowych, takich jak transakcje związane z obsługą koszyka zakupów, „złotym standardem” jest nadal relacyjna baza danych.

Zatwierdzenie i niepodzielność

Relacyjne bazy danych obsługują reguły i zasady biznesowe na bardzo szczegółowym poziomie — obowiązują na przykład bardzo rygorystyczne zasady zatwierdzania (tj. wprowadzania zmian w bazie danych na stałe). Rozważmy na przykład bazę danych dotyczącą stanu zapasów, która monitoruje trzy zawsze używane razem części. Kiedy jedna z tych części jest pobierana z zapasów, muszą zostać pobrane również pozostałe dwie. Jeśli jedna z tych trzech części jest niedostępna, nie może zostać pobrana żadna z nich — aby baza danych dokonała zatwierdzenia, muszą być dostępne wszystkie trzy części. Relacyjna baza danych nie zatwierdzi do wydania jednej części, o ile nie może zatwierdzić wszystkich trzech. Ta wielopłaszczyznowość zatwierdzania jest nazywana niepodzielnością. Niepodzielność jest kluczem do zachowania dokładności danych w bazie danych i zapewnienia ich zgodności z zasadami, przepisami i polityką firmy.

Właściwości ACID a RDBMS

Cztery kluczowe właściwości definiują transakcje w relacyjnych bazach danych: niepodzielność, spójność, izolacja i trwałość — zwykle są one określane jako ACID (ang. atomicity, consistency, isolation, and durability).

- **Niepodzielność** określa wszystkie elementy składające się na kompletną transakcję bazy danych.
- **Spójność** określa zasady utrzymywania elementów danych we właściwym stanie po transakcji.

- **Izolacja** oznacza, że efekt transakcji będzie niewidoczny dla innych, dopóki nie zostanie ona zatwierdzona, aby uniknąć nieporozumień.
- **Trwałość** oznacza, że zmiany w danych stają się trwałe po zatwierdzeniu transakcji.

Procedury składowane i relacyjne bazy danych

Dostęp do danych obejmuje wiele powtarzających się czynności. Na przykład proste zapytanie, które ma zapewnić uzyskanie informacji z tabeli danych, może wymagać powtórzenia setek lub tysięcy razy, aby uzyskać pożądany wynik. Te funkcje dostępu do danych wymagają jakiegoś rodzaju kodu, aby uzyskać dostęp do bazy danych. Twórcy aplikacji nie chcą pisać nowego kodu dla tych funkcji w każdej nowej aplikacji. Na szczęście relacyjne bazy danych obsługują procedury składowane będące blokami kodu, do których można uzyskać dostęp poprzez proste wywołanie aplikacji. Na przykład pojedyncza procedura składowana może zapewnić spójne znakowanie rekordów dla użytkowników wielu aplikacji. Procedury składowane mogą również pomagać programistom w zadbaniu o to, aby określone funkcje danych w aplikacji zostały zaimplementowane w konkretny sposób.

Blokowanie bazy danych i współbieżność

Kiedy wielu użytkowników lub wiele aplikacji próbuje jednocześnie zmienić te same dane, w bazie danych mogą występować konflikty. Techniki blokowania i współbieżności zmniejszają ryzyko konfliktów przy jednoczesnym zachowaniu integralności danych.

Blokowanie uniemożliwia innym użytkownikom i aplikacjom dostęp do danych podczas ich aktualizacji. W niektórych bazach danych blokowanie dotyczy całej tabeli, co ma negatywny wpływ na wydajność aplikacji. Inne bazy danych, takie jak relacyjne bazy danych Oracle, stosują blokady na poziomie rekordu, umożliwiając dostęp do pozostałych rekordów w tabeli, co pomaga zapewnić lepszą wydajność aplikacji.

Współbieżność występuje wówczas, gdy wielu użytkowników lub wiele aplikacji jednocześnie wywołuje zapytania dotyczące tej samej bazy danych. Funkcja ta zapewnia odpowiedni dostęp użytkownikom i aplikacjom, zgodnie ze zdefiniowanymi zasadami kontroli danych.

Na co zwracać uwagę przy wyborze relacyjnej bazy danych

Oprogramowanie używane do przechowywania, wyszukiwania i pobierania danych przechowywanych w relacyjnej bazie danych oraz do zarządzania nimi to tzw. system zarządzania relacyjnymi bazami danych (RDBMS). RDBMS zapewnia interfejs między użytkownikami i aplikacjami a bazą danych, a także funkcje administracyjne do zarządzania przechowywaniem danych, dostępem i wydajnością.

Na wybór określonego typu bazy danych i produktów związanych z relacyjną bazą danych wypływa kilka czynników. Wybór RDBMS zależy od potrzeb biznesowych firmy. Należy sobie wtedy zadać następujące pytania:

- Jakie są nasze wymagania w zakresie dokładności danych? Czy przechowywanie i dokładność danych zależą od logiki biznesowej? Czy nasze dane są objęte rygorystycznymi wymaganiami dotyczącymi dokładności (na przykład dane finansowe i raporty dla organów administracji publicznej)?

- Czy potrzebujemy skalowalności? Jaka jest skala zarządzanych danych i jaki jest oczekiwany wzrost ich ilości? Czy model bazy danych będzie musiał obsługiwać lustrzane kopie bazy danych (jako osobne instancje) w celu skalowania? Jeśli tak, czy może utrzymać spójność danych w tych instancjach?
- Jak ważna jest współbieżność? Czy wielu użytkowników i wiele aplikacji będzie potrzebować jednoczesnego dostępu do danych? Czy oprogramowanie bazy danych obsługuje współbieżność, jednocześnie chroniąc dane?
- Jakie są nasze potrzeby w zakresie wydajności i niezawodności? Czy potrzebujemy produktu o wysokiej wydajności i niezawodności? Jakie są wymagania dotyczące wydajności odpowiedzi na zapytania? Jakie są zobowiązania dostawcy dotyczące umów o poziom usług (SLA) lub nieplanowanych przestojów?

Relacyjna baza danych przyszłości: samoczynna baza danych

Z biegiem lat relacyjne bazy danych stały się lepsze, szybsze, wydajniejsze i łatwiejsze w obsłudze. Ale są także bardziej skomplikowane, a administrowanie bazą danych od dawna jest zajęciem na pełen etat. Zamiast wykorzystywać swoją wiedzę i koncentrować się na opracowywaniu innowacyjnych aplikacji przynoszących firmie korzyści, programiści muszą spędzać większość czasu na działaniach związanych z zarządzaniem, niezbędnych do zapewnienia optymalnego działania bazy danych.

Oferowana obecnie technologia autonomiczna opiera się na mocnych stronach modelu relacyjnego, chmurowej bazy danych oraz uczenia maszynowego, udostępniając nowy typ relacyjnej bazy danych. Samoczynna baza danych (zwana również autonomiczną bazą danych) zachowuje moc obliczeniową i zalety modelu relacyjnego, ale wykorzystuje sztuczną inteligencję (AI), samouczenie się maszyn i automatyzację do monitorowania obsługi zapytań i zadań zarządzania oraz zwiększania jej wydajności. Na przykład aby poprawić wydajność obsługi zapytań, samoczynna baza danych może postawić hipotezę i przetestować indeksy w celu przyspieszenia tego procesu, a następnie przekazać najlepsze wyniki do produkcji — wszystko robi samodzielnie. Samoczynna baza danych wprowadza tego rodzaju usprawnienia nieustannie, nie angażując przy tym człowieka.
Technologia autonomiczna zwalnia programistów z wykonywania żmudnych zadań związanych z zarządzaniem bazą danych. Nie muszą oni na przykład z wyprzedzeniem definiować wymagań dotyczących infrastruktury. Zamiast tego, dzięki samoczynnej bazie danych, mogą swobodnie dodawać zasoby pamięci masowej i moc obliczeniową w miarę rozrastania się bazy danych. Programiści mogą łatwo, w zaledwie kilku krokach, utworzyć autonomiczną relacyjną bazę danych, przyspieszając czas tworzenia aplikacji.