

Wprowadzenie do relacyjnych baz¹

Historia serwera SQL Server

W latach 70-tych firma IBM stworzyła język programowania przeznaczony specjalnie do *zapytań* do bazy danych, zwany SEQUEL, co oznacza Structured English Query Language. Z czasem, język ten został poszerzony, i obecnie nie jest to jedynie język zapytań, ale może być także używany do tworzenia i zarządzania mechanizmami bezpieczeństwa bazy danych. Firma IBM udostępniła ogólnie SEQUEL, który od tej pory znany jest jako SQL. Ze względu na historię powstania słowo SQL można wymawiać jako *sequel* lub przeliterować jako S-Q-L.

Obecne motory baz danych używają różnych wersji języka SQL. Microsoft SQL Server używa wersji zwanej Transact-SQL (T-SQL).

Co to jest baza danych?

SQL Server używa bazy danych typu relacyjnego. W *relacyjnych bazach danych*, dane są zgrupowane w tabelach. Tabele są tworzone poprzez grupowanie danych z tego samego tematu i zawierają kolumny oraz wiersze informacji. Tabele są ze sobą wiązane za pomocą mechanizmów bazy danych, gdy uruchamiane jest zapytanie. Tabele są ściśle związane z pojęciami *relacji* lub *encji* stosowanymi w odpowiednich książkach teoretycznych.

Generalnie bazę danych można przedstawić jako zbiór powiązanych danych. W niektórych wcześniejszych systemach baz danych, baza danych była plikiem – takim jak `pracownicy.txt` zawierającym pojedynczą tabelę danych. Wewnątrz pliku `pracownicy.txt` znajdowały się kolumny odnoszące się do danych pracownika, takich jak pensja, data przyjęcia, nazwisko, numer legitymacji ubezpieczeniowej itp. Plik zawierał wiersz dla każdej osoby w firmie, z odpowiednimi wartościami w odpowiednich kolumnach. Indeksy, użyte do przyspieszenia dostępu do danych, były w odrębnym pliku, tak jak inne elementy dotyczące zabezpieczeń.

W serwerze SQL Server ..., baza danych nie koniecznie jest związana z pojedynczym plikiem; jest to pojęcie bardziej logiczne, oparte na zbiorze powiązanych obiektów. Przykładowo, baza danych na serwerze SQL Server zawiera nie tylko pierwotne dane, ale także strukturę bazy danych, wszelkie indeksy, zabezpieczenia bazy i być może inne obiekty takie jak widoki lub procedury składowane związane z określoną bazą.

Obiekty relacyjnej bazy danych

Relacyjna baza danych składa się z obiektów różnego typu. Poniżej przedstawiono kilka najpowszechniejszych obiektów:

- *Tabele* są obiektami zawierającymi typy danych i aktualne dane.
- *Kolumny* są częścią tabel przechowującą dane. Kolumny muszą posiadać określony typ danych i unikalną nazwę.
- *Typy danych* określają rodzaj przechowywanych danych. Można korzystać z wielu typów danych, takich jak typ znakowy, numeryczny i typ data. Każdej kolumnie w tabeli jest przypisany pojedynczy typ danych.
- *Procedury składowane* są podobne do makr stworzonych w kodzie języka Transact-SQL, które można napisać i zachować pod daną nazwą. Przez wykonywanie procedury zostaje faktycznie uruchomiony kod języka T-SQL zawarty w procedurze. Przykładowo, kod języka T-SQL stosowany do uruchamiania co tygodniowych raportów zostaje zachowany jako procedura, a następnie, w celu generacji raportu kod ten jest uruchamiany już jako procedura. Procedur składowanych można używać także jako mechanizmów zabezpieczeń.
- *Funkcje zdefiniowane przez użytkownika* są kodem języka Transact-SQL i są bardzo podobne do procedur składowanych. Jednak funkcje mogą być wywoływane w zapytaniach do bazy danych w celu modyfikacji kolumn danych, które mają być widoczne jako tabela, bez względu na to czy są one tworzone programowo czy dynamicznie. Przykładowo, można napisać własną funkcję daty, aby modyfikowała kolumny danych typu DataGodzina (*datetime*).
- *Wyzwalacze* są procedurami składowanymi, które są uaktywniane przed lub po dodaniu, zmianą lub usunięciu danych z bazy. Wyzwalacze zapewniają, że reguły biznesowe lub reguły integralności danych w

bazie są przestrzegane. Przykładowo, wyzwalacz może zapewnić, że każda książka w księgarni ma przypisanego prawidłowego wydawcę.

- *Widoki* są to głównie zapytania przechowywane w bazie danych, odnoszące się do jednej lub wielu tabel. Widoki można stworzyć i zachować w celu łatwiejszego użycia w przyszłości. Widoki zwykle wykluczają kolumny z tabeli lub łączą dwie lub więcej tabel. Można ich również używać jako mechanizmów zabezpieczeń.
- *Indeksy* pomagają tak zorganizować dane, że zapytania są wykonywane szybciej.
- *Klucze podstawowe* pomimo tego, że nie są obiektami, mają podstawowe znaczenie dla relacyjnych baz danych. Wymuszają unikalność wierszy oraz umożliwiają unikatową identyfikację każdego przechowywanego elementu.

Projektowanie relacyjnych baz danych

Podstawowe kroki projektowania RBD:

- Przeanalizuj sytuację, aby uzyskać informacje o proponowanej bazie danych.
- Podejmij decyzje o kolumnach, typach danych i długości danych.
- Normalizuj dane w tabelach.
- Stwórz bazę danych i tabele.

Proces projektowania powinien rozpoczynać się od przeglądu sytuacji biznesowej i celów, jakie klient stara się osiągnąć. Burza mózgów na temat różnych zmiennych oraz ich powiązań w tabelach to następny krok w tym procesie. Następnie proces przechodzi do tworzenia raportów i zapytań, które będą potrzebne użytkownikom, jak również innych części projektu, włączając w to dostęp do witryn sieci Web.

TAK	NIE
Pytaj użytkowników o ich potrzeby.	Nie ignoruj użytkowników (zwanych także klientami).
Utwórz listę obiektów.	Nie twórz obiektów, których nigdy nie użyjesz.
Utrzymuj nazwy obiektów możliwie krótkie ale wystarczająco opisowe.	Nie używaj złożonych nazw, nazw ze spacjami lub nietypowymi znakami ponieważ są trudniejsze do wpisywania.
Organizuj właściwości obiektów w odpowiednie grupy.	Nie stosuj kolumn zawierających więcej niż jedną wartość.
Twórz identyczne nazwy kolumn w różnych tabelach, w celu późniejszego skojarzenia ich ze sobą. Kolumny te będą dla Ciebie kluczami podstawowymi i obcymi.	Nie twórz tabel z ogromną ilością kolumn.
Przetestuj swój projekt na kilku danych testowych.	Nie zakładaj, że projekt działa dobrze ponieważ działa prawidłowo przy 5 wierszach, powinien działać równie dobrze przy 500 000 wierszy.
Utwórz co najmniej jeden indeks dla tabel, do których będą kierowane zapytania.	Nie twórz wielu indeksów (więcej niż pięciu) dla jednej tablicy.
Przy tworzeniu tabel bierz pod uwagę bezpieczeństwo.	Nie zapomnij ustawić zabezpieczeń danych.
Dokumentuj nazwy tabel, kolumn oraz kluczy podstawowych i obcych.	Nie zgub dokumentacji.
Zachowuj określoną konwencję nazewnictwa obiektów bazy danych. To znacznie upraszcza pracę z obiektami. Zalecane jest używanie prefiksów. Przykładowo, można używać nazwy	

tabeli <code>tblEmployees</code> dla tabeli obiektów zwanych <code>Employees</code> oraz nazwy <code>idxLastName</code> dla indeksu opartego na nazwiskach.	
---	--

Rozmowa z klientem

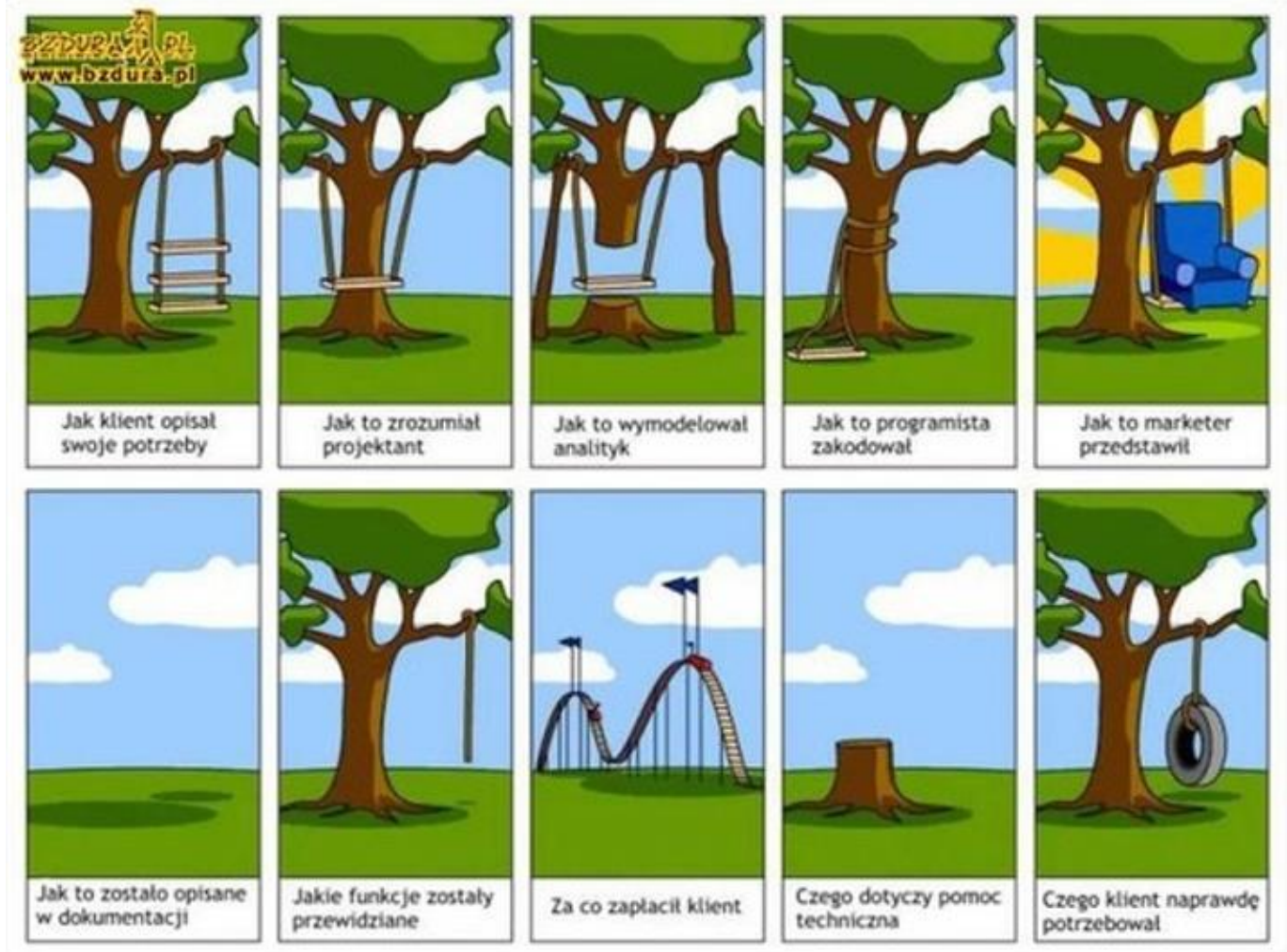
Dobry projekt bazy danych rozpoczyna się od zrozumienia sytuacji klienta i pożądaných wyników. Dlatego ludzie tworzący takie systemy zwani są *analitykami* – analizują oni problem szczegółowo i próbują określić sposoby rozwiązania danego problemu.

Czasem niemożliwy już wywiad jest najlepszym sposobem poznania potrzeb klienta, szczególnie jeśli nie posiada się pełnej informacji o obecnej sytuacji i dążeniach klienta. Aby wybadać potrzeby klienta można skorzystać z poniższych pytań:

- Jaki system działa obecnie u klienta?
- Jakie części obecnego systemu należałoby zastąpić w pierwszej kolejności?
- Czy istnieje potrzeba stworzenia dodatkowych raportów?
- Działanie jakich elementów systemu należałoby śledzić?
- Czy dane są publiczne czy prywatne?
- Kto potrzebuje mieć dostęp do danych i jaki typ dostępu powinien mieć każdy użytkownik lub grupa?
- Czy dane mają być publikowane w Internecie?
- Czy informacje powinny być dostępne publicznie do przeglądu poprzez sieć Internet?
- Czy posiadany przez Klienta sprzęt jest wystarczający do uruchomienia oprogramowania serwera i klienta bazy danych?
- Jeżeli nie ma zastrzeżeń co do technologii i kwoty pieniędzy, co jeszcze powinno być uwzględnione w nowym systemie?

Poprzez zadawanie tego typu pytań można szybko określić, w jakim celu Klientowi jest potrzebna baza danych. Pomimo, że być może nie będzie możliwe zaspokojenie wszystkich wymagań (określone ograniczenia przydzielonego budżetu, ramy czasowe i możliwości sprzętowe), będzie to początek długoterminowego planu wzrostu i rozszerzania bazy danych.

Dla poprawy humoru:



Rysunek 1. Praca zespołowa. Dlaczego duże projekty kończą się sprawami w sądzie?, opóźnieniami w oddawaniu projektu czy karami za nie otrzymanie odpowiedniego programu?

Grupowanie obiektów

Po przeprowadzeniu wywiadu najlepiej jest przeprowadzić „burzę mózgów” na temat możliwych obiektów, włączając w to ich nazwy, typy i długość. Po określeniu obiektów, należy je pogrupować w powiązane tabele.

SQL Server obsługuje kilka różnych typów danych, włączając w to typ znakowy, numeryczny, data oraz waluta.

Po określeniu tabel, należy ustalić właściwości (kolumny) tych tabel. Nazwy kolumn powinny być krótkie, ale wystarczająco opisowe. Długość kolumny powinna być wystarczająca nawet dla wyjątkowych przypadków. W przypadku nazw, ograniczeniem powinno być to ile znaków można wpisać jako adres e-mail – a nie to, ile takich adresów da się przechowywać.

Normalizacja danych

Po określeniu kolumn, należy zebrać dane w powiązane tabele. Proces ten nosi nazwę *normalizacji*. Normalizacja jest procesem organizowania danych w powiązane relacjami tabele.

Normalizacja danych ma na celu eliminację nadmiarowych danych. Przypuśćmy, że ten sam klient kupił dwa samochody. W bazie danych z pojedynczą tabelą, należy wpisać informacje o kliencie dwukrotnie. Co gorsze, jeśli klient zmieni miejsce zamieszkania, trzeba zmienić jego adres w obydwu miejscach lub dane nie będą spójne wewnątrz. Poprzez wpisanie danych klienta jedynie raz w tabeli klientów i połączenie ich z dowolnym zakupem samochodu, eliminuje się nadmiarowość (a czasami konflikt) danych, ale także można zmienić jego dane tylko w jednym miejscu. Rysunek 1.1 pokazuje jak mogą wyglądać przykładowe tablice. Warto zauważyć, że zostały

utworzone odrębne tablice dla klientów i samochodów. W tablicy Cars, pole CustID reprezentuje pojedynczy identyfikator klienta – ID. Jak można zauważyć, Ann posiada dwa samochody a Bob jeden samochód. Cathy nie ma jeszcze samochodu, ale w tym modelu bazy danych jest to możliwe do zapisania. Można także zapisać informację o wielu samochodach Ann, mając w bazie tylko jedną kopię jej adresu.

Rysunek 1.1.
Grupowanie
zmiennych w
relacyjnej bazie
danych.

Customer				
CustID	Name	Address	City	ect...
1	Ann	123 AnyStreet	Redmond	
2	Bob	947 Main	Seattle	
3	Cathy	1010 1st Avenue	Bellevue	
Cars				
CarID	Make	Model	Year	CustID
111AAA	Ford	Explorer	1999	1
222ABC	Honda	Civic	2000	1
112ADE	Toyota	Corolla	1998	2

Reguły są ustanowione w celu normalizacji danych. Reguły te znane są jako pierwsza, druga i trzecia forma normalna:

- *Pierwsza forma normalna (FNF)* mówi, że kolumna nie może zawierać wielu wartości. Przykładowo, nazwa osoby musi być rozbita na nazwisko oraz pierwsze i drugie imię aby spełniać FNF.
- *Druga forma normalna (SNF)* mówi, że każda kolumna nie należąca do klucza musi być oparta na całym kluczu, a nie tylko na części klucza podstawowego. Przykładowo, używając identyfikatora ID klienta i numeru części jako klucza, wszystkie kolumny w tej tabeli muszą odnosić się łącznie do danego klienta (ID) i numeru części. Czyli, kolumna `part_description` (opis części) nie należy do takiej tabeli. Tabela musi spełniać pierwszą formę normalną, żeby przyjąć drugą formę normalną.
- *Trzecia forma normalna (TNF)*, podobnie jak SNF, głosi, że wszystkie nie kluczowe kolumny nie mogą zależeć od innych nie kluczowych kolumn. Przykładowo, mając tabele z adresami, kod pocztowy nie może zależeć od innego nie kluczowego pola takiego jak stan, miasto. Powinien zależeć od całego klucza podstawowego. Tabela musi oczywiście spełniać drugą formę normalną. TNF jest często naruszana dla zachowania wydoby.

Istnieje także czwarta i piąta forma normalna. Większość projektantów baz danych jest usatysfakcjonowana osiągnięciem trzeciej formy normalnej.

Tworzenie bazy danych i tabel

Ponieważ tabele są blokami budującymi bazy danych, jest oczywiste, że dobrze zbudowane tabele (oraz kolumny w tabelach) są najistotniejsze dla sukcesu bazy danych. Jak przy większości rzeczy, planowanie i tworzenie to trudniejsza część; w rzeczywistości tworzenie bazy danych i tabel nie jest rzeczą trudną. Tabela składa się z kolumn, które przechowują własności tabeli.

SQL Server i model klient-serwer

Microsoft SQL Server jest bazą danych typu klient-serwer, dlatego istotne jest zrozumienie architektury tego modelu.

Aplikację *klient-serwer* można zdefiniować jako jedną aplikację rozbitą na dwie części: jedna z nich jest uruchomiona na serwerze a druga na stacji roboczej. Strona serwera zapewnia aplikacji zabezpieczenia, odporność na uszkodzenia, wydajność, współbieżność i wiarygodne kopie zapasowe. Strona klienta dostarcza interfejsu użytkownika i może zawierać puste raporty, zapytania i formularze. Ideą tego rozwiązania jest korzystanie z zalet obydwóch stron i łączenia ich razem.

SQL Server jest w tym modelu stroną serwera; można wybrać różnych klientów do łączenia się z serwerem SQL Server, włączając w to narzędzia dostarczane wraz z tym serwerem, takie jak SQL Server Query Analyzer. SQL Server dostarcza następujących korzyści dla klientów i serwerów:

Korzyści z klienta	Korzyści z serwera
Łatwość użycia	Niezawodność
Wsparcie dla wielu platform sprzętowych	Współbieżność
Wsparcie dla wielu platform programowych	Skomplikowane blokowanie
Przyjazny dla użytkownika	Odporność na uszkodzenia
	Sprzęt o wysokiej wydajności
	Scentralizowana kontrola

W architekturze typu klient-serwer, kiedy zostaje uruchomione zapytanie, serwer przeszukuje bazę danych i wysyła do klienta tylko wiersze pasujące do zapytania. Proces ten nie tylko zmniejsza ilość przesyłanych danych ale może być szybszy, niż w przypadku wykonywania zapytania na stacji roboczej, jeżeli serwer jest wystarczająco mocną maszyną.

ⁱ Cały materiał pochodzi z książki: SQL SERWER DLA KAŻDEGO