1.4.9. Typy danych

Projektując bazę danych, zwracamy uwagę na to, aby wartości przechowywane w polach były wartościami elementarnymi (atomowymi). W praktyce wymóg ten jest realizowany przez określenie typu pola.

Teoria relacyjnych baz danych mówi, że wszystkie wartości danych oparte są na prostych typach danych. Oznacza to, że dla każdego pola powinien zostać wybrany typ danych najmniejszy z możliwych, który spełni związane z nim wymagania projektowe. Typ danych określi, jaki rodzaj informacji może zostać zapisany w wybranym polu. Poza tym definiowanie typów danych znacząco wpływa na optymalizację działania bazy danych.

Jeżeli w tabeli występuje pole *Imię*, dla którego wybierzemy typ `char(100)` — typ tekstowy o długości 100 znaków — to na każde imię zostanie zarezerwowane 100 bajtów. Przyjmując, że najdłuższe imię nie zawiera więcej niż 20 znaków, dodatkowo rezerwujemy 80 bajtów, które nigdy nie zostaną wykorzystane. Gdy w tabeli wystąpi 200 tysięcy rekordów, to zajmą one kilkanaście GB nigdy niewykorzystanej pamięci.

To samo dotyczy pola *Data_urodzenia*. Jeżeli zostanie wybrany typ `date`, to zajmie on 3 bajty, jeżeli zostanie zastosowany typ `datetime`, to dla każdego rekordu zostanie zarezerwowane 8 bajtów na datę urodzenia.

Typy danych stosowane w językach komunikowania się z bazą danych można podzielić na kilka grup. Wybrane typy zostały pokazane w tabeli 1.1.

Tabela 1.1. Typy danych stosowane w MS SQL i MySQL

Grupa	*Typy w MS SQL*	*Typy w MySQL*
Tekstowe	char, varchar, nchar, ntext, nvarchar	char, varchar
Liczbowe	int, smallint, bigint, tinyint, float, real, decimal, numeric	int, smallint, bigint, tinyint, float, real, decimal, numeric
Daty i czasu	date, datetime	date, datetime, timestamp, time
Binarne	binary, varbinary	binary, varbinary
Waluty	money, smallmoney	
Specjalne	text, image, bit	text, bit, enum, set

Typy tekstowe

Typy tekstowe to pola, które mogą zawierać litery, liczby i symbole. Po zadeklarowaniu długości można przechowywać teksty o określonej liczbie znaków.

Typ char(długość) — można przechowywać do 255 znaków. Jest to pole o stałej liczbie znaków, np. `char(10)`. Jeżeli tekst jest krótszy, zostanie uzupełniony do zadeklarowanej

długości spacjami. Jeśli tekst jest dłuższy niż zadeklarowana liczba znaków, zostanie obcięty.

Typ varchar(długość) — można przechowywać do 255 znaków. Jest to pole o zmiennej liczbie znaków, np. `varchar(10)`. Jeżeli tekst jest krótszy niż zadeklarowana długość, nie jest uzupełniany spacjami, co powoduje zmniejszenie zasobów pamięci zajmowanych przez bazę danych. Przy deklarowaniu typów pól zaleca się stosowanie typów o zmiennej liczbie znaków.

Typ nvarchar(długość) — można przechowywać do 255 znaków. Ten typ działa w standardzie Unicode, który zapisuje pojedynczy znak na 2 bajtach. Jest to związane z kodowaniem znaków narodowych. Powoduje to zwiększenie objętości bazy danych, ale jeżeli będzie ona funkcjonowała w środowisku wielojęzycznym, należy zastosować typ, który będzie poprawnie interpretował znaki narodowe, np. `nvarchar(10)`.

Typ ntext(długość) — pozwala na wprowadzanie tekstu o wielkości do 2 GB. Typ stosowany, jeśli w polu będzie przechowywany tekst o dużej liczbie znaków.

Typy liczbowe

Typy liczb całkowitych

Jeżeli w polu będą wprowadzane tylko liczby całkowite, powinien zawsze być wybierany typ liczb całkowitych. Typy całkowite różnią się tylko liczbą bajtów przeznaczonych na zapisanie liczby, a więc również zakresem liczb, które mogą zostać zapisane.

Typ tinyint — 1 bajt, zapis liczb z zakresu od –128 do 127 lub bez znaku z zakresu od 0 do 255.

Typ smallint — 2 bajty, zapis liczb z zakresu od –32 768 do 32 767 lub bez znaku z zakresu od 0 do 65 535.

Typ int — 4 bajty, zapis liczb z zakresu od –2 147 483 648 do 2 147 483 647 lub bez znaku z zakresu od 0 do 4 294 967 295.

Typ bigint — 8 bajtów, zapis liczb z zakresu od –2^63 do 2^63–1.

Typy liczb rzeczywistych

Jeżeli w polu będą przechowywane liczby, dla których należy dokładnie określić liczbę miejsc po przecinku, powinien zostać zastosowany typ `decimal` lub `numeric`.

Typ numeric(precyzja, skala) — dla tego typu należy podać dwa parametry: precyzję i skalę. Pierwszy parametr określa całkowitą liczbę cyfr, drugi mówi, ile cyfr znajduje się po przecinku, np. `numeric(4,2)`. W podanym przykładzie możliwe będzie przechowywanie liczb nie większych od 9 999 z dokładnością do dwóch miejsc po przecinku. Jeśli nie podamy tych parametrów, zostaną one ustawione automatycznie na wartości 10 jako precyzja i 0 jako skala.

Typ decimal(precyzja, skala) — podobnie jak dla typu `numeric` należy podać dwa parametry: precyzję i skalę.

Jeżeli dokładność przechowywanych w bazie liczb jest mniej istotna, natomiast najważniejsza jest ich rozpiętość (np. dane statystyczne), należy stosować typ `float` lub `real`.

Typ float(n) — 4 bajty lub 8 bajtów, liczba zmiennoprzecinkowa. Parametr *n* określa precyzję w bitach. Jeśli *n* jest w zakresie od 1 do 24, to mamy do czynienia z pojedynczą precyzją (dokładność do około 7 cyfr po przecinku), jeżeli wynosi od 25 do 53, to mamy do czynienia z podwójną precyzją (dokładność do około 15 cyfr po przecinku). Używanie typu `float` może powodować problemy przy wykonywaniu obliczeń.

Typ float(m,d) — liczba zmiennoprzecinkowa o pojedynczej precyzji. Typ ten jest niestandardowym rozszerzeniem MySQL.

Typ real — 4 bajty, liczba zmiennoprzecinkowa. Definiowana podobnie jak typ `float`.

Jeżeli nie jest to konieczne, należy unikać używania typów przybliżonych. Jeżeli jest to możliwe, należy stosować typy `decimal` lub `numeric`.

Typy daty i czasu

Typ datetime — 8 bajtów, zapis daty i czasu z dokładnością do sekundy w postaci RRRR-MM-DD HH:MM:SS.

Typ date — 3 bajty, zapis daty w postaci RRRR-MM-DD.

Typ time — 3 bajty, zapis czasu w postaci HH:MM:SS.

Typ daty powinien być dostosowany do rodzaju przechowywanej informacji. Na ogół nie ma potrzeby przechowywania zarówno daty, jak i czasu. Jeśli chcemy przechowywać datę urodzin, w zupełności wystarczy pole typu `date`. Jeśli wymagane jest przechowywanie tylko roku urodzin, wystarczy zastosować pole typu `int`. Dzięki temu zostanie zaoszczędzona znaczna ilość miejsca w pamięci i uzyskamy większą efektywność działań na danych.

Typy binarne

Typy binarne służą do przechowywania danych binarnych. Dane reprezentowane są za pomocą liczb heksadecymalnych.

Typ binary — do przechowywania danych binarnych o stałej długości, długość maksymalna 8000 bajtów.

Typ varbinary — do przechowywania danych binarnych o zmiennej długości, długość maksymalna do 8000 bajtów.

Typy walutowe

Typy walutowe służą do przechowywania wartości walutowych. Mogą również służyć do przechowywania wartości innych niż wartości walutowe.

Typ money — 8 bajtów, do przechowywania wartości walutowych z zakresu od –2^31 do 2^31–1 z dokładnością do jednej dziesięciotysięcznej (cztery cyfry po przecinku).

Typ smallmoney — 4 bajty, do przechowywania wartości walutowych z zakresu od –2^63 do 2^63–1 z dokładnością do jednej dziesięciotysięcznej (cztery cyfry po przecinku).

Typy specjalne

Typ text — do przechowywania łańcuchów tekstowych o zmiennej długości. Może przechowywać do 2 GB danych.

Typ image — do przechowywania danych binarnych o zmiennej długości. Może przechowywać do 2 GB danych.

Mimo że typy `text` i `image` pozwalają na przechowywanie dużej ilości informacji, powinniśmy zastanowić się, czy lepszym rozwiązaniem nie byłoby umieszczenie pliku z tą informacją poza bazą, a w bazie danych przechowywanie jedynie ścieżki do pliku.

Typ bit — 1 bit, do przechowywania wartości logicznych w postaci *0* lub *1*, *włączony/wyłączony*, *tak/nie*.

1.5. Cechy relacyjnej bazy danych

Baza danych powinna charakteryzować się następującymi cechami:

- trwałość danych,
- integralność danych,
- bezpieczeństwo danych,
- współdzielenie danych,
- abstrakcja danych,
- niezależność danych,
- integracja danych.

1.5.1. Trwałość danych

Trwałość danych zapisanych w bazie jest podstawową cechą baz danych. Trwałość danych oznacza, że dane zostały zapisane w bazie w sposób nieulotny. Wszystkie współczesne systemy baz danych muszą spełniać ten wymóg. Trwałość danych jest niezależna od działania aplikacji oraz od platformy sprzętowej i programowej. Dane gromadzone w bazie danych są przechowywane w pamięci zewnętrznej (dyskowej, optycznej, taśmowej). Powinny one być przechowywane w pamięci dopóty, dopóki wymagają tego użytkownicy systemu baz danych.

1.5.2. Integralność, czyli poprawność danych

Integralność (spójność) danych oznacza, że dane muszą:

- wiernie odzwierciedlać dane rzeczywiste (dane są prawdziwe oraz są aktualizowane, gdy ulega zmianie rzeczywistość),

- spełniać ograniczenia nałożone przez użytkowników (istnieje system kontroli danych wejściowych),
- wykazywać brak anomalii wynikających ze współbieżnego dostępu do danych (istnieją mechanizmy, które zabezpieczają dane przed błędami pojawiającymi się podczas współbieżnego dostępu do nich).

Integralność bazy danych to także:

- odporność na błędy i awarie wynikające z zawodności sprzętu i oprogramowania (istnieją mechanizmy, które zabezpieczają dane przed skutkami awarii sprzętu i oprogramowania),
- odporność na błędy użytkowników (istnieją mechanizmy, które zabezpieczają dane przed następstwami błędów logicznych).

Na poziomie danych wyróżniamy dwa rodzaje integralności danych:

- *semantyczna* (spójność logiczna) — oznacza zgodność danych z rzeczywistością, czyli poprawność odwzorowania rzeczywistości (baza danych odpowiada zaprojektowanemu schematowi i zadeklarowanym ograniczeniom),
- *bazowa* (spójność fizyczna) — oznacza poprawność procesów zachodzących w bazie (operacje wykonywane w bazie danych kończą się sukcesem).

Na poziomie struktur bazy danych można wyróżnić następujące rodzaje integralności:

- *referencyjna* — odnosi się do powiązań między tabelami i oznacza, że każdej wartości klucza obcego odpowiada dokładnie jedna wartość klucza podstawowego;
- *encji* — odnosi się do schematu bazy danych i oznacza, że każda encja musi posiadać klucz podstawowy, czyli pole lub zbiór pól o wartościach unikatowych i niebędących wartością NULL. Z tego wynika, że w danej tabeli nie mogą istnieć dwa identyczne wiersze;
- *atrybutu* — oznacza, że wartość każdego atrybutu należy do jego dziedziny.

1.5.3. Bezpieczeństwo danych

Bezpieczeństwo danych oznacza, że dostęp do bazy danych mają tylko jej użytkownicy identyfikowani unikatową nazwą (loginem) i hasłem. Ponadto każdy użytkownik posiada określone uprawnienia w bazie danych.

Podstawowym sposobem zapewnienia bezpieczeństwa danych jest stosowanie procedur uwierzytelniania oraz ograniczania uprawnień dostępu do danych. Sposobem pośrednim może być szyfrowanie danych oraz ograniczenie fizycznego dostępu do systemu komputerowego.

Bezpieczeństwo danych oznacza również, że baza danych jest zabezpieczona przed awarią sprzętu lub oprogramowania.

1.5.4. Współdzielenie danych

Współdzielenie danych oznacza, że istnieje możliwość równoczesnej pracy wielu użytkowników z tą samą bazą danych. Użytkownicy mogą też jednocześnie pracować z tym samym zbiorem danych. Skutkiem takiej pracy mogą być konflikty w dostępie do danych, gdy jeden użytkownik modyfikuje zbiór danych, a drugi próbuje ten sam zbiór odczytać lub zmodyfikować w inny sposób. W bazie danych muszą istnieć mechanizmy zapewniające poprawne rozwiązanie takich konfliktów (np. stosowanie mechanizmów transakcji i współbieżności).

1.5.5. Abstrakcja danych

Abstrakcja to uogólnienie (uproszczenie) rozpatrywanego problemu, które polega na wyodrębnieniu wspólnych jego cech.

Baza danych zawiera pewien model rzeczywistości. Ale przechowywane są w niej tylko niektóre dane o obiektach. Powinny one opisywać wyłącznie istotne cechy obiektów świata rzeczywistego. Baza danych to abstrakcyjny model pewnego wycinka rzeczywistości, a jej struktura powinna poprawnie odzwierciedlać obiekty świata rzeczywistego i powiązania pomiędzy tymi obiektami.

Wyodrębniamy trzy poziomy abstrakcji:

- *poziom wewnętrzny* — określa sposób pamiętania danych;
- *poziom pojęciowy* (*konceptualny*) — wyższy poziom abstrakcji danych. Definiuje dane oraz związki zachodzące między nimi (przedstawia on logiczną strukturę bazy danych);
- *poziom zewnętrzny* — najwyższy poziom abstrakcji danych. Opisuje sposób, w jaki dane są widziane przez użytkownika. Dla każdego użytkownika obraz danych definiuje się za pomocą schematu zewnętrznego (podschematu).

Dla tej samej bazy danych istnieje jeden model pojęciowy i wiele schematów zewnętrznych.

1.5.6. Niezależność danych

Niezależność danych polega na oddzieleniu danych od aplikacji, które używają tych danych. Niezależność danych może być rozpatrywana w dwóch aspektach:

- *Logiczna niezależność danych* — niezależność danych widzianych przez użytkownika (przez jego aplikacje) od logicznej struktury bazy danych (schematu pojęciowego) oraz zmian tej struktury w czasie.
- *Fizyczna niezależność danych* — niezależność logicznej struktury bazy danych (schematu pojęciowego) od sposobu przechowywania danych w pamięci zewnętrznej (na nośnikach zewnętrznych).

We współczesnych bazach danych niezależność danych jest osiągana tylko częściowo.

1.5.7. Integracja danych

Integracja danych polega na zapewnieniu współpracy danych przechowywanych w różnych miejscach i obsługiwanych przez różne systemy. Często bazy danych fizycznie są umieszczane na różnych komputerach połączonych ze sobą w taki sposób, że użytkownik nie wie, iż dane, z którymi pracuje, pochodzą z różnych baz i komputerów. Zmiany zawartości bazy na jednym komputerze powinny być uwzględniane również na innych komputerach. Tak funkcjonujące bazy nazywamy *rozproszonymi bazami danych*. Bazy danych powinny być zaprojektowane tak, aby dane były zawsze dostępne, niezależnie od tego, gdzie i w jakiej postaci zostały zapisane i jak są przechowywane. Serwery bazodanowe powinny zapewniać integrację rozproszonych danych znajdujących się na wielu komputerach. Mechanizmy integracyjne dostępne na nich powinny działać w czasie rzeczywistym i współpracować z różnorodnymi bazami danych.

1.6. Pytania i zadania

1.6.1. Pytania

1. Podaj zalety korzystania z komputerowych baz danych.
2. Podaj definicję bazy danych.
3. Omów poznane modele baz danych.
4. Omów występujące w modelu relacyjnym rodzaje więzów integralności.
5. Podaj podstawowe cechy relacyjnego modelu baz danych.
6. Podaj definicję klucza podstawowego.
7. Jakiego typu relacje mogą wystąpić w bazie danych?
8. Co oznacza występujące w bazach danych pojęcie *encja*?
9. Do czego służą diagramy *ERD*?
10. Jakie zastosowanie w projektowaniu bazy danych mają narzędzia *CASE*?
11. W jaki sposób w tabelach opisywany jest związek „wiele do wielu"?
12. Na czym polega normalizacja tabel?
13. Wymień cechy, którymi powinna charakteryzować się baza danych.
14. Na czym polega integralność referencyjna bazy danych?

1.6.2. Zadania

Zadanie 1.

Zaprojektuj zgodnie z poznanymi zasadami bazę danych dotyczącą Twojej szkoły. Nazwij ją na przykład *Moja_szkoła*. Umieść w niej informacje na temat uczniów, klas, przedmiotów i nauczycieli, sal lekcyjnych i pracowni. Określ funkcje tworzonej bazy danych oraz zdefiniuj zbiory przechowywanych informacji. Wykorzystując diagramy ERD, opracuj model graficzny schematu bazy danych. Zaprojektuj tabele i sprawdź za pomocą reguł normalizacji, czy mają one prawidłową strukturę.