John L. Viescas

Słowo wstępne: Keith W. Hare
wiceprezes amerykańskiej komisji ds. norm SQL

ZAPYTANIA W SQL

PRZYJAZNY PRZEWODNIK

WYDANIE IV

Helion

Tytuł oryginału: SQL Queries for Mere Mortals: A Hands-On Guide to Data Manipulation in SQL (4th Edition)

Tłumaczenie: Piotr Cieślak

ISBN: 978-83-283-6064-8

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/zsqpp4
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

2
Prawidłowa struktura bazy danych

My kształtujemy nasze budowle, potem one kształtują nas.
— Sir Winston Churchill

Zagadnienia omówione w tym rozdziale

- Skąd wziął się tutaj ten rozdział?
- Dlaczego warto się troszczyć o prawidłowe struktury?
- Optymalizacja kolumn
- Optymalizacja tabel
- Definiowanie poprawnych zależności
- I to już wszystko?
- Podsumowanie

Większość czytelników tej książki zapewne pracuje z istniejącą bazą danych o konkretnej strukturze, zaimplementowaną w ulubionym (mam nadzieję) programie RDBMS. Na tym etapie trudno będzie Ci ocenić, czy miałeś — Ty albo inna osoba będąca autorem bazy — wystarczającą wiedzę i umiejętności, by poprawnie zaprojektować jej strukturę. Jeśli założyć najgorsze, to w bazie może się znajdować kilka tabel wymagających optymalizacji. Na szczęście za chwilę poznasz pewne techniki, które pozwolą Ci doprowadzić Twoją bazę do lepszego stanu i zagwarantują, że będziesz mógł bez trudu odczytywać z tabel wszelkie niezbędne informacje.

Skąd wziął się tutaj ten rozdział?

Być może zastanawiasz się, dlaczego zająłem się w tej książce kwestią projektowania baz danych i czemu rozdział jej poświęcony znalazł się na początku. Powód jest prosty: jeśli struktura bazy danych będzie kiepsko zaprojektowana, to wiele spośród instrukcji SQL, których konstruowania zaczniesz się uczyć w dalszej części książki, będzie w najlepszym wypadku trudnych do wykonania, a w najgorszym w ogóle bezużytecznych. Dopiero przy prawidłowo zaprojektowanej strukturze bazy nie będziesz miał problemów z wykorzystaniem umiejętności, jakie zdobędziesz w dalszej części książki.

Ten rozdział nie nauczy Cię tajników projektowania baz danych, ale ułatwi Ci doprowadzenie Twojej bazy do akceptowalnego stanu. Gorąco zalecam przeczytanie całego tego rozdziału, abyś miał pewność, że konstrukcja tabel w Twojej bazie jest prawidłowa.

Uwaga. Należy podkreślić, że poniższe omówienie jest poświęcone *logicznej* strukturze bazy danych. Nie nauczysz się z niego, jak tworzyć lub jak implementować bazę w konkretnym programie do zarządzania bazami danych, obsługującym SQL, ponieważ — zgodnie z tym, o czym przeczytałeś we „Wstępie" — tematyka ta wykracza poza zakres niniejszej książki.

Dlaczego warto się troszczyć o prawidłowe struktury?

Jeśli struktura Twojej bazy danych nie jest w pełni poprawna, to będziesz miał problemy z odczytywaniem z bazy nawet pozornie prostych informacji; trudno Ci będzie pracować z danymi, a konieczność dodania albo usunięcia jakiejś kolumny z tabeli przyprawi Cię o rozpacz. Źle zaprojektowana struktura bazy ma wpływ na inne jej aspekty, takie jak integralność danych, związki między tabelami i dokładność pozyskiwanych informacji. A te przeszkody to tylko wierzchołek góry lodowej. Listę można by wydłużać! Warto więc zadbać o prawidłowe struktury, aby uniknąć podobnych trudności.

Wielu wymienionym problemom można zapobiec przez właściwe zaprojektowanie bazy danych od samego początku. Ale nawet jeśli masz już gotową bazę, nie wszystko jest stracone. Nadal możesz wykorzystać opisane niżej techniki i skorzystać z zalet płynących z prawidłowej struktury. Musisz jednak mieć świadomość tego, że jakość ostatecznych struktur jest wprost proporcjonalna do ilości czasu, jaki poświęcasz na ich optymalizowanie. Z im większą troską i cierpliwością podejdziesz do wdrażania podanych dalej metod, tym większą będziesz miał szansę na sukces.

Zajmijmy się teraz pierwszym etapem modyfikowania struktur: pracą z kolumnami.

Optymalizacja kolumn

Ponieważ kolumny są najprostszą strukturą w bazie danych, musisz zadbać o to, by były w jak najlepszej kondycji przed przystąpieniem do optymalizowania tabel jako całości. Skorygowanie kolumn zwykle eliminuje niektóre wady danej tabeli i pozwala uniknąć potencjalnych kłopotów, które mogłyby one spowodować.

Odpowiednie dać rzeczy słowo (część pierwsza)

Zgodnie z tym, o czym przeczytałeś w poprzednim rozdziale, kolumna reprezentuje pewną cechę tematu tabeli, w której się ona znajduje. Jeśli nadasz kolumnie właściwą nazwę, to będziesz mógł łatwo określić, jaka to jest cecha. Nazwy niejednoznaczne i niejasne zwiastują kłopoty i sugerują, że cel utworzenia kolumny nie został dobrze przemyślany. Zweryfikuj nazwy poszczególnych kolumn na podstawie poniższej listy:

- *Czy nazwa jest rzeczowa i zrozumiała dla całej Twojej organizacji?* Jeśli z bazą danych będą pracowali użytkownicy z różnych działów firmy, to zadbaj o dobranie nazwy zrozumiałej dla wszystkich, którzy będą używali danej kolumny. Semantyka to zdradliwa rzecz, toteż użycie słowa, które będzie miało różne znaczenie dla różnych grup ludzi, oznacza wręcz proszenie się o kłopoty.
- *Czy nazwa kolumny jest jednoznaczna i przystępna?* Na przykład nazwa `NumerTelefonu` może być bardzo myląca. Jakiego rodzaju numery telefonów mają trafiać do danej kolumny? Numer domowy? Do pracy? Komórkowy? Ucz się precyzyjności. Jeśli zamierzasz przechowywać w bazie różne rodzaje numerów telefonów, to utwórz pola `TelefonDomowy`, `TelefonFirmowy` i `TelefonKomorkowy`.

Uwaga. Można powiedzieć, że `TelefonDomowy`, `TelefonFirmowy` i `TelefonKomorkowy` pasują do grupy, którą można byłoby przenieść do osobnej tabeli, zawierającej różne numery telefonów danego klienta albo pracownika. W takiej tabeli można byłoby utworzyć kolumnę określającą rodzaj telefonu; ponadto dla każdej osoby lub firmy można byłoby w ten sposób zdefiniować dowolną liczbę numerów. Więcej informacji na ten temat znajdziesz w kolejnym podrozdziale, poświęconym strukturze tabel.

Po zajęciu się jednoznacznością i przystępnością nazw koniecznie zadbaj o to, by ta sama nazwa kolumny nie powtarzała się w różnych tabelach. Przypuśćmy, że masz trzy tabele o nazwach `Klienci`, `Dostawcy` i `Pracownicy`. Bez wątpienia w każdej z tych tabel znajdą się kolumny `Miasto` i `Adres`, które otrzymają te same nazwy we wszystkich trzech tabelach. To nie stanowi problemu, dopóki nie będziesz musiał się odwołać do jednej, konkretnej kolumny. W jaki sposób odróżnić na przykład kolumnę `Miasto` w tabeli `Dostawcy` od kolumny `Miasto` w tabeli `Klienci` i od kolumny `Miasto` w tabeli `Pracownicy`? Odpowiedź jest prosta: dodaj krótki prefiks albo

przyrostek do nazw tych kolumn. Na przykład `DostMiasto` albo `MiastoDostawcy` w tabeli `Dostawcy`, `KlntMiasto` albo `MiastoKlienta` w tabeli `Klienci` i `PracMiasto` lub `MiastoPracownika` w tabeli `Pracownicy`. W ten sposób będziesz mógł bez trudu odwołać się do dowolnej z tych kolumn. (Tego rodzaju modyfikacjom możesz poddać dowolne kolumny o ogólnym charakterze, takie jak `Imie`, `Nazwisko` czy `Adres`). Podsumowując, trzeba pamiętać o jednym: by każda kolumna w bazie danych miała unikatową nazwę, która występuje tylko raz w całej strukturze bazy. Jedynym wyjątkiem od tej reguły jest użycie kolumny w celu zdefiniowania zależności między dwiema tabelami.

Uwaga. Powszechność zastosowania prefiksów w nazwach pól w tabeli to kwestia stylu. W przypadku tabel zawierających pola o charakterze ogólnym niektórzy projektanci baz danych decydują się na poprzedzenie prefiksem tylko nazw mogących budzić wątpliwości, zaś inni wolą używać prefiksów *we wszystkich* nazwach pól tabeli. Niezależnie od tego, jakie obierzesz podejście, pamiętaj o konsekwentnym przestrzeganiu go w całej strukturze bazy.

- *Czy użyłeś akronimu albo skrótu jako nazwy kolumny?* Jeśli tak, zmień ją! Akronimy mogą być trudne do rozszyfrowania i łatwo wprowadzają w błąd. Wyobraź sobie na przykład kolumnę o nazwie `DUr_PRC`. Jak zgadnąć, czego ona dotyczy? Skróty stosuj z umiarem i rozważnie. Warto ich używać tylko wtedy, gdy uzupełniają lub wzbogacają nazwę kolumny. Nie powinny one utrudniać zrozumienia prawdziwego znaczenia nazwy.
- *Czy użyłeś nazwy, która pośrednio lub bezpośrednio odwołuje się do więcej niż jednej cechy?* Tego rodzaju nazwy łatwo wychwycić, ponieważ zwykle zawierają one spójniki *i* albo *lub*. Podobnie rzecz się ma z kolumnami zawierającymi ukośnik (\), myślnik (–) albo ampersand (&). Jeśli w tabeli znajdują się kolumny o nazwach takich jak `Telefon\Fax` albo `Wojewodztwo_lub_Region`, to przejrzyj zawarte w nich dane i zastanów się, czy nie lepiej byłoby podzielić je na mniejsze, bardziej precyzyjnie zdefiniowane kolumny.

Uwaga. W standardzie SQL za prawidłowy *identyfikator zwykły* uważa się nazwę zaczynającą się od litery i zawierającą tylko litery, cyfry oraz znak podkreślenia. Spacje *nie są* dozwolone. Standard ten przewiduje także użycie nazw w postaci *identyfikatorów z ogranicznikami* — takie nazwy są ujęte w podwójny cudzysłów, muszą zaczynać się od litery i mogą zawierać litery, cyfry, znaki podkreślenia, spacje oraz znaki specjalne z pewnego ściśle określonego zbioru. W odniesieniu do nazw kolumn zalecam jednak używanie tylko pierwszej konwencji, ponieważ wiele implementacji SQL obsługuje wyłącznie identyfikatory zwykłe.

Po zweryfikowaniu nazw kolumn na podstawie powyższej listy zostanie Ci do zrobienia tylko jedno: upewnij się, że nazwa została podana w liczbie pojedynczej. Kolumna o nazwie w liczbie mnogiej, takiej jak `Kategorie`, sugeruje, że może ona zawierać dwie albo większą liczbę wartości w jednym wierszu, co nie jest najlepszym pomysłem. Nazwa kolumny powinna być podana w liczbie pojedynczej, ponieważ odzwierciedla wtedy jedną cechę zagadnienia reprezentowanego przez tabelę, do której należy. Z kolei nazwa tabeli powinna na ogół mieć formę w liczbie mnogiej, ponieważ stanowi ona zbiór wielu podobnych obiektów albo zdarzeń. Jeśli będziesz przestrzegał tej zasady nazywania, to odróżnienie nazw tabel od nazw kolumn przyjdzie Ci z łatwością.

Uwaga. Choć zalecam stosowanie konwencji nazw zgodnej ze standardem SQL, to należy pamiętać, że przy implementowaniu bazy — przez Ciebie albo programistę odpowiedzialnego za obsługę systemu — w konkretnej aplikacji RDBMS nazwy mogą ulec pewnym modyfikacjom. Ostateczna postać nazw powinna być zgodna z konwencją, której przestrzegają programiści pracujący z danym programem RDBMS.

Kosmetyka

Po skorygowaniu nazw skupmy się na strukturze samej kolumny. Nawet jeśli jesteś niemal pewien, że struktura kolumn w bazie danych jest prawidłowa, to dzięki kilku zabiegom możesz zyskać gwarancję, że rzeczywiście będzie ona tak efektywna, jak to możliwe. Przeanalizuj kolumny pod kątem poniższej listy, aby się przekonać, czy wymagają one pewnych przeróbek.

- *Upewnij się, że kolumna odzwierciedla konkretną cechę zagadnienia reprezentowanego przez tabelę*. Chodzi o to, by się upewnić, że dana kolumna rzeczywiście powinna należeć do tej tabeli. Jeśli nie ma ona związku z tematem tabeli, usuń ją albo ewentualnie przenieś do innej tabeli. Jedyne wyjątki od tej reguły dotyczą sytuacji, gdy dana kolumna jest używana do zdefiniowania zależności między daną tabelą a innymi tabelami w bazie bądź gdy kolumna ta została dodana do tabeli w celu wykonania jakiejś operacji, wymaganej przez program do obsługi baz danych. Na przykład w tabeli `Zajecia` przedstawionej na rysunku 2.1 kolumny `NazwiskoPracownika` i `ImiePracownika` są zbędne ze względu na obecność kolumny `IDPracownika`. Kolumna `IDPracownika` służy do zdefiniowania związku między tabelą `Zajecia` a tabelą `Pracownicy`, a dane z obydwu tabel można wyświetlić jednocześnie przy użyciu widoku lub odpowiedniego zapytania SQL z instrukcją `SELECT`. Jeśli znajdziesz w tabelach takie zbędne kolumny, możesz je zupełnie usunąć albo wykorzystać do utworzenia nowej tabeli, jeśli nie występują one w innym miejscu w strukturze bazy danych. (O tym, jak to zrobić, przeczytasz w dalszej części tego rozdziału).

Pracownicy

IDPracownika	ImiePracownika	NazwiskoPracownika	AdresPracownika	MiastoPracownika	StanZamPracownika	<< *inne kolumny* >>
98014	Peter	Brehm	722 Moss Bay Blvd.	Kirkland	WA	...
98019	Mariya	Sergienko	901 Pine Avenue	Portland	OR	...
98020	Jim	Glynn	13920 S.E. 40th Street	Bellevue	WA	...
98021	Tim	Smith	30301 166th Ave. N.E.	Seattle	WA	...
98022	Carol	Viescas	722 Moss Bay Blvd.	Kirkland	WA	...
98023	Alaina	Hallmark	Route 2, Box 203 B	Woodinville	WA	...

Zajecia

IDZajec	TematZajec	IDSali	IDPracownika	NazwiskoPracownika	ImiePracownika	<< *inne kolumny* >>
1031	Historia sztuki	1231	98014	Brehm	Peter	...
1030	Historia sztuki	1231	98014	Brehm	Peter	...
2213	Podstawy biologii	1532	98021	Smith	Tim	...
2005	Chemia	1515	98019	Sergienko	Mariya	...
2001	Chemia	1519	98023	Hallmark	Alaina	...
1006	Rysunek	1627	98020	Glynn	Jim	...
2907	Algebra	3445	98022	Viescas	Carol	...

Rysunek 2.1. Tabela ze zbędnymi kolumnami

- *Upewnij się, że kolumna zawiera tylko jedną wartość.* Kolumna, która może potencjalnie zawierać kilka wystąpień wartości tego samego typu, nosi nazwę kolumny *wielowartościowej*. (Przykładem jest kolumna zawierająca kilka numerów telefonów). Na tej samej zasadzie kolumna, w której potencjalnie można zapisać dwie lub większą liczbę wartości *różnych* typów, nosi nazwę *wieloczęściowej* (kolumna, która zawiera numer przedmiotu oraz opis przedmiotu, jest przykładem kolumny wieloczęściowej). Kolumny wielowartościowe i wieloczęściowe mogą przysparzać niemałych problemów, zwłaszcza podczas prób edytowania, usuwania albo sortowania danych. Jeśli zadbasz o to, by każda kolumna zawierała tylko jedną wartość, zrobisz znaczący krok ku zagwarantowaniu integralności i dokładności danych. Na razie postaraj się wyszukać wszystkie kolumny wielowartościowe lub wieloczęściowe i odnotować je. W dalszej części rozdziału dowiesz się, jak je wyeliminować.
- *Upewnij się, że kolumna nie zawiera rezultatu obliczeń albo konkatenacji.* W prawidłowo zaprojektowanej tabeli nie powinny występować kolumny będące wynikami obliczeń. Problem jest związany z wartością takiej kolumny. Kolumna, w odróżnieniu od komórki w arkuszu kalkulacyjnym, nie zawiera wzoru obliczeń. Gdy wartość dowolnego składnika obliczeń ulegnie zmianie, rezultat przechowywany w kolumnie nie zostanie zaktualizowany. Jedyny sposób na uaktualnienie jego wartości polega na wykonaniu zmiany ręcznie lub na

napisaniu procedur, które będą to robiły automatycznie. Tak czy owak, obowiązek aktualizacji takich kolumn spoczywa na użytkowniku albo na Tobie jako projektancie bazy. Lepszy sposób wykonywania obliczeń polega na zawarciu ich w instrukcji `SELECT`. O zaletach takiego rozwiązania przekonasz się, czytając rozdział 5. „Nie tylko zwykłe kolumny".

- *Upewnij się, że dana kolumna pojawia się tylko raz w całej bazie danych*. Jeśli popełniłeś typowy błąd i umieściłeś tę samą kolumnę (na przykład `NazwaFirmy`) w kilku tabelach w bazie, to możesz napotkać problem niespójności danych. Błąd ten wystąpi w chwili, gdy zmienisz wartość tej kolumny w jednej tabeli, ale zapomnisz dokonać analogicznej poprawki we wszystkich innych miejscach, w których ta kolumna występuje. Aby całkowicie wyeliminować tego rodzaju problemy, zadbaj o to, by dana kolumna występowała tylko raz w całej strukturze bazy danych. (Jedyny wyjątek od tej reguły dotyczy sytuacji, gdy kolumna służy do zdefiniowania związku między dwiema tabelami).

Uwaga. W najnowszych wersjach niektórych komercyjnych systemów baz danych istnieje możliwość zdefiniowania kolumny będącej wynikiem obliczenia. Jeżeli Twój system do obsługi baz danych na to pozwala, możesz definiować kolumny obliczeniowe, pamiętaj jednak, że utrzymanie aktualności wszystkich kolumn, gdy jedna z części wyrażenia ulegnie zmianie, wymaga dodatkowych zasobów dla systemu bazodanowego.

Eliminowanie kolumn wieloczęściowych

Jak wspomniałem wcześniej, kolumny wieloczęściowe i wielowartościowe mogą doprowadzić do krytycznych problemów z integralnością danych, trzeba je więc eliminować, aby uniknąć przykrych niespodzianek. Decyzja, który rodzaj kolumn usunąć najpierw, jest czysto arbitralna, zacznijmy więc od wieloczęściowych.

Kolumny wieloczęściowe można rozpoznać, zadając sobie bardzo proste pytania: „Czy bieżącą zawartość tej kolumny mogę podzielić na mniejsze, bardziej precyzyjne części?", „Czy będę miał problemy z wyodrębnieniem z tej kolumny konkretnej informacji, ponieważ znajduje się ona wśród innych danych w tej samej kolumnie?". Jeśli na dowolne z tych pytań odpowiesz „tak", to znaczy, że masz do czynienia z kolumną wieloczęściową. Rysunek 2.2 przedstawia kiepsko zaprojektowaną tabelę z kilkoma wieloczęściowymi kolumnami.

Pokazana na rysunku tabela `Klienci` zawiera dwie kolumny wieloczęściowe: `NazwaKlienta` oraz `Adres`. Znajduje się w niej też jedna kolumna, które potencjalnie także może składać się z kilku części: `Telefon`. W jaki sposób posortować dane z tej tabeli według nazwiska albo kodu pocztowego? Nie da się, ponieważ wartości te są umieszczone w kolumnach zawierających inne informacje. Jak widać, każda z tych kolumn może zostać podzielona na mniejsze części. Na przykład kolumnę `NazwaKlienta` można podzielić na dwie odrębne kolumny — `ImieKlienta` oraz `NazwiskoKlienta`.

Klienci

IDKlienta	NazwaKlienta	Adres	Telefon	<<inne kolumny>>
1001	Suzanne Viescas	15127 NE 24th, #383, Redmond, WA 98052	425 555-2686	...
1002	William Thompson	122 Spring River Drive, Duvall, WA 98019	425 555-2681	...
1003	Gary Hallmark	Route 2, Box 203B, Auburn, WA 98002	253 555-2676	...
1004	Robert Brown	672 Lamont Ave, Houston, TX 77201	713 555-2491	...
1005	Dean McCrae	4110 Old Redmond Rd., Redmond, WA 98052	425 555-2506	...
1006	John Viescas	15127 NE 24rh, #383, Redmond, WA 98052	425 555-2511	...
1007	Mariya Sergienko	901 Pine Avenue, Portland, OR 97208	503 555-2526	...
1008	Neil Patterson	233 West Valley Hwy, San Diego, CA 92199	619 555-2541	...

Kolumny wieloczęściowe

Rysunek 2.2. Tabela zawierająca kolumny wieloczęściowe

(Zauważ, że przy okazji posłużyłem się regułą nazywania omówioną wcześniej w tym rozdziale i dodałem przyrostek `-Klienta` do kolumn `Imie` i `Nazwisko`). Po zidentyfikowaniu w tabeli kolumny wieloczęściowej określ, z ilu części składa się znajdująca się w niej wartość, a potem podziel tę kolumnę na odpowiednią liczbę mniejszych. Rysunek 2.3 ilustruje sposób rozwiązania problemu z dwiema wieloczęściowymi kolumnami z tabeli `Klienci`.

Klienci

IDKlienta	ImieKlienta	NazwiskoKlienta	AdresKlienta	MiastoKlienta	StanZamKlienta	KodPocztowyKlienta
1001	Suzanne	Viescas	15127 NE 24th, #383	Redmond	WA	98052
1002	William	Thompson	122 Spring River Drive	Duvall	WA	98019
1003	Gary	Hallmark	Route 2, Box 203B	Auburn	WA	98002
1004	Robert	Brown	672 Lamont Ave	Houston	TX	77201
1005	Dean	McCrae	4110 Old Redmond Rd.	Redmond	WA	98052
1006	John	Viescas	15127 NE 24th, #383	Redmond	WA	98052
1007	Mariya	Sergienko	901 Pine Avenue	Portland	OR	97208
1008	Neil	Patterson	233 West Valley Hwy	San Diego	CA	92199

Rysunek 2.3. Rozwiązanie problemu z wieloczęściowymi kolumnami w tabeli Klienci

Uwaga. Oprócz podzielenia kolumn `NazwaKlienta` oraz `Adres` dobrym pomysłem w przypadku bazy danych przechowującej numery telefonów klientów z Ameryki Północnej jest podzielenie kolumny `Telefon` na dwie osobne — numer kierunkowy regionu oraz lokalny numer telefonu. Także w danych klientów z innych krajów można rozważyć oddzielenie numeru kierunkowego miasta od właściwego numeru telefonu. W praktyce jednak w większości biznesowych baz danych przechowuje się całe numery telefonów w jednej kolumnie. Wydzielenie numeru kierunkowego rejonu lub miasta mogłoby się przydać w bazach danych, na podstawie których wykonuje się analizy demograficzne. Niestety ze względu na szczupłość miejsca nie mogłem zademonstrować tego rozwiązania na rysunku 2.3.

Czasami rozpoznanie kolumny wieloczęściowej jest bardziej kłopotliwe. Spójrz na tabelę `Instrumenty`, pokazaną na rysunku 2.4. Na pierwszy rzut oka nie widać w niej żadnych wieloczęściowych kolumn. Jeśli jednak przyjrzysz się jej uważniej, przekonasz się, że kolumna `IDInstrumentu` jest w istocie wieloczęściowa. Wartości przechowywane w tej kolumnie odzwierciedlają dwie oddzielne informacje: kategorię, do której należy instrument — na przykład AMP (wzmacniacz), GUIT (gitara) czy MFX (urządzenie do efektów specjalnych) — oraz numer identyfikacyjny instrumentu. Wyobraź sobie teraz, jakich problemów przysporzyłaby zmiana nazwy kategorii MFX na MFU. Trzeba byłoby napisać specjalny program, który przetworzyłby dane w tej kolumnie: wyszukał ciąg znaków MFX i wszystkie jego wystąpienia zastąpił akronimem MFU. Nie chodzi o to, że *nie da się* tego zrobić, ale po co przysparzać sobie niepotrzebnej pracy, skoro można jej uniknąć dzięki poprawnemu zaprojektowaniu bazy danych? W przypadku kolumn takich jak w omawianym przykładzie dobrze jest podzielić je na mniejsze, aby nadać tabeli prawidłową, efektywną strukturę.

Instrumenty

IDInstrumentu	Producent	OpisInstrumentu	<<*inne kolumny*>>
GUIT2201	Fender	Fender Stratocaster	...
MFX3349	Zoom	Player 2100 Multi-Effects	...
AMP1001	Marshall	JCM 2000 Tube Super Lead	...
AMP5590	Crate	VC60 Pro Tube Amp	...
SFX2227Dunlop	Dunlop	Cry Baby Wah-Wah	...
AMP2766Fender	Fender	Twin Reverb Reissue	...

Rysunek 2.4. Przykład nieoczywistej kolumny wieloczęściowej

Eliminowanie pól wielowartościowych

Rozwiązywanie problemów z kolumnami wieloczęściowymi wcale nie jest trudne, ale eliminowanie kolumn wielowartościowych może być nieco bardziej skomplikowane i wymagać większego nakładu pracy. Na szczęście identyfikacja kolumn wielowartościowych jest prosta. Niemal bez wyjątku dane przechowywane w tego rodzaju kolumnach zawierają przecinki, średniki albo inne znaki często używane w charakterze separatorów. Separatory służą do rozdzielania kolejnych wartości wewnątrz kolumny. Rysunek 2.5 przedstawia przykład tabeli z kolumną wielowartościową.

W tym przykładzie każdy pilot ma uprawnienia do latania różnymi samolotami. Wszystkie te uprawnienia są przechowywane w jednej kolumnie o nazwie `Uprawnienia`. Dane w tej kolumnie zostały jednak zapisane w bardzo kłopotliwy sposób, który może powodować te same problemy z integralnością co w przypadku kolumn wieloczęściowych.

Piloci

IDPilota	ImiePilota	NazwiskoPilota	DataZatrudnienia	Uprawnienia	<<inne kolumny>>
25100	Sam	Alborous	1994-07-11	727, 737, 757, MD80	...
25101	Jim	Wilson	1994-05-01	737, 747, 757	...
25102	David	Smith	1994-09-11	757, MD80, DC9	...
25103	Kathryn	Patterson	1994-07-11	727, 737, 747, 757	...
25104	Michael	Hernandez	1994-05-01	737, 757, DC10	...
25105	Kendra	Bonnicksen	1994-09-11	757, MD80, DC9	...

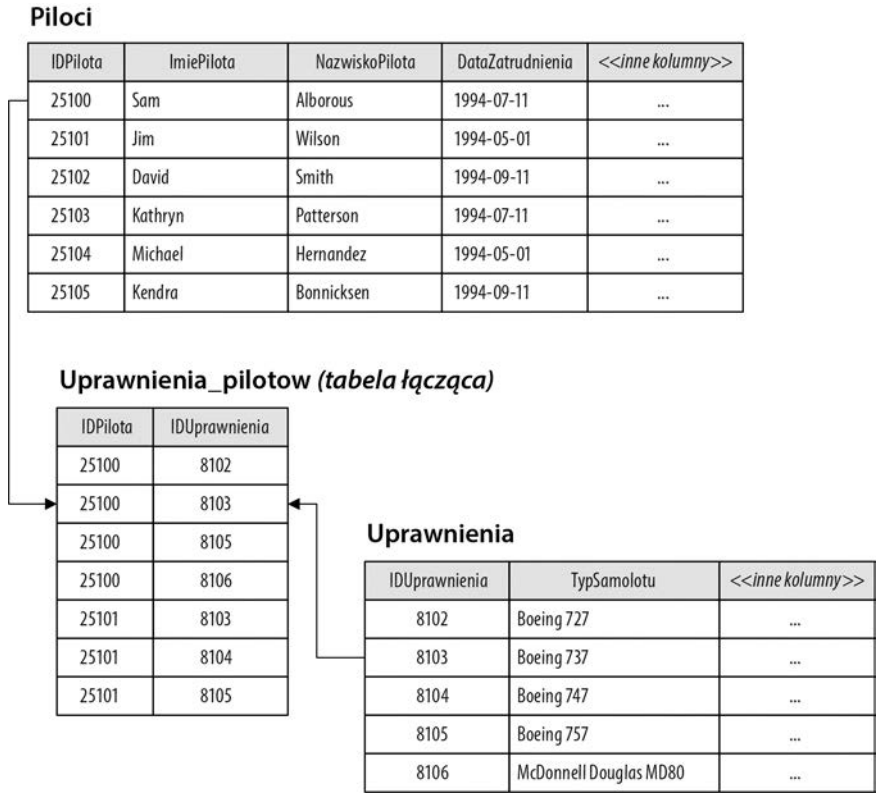
Rysunek 2.5. Tabela z kolumną wielowartościową

Jeśli uważnie przyjrzysz się tym danym, przekonasz się, że za pomocą zapytania SQL trudno Ci będzie wykonywać operacje wyszukiwania i sortowania na podstawie tej kolumny. Zanim jednak rozwiążesz problem z kolumną wielowartościową w prawidłowy sposób, powinieneś przeanalizować prawdziwą zależność między nią a tabelą, w której została ona umieszczona.

Wartości w kolumnie wielowartościowej mogą pozostawać w związku wiele do wielu z każdym z wierszy w macierzystej tabeli: dowolna wartość z kolumny wielowartościowej może być powiązana z wieloma wierszami z tabeli macierzystej, a jeden wiersz z tabeli macierzystej może być powiązany z dowolną liczbą wartości w kolumnie wielowartościowej. Na przykład na rysunku 2.5 konkretne samoloty w polu `Uprawnienia` mogą być powiązane z wieloma pilotami (co należy rozumieć tak, że wielu pilotów ma licencję na dany model maszyny), a jeden pilot może być powiązany z wieloma samolotami z pola `Uprawnienia` (dany pilot może mieć licencje na wiele maszyn). Tego rodzaju zależność wiele do wielu należy potraktować tak samo jak w przypadku innych związków tego typu — utworzyć dla niej tabelę łączącą.

Taką tabelę łączącą można utworzyć przy użyciu kolumny wielowartościowej oraz *kopii* kolumny klucza głównego z tabeli macierzystej — te dwa elementy staną się podstawą nowej tabeli. Nadaj tej tabeli odpowiednią nazwę i obie kolumny oznacz jako złożony klucz główny. (W tym przypadku każdy wiersz w nowej tabeli będzie jednoznacznie identyfikowany przez kombinację wartości z obydwu kolumn). Teraz możesz powiązać wartości obydwu kolumn z tabeli łączącej z odpowiednimi tabelami za pośrednictwem relacji jeden do jednego. Rysunek 2.6 ilustruje przykład takiego rozwiązania na podstawie tabeli `Piloci` z rysunku 2.5.

Porównaj dane o Samie Alborous (`IDPilota` 25100) w starej tabeli `Piloci` i nowej tabeli `Uprawnienia_pilotow`. Znaczącą zaletą nowej tabeli łączącej jest fakt, że pozwala ona powiązać *dowolną* liczbę uprawnień z jednym pilotem. Znacznie łatwiejsze jest też teraz uzyskiwanie odpowiedzi na pewne pytania. Można na przykład określić, którzy piloci mają uprawnienia do latania boeingami 747, bądź pobrać listę uprawnień dla konkretnego pilota. Poza tym dane w tak zmienionej strukturze można łatwo i w dowolny sposób sortować bez skutków ubocznych.



Rysunek 2.6. Rozwiązanie problemu z kolumną wielowartościową przy użyciu tabeli łączącej

Uwaga. Niektóre systemy do zarządzania bazami danych — głównie Microsoft Office Access 2007 i jego nowsze wersje — umożliwiają jawne deklarowanie kolumn wielowartościowych. System realizuje jednak tę operację poprzez utworzenie ukrytego zestawu tabel, podobnego do tabeli łączącej pokazanej na rysunku 2.6. Szczerze mówiąc, wolę pracować z jawną strukturą tabel, zalecam więc tworzenie poprawnych struktur danych samemu, zamiast polegać na funkcjach dostępnych w systemie do obsługi baz danych.

Jeśli będziesz postępował zgodnie z procedurami opisanymi w tej części rozdziału, to kolumny w Twoich tabelach powinny być w doskonałej formie. Po uporaniu się z kolumnami możemy się zająć kolejnym etapem zadania: analizą struktury tabel.

Optymalizacja tabel

Tabele są podstawą wszystkich zapytań SQL. Wkrótce się przekonasz, że źle zaprojektowane tabele mogą się przyczyniać do problemów z integralnością danych i przysparzają trudności przy tworzeniu zapytań SQL obejmujących wiele tabel. To sprawia, że naprawdę warto zadbać o efektywną strukturę tabel, by pozyskiwanie potrzebnych danych było jak najłatwiejsze.

Odpowiednie dać rzeczy słowo (część druga)

W analogicznej części rozdziału poświęconej kolumnom dowiedziałeś się, jak ważne jest nadawanie kolumnom odpowiednich nazw i dlaczego warto tę kwestię dogłębnie przemyśleć. Za chwilę przekonasz się, że to samo dotyczy tabel. Według definicji tabela powinna reprezentować jedno zagadnienie. Jeśli zawiera ona dane dotyczące różnych zagadnień, to należy ją podzielić na mniejsze tabele. Nazwa tabeli musi jasno odzwierciedlać jej temat. Enigmatyczna albo niejednoznaczna nazwa tabeli świadczy o tym, że jej struktura nie została dobrze przemyślana. Upewnij się, że nazwy Twoich tabel są poprawne, kierując się poniższymi zaleceniami.

- *Czy nazwa jest unikatowa i wystarczająco opisowa, by była zrozumiała dla całej Twojej organizacji?* Nadanie tabelom unikatowych nazw gwarantuje, że każda tabela w bazie danych będzie odzwierciedlała inne zagadnienie, a każdy użytkownik w organizacji będzie wiedział, do czego ona służy. Wybranie unikatowej i zarazem opisowej nazwy wymaga trochę pracy, ale na dłuższą metę jest warte zachodu.
- *Czy nazwa precyzyjnie, jasno i jednoznacznie odzwierciedla temat tabeli?* Jeśli nazwa tabeli jest niejasna albo wieloznaczna, to można z dużym prawdopodobieństwem założyć, że tabela ta reprezentuje kilka zagadnień. Niejednoznaczna jest na przykład nazwa `Daty`. Trudno się domyślić, czego konkretnie dotyczą dane w takiej tabeli, jeśli nie ma się pod ręką jej opisu. Powiedzmy, że tabela `Daty` znajduje się w bazie danych agencji organizującej imprezy rozrywkowe. Po uważniejszej analizie jej zawartości zauważasz, że zawiera ona daty spotkań z klientami i daty rezerwacji związane z występami wykonawców współpracujących z agencją. To oznacza, że tabela ta zawiera informacje o dwóch różnych sprawach. Taki problem można rozwiązać poprzez podzielenie tabeli na dwie nowe i nadanie każdej z nich odpowiedniej nazwy, na przykład `Daty_spotkan` i `Daty_wystepow`.
- *Czy nazwa zawiera słowa opisujące fizyczne właściwości danych?* Unikaj używania słów takich jak *Plik*, *Rekord* albo *Tabela* w nazwach tabel, ponieważ wprowadzają one zamieszanie. Nazwa tabeli zawierająca tego rodzaju określenia zapewne reprezentuje kilka różnych zagadnień. Weźmy na przykład nazwę `Rekordy_pracowników`. Pozornie nie budzi ona zastrzeżeń.

Ale jeśli zastanowisz się, co powinien zawierać wiersz pracownika, uświadomisz sobie, że taka nazwa może powodować komplikacje. Zawiera ona słowo, którego należy zdecydowanie unikać, i nie wyjaśnia, jaki rodzaj informacji został zgromadzony w tabeli: dane teleadresowe pracowników, ich funkcje w działach firmy czy może lista płac. Biorąc to pod uwagę, można podzielić oryginalną tabelę (`Rekordy_pracowników`) na trzy nowe, po jednej dla każdego z wymienionych zagadnień.

- *Czy użyłeś akronimu albo skrótu jako nazwy tabeli?* Jeśli odpowiedź brzmi „tak”, to od razu zmień nazwę! Skróty rzadko dobrze oddają tematykę tabeli, a akronimy są na ogół trudne do rozszyfrowania. Przypuśćmy, że w bazie danych Twojej firmy istnieje tabela o nazwie `KB`. Jak się domyślić, co ta tabela reprezentuje, bez znajomości znaczenia tych liter? Odgadnięcie roli takiej tabeli wcale nie jest proste. Co więcej, taka tabela różnym działom firmy może się kojarzyć z zupełnie innymi rzeczami. (A to jeszcze gorzej). Pracownicy z działu PR mogą sądzić, że tabela ta zawiera `Kontakty_biznesowe`, administratorzy mogą wyjść z założenia, że to pewnie `Konfiguracja_bazy`, zaś spece od zabezpieczeń mogą być przekonani, że znajdą w niej `Kody_bezpieczeństwa`. Ten przykład dobitnie dowodzi, dlaczego lepiej unikać skrótów i akronimów w nazwach tabel.
- *Czy użyłeś nazwy, która pośrednio lub bezpośrednio odwołuje się do więcej niż jednego tematu?* Jest to jeden z częstszych błędów przy nazywaniu tabel, który zarazem dość łatwo wychwycić. Tego rodzaju nazwy zwykle zawierają słowa *i* albo *lub* oraz znaki takie jak ukośnik (\), myślnik (–) albo ampersand (&). Przykładami takich nazw są choćby `Infrastruktura\Budynek` albo `Pion lub Dział`. Jeśli nazwałeś tabelę w taki sposób, musisz sprecyzować, czy rzeczywiście odzwierciedla ona więcej niż jeden temat. Jeśli tak jest w istocie, rozdziel ją na mniejsze tabele i nadaj im odpowiednie nazwy.

Uwaga. Przypominam, że standard SQL definiuje prawidłowy *identyfikator zwykły* jako nazwę zaczynającą się od litery, zawierającą tylko litery, cyfry oraz znak podkreślenia. Spacje *nie są* dozwolone. Standard ten przewiduje także użycie nazw w postaci *identyfikatorów z ogranicznikami* — takie nazwy są ujęte w podwójny cudzysłów, muszą zaczynać się od litery i mogą zawierać litery, cyfry, znaki podkreślenia, spacje oraz znaki specjalne z pewnego ściśle określonego zbioru. W odniesieniu do nazw tabel zalecam jednak używanie tylko pierwszej metody tworzenia identyfikatorów, ponieważ wiele implementacji SQL obsługuje wyłącznie identyfikatory zwykłe.

Po wstępnym przeanalizowaniu nazw tabel masz przed sobą jeszcze tylko jedno zadanie: przyjrzyj się im ponownie i upewnij się, że użyłeś liczby mnogiej. Nazywanie tabel w liczbie mnogiej jest zalecane, ponieważ przechowują one *zbiór wystąpień* zagadnienia, któremu

są poświęcone. Na przykład tabela `Pracownicy` zawiera dane nie o jednym, ale o wielu pracownikach. Zastosowanie liczby mnogiej ułatwia też odróżnienie nazwy tabeli od nazwy kolumny.

Uwaga. Zalecenie użycia liczby mnogiej w nazwach tabel jest szczególnie istotne podczas konstruowania logicznej struktury bazy danych. Takie rozwiązanie bardzo ułatwia odróżnienie nazw tabel od nazw kolumn, zwłaszcza jeśli wyświetlasz je na rzutniku albo wypiszesz mazakiem na białej tablicy w sali konferencyjnej.

Należy jednak pamiętać, że przy implementowaniu bazy — przez Ciebie albo programistę odpowiedzialnego za obsługę systemu — w konkretnej aplikacji RDBMS nazwy tabel mogą ulec pewnym modyfikacjom. Ostateczna postać nazw powinna być zgodna z konwencją, której przestrzegają programiści pracujący z danym programem RDBMS.

Zapewnianie prawidłowej struktury

Po przeanalizowaniu kwestii nazw skupmy się na strukturze tabel. To bardzo ważne, by tabele były poprawnie skonstruowane, gdyż dzięki temu możesz efektywniej przechowywać dane i pozyskiwać precyzyjne informacje. Czas poświęcony na poprawne opracowanie struktur tabel opłaci Ci się z nawiązką przy tworzeniu złożonych zapytań SQL, odwołujących się do wielu tabel. Użyj poniższej listy, aby zweryfikować poprawność strukturalną tabel.

- *Upewnij się, że dana tabela reprezentuje tylko jedno zagadnienie.* Tak, zdaję sobie sprawę, że pisałem o tym już kilkakrotnie, ale jest to naprawdę aż tak ważne. Pewność, że każda tabela odpowiada tylko jednemu zagadnieniu, znacznie zmniejsza ryzyko potencjalnych problemów z integralnością danych. Przypominam też, że tematem tabeli może być obiekt albo zdarzenie. Przez „obiekt" rozumiem coś namacalnego, na przykład pracowników, dostawców, maszyny, budynki, działy firmy itp., zaś „zdarzenie" jest czymś, co nastąpiło w określonym czasie i ma pewne właściwości, które chciałbyś odnotować. Dobrym przykładem takiego zdarzenia, które każdemu łatwo sobie wyobrazić, jest wizyta u lekarza. Choć sama wizyta jako taka nie jest „namacalna", to ma pewne właściwości, które warto zarejestrować, takie jak data wizyty, godzina, ciśnienie krwi i temperatura ciała pacjenta.

- *Upewnij się, że każda tabela ma klucz główny.* W każdej tabeli należy koniecznie zdefiniować klucz główny. Trzeba to zrobić z dwóch powodów. Po pierwsze, klucz główny jednoznacznie identyfikuje każdy wiersz w tabeli, po drugie zaś, służy on do definiowania związków między tabelami. Jeśli nie zdefiniujesz klucza głównego we wszystkich tabelach, to na pewnym etapie napotkasz problemy z integralnością danych, a także z realizacją niektórych zapytań SQL, obejmujących wiele tabel. W dalszej części tego rozdziału poznasz kilka wskazówek dotyczących prawidłowego definiowania kluczy głównych.

- *Upewnij się, że tabela nie zawiera kolumn wieloczęściowych albo wielowartościowych.* Teoretycznie tego rodzaju problemy powinieneś wyeliminować podczas analizowania struktury kolumn. Niemniej jednak warto raz jeszcze przyjrzeć się tabeli pod tym kątem, aby mieć absolutną pewność, że pozbyłeś się wszelkich kolumn wieloczęściowych i wielowartościowych.
- *Upewnij się, że w tabeli nie ma kolumn obliczeniowych.* Nawet jeśli masz wrażenie, że w bieżącej strukturze tabel nie ma kolumn obliczeniowych, to podczas ich optymalizowania jedna czy dwie mogły umknąć Twojej uwadze. To dobra okazja, by przyjrzeć się strukturze tabeli jeszcze raz i usunąć dowolne kolumny obliczeniowe, które przegapiłeś.
- *Upewnij się, że w tabeli nie ma zbędnych, powtarzających się kolumn*. Jedną z zasadniczych cech źle zaprojektowanej tabeli jest umieszczenie w niej kolumn powtarzających się w innych tabelach. Do powielania kolumn między tabelami mogą skłaniać następujące dwa powody: 1) chęć zamieszczenia informacji pomocniczej lub 2) chęć uwzględnienia wielu wystąpień danego rodzaju wartości. Zapewne pamiętasz o wspomnianym wcześniej ryzyku powielania danych w kolumnach `TelefonDomowy`, `TelefonFirmowy` i `TelefonKomorkowy`. Duplikaty kolumn przysparzają rozmaitych problemów podczas pracy z danymi i prób pozyskiwania informacji z tabel. Przyjrzyjmy się, jak poradzić sobie z powtarzającymi się kolumnami.

Usuwanie zbędnych, powtarzających się kolumn

Chyba najtrudniejszym aspektem opracowania poprawnej struktury bazy jest rozwiązanie problemu z duplikatami kolumn. Oto dwa przykłady, które demonstrują właściwy sposób korygowania tabel, w których znajdują się powtarzające się kolumny.

Rysunek 2.7 przedstawia przykładową tabelę zawierającą duplikaty kolumn, w których znajdują się informacje pomocnicze.

W tym przypadku kolumny `ImiePracownika` i `NazwiskoPracownika` znajdują się w tabeli `Zajecia` po to, by osoba przeglądająca tę tabelę mogła zobaczyć imię i nazwisko wykładowcy prowadzącego dane zajęcia. Kolumny te są niepotrzebne ze względu na istniejącą relację jeden do wielu między tabelami `Zajecia` i `Pracownicy`. (Jeden wykładowca może prowadzić dowolną liczbę zajęć, ale dane zajęcia prowadzi jeden, konkretny wykładowca). Za utworzenie relacji między tymi kolumnami odpowiada kolumna `IDPracownika`, zaś relacja ta umożliwia wyświetlanie danych z obu tabel jednocześnie przy użyciu odpowiedniego zapytania SQL. Mając to na uwadze, możemy bezpiecznie usunąć kolumnę `ImiePracownika` i `NazwiskoPracownika` z tabeli `Zajecia`, bez ryzyka efektów ubocznych. Rysunek 2.8 przedstawia skorygowaną strukturę tabeli `Zajecia`.

Pracownicy

IDPracownika	ImiePracownika	NazwiskoPracownika	AdresPracownika	MiastoPracownika	StanZamPracownika	<<inne kolumny>>
98014	Peter	Brehm	722 Moss Bay Blvd.	Kirkland	WA	...
98019	Mariya	Sergienko	901 Pine Avenue	Portland	OR	...
98020	Jim	Glynn	13920 S.E. 40th Street	Bellevue	WA	...
98021	Tim	Smith	30301 166th Ave. N.E.	Seattle	WA	...
98022	Carol	Viescas	722 Moss Bay Blvd.	Kirkland	WA	...
98023	Alaina	Hallmark	Route 2, Box 203 B	Woodinville	WA	...

Te kolumny są zbędne

Zajęcia

IDZajec	TematZajec	IDSali	IDPracownika	NazwiskoPracownika	ImiePracownika	<<inne kolumny>>
1031	Historia sztuki	1231	98014	Brehm	Peter	...
1030	Historia sztuki	1231	98014	Brehm	Peter	...
2213	Podstawy biologii	1532	98021	Smith	Tim	...
2005	Chemia	1515	98019	Sergienko	Mariya	...
2001	Chemia	1519	98023	Hallmark	Alaina	...
1006	Rysunek	1627	98020	Glynn	Jim	...
2907	Algebra	3445	98022	Viescas	Carol	...

Rysunek 2.7. Tabela z powielonymi kolumnami, dodanymi tylko w celach poglądowych

Pracownicy

IDPracownika	ImiePracownika	NazwiskoPracownika	AdresPracownika	MiastoPracownika	StanZamPracownika	<<inne kolumny>>
98014	Peter	Brehm	722 Moss Bay Blvd.	Kirkland	WA	...
98019	Mariya	Sergienko	901 Pine Avenue	Portland	OR	...
98020	Jim	Glynn	13920 S.E. 40th Street	Bellevue	WA	...
98021	Tim	Smith	30301-166th Ave. N.E.	Seattle	WA	...
98022	Carol	Viescas	722 Moss Bay Blvd.	Kirkland	WA	...
98023	Alaina	Hallmark	Route 2, Box 203 B	Woodinville	WA	...

Zajęcia

IDZajec	TematZajec	IDSali	IDPracownika	<<inne kolumny>>
1031	Historia sztuki	1231	98014	...
1030	Historia sztuki	1231	98014	...
2213	Podstawy biologii	1532	98021	...
2005	Chemia	1515	98019	...
2001	Chemia	1519	98023	...
1006	Rysunek	1627	98020	...
2907	Algebra	3445	98022	...

Rysunek 2.8. Rozwiązanie problemu z powtarzającymi się, poglądowymi kolumnami

Pozostawienie zbędnych kolumn w tabeli automatycznie powoduje poważny problem z integralnością danych. Otóż przy takiej strukturze trzeba zadbać o to, by wartości w kolumnach `NazwiskoPracownika` i `ImiePracownika` w tabeli `Zajecia` były zawsze zgodne z ich odpowiednikami w tabeli `Pracownicy`. Przypuśćmy na przykład, że pewna nauczycielka wychodzi za mąż i decyduje się przyjąć nowe nazwisko, którym odtąd będzie się posługiwać. Wówczas nie tylko musisz wprowadzić odpowiednią zmianę w jej wierszu w tabeli `Pracownicy`, ale też poprawić wszystkie wystąpienia jej nazwiska w tabeli `Zajecia`. Owszem, jest to wykonalne (przynajmniej z technicznego punktu widzenia), ale wymaga zbędnego nakładu pracy. Ponadto jednym z głównych założeń używania relacyjnej bazy danych jest możliwość przechowywania danej informacji tylko w jednym miejscu. (Jedynym wyjątkiem jest użycie kolumny do zdefiniowania związku między dwiema tabelami). Jak zwykle w takich przypadkach najlepszym wyjściem jest usunięcie wszystkich powtarzających się kolumn z tabel w bazie danych.

Rysunek 2.9 ilustruje inny typowy przykład tabeli zawierającej duplikaty kolumn. Ten przykład przedstawia sytuację, w której powtarzające się kolumny zostały błędnie użyte do określenia wielu wystąpień konkretnego typu wartości. W tym przypadku mamy do czynienia z trzema kolumnami o nazwie `Komisja`, które zostały użyte do zapisywania nazw komisji, w których uczestniczą różni pracownicy.

Pracownicy

IDPracownika	NazwiskoPracownika	ImiePracownika	Komisja1	Komisja2	Komisja3	*<<inne kolumny>>*
7004	Gehring	Darren	ds. rozwoju			...
7005	Kennedy	John	ISO 9000	ds. bezpieczeństwa		...
7006	Thompson	Sarah	ds. bezpieczeństwa	ISO 9000	ds. rozwoju	...
7007	Wilson	Jim				...
7008	Seidel	Manuela	ISO 9000			...
7009	Smith	David	ds. rozwoju	ds. bezpieczeństwa	ISO 9000	...
7010	Patterson	Neil				...
7011	Viescas	Michael	ISO 9000	ds. rozwoju	ds. bezpieczeństwa	...

Rysunek 2.9. Tabela z powtarzającymi się kolumnami użytymi w celu odnotowania wielu wystąpień danego typu wartości

Nietrudno się domyślić, w jakich przypadkach te powtarzające się kolumny mogą rodzić komplikacje. Pierwszy problem dotyczy liczby pól `Komisja` umieszczonych w tabeli. A co będzie, jeśli kilku pracowników zacznie udzielać się w czterech komisjach? A skoro już o tym mowa, to jaka liczba kolumn `Komisja` zaspokoiłaby przyszłe potrzeby bazy? Okazuje się, że jeśli kilku pracowników brałoby udział w więcej niż trzech komisjach, tabelę `Pracownicy` trzeba byłoby rozbudować o kolejne kolumny `Komisja`.

Drugi problem dotyczy pozyskiwania informacji z tej tabeli. W jaki sposób wybrać z niej tylko tych pracowników, którzy aktualnie biorą udział w pracach komisji ISO 9000? Nie jest to niemożliwe, ale pozyskanie takiej informacji byłoby dość kłopotliwe.

Aby prawidłowo odpowiedzieć na to pytanie, trzeba byłoby wykonać trzy osobne zapytania (albo zdefiniować kryteria wyszukiwania, które umożliwią przeszukanie zawartości trzech osobnych kolumn), ponieważ nie ma pewności, w której z kolumn `Komisja` znajduje się wartość ISO 9000. To zaś oznacza większy nakład czasu i pracy, niż to konieczne.

Trzeci problem dotyczy sortowania danych. W zaistniałej sytuacji nie da się posortować danych względem komisji i nie ma też sposobu na eleganckie wyświetlenie nazw komisji w kolejności alfabetycznej. Choć pozornie są to mało istotne przeszkody, to przy próbach pozyskania ogólnych i w jakiś sposób uporządkowanych informacji na podstawie danych z tej tabeli mogą one okazać się bardzo irytujące.

Jeśli uważnie przyjrzałeś się tabeli `Pracownicy` na rysunku 2.9, to zapewne zwróciłeś uwagę na fakt, że między pracownikami a komisjami, do których należą, występuje związek wiele do wielu. Jeden pracownik może należeć do dowolnej liczby komisji, a jedna komisja może się składać z dowolnej liczby pracowników. To zaś oznacza, że problem z powtarzającymi się kolumnami można rozwiązać w podobny sposób jak w przypadku dowolnej innej relacji wiele do wielu — przez utworzenie tabeli łączącej. W przypadku tabeli `Pracownicy` tabelę łączącą można utworzyć na podstawie kopii klucza głównego (`IDPracownika`) oraz jednej kolumny o nazwie `Komisja`. Tak otrzymanej tabeli należy nadać właściwą nazwę, na przykład `Czlonkowie_komisji`, obydwa umieszczone w niej kolumny — `IDPracownika` oraz `Komisja` — zdefiniować jako złożony klucz główny, usunąć numerowane kolumny `Komisja`... z tabeli `Pracownicy` — i gotowe. (O kluczach głównych przeczytasz w dalszej części tego rozdziału). Rysunek 2.10 przedstawia poprawioną tabelę `Pracownicy` oraz nową tabelę `Czlonkowie_komisji`.

Pracownicy

IDPracownika	NazwiskoPracownika	ImiePracownika	MiastoPracownika	*<<inne kolumny>>*
7004	Gehring	Darren	Chico	...
7005	Kennedy	John	Portland	...
7006	Thompson	Sarah	Lubbock	...
7007	Wilson	Jim	Salem	...
7008	Seidel	Manuela	Medford	...
7009	Smith	David	Fremont	...
7010	Patterson	Neil	San Diego	...
7011	Viescas	Michael	Redmond	...

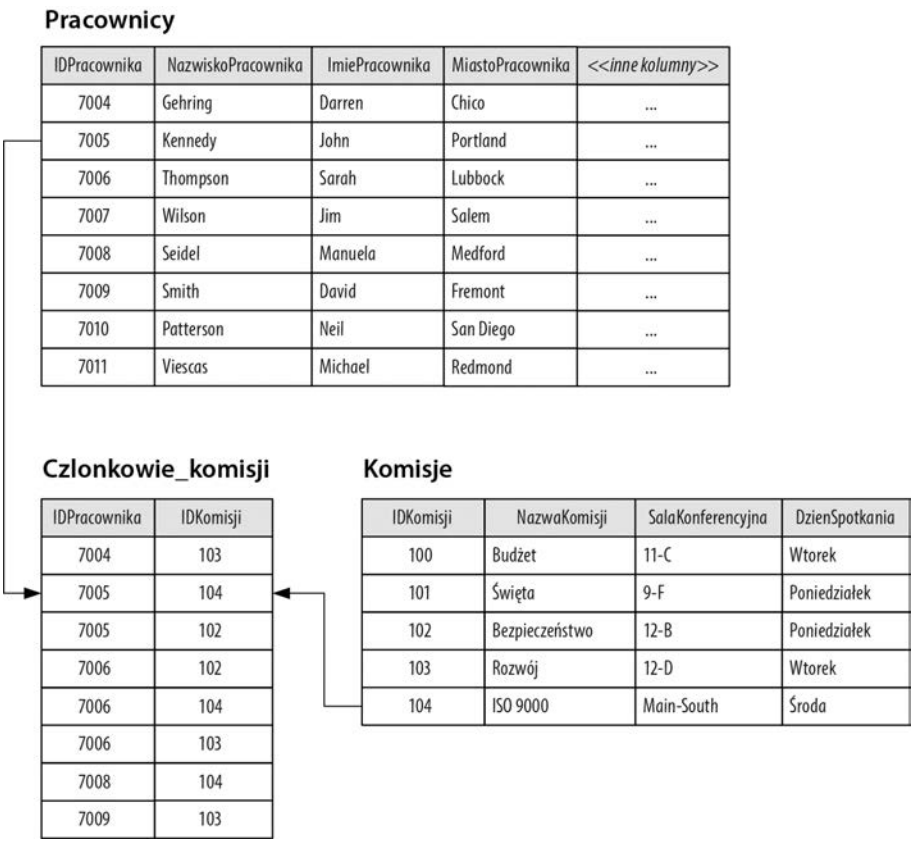
Czlonkowie_komisji

IDPracownika	Komisja
7004	ds. rozwoju
7005	ISO 9000
7005	ds. bezpieczeństwa
7006	ds. bezpieczeństwa
7006	ISO 9000
7006	ds. rozwoju
7008	ISO 9000
7009	ds. rozwoju

Rysunek 2.10. Poprawiona tabela Pracownicy oraz nowa tabela Czlonkowie_komisji

Problem z powtarzającymi się kolumnami, które znajdowały się w oryginalnej tabeli `Pracownicy`, został rozwiązany, ale to jeszcze nie koniec. Pamiętając o fakcie, że między pracownikami a komisjami, do których oni należą, istnieje związek wiele do wielu, mógłbyś zadać słuszne pytanie: „a gdzie jest tabela `Komisje`?”. Nie ma takiej... na razie! Całkiem możliwe, że komisje mają pewne cechy, które warto odnotowywać, na przykład nazwę sali, w której są organizowane spotkania, albo dzień tygodnia, w którym te

spotkania się odbywają. W takim przypadku dobrze byłoby utworzyć tabelę `Komisje`, która zawierałaby pola takie jak `IDKomisji`, `NazwaKomisji`, `SalaKonferencyjna` i `DzienSpotkania`. Po utworzeniu takiej tabeli kolumnę `Komisja` w tabeli `Czlonkowie_komisji` można będzie zastąpić kolumną `IDKomisji` z nowej tabeli `Komisje`. Ostateczna wersja tej struktury została pokazana na rysunku 2.11.



Rysunek 2.11. Ostateczna struktura tabel Pracownicy, Czlonkowie_komisji i Komisje

Opracowanie tabel w podany sposób ma konkretne zalety, choćby z tego względu, że umożliwia powiązanie jednego pracownika z dowolną liczbą komisji, jak również jednej komisji z dowolną liczbą pracowników. Za pomocą zapytania SQL możesz wyświetlić informacje zaczerpnięte ze wszystkich trzech tabel.

Wróćmy do wspomnianego wcześniej problemu dotyczącego wielu, choć niejednakowo nazwanych kolumn, które ewentualnie mogą być duplikatami. Weźmy na przykład tabelę pokazaną na rysunku 2.12.

Pracownicy

IDpracownika	NazwiskoPracownika	ImiePracownika	MiastoPracownika	TelDomPracownika	TelSluzbPracownika	TelKomPracownika	<<inne kolumny>>
7004	Gehring	Darren	Chico	555-1234	556-1234	889-1234	...
7005	Kennedy	John	Portland	555-2345	556-2345	889-2345	...
7006	Thompson	Sarah	Lubbock	555-3456	556-3456	889-3456	...
7007	Wilson	Jim	Salem	555-4567	556-4567		...
7008	Seidel	Manuela	Medford		556-5678	889-5678	...
7009	Smith	David	Fremont	555-5689	556-5689	889-6789	...
7010	Patterson	Neil	San Diego	555-7890	556-7890	889-7890	...
7011	Viescas	Michael	Redmond	555-4321	555-4321		...

Rysunek 2.12. Powtarzające się numery telefonów w kolumnach tabeli Pracownicy

Jaki potencjalny problem dostrzegasz? Przede wszystkim, jeśli pracownik nie ma któregoś rodzaju telefonu, w tabeli marnuje się miejsce. Jest jednak jeszcze większy problem. Domyślasz się jaki? Pomyśl, co należałoby zrobić w sytuacji, gdyby któryś z pracowników miał dwa numery domowe albo osobny numer faksu? A co, jeśli najważniejsi pracownicy mają nie tylko prywatną komórkę, ale też służbową? Rozwiązanie polega na utworzeniu osobnej tabeli `Numery_Telefonow` i powiązaniu jej z tabelą `Pracownicy` w sposób pokazany na rysunku 2.13.

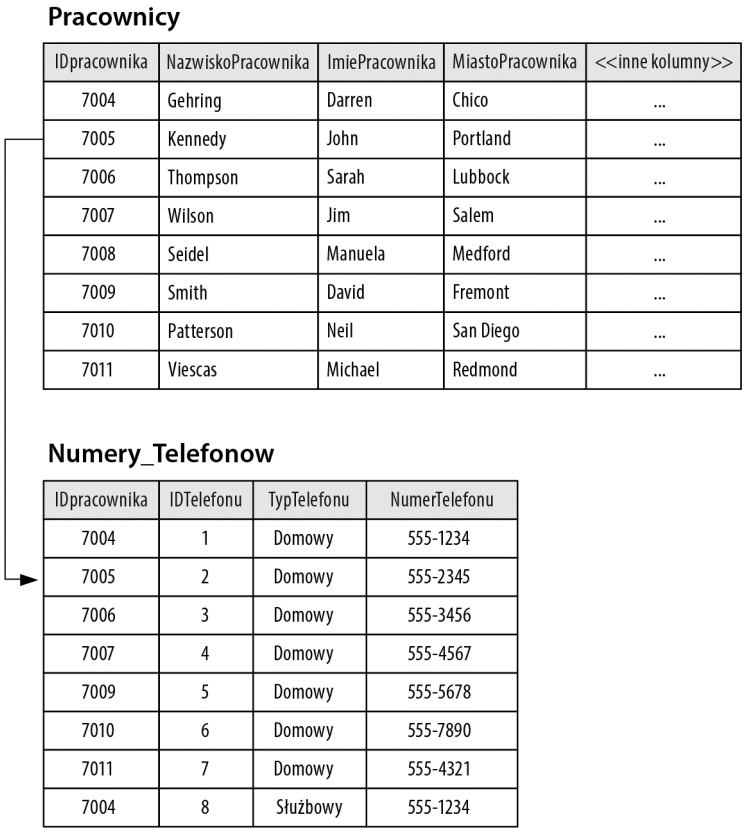
Dzięki takiemu projektowi można zapisać dowolną liczbę numerów telefonów każdego pracownika. Jeśli musiałbym wprowadzić nowy typ telefonu, wystarczyłoby zdefiniować odpowiednią wartość w kolumnie `TypTelefonu`. Zauważ, że w tym przypadku nie marnujemy miejsca w tabeli na nieistniejący numer telefonu domowego pracownika o identyfikatorze 7008 — w tabeli po prostu nie ma wiersza z takim numerem. Zauważ też, że każdy wiersz w tabeli `Numery_Telefonow` zawiera pole `IDTelefonu` z unikatową wartością. Więcej informacji o roli unikatowego identyfikatora wiersza znajdziesz w następnym podrozdziale.

Powoli zbliżamy się do końca procesu optymalizowania struktury tabel. Ostatnie zadanie będzie polegało na sprawdzeniu, czy każdy wiersz w tabeli można jednoznacznie zidentyfikować i czy sama tabela może być jednoznacznie zidentyfikowana w ramach całej bazy danych.

Identyfikacja to klucz

W rozdziale 1. „Co to znaczy »relacyjna«?" dowiedziałeś się, że klucz główny jest jednym z najważniejszych kluczy w tabeli, ponieważ pozwala on na zidentyfikowanie każdego wiersza w tabeli, jak również identyfikuje samą tabelę w bazie danych. Klucz tego rodzaju pozwala też na ustanowienie związku między dwiema tabelami. Doprawdy trudno przecenić rolę kluczy głównych — każda tabela w Twojej bazie powinna zawierać taki klucz!

Z definicji klucz główny jest kolumną albo grupą kolumn jednoznacznie identyfikującą każdy wiersz w tabeli. Klucz główny składający się z jednej kolumny jest nazywany *kluczem głównym prostym* (albo zwyczajnie kluczem głównym). Klucz główny składający się z dwóch



Pracownicy

IDpracownika	NazwiskoPracownika	ImiePracownika	MiastoPracownika	<<inne kolumny>>
7004	Gehring	Darren	Chico	...
7005	Kennedy	John	Portland	...
7006	Thompson	Sarah	Lubbock	...
7007	Wilson	Jim	Salem	...
7008	Seidel	Manuela	Medford	...
7009	Smith	David	Fremont	...
7010	Patterson	Neil	San Diego	...
7011	Viescas	Michael	Redmond	...

Numery_Telefonow

IDpracownika	IDTelefonu	TypTelefonu	NumerTelefonu
7004	1	Domowy	555-1234
7005	2	Domowy	555-2345
7006	3	Domowy	555-3456
7007	4	Domowy	555-4567
7009	5	Domowy	555-5678
7010	6	Domowy	555-7890
7011	7	Domowy	555-4321
7004	8	Służbowy	555-1234

Rysunek 2.13. Rozwiązanie problemu z numerami telefonów za pomocą osobnej tabeli tematycznej

lub większej liczby kolumn nazywa się *kluczem głównym złożonym*. Zawsze gdy to tylko możliwe, należy definiować klucze główne proste, ponieważ tworzenie związków za ich pomocą jest bardziej efektywne i łatwiejsze. Kluczy złożonych należy używać tam, gdzie to konieczne, na przykład przy tworzeniu i definiowaniu tabeli łączącej.

Do utworzenia klucza głównego może posłużyć dowolna kolumna lub kombinacja kolumn spełniająca podane niżej kryteria. Jeśli kolumna lub kolumny, jakich zamierzasz użyć w roli klucza głównego, nie spełniają *wszystkich* podanych kryteriów, to użyj innej kolumny lub zdefiniuj nową, która będzie pełniła funkcję klucza głównego tabeli. Poświęć teraz trochę czasu i na podstawie poniższej listy określ, czy wszystkie klucze główne w Twojej bazie danych zostały prawidłowo zdefiniowane.

- *Czy kolumny klucza w sposób jednoznaczny identyfikują dowolny wiersz w tabeli?* Każdy wiersz w tabeli reprezentuje pewne wystąpienie tematu tej tabeli. Dobry klucz główny gwarantuje możliwość zidentyfikowania lub odwołania się do

każdego wiersza z tej tabeli z poziomu innych tabel w bazie danych. Pomaga on też w uniknięciu duplikowania wierszy w tabeli.

- *Czy wybrana kolumna lub kombinacja kolumn zawiera unikatowe wartości?* Dopóki wartości klucza głównego są unikatowe, masz możliwość sprawdzenia, że w tabeli nie ma powtarzających się wierszy.
- *Czy wybrane kolumny będą mogły zawierać nieznane wartości?* To bardzo ważne pytanie, ponieważ klucz główny nie może zawierać nieznanych wartości. Kolumnę, co do której zachodzi najlżejsze podejrzenie, że mogą się w niej kiedyś znaleźć niesprecyzowane wartości, należy od razu zdyskwalifikować.
- *Czy wartość wybranych kolumn może być opcjonalna?* Jeśli odpowiedź brzmi tak, to nie należy używać takich kolumn w charakterze klucza głównego. Opcjonalna wartość kolumny oznacza, że na pewnym etapie mogą się w niej pojawić nieznane wartości, a zgodnie z tym, o czym przeczytałeś w poprzednim punkcie, klucz główny nie może takich wartości zawierać.
- *Czy jest to kolumna wieloczęściowa?* Dobrze jest zadać sobie to pytanie, choć na tym etapie pracy powinieneś już wyeliminować z bazy wszystkie kolumny wieloczęściowe. Jeśli jednak przypadkiem przegapiłeś tego rodzaju kolumnę, rozpraw się z nią od razu, a w charakterze klucza głównego użyj innej kolumny bądź zdefiniuj złożony klucz na podstawie kolumn otrzymanych po rozdzieleniu kolumny wieloczęściowej.
- *Czy wartości w wybranych kolumnach mogą kiedyś ulec zmianie?* Wartości w kolumnach klucza głównego powinny pozostać stałe. Nigdy nie należy zmieniać wartości w kolumnie klucza głównego; chyba że ma się wyjątkowy powód. Jeśli wartość kolumny może podlegać arbitralnym zmianom, to taka kolumna raczej nie będzie spełniała warunków podanych we wcześniejszych punktach listy.

Zgodnie z tym, o czym wspomniałem wcześniej, kolumna albo kombinacja kolumn muszą śpiewająco spełnić wszystkie warunki z powyższej listy, jeśli mają one być użyte w roli klucza głównego. Na rysunku 2.14 kolumna `IDPilota` jest kluczem głównym tabeli `Piloci`. Powstaje jednak pytanie: czy kolumna `IDPilota` spełnia wszystkie warunki z podanej listy? Jeśli tak, to taki klucz główny będzie prawidłowy, a jeśli nie, to kolumnę trzeba będzie zmodyfikować tak, by spełniała wszystkie podane warunki, lub utworzyć klucz główny na podstawie innej kolumny.

W istocie `IDPilota` jest prawidłowym kluczem głównym, gdyż spełnia wszystkie warunki z powyższej listy. Co jednak zrobić, jeśli w tabeli nie ma kolumny, której można byłoby użyć w charakterze klucza głównego? Weźmy na przykład tabelę `Pracownicy` z rysunku 2.15. Czy w tej tabeli istnieje kolumna, które może pełnić funkcję klucza głównego?

Piloci

IDPilota	ImiePilota	NazwiskoPilota	DataZatrudnienia	Funkcja	NumerKierPilota	TelefonPilota
25100	Sam	Alborous	1994-07-11	Kapitan	206	555-3982
25101	Jim	Wilson	1994-05-01	Kapitan	206	555-6657
25102	David	Smith	1994-09-11	Pierwszy oficer	915	555-1992
25103	Kathryn	Patterson	1994-07-11	Nawigator	972	555-8832
25104	Michael	Hernandez	1994-05-01	Nawigator	360	555-9901
25105	Kendra	Bonnicksen	1994-09-11	Kapitan	206	555-1106

Rysunek 2.14. Czy IDPilota jest dobrym kluczem głównym?

Pracownicy

NazwiskoPrac	ImiePrac	MiastoPrac	StanZamPrac	KodPocztPrac	NumKierPrac	TelefonPrac	DataZatr
Gehring	Darren	Chico	CA	95926			1998-12-31
Kennedy	John	Portland	OR	97208	503	555-2621	1998-05-01
Thompson	Sarah	Redmond	WA	98052	425	555-2626	1998-09-11
Wilson	Jim	Salem	OR				1998-12-27
Seidel	Manuela	Medford	OR	97501	541	555-2641	1998-05-01
Smith	David	Fremont	CA	94538	510	555-2646	1998-09-11
Patterson	Neil	San Diego	CA	92199	619	555-2541	1998-05-01
Viescas	Michael	Redmond	WA	98052	425	555-2511	1998-09-11
Viescas	David	Portland	OR	97207	503	555-2633	1998-10-15

Rysunek 2.15. Czy ta tabela ma klucz główny?

Jest oczywiste, że ta tabela nie zawiera kolumny (albo grupy kolumn), która może być użyta w roli klucza głównego. Z wyjątkiem kolumny `TelefonPrac` każda kolumna zawiera powtarzające się wartości, zaś w kolumnach `KodPocztPrac`, `NumKierPrac` i we wspomnianym `TelefonPrac` mogą się pojawić wartości nieznane. Choć kuszące wydaje się użycie kombinacji kolumn `NazwiskoPrac` i `ImiePrac`, to nie ma gwarancji, że firma w przyszłości nie zatrudni kolejnej osoby o imieniu i nazwisku Jim Wilson albo David Smith. W tej sytuacji okazuje się, że w tej tabeli nie ma kolumny, której można byłoby użyć jako klucza głównego: wartość każdej z zawartych w niej kolumn może bowiem ulegać arbitralnym zmianom.

Co zrobić? Można byłoby się pokusić o użycie jakiegoś ogólnokrajowego numeru identyfikacyjnego, który posiada każdy pracownik, na przykład numeru ubezpieczenia albo PESEL. Istnieje jednak bardzo niewielkie (ale zawsze) ryzyko, że ten sam numer będą miały dwie osoby lub więcej. W razie takich wątpliwości rozwiązanie polega na utworzeniu sztucznego klucza głównego. Jest to arbitralna kolumna, którą należy zdefiniować i dodać do tabeli tylko po to, by można było używać jej jako klucza głównego. Zaletą dodania arbitralnych kolumn jest gwarancja spełnienia wszystkich warunków z wcześniejszej

listy. Po dodaniu kolumny do tabeli nadaj jej status klucza głównego — i gotowe! To naprawdę wystarczy. Rysunek 2.16 przedstawia tabelę `Pracownicy` z dodanym „sztucznym" kluczem głównym o nazwie `IDPracownika`.

Pracownicy

IDPracownika	NazwiskoPrac	ImiePrac	MiastoPrac	StanZamPrac	KodPocztPrac	*<< inne kolumny >>*
98001	Gehring	Darren	Chico	CA	95926	...
98002	Kennedy	John	Portland	OR	97208	...
98003	Thompson	Sarah	Redmond	WA	98052	...
98004	Wilson	Jim	Salem	OR		...
98005	Seidel	Manuela	Medford	OR	97501	...
98006	Smith	David	Fremont	CA	94538	...
98007	Patterson	Neil	SanDiego	CA	92199	...
98008	Viescas	Michael	Redmond	WA	98052	...
98009	Viescas	David	Portland	OR	97207	...

Rysunek 2.16. Tabela Pracownicy z dodanym sztucznym kluczem głównym

Uwaga. Choć sztuczne klucze główne są bardzo łatwym sposobem na rozwiązanie problemu, tak naprawdę nie gwarantują one, że dane w tabeli nie będą się powtarzać. Na przykład, jeśli ktoś doda nowy wiersz dla osoby o imieniu i nazwisku John Kennedy i nada temu wierszowi nową, unikatową wartość klucza `IDPracownika`, to skąd będzie wiadomo, że ten drugi John Kennedy nie jest tym samym co pracownik o numerze 98002, który już znajduje się w tabeli?

Rozwiązanie polega na wyposażeniu aplikacji w kod weryfikujący, który będzie wyszukiwał potencjalne duplikaty i ostrzegał użytkownika przed ich utworzeniem. W wielu systemach do obsługi baz danych istnieje możliwość napisania takiego kodu w postaci tzw. wyzwalacza (ang. *trigger*), uruchamianego za każdym razem, gdy jakiś wiersz ulegnie zmianie, zostanie dodany albo usunięty. Omówienie wyzwalaczy wykracza jednak daleko poza ramy materiału przedstawionego w tej książce. Więcej informacji na ich temat znajdziesz w dokumentacji do używanego systemu baz danych.

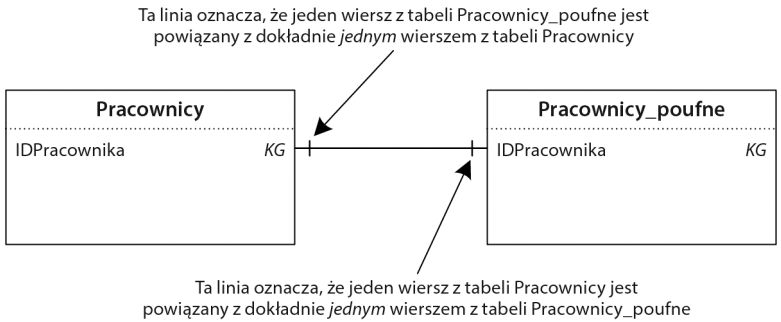
Na tym etapie zrobiłeś już wszystko, co było możliwe, by usprawnić i zoptymalizować strukturę tabel. Teraz zastanówmy się, jak zagwarantować poprawność związków między tabelami.

Definiowanie poprawnych zależności

W rozdziale 1. przeczytałeś o tym, że można mówić o związku między dwiema tabelami, jeśli wiersze w pierwszej tabeli są w jakiś sposób powiązane z wierszami w drugiej tabeli. Dowiedziałeś się też, że wyróżnia się trzy rodzaje takich zależności: jeden do jednego, jeden do wielu i wiele do wielu. Przeczytałeś też o tym, że każdą z tych zależności definiuje się w określony sposób. Wróćmy na chwilę do tych rozważań.

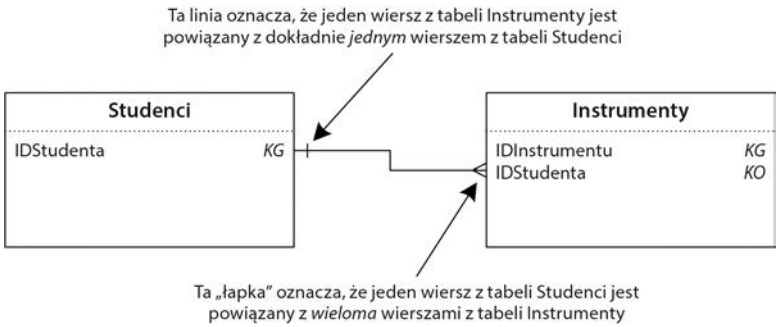
Uwaga. Symbole pokazane na diagramach w tej sekcji rozdziału zostały zaczerpnięte z metody kreślenia diagramów opisanej przez Mike'a Hernandeza w książce *Projektowanie baz danych dla każdego. Przewodnik krok po kroku* (wyd. III, Helion 2014). Zastosowałem ponadto symbol KG do oznaczenia kolumny klucza głównego, KO dla kolumny klucza obcego, zaś ZKG do oznaczenia kolumny stanowiącej jeden z elementów złożonego klucza głównego.

- **Związek jeden do jednego** definiuje się poprzez wzięcie klucza głównego z tabeli głównej i wstawienie go do tabeli drugorzędnej, w której staje się on kluczem obcym. Jest to szczególny rodzaj zależności, ponieważ w wielu przypadkach klucz obcy będzie zarazem pełnił funkcję klucza głównego tabeli podrzędnej. Rysunek 2.17 ilustruje sposób przedstawienia takiej relacji na diagramie.



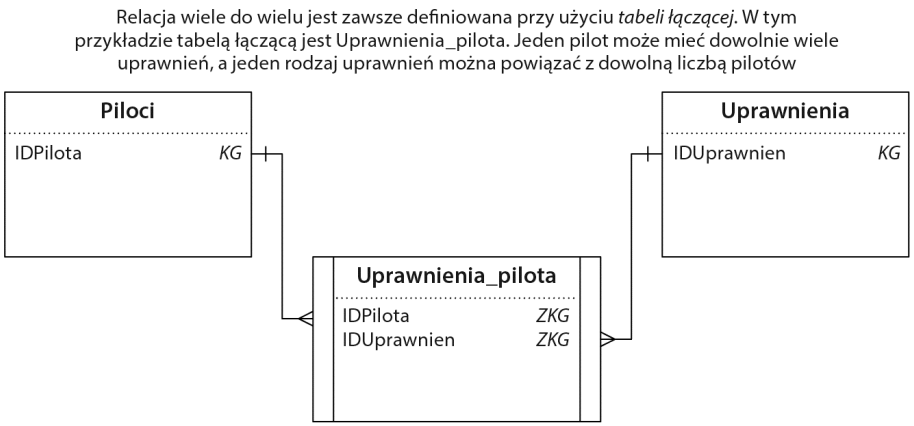
Rysunek 2.17. Przedstawienie zależności jeden do jednego na diagramie

- Związek **jeden do wielu** definiuje się poprzez wzięcie klucza głównego z tabeli po stronie „jeden" i wstawienie go do tabeli po stronie „wiele", gdzie staje się on kluczem obcym. Rysunek 2.18 ilustruje sposób przedstawienia takiej zależności na diagramie.



Rysunek 2.18. Przedstawienie zależności jeden do wielu na diagramie

- Związek **wiele do wielu** definiuje się przez utworzenie tabeli łączącej. Tabelę taką tworzy się na podstawie kopii kluczy głównych należących do każdej z powiązanych tabel. Kolumny te zwykle pełnią dwie odrębne funkcje: łącznie stanowią złożony klucz główny tabeli łączącej, zaś osobno są kluczami obcymi. Rysunek 2.19 ilustruje sposób przedstawienia takiej zależności na diagramie.



Rysunek 2.19. Przedstawienie zależności wiele do wielu na diagramie

Aby mieć pewność, że związki między tabelami w bazie danych są bez zarzutu, trzeba określić właściwości każdego z nich. Właściwości te będą decydowały o tym, co stanie się w przypadku usunięcia wiersza, jaką rolę dana tabela odgrywa w związku i w jakim stopniu w nim uczestniczy.

Należy zauważyć, że kolumny wykorzystane do połączenia dwóch tabel muszą być tego samego typu. Na przykład klucz główny typu *Int* (*Integer*) można połączyć tylko z kluczem obcym typu *Int*. Nie da się powiązać liczby z kolumną znakową albo datą. Jedynym wyjątkiem od tej reguły są klucze główne automatycznie generowane przez system baz danych, znane pod różnymi nazwami, w zależności od systemu (`AutoNumber`, `Identity`,

`Serial` albo `Auto_Increment`). Każdy z tych kluczy ma określony typ danych numerycznych — w przypadku programu Microsoft Access jest to typ *Long Integer*, w większości pozostałych typ *Int* — który można łączyć z kolumnami typu numerycznego. Da się więc połączyć kolumnę `AutoNumber` w Accessie z kluczem obcym typu *Number* lub *Long Integer* w skojarzonej tabeli, a w systemie Microsoft SQL Server da się połączyć kolumnę typu `Identity` z kolumną typu *Int*.

Przed przystąpieniem do omówienia właściwości związków muszę wyraźnie podkreślić jedną rzecz: poniższe właściwości zostały wymienione w pewnej ogólnej, logicznej kolejności. Właściwości te są ważne, ponieważ decydują o integralności związków (w niektórych systemach baz danych nazywanej *integralnością referencyjną*). Sposób ich implementacji może jednak być różny w zależności od używanego programu bazodanowego. Aby określić, czy dane właściwości są obsługiwane przez Twój system baz danych i jak je skonfigurować, musisz sięgnąć do jego dokumentacji.

Definiowanie reguły usuwania

Reguła usuwania decyduje o tym, co dzieje się w chwili, gdy użytkownik postanowi usunąć wiersz z tabeli głównej pozostającej z inną tabelą w zależności jeden do jednego bądź wiersz z tabeli po stronie „jeden" ze związku jeden do wielu. Zdefiniowanie tej reguły pozwala uniknąć powstawania osieroconych wierszy. (*Osierocone wiersze* to te wiersze z tabeli drugorzędnej w związku jeden do jednego, które nie mają odpowiadających im wierszy z tabeli głównej, bądź wiersze z tabeli po stronie „wiele" w związku jeden do wielu, które nie mają odpowiadających im wierszy w tabeli po stronie „jeden").

Dla danej zależności można zdefiniować jedną z dwóch reguł usuwania: *restrykcyjną* i *kaskadową*.

- Restrykcyjna reguła usuwania nie pozwala na usunięcie wskazanego wiersza, jeśli w tabeli drugorzędnej (w związku jeden do jednego) lub w tabeli po stronie „wiele" (w związku jeden do wielu) istnieją wiersze z nim powiązane. Wszelkie powiązane wiersze należy usunąć *przed* usunięciem żądanego wiersza. Tej reguły usuwania używa się na ogół bezwiednie. W systemach baz danych, które umożliwiają definiowanie reguł relacji, jest to zwykle opcja domyślna; w pozostałych — jedyna.

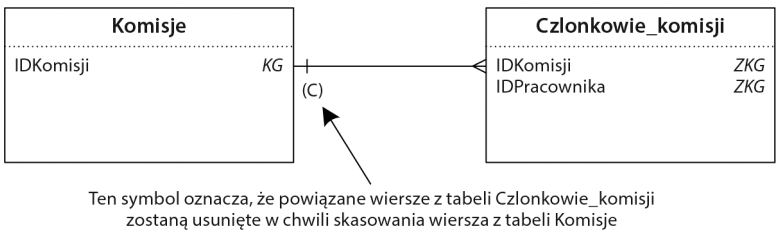
- Jeśli obowiązuje kaskadowa reguła usuwania, to skasowanie wiersza po stronie „jeden" zależności powoduje automatyczne usunięcie wszystkich powiązanych z nim wierszy w tabeli drugorzędnej (w związku jeden do jednego) lub z tabeli po stronie „wiele" (w związku jeden do wielu). Tej reguły należy używać z rozwagą, bo może ona spowodować usunięcie wierszy, które tak naprawdę chciałeś zostawić! Nie wszystkie systemy baz danych obsługują kaskadową regułę usuwania.

Niezależnie od bieżącej reguły usuwania zawsze bardzo uważnie przyjrzyj się zależnościom w bazie danych, aby określić, która z nich będzie właściwsza. Aby ułatwić sobie podjęcie właściwej decyzji, zadaj sobie bardzo proste pytanie. Najpierw wybierz parę tabel, a potem zastanów się nad następującą kwestią: „Jeśli wiersz w [`nazwa tabeli głównej lub`

po stronie „jeden"] zostanie usunięty, to czy powiązane z nim wiersze w [nazwa tabeli drugorzędnej lub po stronie „wiele"] również powinny zostać skasowane?".

To pytanie zostało sformułowane w postaci ogólnej, aby ułatwić Ci zrozumienie jego idei. Aby zastosować je w praktyce, zastąp wyrażenia w nawiasach kwadratowych odpowiednimi nazwami tabel. Pytanie powinno wtedy wyglądać na przykład tak: „Jeśli wiersz w tabeli `Komisje` zostanie usunięty, to czy powiązane z nim wiersze w tabeli `Czlonkowie_komisji` również powinny zostać skasowane?".

Jeśli odpowiedź na to pytanie brzmi „nie", zastosuj restrykcyjną regułę usuwania. W przeciwnym razie zastosuj kaskadową regułę usuwania. W praktyce odpowiedź na to pytanie w dużej mierze zależy od sposobu wykorzystania danych przechowywanych w bazie. Właśnie z tego względu warto uważnie analizować zależności między tabelami i wybrać właściwą regułę. Rysunek 2.20 przedstawia sposób przedstawienia na diagramie reguły usuwania dla omawianej zależności. Zauważ, że reguła restrykcyjna jest oznaczana na diagramach symbolem (R), zaś kaskadowa symbolem (C).



Rysunek 2.20. Przedstawienie reguły usuwania dla tabel Komisje i Czlonkowie_komisji na diagramie

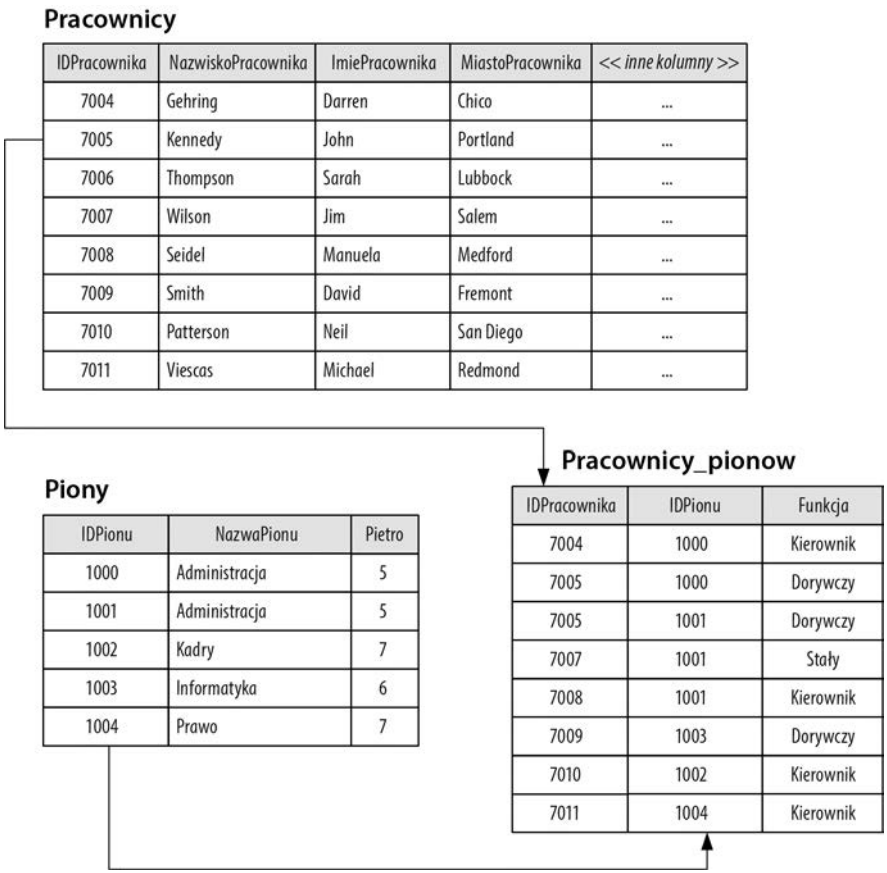
Definiowanie rodzaju uczestnictwa

Przy określaniu związku między dwiema tabelami należy wziąć pod uwagę fakt, że każda z nich bierze w nim udział w określony sposób. *Rodzaj uczestnictwa* przypisany do danej tabeli decyduje o tym, czy w tabeli tej musi istnieć wiersz, by można było wprowadzić wiersz do drugiej tabeli. Wyróżnia się dwa rodzaje uczestnictwa:

- Obowiązkowy — co najmniej jeden wiersz musi istnieć w danej tabeli, zanim wprowadzisz dowolny wiersz w drugiej tabeli.
- Opcjonalny — nie ma wymogu istnienia wierszy w danej tabeli, by można było wprowadzić dowolne wiersze w drugiej tabeli.

Rodzaj uczestnictwa, jaki wybierzesz dla dwóch tabel, jest uzależniony głównie od logiki biznesowej Twojej organizacji. Przyjmijmy na przykład, że pracujesz w dużej firmie składającej się z kilku pionów. Załóżmy też, że w bazie utworzonej dla tej firmy masz tabele `Pracownicy`, `Piony` i `Pracownicy_pionow`. Wszystkie informacje dotyczące pracownika znajdują się w tabeli `Pracownicy`, zaś wszystkie istotne informacje dotyczące pionów firmy znajdują się w tabeli `Piony`. Tabela `Pracownicy_pionow` jest tabelą łączącą, która umożliwia powiązanie

dowolnej liczby pionów z danym pracownikiem. Wymienione tabele zostały zilustrowane na rysunku 2.21. (Na tym rysunku użyłem zwykłych strzałek, skierowanych grotem w stronę „wiele" danej zależności).



Pracownicy

IDPracownika	NazwiskoPracownika	ImiePracownika	MiastoPracownika	<< inne kolumny >>
7004	Gehring	Darren	Chico	...
7005	Kennedy	John	Portland	...
7006	Thompson	Sarah	Lubbock	...
7007	Wilson	Jim	Salem	...
7008	Seidel	Manuela	Medford	...
7009	Smith	David	Fremont	...
7010	Patterson	Neil	San Diego	...
7011	Viescas	Michael	Redmond	...

Pracownicy_pionow

IDPracownika	IDPionu	Funkcja
7004	1000	Kierownik
7005	1000	Dorywczy
7005	1001	Dorywczy
7007	1001	Stały
7008	1001	Kierownik
7009	1003	Dorywczy
7010	1002	Kierownik
7011	1004	Kierownik

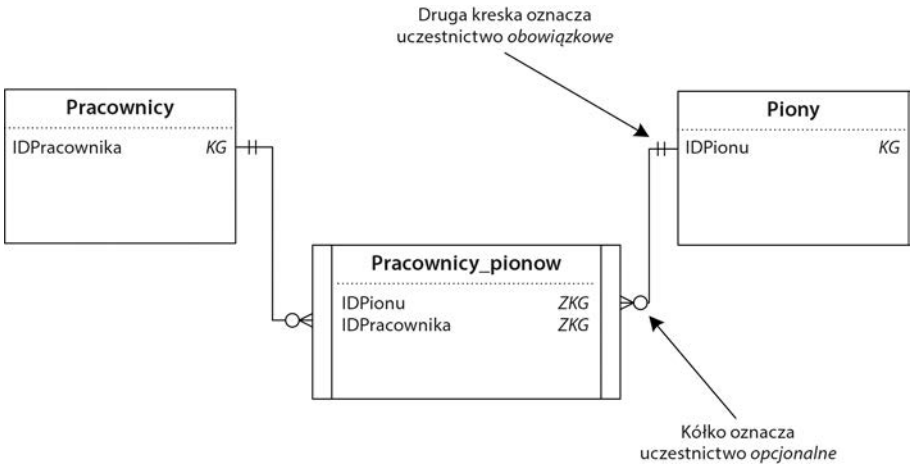
Piony

IDPionu	NazwaPionu	Pietro
1000	Administracja	5
1001	Administracja	5
1002	Kadry	7
1003	Informatyka	6
1004	Prawo	7

Rysunek 2.21. Tabele Pracownicy, Piony i Pracownicy_pionow

Na niedawnym spotkaniu powiedziano Ci, że trzeba przydzielić niektórych pracowników do nowego pionu o nazwie `Badania i Rozwój`. Tutaj powstaje problem: chcesz mieć pewność, że najpierw dodasz nowy pion do tabeli `Piony`, aby potem można było przydzielić do niego pracowników za pośrednictwem tabeli `Pracownicy_pionow`. W takich przypadkach do głosu dochodzi rodzaj uczestnictwa tabeli w związku. Zmień rodzaj uczestnictwa dla tabeli `Piony` na obowiązkowy, zaś dla tabeli `Pracownicy_pionu` na opcjonalny. Dzięki takim ustawieniom będziesz miał pewność, że dany pion musi najpierw zaistnieć w tabeli `Piony`, zanim dowolni pracownicy zostaną do niego przypisani przy użyciu tabeli `Pracownicy_pionow`.

Podobnie jak w przypadku reguły usuwania uważnie przeanalizuj każdy związek, aby wybrać właściwy rodzaj uczestnictwa dla każdej tabeli w tym związku. Rodzaj uczestnictwa można przedstawić na diagramie, tak jak zostało to zrobione na rysunku 2.22.



Rysunek 2.22. Diagram ilustrujący rodzaj uczestnictwa dla tabel Piony i Pracownicy_pionow

Określanie stopnia uczestnictwa

Po zadecydowaniu, *jak* tabela będzie uczestniczyła w związku, trzeba określić, *do jakiego stopnia* każda z tabel będzie w nim uczestniczyć. Stopień ten określa się przez zdefiniowanie minimalnej i maksymalnej liczby wierszy z jednej tabeli, które mogą się odwoływać do jednego wiersza z drugiej tabeli. Ten proces nazywa się określaniem *stopnia uczestnictwa* tabeli w związku. Stopień uczestnictwa dla danej tabeli odzwierciedlają dwie liczby rozdzielone przecinkiem i ujęte w nawiasy zwykłe. Pierwsza liczba oznacza minimalną możliwą liczbę powiązanych wierszy, a druga maksymalną możliwą liczbę powiązanych wierszy. Na przykład stopień uczestnictwa wyrażony w postaci „(1,12)" oznacza, że minimalna liczba powiązanych wierszy wynosi 1, a maksymalna 12.

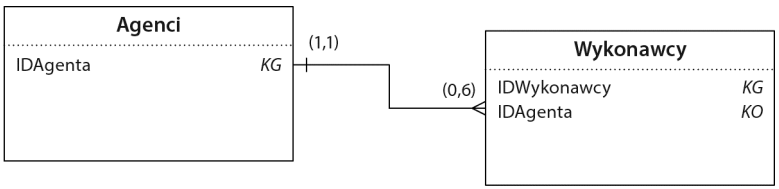
Stopień uczestnictwa, jaki określisz dla różnych tabel w bazie, w dużej mierze zależy od tego, jak Twoja organizacja przegląda dane i jak z nich korzysta. Przypuśćmy, że jesteś łowcą talentów i pracujesz dla agencji artystycznej. W bazie danych Twojej firmy są dwie tabele o nazwach `Agenci` i `Wykonawcy`. Załóżmy ponadto, że między tymi tabelami zachodzi zależność jeden do wielu: jeden wiersz z tabeli `Agenci` może być powiązany z wieloma wierszami z tabeli `Wykonawcy`, ale jeden wiersz z tabeli `Wykonawcy` można powiązać tylko z jednym wierszem z tabeli `Agenci`. To oznacza, że zagwarantowaliśmy (w sensie ogólnym), że dany wykonawca może być powiązany tylko z jednym agentem („łowcą talentów"). (W ten sposób uniknęliśmy zarazem sytuacji, w których jeden wykonawca może „rozgrywać" agentów między sobą, co jest ze wszech miar słuszne).

Niemal we wszystkich przypadkach maksymalna liczba wierszy po stronie „wiele” zależności będzie nieskończona. Jednak w niektórych sytuacjach reguły panujące w przedsiębiorstwie mogą narzucać ograniczenie tego limitu. Przykładem jest ograniczenie liczby uczniów, którzy mogą zapisać się do jednej klasy. Zaś we wspomnianym przed chwilą przypadku możemy założyć, że szef chce zagwarantować swoim agentom porównywalne prowizje i zarazem do minimum ograniczyć wewnętrzną rywalizację między nimi. W tym celu ogłasza, że jeden agent może reprezentować najwyżej sześciu wykonawców. (Choć ma obawy, że na dłuższą metę to rozwiązanie się nie sprawdzi, to jednak i tak zamierza spróbować). Aby wdrożyć nowe ustalenie, określa następujący stopień uczestnictwa dla obydwu tabel:

`Agenci` (1,1) — wykonawca może być powiązany z jednym i tylko jednym agentem.

`Wykonawcy` (0,6) — choć agent może nie być powiązany z jakimkolwiek wykonawcą, to zarazem w danej chwili nie może być powiązany z więcej niż sześcioma wykonawcami.

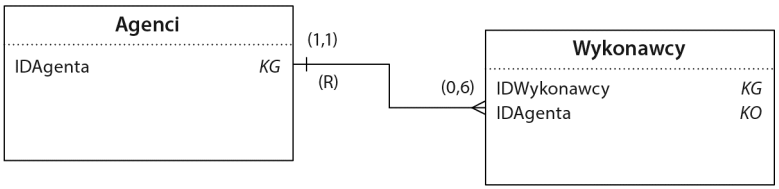
Rysunek 2.23 pokazuje, w jaki sposób przedstawić stopień uczestnictwa dla tych tabel na diagramie.



Rysunek 2.23. Diagram ilustrujący stopień uczestnictwa dla tabel Agenci i Wykonawcy

Po określeniu stopnia uczestnictwa należy określić, w jaki sposób zdefiniować taką zależność w systemie baz danych. Sposób konfiguracji zależy od funkcji dostępnych w systemie, którego używasz. Najprostsza metoda narzucenia wybranych parametrów, obsługiwana przez większość systemów baz danych, polega na ograniczeniu wartości klucza obcego w tabeli po stronie „wiele” tak, by użytkownik nie mógł wprowadzić wartości, która nie istnieje w powiązanej tabeli po stronie „jeden”. Na diagramie można to przedstawić przez umieszczenie litery R (od „restrykcji”) w nawiasach zwykłych obok linii relacji wskazującej na tabelę po stronie „jeden”, jak na rysunku 2.24.

Niektóre systemy baz danych umożliwiają zdefiniowanie reguły, która aktualizuje („kaskaduje”; symbol C) wartość klucza tabeli po stronie „wiele”, jeśli użytkownik zmieni wartość klucza głównego w tabeli po stronie „jeden”. Polega to po prostu na skorygowaniu przez system bazodanowy wartości klucza obcego w odpowiednich wierszach tabeli „wiele”, gdy zmieni się wartość klucza głównego w tabeli „jeden”. Zaś niektóre systemy baz danych



Rysunek 2.24. Diagram przedstawiający wszystkie właściwości związku między tabelami Agenci i Wykonawcy

obsługują funkcję umożliwiającą automatyczne usunięcie (symbol D) wiersza w tabeli po stronie „wiele", gdy usunie się wiersz w tabeli po stronie „jeden". Szczegółowe informacje na ten temat znajdziesz w dokumentacji systemu.

Uwaga. Aby wymusić ograniczenie stopnia uczestnictwa, trzeba zdefiniować jeden lub więcej wyzwalaczy lub ograniczeń w definicji bazy danych (jeśli Twój system do obsługi baz danych na to pozwala).

I to już wszystko?

Dzięki metodom postępowania opisanym w tym rozdziale możesz poczynić podstawowe, niezbędne kroki w kierunku zabezpieczenia integralności danych w bazie. Kolejny etap polega na przeanalizowaniu sposobu, w jaki Twoja organizacja przegląda dane i z nich korzysta, dzięki czemu będziesz mógł określić i zdefiniować dla bazy danych praktyczne reguły biznesowe. Ale aby naprawdę wydobyć z bazy danych maksimum możliwości, powinieneś zacząć od początku i zaprojektować ją zgodnie ze sprawdzoną, skuteczną metodologią projektową. Niestety, omówienie tych zagadnień wykracza poza ramy tej książki. Możesz jednak opanować metodologię projektową na podstawie książek takich jak *Projektowanie baz danych dla każdego. Przewodnik krok po kroku* Michaela J. Hernandeza (wyd. III, Helion 2014) albo *Database Systems: A Practical Approach to Design, Implementation, and Management* Thomasa Connolly'ego i Carolyn Begg (wyd. VI, Addison-Wesley, 2014). Najważniejsza rzecz do zapamiętania to że im solidniejsza będzie struktura bazy, tym łatwiej będzie czerpać informacje z zawartych w niej danych i tworzyć dla niej aplikacje.

Podsumowanie

Ten rozdział rozpocząłem krótkim omówieniem tego, dlaczego warto troszczyć się o poprawne struktury danych. Dowiedziałeś się, że źle zaprojektowane tabele mogą być przyczyną wielu problemów, z których wśród poważniejszych należy wymienić integralność danych.

Następnie omówiłem kwestię optymalizacji kolumn w tabelach. Dowiedziałeś się, że nadawanie kolumnom właściwych nazw jest bardzo ważne, ponieważ dzięki temu nazwy te są zrozumiałe, co z kolei ułatwia wykrywanie potencjalnych problemów ze strukturą kolumn. Wiesz też już, w jaki sposób optymalizować strukturę kolumn poprzez zaadaptowanie ich do kilku prostych reguł. Te reguły mają między innymi gwarantować, że każda kolumna odzwierciedla jedną cechę tematu tabeli, zawiera tylko jedną wartość i nie zawiera obliczeń. Omówiłem też problemy związane z istnieniem wieloczęściowych i wielowartościowych kolumn oraz pokazałem Ci, jak je prawidłowo rozwiązywać.

Kolejnym tematem rozdziału była optymalizacja tabel. Dowiedziałeś się, że nazwy tabel są równie istotne jak nazwy kolumn, właściwie z tych samych powodów. Wiesz już, w jaki sposób nadawać tabelom poprawne nazwy i jak zyskać pewność, że każda tabela reprezentuje tylko jedno zagadnienie. Następnie przedstawiłem zbiór reguł, dzięki którym można zweryfikować poprawność struktury tabel. Choć niektóre z tych reguł zdają się powielać wskazówki dotyczące optymalizowania kolumn, to przekonałeś się, że reguły dotyczące optymalizacji tabel stanowią dodatkowe zabezpieczenie, dające absolutną pewność co do prawidłowości ich struktury.

W dalszej kolejności zająłem się kwestią kluczy głównych. Przekonałeś się, jak ważne jest zdefiniowanie klucza głównego dla każdej tabeli w bazie. Wiesz już, że klucz główny musi spełniać określone kryteria i że kolumnę pełniącą funkcję takiego klucza należy wybrać bardzo uważnie. Dowiedziałeś się ponadto, że jeśli w tabeli nie ma kolumny, która spełniałaby wszystkie kryteria prawidłowości klucza głównego, możesz sztucznie utworzyć kolumnę, która będzie odgrywała taką rolę.

Rozdział zamknąłem omówieniem definiowania poprawnych zależności między tabelami. Po zapoznaniu się z trzema rodzajami zależności dowiedziałeś się, w jaki sposób zilustrować je na diagramie. Przeczytałeś także, jak należy zdefiniować i ująć na diagramie regułę usuwania. Reguła ta jest ważna, ponieważ pozwala uniknąć powstawania osieroconych wierszy. Ostatnie dwa zagadnienia omówione w tym rozdziale były poświęcone kwestii rodzaju i poziomu uczestnictwa poszczególnych tabel w związku. Dowiedziałeś się, że uczestnictwo tabeli może mieć charakter obowiązkowy lub opcjonalny i że istnieje możliwość określenia dopuszczalnego progu liczby powiązanych wierszy między poszczególnymi tabelami.

W następnym rozdziale pokrótce zapoznasz się z historią SQL i jego ewolucją, która doprowadziła do powstania bieżącej (w chwili pisania tej książki) wersji standardu — SQL:2016.


PROGRAM
PARTNERSKI
GRUPY HELION
1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ
Zmień swoją stronę WWW w działający bankomat!
Dowiedz się więcej i dołącz już dzisiaj!
http://program-partnerski.helion.pl
GRUPA
Helion

Działające zapytania SQL. Prościej wytłumaczyć się nie da!

Od ponad 30 lat język SQL jest głównym narzędziem do pracy na bazach danych i nic nie wskazuje na to, aby jakakolwiek inna technologia mogła zyskać porównywalne znaczenie. Sam SQL wciąż jest unowocześniany i rozwijany. Jego ogromne możliwości w zakresie przetwarzania danych znajdują zastosowanie niemal wszędzie, gdzie trzeba zbudować nową lub wykorzystać istniejącą bazę danych. Oznacza to, że nie tylko programista czy architekt, ale także zaawansowany użytkownik systemów informatycznych powinien poznać ten język. Bez tego nie da się naprawdę zrozumieć działania istniejących aplikacji ani tworzyć własnych!

Oto przystępny przewodnik, dzięki któremu nauczysz się krok po kroku pisać zapytania SQL. Poznasz też narzędzia, które umożliwiają zrozumienie, edytowanie i tworzenie zapytań SQL. Nową wiedzę utrwalisz poprzez analizę setek szczegółowo wyjaśnionych przykładów. Niepostrzeżenie zaczniesz rozwiązywać tak trudne problemy jak złożone wyrażenia warunkowe czy operacje logiczne i nauczysz się nieszablonowego podejścia do zadań wymagających użycia niepowiązanych tabel. Dowiesz się, jak wykonywać skomplikowane operacje na grupach danych, co umożliwi Ci tworzenie wyrafinowanych raportów, oraz jak zwiększać elastyczność mechanizmów agregowania. Zawarty tu materiał jest w pełni niezależny od implementacji SQL, co pozwoli Ci na mistrzowskie operowanie zapytaniami w wielu różnych systemach!

Dzięki tej książce:

- zrozumiesz, czym są relacyjne bazy danych i jak powinny być zbudowane
- dowiesz się, jak poprawnie używać instrukcji SELECT
- nauczysz się wybierać dane z wielu tabel i modyfikować zbiory danych
- będziesz płynnie posługiwać się rozszerzeniami klauzuli GROUP BY
- nauczysz się uzyskiwać odpowiedzi na naprawdę skomplikowane pytania

John L. Viescas od ponad 50 lat zajmuje się bazami danych. Był analitykiem systemów, projektował duże bazy danych i prowadził seminaria techniczne dotyczące relacyjnych baz danych. Obecnie specjalizuje się w zarządzaniu systemami baz danych Microsoft Access i SQL Server. W latach 1993 – 2015 był wielokrotnie wyróżniany tytułem MVP. Napisał też kilka świetnie przyjętych książek o tworzeniu bazodanowych systemów informatycznych.

ISBN 978-83-283-6064-8
9 788328 360648
Cena: 129,00 zł
Addison-Wesley
ALWAYS LEARNING
PEARSON