

Zacznij bezboleśnie tworzyć bazy danych w języku SQL

Rusz głową!

SQL



Pomóż Grześkowi poprawić relacje ze swymi danymi



Przestań w końcu myśleć klucz główny z kluczem obcym



Raz na zawsze naucz się, co jest normalne



Wczytaj ważne pojęcia dotyczące zapytań SQL prosto do swego mózgu



Uchron się przed wstydliwą koniecznością modyfikacji swoich tabel



Sprawdź swą znajomość SQL, wykonując dziesiątki ćwiczeń



O'REILLY®

Lynn Beighley

Hellon

» Idź do

- Spis treści
- Przykładowy rozdział

» Katalog książek

- Katalog online
- Zamów drukowany katalog

» Twój koszyk

- Dodaj do koszyka

» Cennik i informacje

- Zamów informacje o nowościach
- Zamów cennik

» Czytelnia

- Fragmenty książek online

» Kontakt

Helion SA
ul. Kościuszki 1c
44-100 Gliwice
tel. 32 230 98 63
e-mail: helion@helion.pl
© Helion 1991–2010

SQL. Rusz głową!

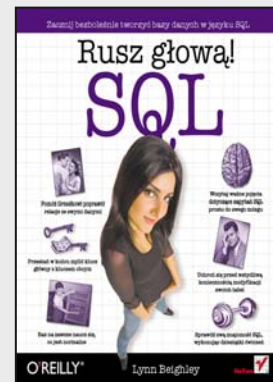
Autor: Lynn Beighley

Tłumaczenie: Piotr Rajca

ISBN: 978-83-246-2828-5

Tytuł oryginału: [Head First SQL: Your Brain on SQL – A Learner](#)

Format: 200×230, stron: 592



Teraz i Ty możesz dołączyć do grona osób znających SQL.

Wystarczy, że sięgniesz po nasz bestseller

Co w nim znajdziesz? Wszystko co najlepsze w serii Rusz głową! Konkretna wiedza, dużo humoru oraz wszystko wyjaśniające rysunki. Na sam początek dowiesz się co to jest język SQL oraz stworzysz swoją pierwszą bazę danych (polecenie CREATE). W kolejnych rozdziałach nauczysz się uzupełniać te bazy danymi, usuwać je oraz modyfikować (polecenia INSERT, UPDATE oraz DELETE). Jak już przebrniesz przez podstawy, zabierzesz się za to co bazodanowcy lubią najbardziej - indeksy, klucze, złączenia oraz podzapytania. Na sam koniec sprawdzisz jak zagwarantować swojej bazie bezpieczeństwo oraz wykrzesać z niej jeszcze więcej możliwości. Brzmi skomplikowanie? Nie martw się, jesteś w dobrych rękach! A więc...

Naprzód, głowo!

Nikt ci tego nie potrafił wytłumaczyć? Nie potrzebujesz elektrowstrząsów, żeby pobudzić swój mózg do aktywnego działania. Tylko żadnych gwałtownych gestów! Usiądź wygodnie, otwórz książkę, dopiero teraz się zaczniesz. Na początek – rusz głową!

Precz z nudnymi wykładami i zakuwaniem bez zrozumienia!

Nauka to znacznie więcej niż tylko czytanie suchego tekstu. Twój mózg jest niczym głodny rekin, cały czas pracy naprzód w poszukiwaniu nowej, apetycznej przekąski.

Jak karmimy Twój wygłodniały umysł?

Używamy rysunków, bo obraz wart jest 1024 słów. Stosujemy powtórzenia, by zakodować na stałe dane w Twojej chłonnej głowie. Oddziałujemy na emocje, jesteśmy nieprzewidywalni, zaskakujący i zabawni. Cały czas pobudzamy Twój umysł do aktywnego działania, zmuszamy go do posłuszeństwa... a za ciężką pracę nagrodzimy go smakowitym ciasteczkiem w postaci wiedzy – wisienka gratis!

Rozgrzyź to sam!

- Tworzenie bazy danych i tabel (CREATE)
- Pobieranie danych (SELECT)
- Usuwanie (DELETE) i aktualizowanie danych (UPDATE)
- Wykorzystywanie kluczy i indeksów
- Sortowanie danych
- Operacje na danych – sumy, średnie, elementy maksymalne i minimalne
- Pobieranie unikalnych danych
- Sposób pobierania określonej liczby wierszy
- Stosowanie podzapytań
- Zapewnianie spójności danych
- Ograniczanie dostępu do zgromadzonych danych
- Zapewnianie bezpieczeństwa bazy danych

Spis treści (skrótowy)

	Wprowadzenie	25
1	Dane i tabele: <i>Na wszystko znajdzie się odpowiednie miejsce</i>	37
2	Polecenie SELECT: <i>Pobieranie podarowanych danych</i>	87
3	DELETE i UPDATE: <i>Są szanse, że wszystko będzie w porządku</i>	153
4	Projektowanie dobrych tabel: <i>Po co być normalnym?</i>	193
5	Polecenie ALTER: <i>Korygowanie przeszłości</i>	231
6	Zaawansowane zastosowanie polecenia SELECT: <i>Nowy sposób spojrzenia na dane</i>	267
7	Projektowanie baz danych składających się z wielu tabel: <i>Wyrastamy z naszych starych tabel</i>	311
8	Złączenia i operacje na wielu tabelach: <i>Czy nie możemy się wszyscy dogadać?</i>	373
9	Podzapytania: <i>Zapytania w zapytaniach</i>	409
10	Złączenia zewnętrzne, złączenia zwrotne oraz unie: <i>Nowe manewry</i>	447
11	Ograniczenia, widoki i transakcje: <i>Zbyt wielu kucharzy psuje bazę danych</i>	483
12	Bezpieczeństwo: <i>Zabezpieczanie swych dóbr</i>	521
	Dodatek A <i>Pozostałości</i>	551
	Dodatek B <i>Instalacja MySQL-a</i>	569
	Dodatek C <i>Przypomnienie narzędzi</i>	575
	Skorowidz	583

Spis treści (z prawdziwego zdarzenia)

Wprowadzenie

Twój mózg myśli o SQL-u. Czytając książkę, Ty starasz się czegoś nauczyć, natomiast Twój mózg wyświadcza Ci przysługę, dbając o to, by te informacje nie zostały zbyt długo w Twej głowie. Twój mózg myśli sobie: „Lepiej zostawić miejsce na jakieś ważne rzeczy, takie jak: których dzikich zwierząt należy unikać albo czy jeżdżenie nago na snowboardzie jest dobrym pomysłem, czy nie”. Zatem w jaki sposób możesz przekonać swój mózg, by uznał, że poznanie SQL-a to dla Ciebie kwestia życia lub śmierci?

	Dla kogo jest ta książka	26
	Wiemy, co sobie myślisz	27
	Metapoznanie: myślenie o myśleniu	29
	Oto co możesz zrobić, aby zmusić swój mózg do posłuszeństwa	31
	Przeczytaj to	32
	Nasi wspaniali recenzenci	34
	Podziękowania	35

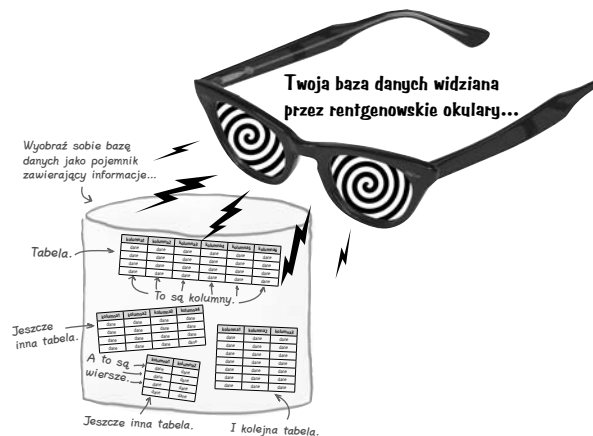
Dane i tabele

1

Na wszystko znajdzie się odpowiednie miejsce

Czy także i Ty nie cierpisz gubienia czegokolwiek? Niezależnie od tego, czy są to kluczyki do samochodu, bon uprawniający do zakupu mieszkania z 25-procentową zniżką, dane używanej aplikacji, nie ma nic gorszego niż **niemożność sprostania własnym potrzebom...** wtedy gdy tego najbardziej potrzebujemy. A jeśli chodzi o używane aplikacje, to trzeba wiedzieć, że nie ma lepszego miejsca na przechowywanie ważnych informacji niż **tabele**. A zatem przewróć kartkę i zacznij, krok za krokiem, poznawać świat **relacyjnych baz danych**.

Definiowanie danych	38
Przeanalizuj swoje dane pod względem kategorii	43
Co znajduje się w bazie danych?	44
Twoja baza danych widziana przez rentgenowskie okulary...	46
Bazy danych zawierają powiązane ze sobą informacje	48
Tabele w zbliżeniu	49
Przejmij kontrolę!	53
Tworzenie tabeli: Polecenie CREATE TABLE	55
Tworzenie bardziej złożonych tabel	56
Przekonajmy się, jak łatwo można pisać kod SQL	57
Utwórzmy w końcu tabelę moje_kontakty	58
Twoja tabela jest gotowa	59
Spotkajmy się z niektórymi typami danych SQL	60
Twoja tabela bez tajemnic	64
Nie można ponownie stworzyć już istniejącej bazy danych lub tabeli!	66
Do kosza ze starą tabelą — czas na nową	68
Aby dodać dane do tabeli, będziesz musiał skorzystać z polecenia INSERT	70
Utworzenie polecenia INSERT	73
Wariacje na temat polecenia INSERT	77
Kolumny bez wartości	78
Zerknij na swoją tabelę, używając polecenia SELECT	79
SQL bez tajemnic: Wyznania wartości NULL	80
Kontrola wewnętrznych wartości NULL	81
NOT NULL pojawia się w wynikach polecenia DESC	83
Wypełnij pustkę słowem kluczowym DEFAULT	84
Przybornik SQL	86



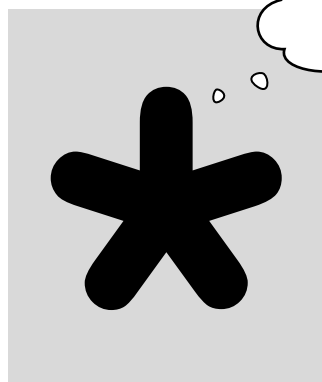
Polecenie SELECT

2

Pobieranie podarowanych danych

Czy naprawdę lepiej jest dawać, niż brać? W przypadku korzystania z baz danych najprawdopodobniej częściej będziesz musiał **pobierać z nich dane**, niż je zapisywać. I właśnie w tych wszystkich sytuacjach przydadzą Ci się informacje przedstawione w tym rozdziale: poznasz w nim bardzo przydatne polecenie **SELECT** i dowiesz się, jak **uzyskać dostęp do tych wszystkich ważnych informacji**, które wcześniej zapisywałeś w swoich tabelach. Co więcej, dowiesz się także, jak stosować klauzulę **WHERE** i operatory **AND** i **OR**, by nie tylko pobierać dane, lecz wyświetlać te, które **śq** Ci potrzebne.

Pobierać dane czy nie pobierać?	88
Lepsza postać polecenia SELECT	91
Co oznacza gwiazdka (*)	92
Jak poszukiwać wartości różnych typów?	98
Kolejne problemy ze znakami przestankowymi	99
Niedopasowane apostrofy	100
Polecenia INSERT z danymi zawierającymi apostrofy	102
Pobieranie konkretnych kolumn w celu ograniczenia wyników	107
Określanie kolumn w celu zwiększenia szybkości zapytania	107
Łączenie zapytań	114
Odnajdywanie wartości liczbowych	117
Łagodne operatory porównania	120
Odnajdywanie wartości liczbowych przy użyciu operatorów porównania	122
Odnajdywanie danych tekstowych przy użyciu operatorów porównania	125
Być ALBO nie być	127
Różnica pomiędzy operatorem AND a OR	130
By odszukać NULL, użyj operatora IS NULL	133
Jak można zaoszczędzić czas dzięki jednemu słowu: LIKE	135
Zew wieloznaczności	135
Pobieranie zakresów przy użyciu operatora AND i operatorów porównania	139
Lepszy sposób — operator BETWEEN	140
Operator IN — w kręgu zainteresowania...	143
...lub poza nim — NOT IN	144
Więcej o operatorze NOT	145
Przybornik SQL	150



Jestem gwiazdą!

DELETE i UPDATE

3

Są szanse, że wszystko będzie w porządku

Cały czas zmieniasz zdanie? Teraz nie przysporzy Ci to najmniejszego problemu! Dzięki poleceniom **DELETE** i **UPDATE**, które poznasz w tym rozdziale, nie będziesz już dłużej musiał ponosić konsekwencji decyzji podjętych pół roku temu, kiedy to zapisałeś w bazie dane o spodniach w kształcie dzwonów, które właśnie z powrotem zaczynały być modne. Dzięki poleceniu **UPDATE** będziesz mógł **zmieniać dane**, natomiast polecenie **DELETE** pozwoli **usunąć z bazy dane**, które nie będą Ci już dłużej potrzebne. Jednak w tym rozdziale nie tylko pokażemy Ci te dwa nowe polecenia SQL, lecz także nauczymy, jak można używać ich w precyzyjny sposób, by przez przypadek nie usunąć danych, które cały czas są potrzebne.

Klowni są przerażający	154
Śledzenie kłownów	155
Klowni są w ciągłym ruchu	156
Jak są zapisywane informacje o kłownach?	160
Gonzo, mamy problem	162
Jak pozbyć się rekordu — polecenie DELETE	163
Stosowanie naszego nowego polecenia DELETE	165
Reguły polecenia DELETE	166
Dwa kroki — INSERT i DELETE	169
Stosuj polecenie DELETE rozważnie	174
Problemy z nieprecyzyjnymi poleceniami DELETE	178
Modyfikowanie danych przy użyciu polecenia UPDATE	180
Reguły stosowania polecenia UPDATE	181
UPDATE odpowiada kombinacji INSERT-DELETE	182
Polecenie UPDATE w akcji	183
Aktualizacja miejsc wystąpień kłownów	186
Zaktualizuj ceny drinków	188
A chcemy tylko jednego polecenia UPDATE	190
Przybornik SQL	192



Czy Cię
przerażamy?

Projektowanie dobrych tabel

4

Po co być normalnym?

Dotychczas tworzyłeś tabele bez zwracania na nie szczególnej uwagi.

I wszystko było w porządku, tabele działały bez problemów. Mogłeś w nich zapisywać, modyfikować, usuwać i pobierać dane. Jednak wraz ze zwiększaniem się ilości danych w tabelach zaczniesz zauważać, że są rzeczy, które mogłeś zrobić wcześniej, by ułatwić sobie w przyszłości tworzenie klauzul WHERE. Innymi słowy, musisz **znormalizować** swoje tabele.

Dwie wędkarskie tabele	194
Tabele dotyczą związków	198
Dane atomowe	202
Dane atomowe a Twoje tabele	204
Reguły danych atomowych	205
Dlaczego warto być normalnym?	208
Zalety normalizacji tabel	209
Klowni nie są normalni	210
W połowie drogi do 1NF	211
Reguły KLUCZA GŁÓWNEGO	212
Dążenie do pierwszej postaci NORMALNEJ	215
Poprawianie tabeli Grzeška	216
Oryginalna postać polecenia CREATE TABLE	217
Pokażcie mi moją kasę ^{tabele}	218
Polecenie oszczędzające czas	219
Tworzenie tabeli z KLUCZEM GŁÓWNYM	220
1, 2, 3... automatycznie inkrementowane	222
Dodawanie KLUCZA GŁÓWNEGO do istniejącej tabeli	226
Modyfikacja tabeli i dodanie KLUCZA GŁÓWNEGO	227
Przybornik SQL	228

Chwileczkę, ale ja już mam tabelę z całą masą informacji. Chyba nie jesteście poważni, jeśli oczekujecie, że usunę tabelę jak w pierwszym rozdziale i będę ponownie wpisywał całą jej zawartość tylko i wyłącznie po to, żeby stworzyć kolumnę klucza głównego.



Polecenie ALTER

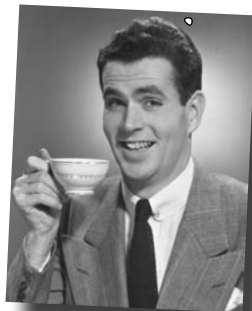
5

Korygowanie przeszłości

Czy kiedykolwiek marzyłeś, by mieć możliwość naprawiania błędów popełnionych w przeszłości? Cóż, teraz masz taką możliwość. Dzięki poleceniu ALTER możesz zastosować wszystkie nowe wiadomości i doświadczenia, by zmodyfikować strukturę tabel utworzonych kilka dni, miesięcy lub nawet lat wcześniej. Ale to jeszcze nie wszystko — możesz to zrobić bez wprowadzania jakichkolwiek zmian w istniejących danych. Skoro tu dotarłeś, zapewne już wiesz, co naprawdę oznacza **normalizacja**, i będziesz mógł zastosować tę wiedzę do zapewnienia odpowiedniej struktury swym tabelom; tym starym i nowym.

Odpicuj
moją
tabelę

Nadszedł czas,
by zmienić Twoją wypróbowaną,
lecz przestarzałą tabelę na cudętko,
jakiemu nie oprą się żadne informacje,
i przenieść ją na taki poziom
bazodanowego odpicowania, o którym
nawet Ci się nie śniło.



Musimy wprowadzić kilka zmian	232
Modyfikowanie tabel	237
Ekstremalne metamorfozy tabel	238
Zmiana nazwy tabeli	239
Musimy poczynić pewne plany	241
Przezbajanie kolumn	242
Zmiany strukturalne	243
Polecenia ALTER i CHANGE	244
Zmiana dwóch kolumn przy użyciu jednego polecenia SQL	245
Szybko! Usuń tę kolumnę	249
Dokładniejsza analiza nieatomowej kolumny lokalizacji	256
Poszukaj wzorca	257
Kilka wygodnych funkcji łańcuchowych	258
Użyj bieżącej kolumny do zapisania wartości w innej kolumnie	263
Sposób działania połączonych poleceń UPDATE i SET	264
Przybornik SQL	266

Zaawansowane zastosowanie polecenia SELECT

6

Nowy sposób spojrzenia na dane

Nadszedł czas, byś dodał do swojego SQL-owego przybornika nieco finezji.

Już wiesz, jak pobierać dane przy użyciu polecenia SELECT i klauzuli WHERE. Jednak czasami będziesz potrzebował nieco większej **precyzji** niż ta, jaką one zapewniają. W tym rozdziale dowiesz się o **określaniu kolejności oraz grupowaniu** danych, jak również o **wykonywaniu operacji matematycznych** na wynikach zapytań.

Reorganizacja Filmoteki Bazodanowa	268
Kilka problemów z bieżącą tabelą	269
Dopasowywanie istniejących danych	270
Określanie zawartości nowej kolumny	271
Polecenie UPDATE z wyrażeniem CASE	274
Wygląda na to, że mamy problem	276
Do tabel może się wkraść bałagan	281
Potrzebujemy możliwości organizowania danych zwracanych przez polecenie SELECT	282
Wypróbuj klauzulę ORDER BY	285
Sortowanie według jednej kolumny	286
Klauzula ORDER z dwoma kolumnami	289
Klauzula ORDER operująca na wielu kolumnach	290
Uporządkowana tabela filmów	291
Zmiana kolejności dzięki użyciu DESC	293
Problem najlepszej sprzedawczyni grupy Młode Gosposie	295
Funkcja SUM zsumuje wszystko za nas	297
Zsumuj wszystko za jednym razem dzięki użyciu GROUP BY	298
Funkcja AVG z klauzulą GROUP BY	299
Funkcje MIN i MAX	300
Liczmy dni	301
Pobieranie unikalnych wartości	303
LIMIT-owanie ilości wyników	306
Ograniczenie tylko do drugiego miejsca	307
Przyborek SQL	310

FILMOTEKA BAZODANOWA



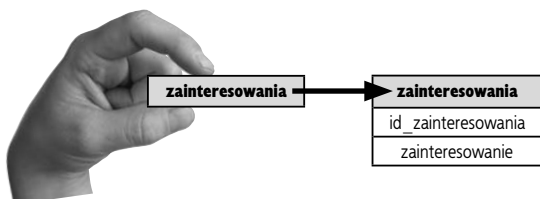
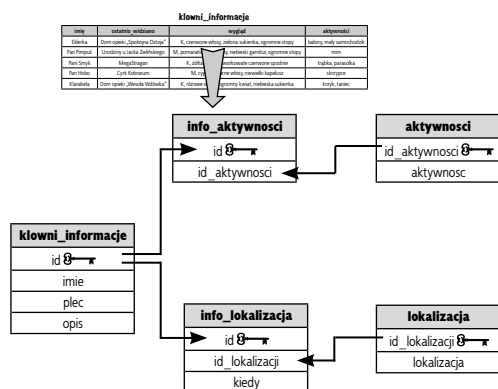
Projektowanie baz danych składających się z wielu tabel

7

Wyrastamy z naszych starych tabel

Kiedyś może nadejść moment, kiedy pojedyncza tabela przestanie Ci wystarczać. Twoje dane stały się bardziej złożone i jedna tabela, której do tej pory używałeś, nie jest już w stanie sprostać Twoim potrzebom. W tabeli jest bardzo dużo powtarzających się, nadmiarowych informacji, które niepotrzebnie zajmują miejsce na dysku i spowalniają zapytania. Wytrwałeś, używając tej jednej tabeli, tak długo, jak to było możliwe. Jednak świat, w którym żyjemy, jest ogromny i czasami potrzeba będzie **więcej niż jednej tabeli**, by zapisać wszystkie dane, zapanować nad nimi i w ostatecznym rozrachunku cały czas być panem własnej bazy danych.

Szukamy partnerki na randkę dla Wieśka	312
Wszystko stracone... ale zaraz	323
Wyobraź sobie dodatkową tabelę	324
Nowe tabele w bazie danych z informacjami o kłownach	325
Schemat bazy danych kłowni_informacje	326
W jaki sposób z jednej tabeli zrobić dwie	328
Łączenie tabel	330
Ograniczanie klucza obcego	335
Dlaczego należy zwracać sobie głowę kluczami obcymi?	336
TWORZENIE tabeli z KLUCZEM OBCYM	337
Zależności pomiędzy tabelami	339
Wzorce danych: jeden-do-jednego	339
Wzorce danych: kiedy używać tabel połączonych zależnością jeden-do-jednego	340
Wzorce danych: jeden-do-wielu	341
Wzorce danych: dochodzimy do zależności wiele-do-wielu	342
Wzorce danych: potrzebujemy tabeli łączącej	345
Wzorce danych: wiele-do-wielu	346
W końcu w 1NF	351
Klucz złożony korzysta z wielu kolumn	352
Zapis uproszczony	354
Częściowa zależność funkcjonalna	355
Przechoźnia zależność funkcjonalna	356
Druga postać normalna	360
Trzecia postać normalna (w końcu)	366
I tak oto Robert (i lista_grzesia) żyli od tej pory długo i szczęśliwie	369
Przybornik SQL	372



Złączenia i operacje na wielu tabelach

8

Czy nie możemy się wszyscy dogadać?

Witam w skomplikowanym świecie wielu tabel. Wspaniale jest mieć w bazie **więcej niż jedną tabelę**, jednak abyś mógł się nimi posługiwać, będziesz musiał poznać **nowe narzędzia i techniki**. Jednoczesne korzystanie z wielu tabel może być kłopotliwe, dlatego też, by wszystko było jasne, będziesz potrzebował nazw zastępczych (nazywanych także „aliasami”). Oprócz tego **złączenia** pomogą Ci połączyć tabele, dzięki czemu będziesz mógł korzystać ze wszystkich danych, zapisanych w wielu różnych miejscach. Przygotuj się — teraz ponownie **odzyskasz pełną kontrolę nad swoją bazą danych**.

Powtarzamy się, cały czas się powtarzamy...	374
Wypełnianie tabel	375
Zainteresowania — kłopotliwa kolumna	378
Wciąż wykazujemy zainteresowanie	379
Aktualizacja wszystkich zainteresowań	380
Pobieranie wszystkich zainteresowań	381
Wiele dróg prowadzących w to samo miejsce	382
CREATE, SELECT oraz INSERT — (prawie) jednocześnie	382
CREATE, SELECT i INSERT — jednocześnie	383
O co chodzi z tym AS?	384
Nazwy zastępcze kolumn	385
Nazwy zastępcze, a kto by ich potrzebował?	386
Wszystko co chciałbyś wiedzieć o złączeniach wewnętrznych	387
Złączenie kartezjańskie	388
Zrozumienie złączeń wewnętrznych	393
Złączenie wewnętrzne w akcji: złączenie równościowe	394
Złączenia wewnętrzne w akcji: złączenie różnościowe	397
Ostatni rodzaj złączeń wewnętrznych: złączenia naturalne	398
Złączone zapytania?	405
Nazwy zastępcze tabel i kolumn bez tajemnic: Dlaczego się ukrywacie?	406
Przybornik SQL	408

... i tak naprawdę to właśnie stąd się biorą małe tablice z wynikami.



Podzapytania

9

Zapytania w zapytaniach

Janie, dwuczęściowe zapytanie, proszę. Złączenia są wspaniałe, jednak czasami musisz *zadać swojej bazie danych więcej niż jedno zapytanie*. Bądź *użyć wyników jednego zapytania jako danych wejściowych dla drugiego*. I właśnie w takich sytuacjach przydają się **podzapytania**. Pozwolą Ci one **uniknąć powielania danych**, sprawią, że Twoje **zapytania staną się bardziej dynamiczne**, a może nawet pozwolą się wkręcać na koncerty i przyjęcia dla wyższych sfer. (No, na to ostatnie bym nie liczyła, ale dwa punkty z trzech to i tak nieźle).

Grzesiek wchodzi na rynek pracy	410
Baza Grzeška wzbogacona o nowe tabele	411
Grzesiek używa złączenia wewnętrznego	412
Ale Grzesiek chce użyć innych zapytań	414
Podzapytania	416
Łączymy dwa zapytania w zapytanie z podzapytaniem	417
Jakby jedno pytanie nie wystarczało: poznajcie podzapytanie	418
Podzapytanie w działaniu	419
Podzapytania i ich reguły	421
Podstawowe informacje o tworzeniu podzapytań	424
Podzapytanie jako kolumna polecenia SELECT	427
Inny przykład: Podzapytanie ze złączeniem naturalnym	428
Podzapytania nieskorelowane	429
SQL bez tajemnic: Wybór optymalnego sposobu realizacji zapytania, jeśli dostępna jest większa ilość możliwości	430
Nieskorelowane podzapytania zwracające wiele wartości: IN oraz NOT IN	433
Podzapytania skorelowane	438
(Przydatne) Podzapytanie skorelowane używające operatora NOT EXISTS	439
EXISTS i NOT EXISTS	440
Usługi pośrednictwa pracy Grzeška — zaproszenie do współpracy!	442
W drodze na imprezę...	443
Przybornik SQL	444

zapytanie ZEWNĘTRZNE

zapytanie WEWNĘTRZNE

Zapytanie zewnętrzne.

```
SELECT jakas_kolumna, inna_kolumna
FROM tabela
WHERE kolumna = (SELECT kolumna FROM tabela );
```

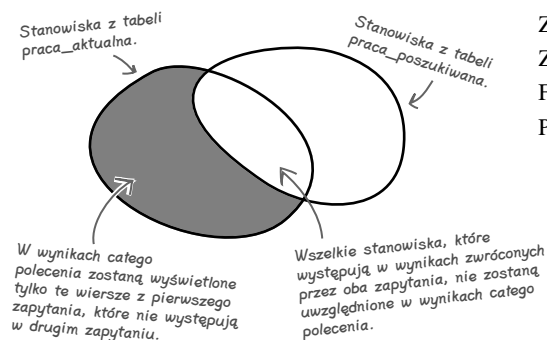
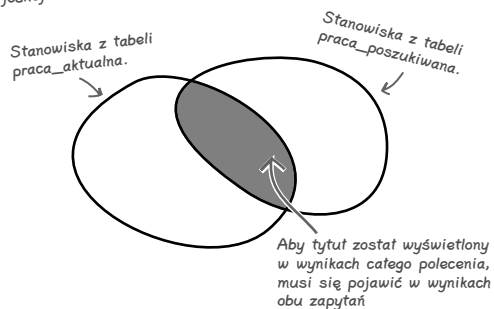
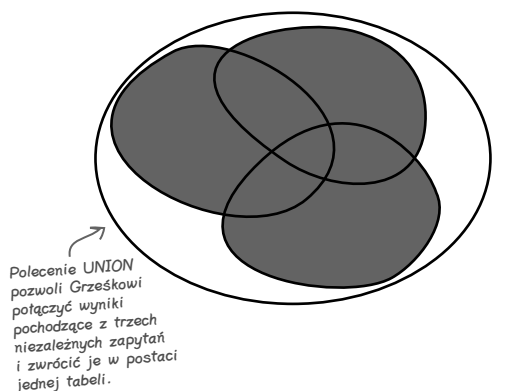
Zapytanie wewnętrzne.

Złączenia zewnętrzne, złączenia zwrotne oraz unie

10

Nowe manewry

To wszystko, czego do tej pory dowiedziałeś się o złączeniach, to jedynie pół prawdy na ich temat. Widziałeś już złączenia krzyżowe tworzące wszystkie możliwe pary rekordów pochodzących z dwóch tabel oraz złączenia wewnętrzne zwracające jedynie pasujące do siebie rekordy obu tabel. Nie spotkałeś się jeszcze natomiast ze **złączeniami zewnętrznymi**, zwracającymi także te wiersze, które *nie mają pasujących odpowiedników w drugiej tabeli*, **złączeniami zwrotnymi**, które łączą *tabelę z nią samą*, oraz **uniami**, które *scalają wyniki z kilku różnych zapytań*. Kiedy poznasz te wszystkie sztuczki, będziesz w stanie pobierać dane z bazy dokładnie w taki sposób, jaki będzie Ci potrzebny. (Nie zapomnimy także ujawnić całej szokującej prawdy o podzapytaniach!).



Porządki w starych danych	448
Kluczem są dwie strony złączenia — lewa i prawa	449
Oto lewostronne złączenie wewnętrzne	450
Złączenia zewnętrzne i wielokrotne dopasowania	455
Prawostronne złączenie zewnętrzne	456
Podczas gdy my radośnie złączaliśmy zewnętrznie...	459
Moglibyśmy utworzyć nową tabelę	460
Gdzie w schemacie umieścimy nową tabelę?	461
Klucz obcy odwołujący się do tej samej tabeli	462
Łączenie tabeli z nią samą	463
Potrzebujemy złączenia zwrotnego	465
Inny sposób zwracania informacji z wielu tabel	466
Można zastosować polecenie UNION	467
Polecenie UNION ma swoje ograniczenia	468
Reguły stosowania poleceń UNION w działaniu	469
UNION ALL	470
Utworzenie tabeli na podstawie wyników polecenia UNION	471
Polecenia INTERSECT i EXCEPT	472
Skończyliśmy ze złączeniami, czas zająć się czymś nowym	473
Podzapytania i złączenia — studium porównawcze	473
Zamiana podzapytania na złączenie	474
Złączenie zwrotne jako podzapytanie	479
Firma Grześka rozwija się	480
Przybornik SQL	482

Ograniczenia, widoki i transakcje

11

Zbyt wielu kucharzy psuje bazę danych

Twoje bazy danych rozrosły się i muszą z nich korzystać także inne osoby.

Problem polega na tym, że niektórzy z nich nie będą znały języka SQL równie dobrze jak Ty. Musisz zatem mieć jakieś sposoby, by **uniemożliwić im wprowadzanie nieprawidłowych danych**, techniki **pozwalające ukryć przed nimi wybrane dane** oraz mechanizmy, które **uchronią te osoby przed wzajemnym przeszkadzaniem sobie, gdy jednocześnie będą modyfikować zawartość bazy.**

W tym rozdziale zaczniemy ochraniać nasze dane przed błędami popełnianymi przez inne osoby.

Witamy w Bazie Obronnej, w części 1.

Grzesiek zatrudnił pomocników	484
Pierwszy dzień Kuby: Dopisywanie nowego klienta	485
Kuba unika wartości NULL	486
Trzy miesiące później	487
Uwaga, KONTROLA: dodawanie OGRANICZEŃ SPRAWDZAJĄCYCH	488
Sprawdzanie płci	489
Praca Franka staje się nużąca	491
Tworzenie widoku	493
Oglądanie własnych widoków	494
Jak właściwie działa widok?	495
Czym są widoki	496
Wstawianie, aktualizacja i usuwanie danych przy wykorzystaniu widoków	499
Sekret polega na tym, by udawać, że widok jest prawdziwą tabelą	500
Widoki z klauzulą CHECK OPTION	503
Twój widok może pozwalać na aktualizację danych, jeśli...	504
Kiedy widok przestanie być potrzebny	505
Kiedy dobrej bazie przydarzy się coś złego	506
Co się stało w bankomacie	507
Kolejne kłopoty z bankomatami	508
To nie marzenia, to transakcje	510
Klasyczny test ACID	511
SQL pomaga nam zarządzać swoimi transakcjami	512
Co powinno się stać w bankomacie	513
Jak umożliwić korzystanie z transakcji w MySQL-u	514
Wypróbuj transakcje samodzielnie	515
Przybornik SQL	518



Bezpieczeństwo

12

Zabezpieczanie swych dóbr

Włożyłeś niezwykle dużo wysiłku i czasu w utworzenie swojej bazy danych.

I na pewno byłbyś zdruzgotany, gdyby przydarzyło się jej coś złego. Jednak musisz zapewnić **dostęp do swoich danych** innym osobom i obawiasz się, że mogłyby zapisać w bazie niewłaściwe dane lub, co gorsza, **usunąć nie te dane, które powinny**. Na szczęście w tym rozdziale dowiesz się, w jaki sposób można dodatkowo **zabezpieczyć** bazę danych oraz obiekty w niej umieszczone oraz w jaki sposób uzyskać pełną kontrolę nad tym, **kto i jakie operacje na bazie może wykonywać**.

Problemy użytkowników	522
Zapobieganie błędom w bazie kłownów	523
Zabezpieczanie konta administratora	525
Dodanie nowego użytkownika	526
Dokładnie określ, czego poszczególni użytkownicy potrzebują	527
Prosta postać polecenia GRANT	528
Różne wersje polecenia GRANT	531
Usuwanie uprawnień	532
Usuwanie uprawnień przydzielonych dzięki GRANT OPTION	533
Precyzyjne usuwanie	534
Współużytkowane konta przysparzają problemów	538
Stosowanie roli	540
Usuwanie roli	540
Stosowanie roli z klauzulą WITH ADMIN OPTION	542
Łączenie poleceń CREATE USER i GRANT	547
Lista Grześka stała się usługą globalną!	548
Czy już coś wiesz o filii Listy Grzesia w swoim mieście?	549
Korzystaj z SQL-a w swoich własnych projektach, ponieważ także Ty możesz odnieść taki sukces jak Grzesiek	549
Przybornik SQL	550



Root



Ważniak



Doktorek



Głupek



Pyskacz



Szczęściarz



Śpioch



Ziewacz

Pozostałości

A

Część najważniejszych zagadnień
(których nie opisaliśmy wcześniej)

Nawet po tym wszystkim jest jeszcze coś więcej. Jest jeszcze dosłownie kilka rzeczy, o których, jak sądzimy, powinieneś wiedzieć. Nie czuliśmy się w porządku, gdybyśmy je całkowicie zignorowali i nie poświęcili im choćby krótkiej wzmianki. A zatem: nim będziesz mógł odłożyć tę książkę na półkę, przeczytaj o tych **drobnych, lecz ważnych sprawach**.

Poza tym kiedy skończysz lekturę tego dodatku, pozostaną Ci jeszcze dwa następne... oraz indeks... no i może jeszcze jakieś reklamy... a potem już naprawdę skończysz. Obiecujemy!

Nr 1.	Znajdź i zainstaluj graficzny program do obsługi używanego systemu zarządzania bazami danych	552
Nr 2.	Słowa zastrzeżone i znaki specjalne	554
Nr 3.	ALL, ANY oraz SOME	556
Nr 4.	Dodatkowe informacje o typach danych	558
Nr 5.	Tabele tymczasowe	560
Nr 6.	Rzutowanie typów	561
Nr 7.	Kim jesteś? Która jest godzina?	562
Nr 8.	Przydatne funkcje matematyczne	563
Nr 9.	Indeksowanie dla poprawy szybkości działania zapytań	565
Nr 10.	Dwu-minutowy kurs PHP i MySQL-a	566

A	ABSOLUTE ACTION ADD ADMIN AFTER AGGREGATE ALIAS ALL ALLOCATE ALTER AND ANY ARE ARRAY AS ASC ASSERTION AT AUTHORIZATION
B	BEFORE BEGIN BINARY BIT BLOB BOOLEAN BOTH BREADTH BY
C	CALL CASCADE CASCADED CASE CAST CATALOG CHAR CHARACTER CHECK CLASS CLOB CLOSE COLLATE COLLATION COLUMN COMMIT COMPLETION CONNECT CONNECTION CONSTRAINT CONSTRAINTS CONSTRUCTOR CONTINUE CORRESPONDING CREATE CROSS CUBE CURRENT CURRENT_DATE CURRENT_PATH CURRENT_ROLE CURRENT_TIME CURRENT_TIMESTAMP CURRENT_USER CURSOR CYCLE
D	DATA DATE DAY DEALLOCATE DEC DECIMAL DECLARE DEFAULT DEFERRABLE DEFERRED DELETE DEPTH DEREF DESC DESCRIBE DESCRIPTOR DESTROY DESTRUCTOR DETERMINISTIC DICTIONARY DIAGNOSTICS DISCONNECT DISTINCT DOMAIN DOUBLE DROP DYNAMIC
E	EACH ELSE END END_EXEC EQUALS ESCAPE EVERY EXCEPT EXCEPTION EXEC EXECUTE EXTERNAL
F	FALSE FETCH FIRST FLOAT FOR FOREIGN FOUND FROM FREE FULL FUNCTION
G	GENERAL GET GLOBAL GO GOTO GRANT GROUP GROUPING
H	HAVING HOST HOUR
I	IDENTITY IGNORE IMMEDIATE IN INDICATOR INITIALIZE INITIALLY INNER INOUT INPUT INSERT INT INTEGER INTERSECT INTERVAL INTO IS ISOLATION ITERATE
J	JOIN
K	KEY
L	LANGUAGE LARGE LAST LATERAL LEADING LEFT LESS LEVEL LIKE LIMIT LOCAL LOCALTIME LOCALTIMESTAMP LOCATOR
M	MAP MATCH MINUTE MODIFIES MODIFY MODULE MONTH
N	NAMES NATIONAL NATURAL NCHAR NCLOB NEW NEXT NO NONE NOT NULL NUMERIC
O	OBJECT OF OFF OLD ON ONLY OPEN OPERATION OPTION OR ORDER ORDINALITY OUT OUTER OUTPUT
P	PAD PARAMETER PARAMETERS PARTIAL PATH POSTFIX PRECISION PREFIX PREORDER PREPARE PRESERVE PRIMARY PRIOR PRIVILEGES PROCEDURE PUBLIC
Q	
R	READ READS REAL RECURSIVE REF REFERENCES REFERENCING RELATIVE RESTRICT RESULT RETURN RETURNS REVOKE RIGHT ROLE ROLLBACK ROLLUP ROUTINE ROW ROWS
S	SAVEPOINT SCHEMA SCROLL SCOPE SEARCH SECOND SECTION SELECT SEQUENCE SESSION SESSION_USER SET SETS SIZE SMALLINT SOME SPACE SPECIFIC SPECIFICTYPE SQL SOLEXCEPTION SQLSTATE SQLWARNING START STATE STATEMENT STATIC STRUCTURE SYSTEM USER
T	TABLE TEMPORARY TERMINATE THAN THEN TIME TIMESTAMP TIMEZONE_HOUR TIMEZONE_MINUTE TO TRAILING TRANSACTION TRANSLATION TREAT TRIGGER TRUE
U	UNDER UNION UNIQUE UNKNOWN UNNEST UPDATE USAGE USING
V	VALUE VALUES VARCHAR VARIABLE VARYING VIEW
W	WHEN WHENEVER WHERE WITH WITHOUT WORK WRITE
X	
Y	YEAR
Z	ZONE

```
C:\Program Files\MySQL\MySQL Server 5.0\bin>mysql.exe
mysql> SELECT CURRENT_DATE;
+-----+
| CURRENT_DATE |
+-----+
| 2008-03-09   |
+-----+
1 row in set (0.00 sec)

mysql>
```

```
C:\Program Files\MySQL\MySQL Server 5.0\bin>mysql.exe
mysql> SELECT CURRENT_TIME;
+-----+
| CURRENT_TIME |
+-----+
| 00:55:07     |
+-----+
1 row in set (0.00 sec)

mysql>
```

```
C:\Program Files\MySQL\MySQL Server 5.0\bin>mysql.exe
mysql> SELECT CURRENT_USER;
+-----+
| CURRENT_USER |
+-----+
| root@localhost |
+-----+
1 row in set (0.00 sec)

mysql> _
```


Instalacja MySQL-a

Spróbuj to zrobić samemu

Cała zdobyta przez Ciebie wiedza i umiejętności nie na wiele się zdadzą, jeśli nie wykorzystasz ich w praktyce. W tym dodatku znajdziesz instrukcje dotyczące instalacji swojego własnego serwera MySQL, którego będziesz mógł używać do zabawy i pracy.

B



- Zacznij działać i to szybko! 570
- Instrukcje i rozwiązywanie problemów 570
- Proces instalacji MySQL-a w systemie Windows 571

Przypomnienie narzędzi

Wszystkie nowe narzędzia SQL

W tym dodatku, w jednym jedynym miejscu, zostały po raz pierwszy zebrane wszystkie podpowiedzi dotyczące SQL-a... ale będą tu tylko przez jedną noc (żartujemy)! Znajdziesz tu podsumowanie wszystkich porad i informacji o SQL-u, jakie zamieściliśmy w całej książce. Poświęć chwilę, by *przejrzeć listę, i rozkoszuj się poczuciem dumy* — już znasz wszystkie te rady i informacje.

C



- Symbole 576

S

- Skorowidz 583

4. Projektowanie dobrych tabel

Po co być normalnym?



Dotychczas tworzyłeś tabele bez zwracania na nie szczególnej uwagi.

I wszystko było w porządku, tabele działały bez problemów. Mogłeś w nich zapisywać, modyfikować, usuwać i pobierać dane. Jednak w miarę zwiększania się ilości danych w tabelach zaczniesz zauważać, że są rzeczy, które mogłeś zrobić wcześniej, by ułatwić sobie w przyszłości tworzenie klauzul WHERE. Innymi słowy, musisz *znormalizować* swoje tabele.

Dwie wędkarskie tabele

Dwóch znajomych wędkarzy, Jacek i Marek, stworzyło tabele do gromadzenia danych o rekordowych połowach. Tabela Marka zawiera kolumny pozwalające na zapisanie łacińskiej nazwy gatunku ryby, nazwy polskiej, wagi złowionej ryby oraz miejsca dokonania połowu. Nie zawiera jednak kolumn pozwalających na zapisanie imienia i nazwiska osoby, która ustanowiła rekord.

połowu_informacje

nazwa	nazwa_gatunkowa	miejsce	waga
bass	M. salmoides	Wigry, PD	1,23 kg
sandacz	S. vitreus	Dziubiele, WM	2,75 kg
pstrąg	O. Clarki	Mrzyglód, PK	1,20 kg
okoń	P. Flavescens	Pisz, WM	0,85 kg
plotka	R. rutilus	Charzykowy, PM	0,65 kg
łuskot	L. Osseus	Czaplinek, ZP	1,10 kg
węgorz	A. anguilla	Swornegacie, PM	1,45 kg
szczupak	E. americanus	Karwica, WM	3,34 kg
złota rybka	C. auratus	Warszawa, MZ	0,35 kg
łosoś	O. Tshawytscha	Toruń, KP	3,10 kg

Ta tabela ma jedynie cztery kolumny. Porównaj ją z tabelą rekordowe_połowu przedstawioną na następnej stronie.

Jestem ichtiologiem. Chcę szukać w tabeli wyłącznie łacińskich nazw gatunkowych i nazw używanych potocznie, by określić wagę rekordowej ryby oraz gdzie ją złowiono.



Marek

Tabela Jacka także zawiera polską nazwę złowionej ryby oraz jej wagę; jednak oprócz tego Jacek umieścił w niej kolumny pozwalające na zapisanie imienia i nazwiska szczęśliwego wędkarza oraz nazwy województwa, w którym dokonano połowu.

Także ta tabela służy do rejestrowania rekordowych połowów wędkarskich, jednak zawiera niemal dwukrotnie więcej kolumn.

rekordowe_połowu

imie	nazwisko	nazwa	miejsce	województwo	waga	data
Jan	Kowalski	bass	Wigry	PD	1,23 kg	5.9.1947
Adrian	Bródka	sandacz	Dziubiele	WM	2,75 kg	16.8.1960
Zenon	Krawczyk	pstrąg	Mrzygłód	PK	1,20 kg	23.6.1978
Maria	Popieła	okoń	Pisz	WM	0,85 kg	18.5.1934
Piotr	Drymza	płatka	Charzykowy	PM	0,65 kg	1.8.1965
Ignacy	Wikorczyk	łuskot	Czaplinek	ZP	1,10 kg	31.9.1988
Krzysztof	Dubała	węgorz	Swornegacie	PM	1,45 kg	12.8.1973
Paweł	Wronek	szczupak	Karwica	WM	3,34 kg	11.6.1995
Andrzej	Książewicz	złota rybka	Warszawa	MZ	0,35 kg	25.9.2003
Roman	Wirotek	łosoś	Toruń	KP	3,10 kg	17.8.1991

Zaostrz ołówki



Dla **obu tabel** napisz zapytanie, które pobierze wszystkie rekordowe połowy dokonane w województwie podkarpackim.

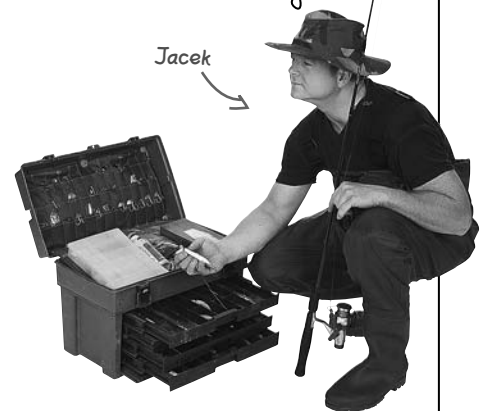
.....

.....

.....

.....

Piszę artykuły dla magazynu „Weekend z wędką”. Muszę znać imiona i nazwiska wędkarzy, którzy ustanowili rekordowe połowy, daty tych połowów oraz ich miejsca.



Jacek

Zaostrz ołówek
Rozwiązanie

Dla każdej z tabel napisz zapytanie, które pobierze informacje o rekordowych połowach dokonanych na terenie województwa podkarpackiego.

Niemal nigdy nie muszę wyszukiwać informacji na podstawie województwa. Dlatego informacje o województwie zapisuję w tabeli w tej samej kolumnie, w której umieszczam nazwę miejsca, gdzie dokonano rekordowego połowu.

Musimy zastosować operator *LIKE* i odszukać interesujące nas rekordy na podstawie pola zawierającego połączoną nazwę miejscowości i oznaczenie województwa.

SELECT * FROM połowy_informacje

WHERE miejsce LIKE '%PK';



nazwa	nazwa_gatunkowa	miejsce	waga
pstrąg	O. Clarki	Mrzygłód, PK	1,20 kg

Często muszę przeszukiwać informacje na podstawie województwa, dlatego też utworzyłem w tabeli odrębną kolumnę określającą województwo, w którym dokonano rekordowego połowu.

To zapytanie odwołuje się bezpośrednio do kolumny „województwo”.

SELECT * FROM rekordowe_połowu

WHERE województwo = 'PK';



imie	nazwisko	nazwa	miejsce	województwo	waga	data
Zenon	Krawczyk	pstrąg	Mrzygłód	PK	1,20 kg	23.6.1978

Nie istnieją
grupie pytania

P: A zatem tabela Jacka jest lepsza od tabeli Marka?

U: Nie. To dwie różne tabeli, stworzone w innych celach. W praktyce Marek rzadko kiedy będzie musiał poszukiwać w swojej tabeli informacji na podstawie województwa, gdyż tak naprawdę interesują go jedynie nazwy gatunkowe i potoczne złowionych ryb oraz, oczywiście, ich waga.

Z drugiej strony, Jacek *będzie* musiał korzystać z informacji o województwie podczas poszukiwania danych w swojej tabeli. To właśnie z tego powodu w jego tabeli informacje o województwie, w jakim dokonano rekordowych połowów, znalazły się w osobnej kolumnie. To mu ułatwi poszukiwanie danych na podstawie województwa.

P: Czy podczas przeszukiwania tabel powinniśmy unikać stosowania operatora LIKE?

U: Nic nie przemawia za tym, by nie stosować operatora LIKE, niemniej jednak może to przysporzyć pewnych problemów oraz prowadzić do otrzymywania niepożądanych wyników. Jeśli w kolumnach są zapisywane złożone informacje, to operator LIKE nie będzie dostatecznie precyzyjny, by je pobierać.

P: Dlaczego krótsze zapytania są lepsze do długich?

U: Im prostsze jest zapytanie, tym lepiej. Zapytania będą się stawać coraz bardziej skomplikowane wraz z powiększaniem się bazy danych i ilości umieszczonych w niej tabel. Jeśli na samym początku zaczniesz od możliwie jak najprostszyc zapytań, to w przyszłości na pewno tego nie pożałujesz.

P: Chcesz przez to powiedzieć, że zawsze powinienem przechowywać w kolumnach bardzo małe fragmenty informacji?

U: Niekoniecznie. Analizując tabeli Jacka i Marka, mogeś już zauważyć, że wszystko zależy od tego, w jaki sposób chcesz *korzystać* z tabel. Na przykład wyobraź sobie tabeli dotyczące samochodów — jedną, z której korzysta warsztat mechaniczny, i drugą, używaną przez właściciela komis samochodowego. Mechanicy mogą potrzebować szczegółowych informacji o każdym samochodzie; z kolei właścicielowi komis wystarczy marka, model, rok produkcji oraz numer nadwozia.

P: A wyobraźmy sobie adres pocztowy. Czy nie moglibyśmy utworzyć jednej kolumny, w której byłby zapisany cały adres, oraz kilku innych, w których umieścilibyśmy jego poszczególne elementy?

U: Choć takie powielanie danych może Ci się obecnie wydawać całkiem dobrym pomysłem, to jednak pomyśl, o ile więcej miejsca na dysku zajmie taka baza, gdy rozrośnie się do potężnych rozmiarów. Poza tym w przypadku powielania danych w poleceniach INSERT i UPDATE pojawią się dodatkowe kolumny, o których będziesz musiał pamiętać.

Przyjrzyjmy się dokładniej, jak należy projektować tabeli, by optymalnie pasowały do sposobów, w jakie będziemy z nich korzystać.

Sposób, w jaki masz zamiar korzystać z danych, będzie determinował postać tworzonych tabeli.



WYSIL SZARE KOMÓRKI

SQL jest językiem używanym w relacyjnych bazach danych.
Jak uważasz, czym są te „relacje” w świecie baz danych?

Tabele dotyczą związków

Bazy danych obsługiwane przy użyciu języka SQL są nazywane systemami zarządzania relacyjnymi bazami danych (ang. Relational Database Management System, w skrócie RDBMS). Ale nie zaprzątaj sobie głowy zapamiętywaniem tych nazw. Dla nas najważniejsze jest tylko jedno słowo: RELACYJNE*. Dla Ciebie oznacza ono mniej więcej tyle, że aby zaprojektować odnotowaną tabelę, musisz zastanowić się i określić, **jak poszczególne kolumny są ze sobą powiązane** i wspólnie **opisują zagadnienie**, jakiego dotyczy tabela.

Całe wyzwanie i sztuka polega na tym, by opisać zagadnienie, wykorzystując do tego celu kolumny, które zagwarantują łatwość pobierania danych z tabeli. A to oczywiście zależy od tego, jakie informacje chcemy pobierać z tabeli. Można wyróżnić kilka bardzo ogólnych kroków, jakie należy wykonać, projektując tabele.

1. Wybierz jedno zagadnienie, które ma opisywać tabela.

Czego mają dotyczyć informacje zapisywane w tabeli?

2. Utwórz listę informacji o danym zagadnieniu, których będziesz potrzebował podczas korzystania z tabeli.

Jak będziesz korzystał z tabeli?

3. Na podstawie tej listy podziel informacje o zagadnieniu na elementy, których następnie będziesz mógł użyć, określając organizację tabeli.

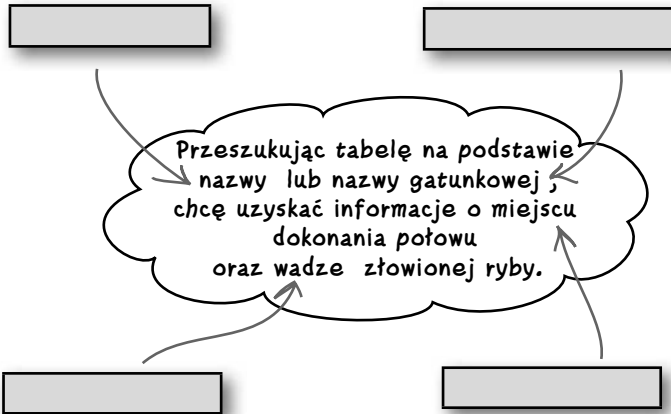
Jak będzie Ci najłatwiej przeszukiwać i pobierać dane z tabeli?

* Niektórzy uważają, że słowo „relacyjny” odnosi się do wielu tabel powiązanych ze sobą. Jednak jest to błędna opinia.

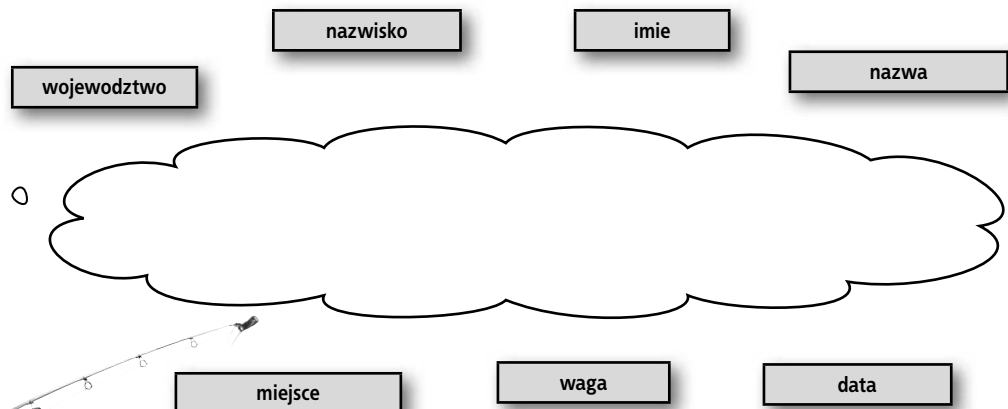


Ćwiczenie

Czy na podstawie przedstawionego poniżej zdania, określającego, w jaki sposób ichtiolog Marek chce przeszukiwać i pobierać dane z tabeli, potrafisz określić, jakie powinny być jej kolumny? Wpisz nazwy kolumn w pustych prostokątach na poniższym rysunku.



Twoja kolej. Napisz analogiczne zdanie dotyczące Jacka — autora artykułów dla magazynu „Weekend z wędką”, który korzysta z bazy, by notować w niej szczegółowe informacje na potrzeby swoich artykułów. Następnie narysuj strzałki prowadzące od nazw kolumn do miejsca w zdaniu, w którym Jacek nawiązuje do danej kolumny.

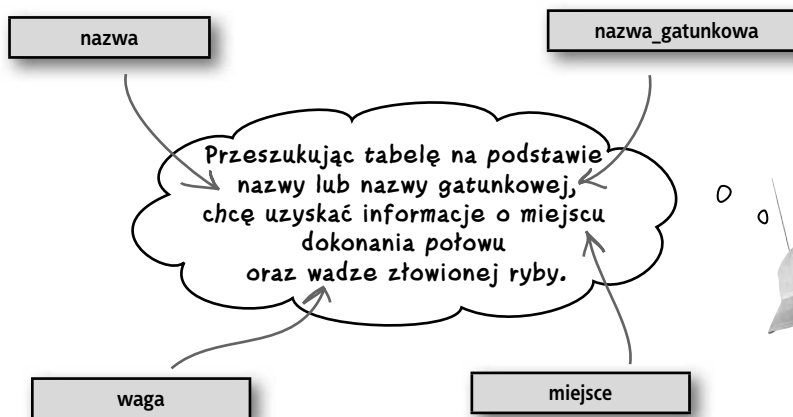


Ćwiczenie. Rozwiązanie

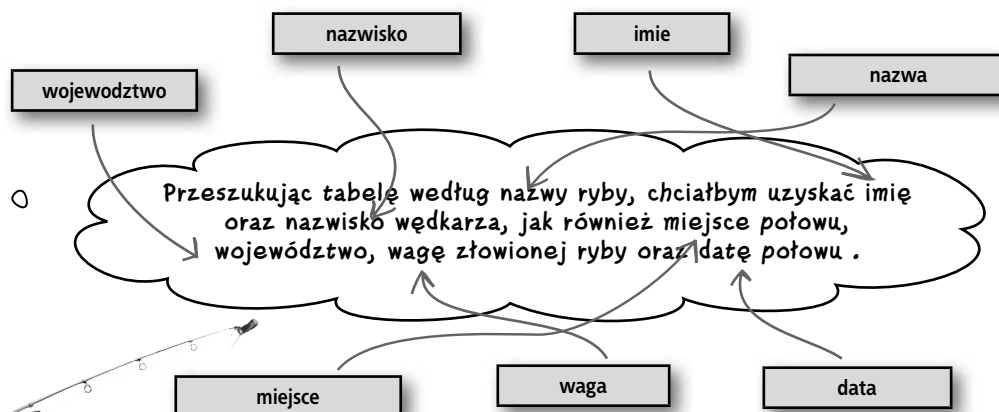


Rozwiązanie ćwiczenia

Czy na podstawie przedstawionego poniżej zdania określającego, w jaki sposób ichtiolog Marek chce przeszukiwać i pobierać dane z tabeli, potrafisz określić, jakie powinny być jej kolumny? Wpisz nazwy kolumn w pustych prostokątach na poniższym rysunku.



Twoja kolej. Napisz analogiczne zdanie dotyczące Jacka — autora artykułów dla magazynu „Weekend z wędką”, który korzysta z bazy, by notować w niej szczegółowe informacje na potrzeby swoich artykułów. Następnie narysuj strzałki prowadzące od nazw kolumn do miejsca w zdaniu, w którym Jacek nawiązuje do danej kolumny.



Ale dlaczego Jacek ma poprzestawać na tym? Dlaczego nie podzielił daty na kolumny dni, miesięcy i lat? Nawet kolumnę określającą miejsce można by dalej podzielić na osobne kolumny z nazwą ulicy i numerem domu.



Oczywiście, że można by tak zrobić. Jednak nasze dane nie muszą być podzielone aż tak dokładnie.

Przynajmniej nie w tym przypadku. Gdyby jednak Jacek miał zamiar napisać artykuł o tym, gdzie pojechać na wakacje, by złapać dużą rybę, to w *takim przypadku* mógłby podzielić kolumnę „miejsce” na nazwę ulicy i numer, tak by czytelnicy mogli znaleźć nocleg jak najbliżej rekordowego łowiska.

Jednak Jacek potrzebuje wyłącznie informacji o miejscu i województwie, dlatego utworzył tylko tyle kolumn, ile jest koniecznych, by niepotrzebnie nie powiększać rozmiaru bazy danych. Uznał, że w jego sytuacji nie ma sensu bardziej dzielić danych; innymi słowy, uznał, że jego informacje są danymi *atomowymi*.



WYSIL SZARE KOMÓRKI

Jak myślisz, co oznacza termin „dane atomowe” w kontekście informacji zapisywanych w relacyjnych bazach danych?

Dane atomowe

Czym jest atom? To niewielki fragment materii, którego nie można lub nie należy dalej dzielić. To samo dotyczy danych. Kiedy zostaną one uznane za dane **ATOMOWE**, oznacza to, że zostały one już podzielone na **najmniejsze elementy**, których nie należy dalej dzielić.

Dostawa w 30 minut lub gratis

Przyjrzyjmy się na przykład dostarczycielowi pizzy. Aby dostarczyć zamówienie w odpowiednie miejsce, wystarczy, że w jednej kolumnie zapiszemy nazwę ulicy i numer domu. Na jego potrzeby są to dane atomowe. Dostawca nigdy nie będzie poszukiwał samego numeru domu.

W rzeczywistości, jeśli miejsce dostawy zostałyby podzielone na nazwę ulicy i numer domu, to zapytania, które musiałby zadawać dostawca, byłyby dłuższe i bardziej złożone, a to spowodowałoby wydłużenie czasu dostarczania pizzy do klienta.



Na potrzeby dostawcy pizzy adres zapisany w jednej kolumnie jest wystarczająco atomową informacją.

```
C:\Program Files\MySQL\MySQL Server 5.0\bin\mysql> mysql>
+-----+-----+
| numer_zamowienia | adres                |
+-----+-----+
| 250               | Mickiewicza 46      |
| 251               | Rozanskiego 15/2    |
| 253               | Podlesnicka 23      |
| 254               | Aleje Zwyciestwa 28/5 |
| 255               | Szeroka 16          |
| 256               | Targowa 87/9        |
| 257               | Zolkiewskiego 15    |
| 258               | Wilcza 26/2         |
| 259               | Skowronkow 2/2      |
+-----+-----+
9 rows in set (0.00 sec)

mysql> SELECT adres FROM pizze_nawynos WHERE numer_zamowienia = 253;
+-----+
| adres                |
+-----+
| Podlesnicka 23      |
+-----+
1 row in set (0.01 sec)

mysql>
```

Lokalizacja, lokalizacja, lokalizacja

A teraz przeanalizujmy przykład pośrednika handlu nieruchomościami. Taki pośrednik mógłby chcieć, by nazwa ulicy została zapisana w osobnej kolumnie. Mógłby bowiem chcieć podać w zapytaniu nazwę ulicy, by uzyskać informacje o wszystkich domach na sprzedaż położonych przy niej. A zatem w przypadku pośrednika handlu nieruchomościami informacjami atomowymi są nazwa ulicy oraz numer domu.

Z kolei w przypadku pośrednika handlu nieruchomościami rozdzielenie nazwy ulicy od numeru domu i zapisanie ich w osobnych kolumnach pozwoli mu odnajdywać wszystkie domy na sprzedaż położone przy konkretnej ulicy za pomocą jednego, prostego zapytania.



```
C:\Program Files\MySQL\MySQL Server 5.0\bin\mysql.exe
+-----+-----+-----+-----+
| ulica_numer | ulica_nazwa | typ_lokalu | cena |
+-----+-----+-----+-----+
| 23          | Al. Niepodleglosci | dom | 450000 |
| 45/11      | Kormoranow | m. wlasnosciove | 230000 |
| 90         | Warszawska | dom, blizniak | 460000 |
| 53/2       | Olimpijska | apartament | 500000 |
| 1011      | Warszawska | dom | 700000 |
| 14/1       | Pszczyńska | m. wlasnosciove | 320000 |
| 15         | Sowinskiego | dom | 380000 |
| 16/2       | Krucza | apartament | 380000 |
| 156        | Kosynierow | dom | 510000 |
| 89/3       | Sylabusa | apartament | 550000 |
+-----+-----+-----+-----+
10 rows in set (0.00 sec)

mysql> SELECT cena, typ_lokalu FROM nieruchomosci WHERE ulica_nazwa = 'Warszawsk
a';
+-----+-----+
| cena | typ_lokalu |
+-----+-----+
| 460000 | dom, blizniak |
| 700000 | dom |
+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

Dane atomowe a Twoje tabele

Poniżej przedstawiliśmy kilka pytań, które możesz sobie zadać, aby ułatwić sobie określenie danych, jakie należy umieścić w tworzonych tabelach.



1. Co jest podstawowym zagadnieniem opisywanym przez tabelę?

Czy tabela opisuje kłownów, krowy, pączki, czy też polityków?



2. W jaki sposób będziesz korzystał z tabeli, by uzyskiwać informacje o tym zagadnieniu?

Projektuj tabele w taki sposób, aby przeszukiwanie ich było jak najprostsze.



3. Czy kolumny tabeli zawierają dane atomowe, dzięki czemu używane zapytania mogą być krótkie i precyzyjne?

..... Nie istnieją
grupy pytań

P.: Czyż atomy nie są bardzo małe? Czy nie powinienem zatem dzielić swoich informacji na naprawdę bardzo małe elementy?

O.: Nie. Tworzenie danych atomowych oznacza podzielenie ich na najmniejsze elementy konieczne do stworzenia wydajnych tabel, a nie na podzielenie ich na najmniejsze możliwe elementy.

Nie należy dzielić danych bardziej niż to konieczne. Jeśli nie potrzebujesz dodatkowych kolumn, to nie twórz ich tylko i wyłącznie dlatego, że mógłbyś to zrobić.

P.: W czym może mi pomóc stosowanie danych atomowych?

O.: Może Ci pomóc w zapewnieniu, że informacje przechowywane w tabeli będą precyzyjne. Jeśli na przykład utworzysz osobną kolumnę przeznaczoną do przechowywania numeru domu, to będziesz mógł upewnić się, że będą w niej zapisywane wyłącznie liczby.

Stosowanie danych atomowych pozwala także poprawić efektywność wykonywanych zapytań, gdyż same zapytania są łatwiejsze do napisania, a czas ich wykonania jest krótszy. Korzyści, jakie zapewnia nam stosowanie danych atomowych, są tym wyraźniejsze, im więcej jest danych przechowywanych w tabeli.



Poniżej przedstawiliśmy dwie oficjalne reguły dotyczące danych atomowych. Dla każdej z nich narysuj dwie hipotetyczne tabele, które nie będą spełniać wytycznych reguły.

Reguła 1.: W kolumnie z danymi atomowymi w jednym wierszu tabeli nie może się znajdować kilka wartości tego samego typu.

Tej reguły nie spełnia tabela Grześka – moje_kontakty, a konkretnie jej pole „zainteresowania”.

Reguła 2.: W tabeli zawierającej dane atomowe nie może być kilku kolumn zawierających dane tego samego typu.

Tej reguły nie spełnia tabela proste_drinki.

Zaostrz ołówek

**Rozwiązanie**

Poniżej przedstawiliśmy dwie oficjalne reguły dotyczące danych atomowych. Dla każdej z nich narysuj dwie hipotetyczne tabele, które nie będą spełniać wytycznych reguły.

Reguła 1.: W kolumnie z danymi atomowymi w jednym wierszu tabeli nie może się znajdować kilka wartości tego samego typu.

Oczywiście Twoja odpowiedź na pewno będzie inna, ale oto jeden z możliwych przykładów:

pożywienie	składniki
chleb	drożdże, mleko, jajka, mąka
sałatka	pomidor, ogórek, sałata

Czy pamiętasz tabelę Grześka? Zawierała ona kolumnę zainteresowania, w której Grzesiek zapisywał niejednokrotnie kilka różnych zainteresowań danej osoby, przez co przeszukiwanie tabeli było prawdziwym koszmarem.

Ta sama sytuacja występuje w tej tabeli. Wyobraź sobie odszukanie pomidorów wśród tych wszystkich pozostałych składników.

Reguła 2.: W tabeli zawierającej dane atomowe nie może być kilku kolumn zawierających dane tego samego typu.

nauczyciel	student1	student2	student3
Pani Martini	Janek	Romek	Kasia
Pan Grog	Sonia	Tymon	Julia

Zbyt wiele kolumn do podawania studentów!

**Ćwiczenie**

Skoro już znasz oficjalne reguły oraz trzy kroki pozwalające na stosowanie danych atomowych, przeanalizuj wszystkie tabele przedstawione do tej pory w książce i wyjaśnij, dlaczego zawierają one dane atomowe, albo co sprawia, że zapisywane w nich informacje nie są danymi atomowymi.

Tabela Grzeška, ze strony 83

Tabela z ocenami pączków, ze strony 112

Tabela z informacjami o kłownach, strona 155

Tabela drinków, ze strony 93

Tabela połowów wędkarskich, strona 194

Dlaczego warto być normalnym?

Kiedy wyczerpie się Twój limit godzin na konsultacje ze specjalistą do spraw baz danych i będziesz musiał zatrudnić projektantów baz SQL, fajnie by było, gdybyś nie musiał tracić cennych godzin na tłumaczenie im, jak działają Twoje tabele.

Cóż, tworzenie tabel ZNORMALIZOWANYCH oznacza, że są one zgodne z pewnymi standardami, które projektanci baz danych będą rozumieć. Co więcej, na pewno ucieszy Cię fakt, iż nasze tabele zawierające dane atomowe są już w połowie drogi do owej „normalności”.

Zapisywanie w tabeli danych atomowych jest pierwszym krokiem na drodze do tworzenia tabel ZNORMALIZOWANYCH.



Rozwiązanie ćwiczenia

Skoro już znasz oficjalne reguły oraz trzy kroki pozwalające na stosowanie danych atomowych, przeanalizuj wszystkie tabele przedstawione do tej pory w książce i wyjaśnij, dlaczego zawierają one dane atomowe, albo co sprawia, że zapisywane w nich informacje nie są danymi atomowymi.

Tabela Grzeška, ze strony 83 Nie jest atomowa. Kolumny „zainteresowania” i „szuka” nie są zgodne z regułą 1.

Tabela z ocenami pączków, ze strony 112 Tabela jest atomowa. W odróżnieniu od kolumn tabeli `proste_drinki` każda kolumna tej tabeli przechowuje informacje różnego typu. Oprócz tego, w odróżnieniu od kolumny „aktywnosci” tabeli `klownów` w każdej z kolumn tej tabeli przechowywany jest tylko jeden element informacji.

Tabela z informacjami o klownach, strona 155 Tabela nie jest atomowa. W niektórych rekordach w kolumnie „aktywnosci” zapisywanych jest więcej czynności niż jedna, co jest sprzeczne z regułą 1.

Tabela drinków, ze strony 93 Tabela nie jest atomowa. Zawiera ona więcej niż jedną kolumnę „składnik”, co jest sprzeczne z regułą 2.

Tabela połowów wędkarskich, strona 194 Tabela jest atomowa. W każdej kolumnie są zapisywane informacje innego typu. Każda kolumna zawiera także tylko jedną informację.

Zalety normalizacji tabel

1. W tabelach znormalizowanych dane się nie powielają, co pozwala ograniczyć wielkość bazy.

Unikanie powielania danych pozwoli Ci zaoszczędzić miejsce na dysku.

2. Dzięki mniejszej ilości informacji w bazie wszelkie zapytania będą wykonywane szybciej.



Moje tabele nie są aż tak duże. Dlaczego zatem mam sobie zawracać głowę jakąś normalizacją?



Ponieważ nawet w przypadku małych tabel można na tym zyskać.

Poza tym w miarę upływu czasu tabele stają się coraz większe. Jeśli od samego początku Twoje tabele będą znormalizowane, to w przyszłości, gdy zapytania zaczną być wykonywane zbyt wolno, nie będziesz musiał ich modyfikować.

Klowni nie są normalni

Czy pamiętasz tabelę z informacjami o publicznych wystąpieniach kłownów? Śledzenie poczyznań kłownów stało się ogólnokrajowym szaleństwem i nasza stara tabela może nie sprostać zwiększonym wymaganiom, ponieważ kolumny wygląd i aktywnosci zawierają *tak* wiele danych. Na nasze potrzeby ta tabela nie jest atomowa.

Wyszukiwanie danych w tych dwóch kolumnach jest naprawdę trudne, gdyż zawierają one tak wiele informacji!

klowni_informacje

imie	ostatnio_widziano	wyglad	aktywnosci
Eklerka	Dom opieki „Spokojna Ostoja”	K, czerwone włosy, zielona sukienka, ogromne stopy	balony, mały samochodzik
Pan Pimpus	Urodziny u Jacka Zielińskiego	M, pomarańczowe włosy, niebieski garnitur, ogromne stopy	mim
Pani Smyk	MegaStragan	K, żółta koszula, workowate czerwone spodnie	trąbka, parasolka
Pan Hobo	Cyrk Koloseum	M, cygaro, czarne włosy, niewielki kapelusz	skrzypce
Klarabela	Dom opieki „Wesoła Wdówka”	K, różowe włosy, ogromny kwiat, niebieska sukienka	krzyk, taniec
Skuter	Szpital Miejski	M, niebieskie włosy, czerwony garnitur, ogromny nos	balony
Zippo	Centrum Handlowe Skarbiec	K, pomarańczowy garnitur, workowate spodnie	taniec
Balbina	Auta Edwarda	K, cała na pomarańczowo i w cekinach	utrzymywanie równowagi, mały samochodzik
Gonzo		M, w przebraniu kobiety, kostium w plamki	śpiew, taniec
Pan Smyk	Zabawex	M, zielono-fioletowy kostium, szpiczasty nos	wchodzenie do malutkiego samochodu

Zaostrz ołówek



Spróbujmy tak zmodyfikować tabelę klowni_informacje, by stała się ona tabelą atomową. Spróbuj zaproponować inną strukturę tabeli, zakładając przy tym, że chcemy wyszukiwać w niej informacje na podstawie kolumn wyglad, aktywnosci oraz ostatnio_widziano.

Odpowiedzi znajdziesz na stronie 229

W połowie drogi do 1NF

Pamiętasz zapewne, że umieszczenie w tabeli danych atomowych to jedynie połowa drogi do znormalizowania tabeli. Kiedy tabela będzie całkowicie znormalizowana, przyjmie ona **PIERWSZĄ POSTAĆ NORMALNĄ** (ang. *first normal form*, w skrócie 1NF).

Aby tabela miała pierwszą postać normalną, musi spełniać dwa warunki:

Już wiem, jak zaspokoić ten wymóg.



Każdy wiersz danych musi zawierać wartości atomowe

Aby całkowicie znormalizować tabelę, do każdego z jej rekordów musimy dodać klucz główny.



Każdy wiersz danych musi mieć unikalny identyfikator, nazywany kluczem głównym (ang. primary key).



WYSIL SZARE KOMÓRKI

Jak sądzisz, jakiego typu kolumny będą się nadawały do tworzenia klucza głównego?

Reguły KLUCZA GŁÓWNEGO

Kolumnę, która ma pełnić rolę klucza głównego tabeli, należy utworzyć specjalnie w tym celu od razu podczas projektowania tabeli. Na kilku kolejnych stronach utworzymy tabelę i wskażemy kolumnę, która będzie jej kluczem głównym. Jednak zanim to zrobimy, przyjrzymy się, czym ten klucz główny jest.



Kluczem głównym tabeli nazywamy kolumnę, która sprawia, że każdy wiersz tabeli jest unikalny.



Klucz główny służy do zapewnienia unikalności wszystkich rekordów tabeli.

A to oznacza, że wartości w kolumnie klucza głównego nie mogą się powtarzać. Przeanalizujemy przykład tabeli przedstawionej poniżej. Jak myślisz, czy któraś z jej kolumn nadawałaby się na klucz główny?

PESEL	nazwisko	numer	telefon
-------	----------	-------	---------

↑
Numery PESEL mają unikalne wartości i są przypisywane konkretnym osobom. Może zatem ta kolumna nadawałaby się na klucz główny?

↖ ↗ ↗ ↗
W tych trzech kolumnach wartości mogą się powtarzać; na przykład z dużą dozą prawdopodobieństwa możemy założyć, że w tabeli znajdzie się więcej niż jedna osoba o imieniu Jan; podobnie może się zdarzyć, że kilka osób będzie mieszkać razem i używać tego samego numeru telefonu. Dlatego te trzy kolumny raczej nie będą się nadawały na pełnienie roli klucza głównego tabeli.



Zadbaj o swoje rekordy, używając kolumny PESEL jako klucza głównego tabeli.

Przy coraz częstszych kradzieżach tożsamości coraz mniej osób zgodzi się na podanie swojego numeru PESEL, co jest zresztą całkowicie zrozumiałe. Jest to zbyt ważna informacja, by ryzykować jej kradzież. Czy możesz z całkowitą pewnością stwierdzić, że Twoja baza jest bezpieczna? Jeśli nie, to wszystkie te numery PESEL mogą zostać skradzione wraz z tożsamością ich właścicieli.



W kolumnie klucza głównego nie mogą się pojawiać wartości NULL.

Gdyby w kolumnie klucza głównego można było zapisywać wartości NULL, to rekordy nie byłyby unikalne, gdyż wartość NULL mogłaby się pojawić w kilku z nich.



Wartość kolumny klucza głównego musi zostać określona w momencie dodawania rekordu do tabeli.

Jeśli wartość kolumny klucza głównego nie zostanie określona w momencie dodawania rekordu, to ryzykujemy, że pojawi się w niej wartość NULL, co może doprowadzić do powtórzenia się tego samego rekordu w tabeli i naruszenia zasad pierwszej postaci normalnej.



Klucz główny musi być krótki.

Klucz główny musi zawierać tylko te informacje, które są niezbędne dla zapewnienia jego unikalności, i nic więcej.



Wartości w kolumnie klucza głównego nie mogą się zmieniać.

Gdyby można było zmieniać wartości zapisane w kolumnie klucza głównego, to przypadkowo można by podać wartość, która została już użyta. Pamiętaj, wszystkie wartości w kolumnie klucza głównego zawsze muszą być unikalne.



WYSIŁ SZARE KOMÓRKI

Czy na podstawie tych wszystkich informacji możesz podać przykład dobrego klucza głównego?

Przejrzyj przykładowe tabele, które przedstawiliśmy we wcześniejszej części książki. Czy w którejś z nich jest kolumna zawierająca naprawdę unikalne wartości?

Chwila, a zatem, skoro nie mogę użyć numeru PESEL jako klucza głównego, a jednocześnie wartości w tej kolumnie wciąż muszą być unikalne, krótkie i niezmiennie, to co to może być?



Najlepszym kluczem głównym może być nowy klucz główny.

W przypadku tworzenia kluczy głównych najlepszym i najprostszym rozwiązaniem jest utworzenie kolumny, która będzie zawierać unikalne liczby. Wyobraź sobie tabelę zawierającą informacje o osobach, w której umieścisz dodatkową kolumnę liczbową. W przykładzie przedstawionym poniżej jest to kolumna `id`.

Gdyby nie kolumna `id`, to dwa rekordy Janka Kowalskiego byłyby identyczne. Jednak ponieważ istnieje klucz główny, oba te rekordy reprezentują różne osoby. A zatem klucz główny zapewnia unikalność tych dwóch rekordów. Poniższa tabela jest w pierwszej postaci normalnej.

id	nazwisko	imie	pseudonim
1	Kowalski	Janek	Pierun
2	Paciorek	Krystyna	Krycha
3	Kowalski	Janek	Pierun
4	Ślusarczyk	Anna	Ania
5	Paprocki	Krzysztof	Krzych

← Rekord Janka Kowalskiego.

← Ten rekord także zawiera dane Janka Kowalskiego, jednak wartość w kolumnie klucza głównego pokazuje, że jest to unikalny rekord i dotyczy całkowicie innej osoby niż pierwszy Janek Kowalski.



Dla maniaków

W świecie SQL-a trwa niekończąca się debata na temat tego, czy należy używać **syntetycznych** kluczy głównych, czyli specjalnie utworzonych (takich jak nasza kolumna `id`), czy też kluczy **naturalnych** — tworzonych na podstawie informacji, które już są zapisane w tabeli (takich jak numer nadwozia lub PESEL). Nie preferujemy żadnego z tych rozwiązań, a zagadnieniami związanymi z kluczami głównymi zajmiemy się bardziej szczegółowo w **rozdziale 7**.

..... Nie istnieją
grupy pytania

P: Cały czas mówicie o „pierwszej” postaci normalnej. Czy to oznacza, że istnieje też druga postać normalna? A może i trzecia?

U: Owszem, faktycznie istnieją także druga i trzecia postać normalna, a każda z nich narzuca coraz to bardziej wymagające warunki na strukturę i zawartość tabel. Zajmiemy się nimi dokładniej w rozdziale 7.

P: No dobrze, zatem zmieniliśmy nasze tabele w taki sposób, by zawierały dane atomowe. Czy któraś z nich znajduje się już w pierwszej postaci normalnej?

U: Nie. Jak na razie żadna ze stworzonych przez nas tabel nie zawiera ani klucza głównego, ani wartości unikalnych.

P: Wydaje mi się, że kolumna „komentarze” w tabeli o pączkach nie zawiera wartości atomowych. Chodzi mi o to, że nie ma żadnego prostego sposobu przeszukiwania jej zawartości.

U: Masz całkowitą rację. Ta kolumna raczej nie jest szczególnie atomowa. Choć z drugiej strony, projekt naszej tabeli nie narzucał takiej konieczności. Gdybyśmy jednak zdecydowali się na ograniczenie zawartości opinii do ściśle określonej grupy słów, to to pole mogłoby być atomowe. Jednak w takim przypadku zapisywane w bazie opinie nie byłyby spontaniczne.

Dążenie do pierwszej postaci NORMALNEJ

Nadszedł czas, by nieco się cofnąć i zająć się normalizacją naszych tabel. Musimy postarać się, by zapisane w nich informacje były atomowe, i dodać do nich klucze główne. Klucz główny tabeli jest zazwyczaj określany podczas tworzenia tabeli, czyli pisania polecenia CREATE TABLE.



WYSIL SZARE KOMÓRKI

Czy pamiętasz, jak można dodawać kolumny do istniejących tabel?

Poprawianie tabeli Grzeška


W jaki sposób — bazując na zdobytych już informacjach — powinieneś poprawić tabelę Grzeška:

Poprawa tabeli Grzeška — krok 1.: Pobierz wszystkie informacje zapisane w tabeli, używając w tym celu polecenia SELECT, i zapisz je gdzieś.

Poprawa tabeli Grzeška — krok 2.: Utwórz nową, znormalizowaną tabelę.

Poprawa tabeli Grzeška — krok 3.: Zapisz wszystkie stare dane w nowej tabeli, modyfikując dane w każdym z wierszy w taki sposób, by odpowiadały one nowej strukturze tabeli.

Teraz możesz usunąć starą wersję tabeli poleceniem DROP.



Chwileczkę, ale ja już mam tabelę z całą masą informacji. Chyba nie jesteście poważni, jeśli oczekujecie, że usunę tabelę jak w pierwszym rozdziale i będę ponownie wpisywał całą jej zawartość tylko i wyłącznie po to, żeby utworzyć kolumnę klucza głównego.

Mamy pewność, że tabela Grzeška nie jest doskonała.

Tabela Grzeška nie jest atomowa i nie ma klucza głównego. Jednak, na szczęście dla Grzeška, *nie jesteśmy skazani* na starą tabelę ani *nie musimy kopiować i ponownie wpisywać jej zawartości*.

Możemy dodać do tabeli Grzeška klucz główny i zapewnić atomowość danych przy użyciu jednego, nowego polecenia SQL. Jednak zanim to zrobimy, cofnijmy się nieco w przeszłość...

Oryginalna postać polecenia CREATE TABLE

Grzesiek musi dodać do swojej tabeli klucz główny; oprócz tego zdaje sobie sprawę, że jest kilka rzeczy, które może zrobić, by poprawić atomowość informacji zapisywanych w tabeli. Zanim przekonamy się, w jaki sposób można poprawić istniejącą tabelę, przypomnij sobie, jak ją utworzyliśmy.

Oto postać polecenia CREATE TABLE, którego użyliśmy do utworzenia tabeli Grzeška dawno temu, w **rozdziale 1.**:

CREATE TABLE moje_kontakty

```
(
nazwisko VARCHAR(30),
imie VARCHAR(20),
email VARCHAR(50),
plec CHAR(1),
data_urodzenia DATE,
zawod VARCHAR(50),
lokalizacja VARCHAR(50),
stan VARCHAR(20),
zainteresowania VARCHAR(100),
szuka VARCHAR(100)
);
```

*W tej tabeli
nie ma
kolumny
klucza
głównego.*

*Czy tworząc tabelę,
możemy poprawić
atomowość tych kolumn?*

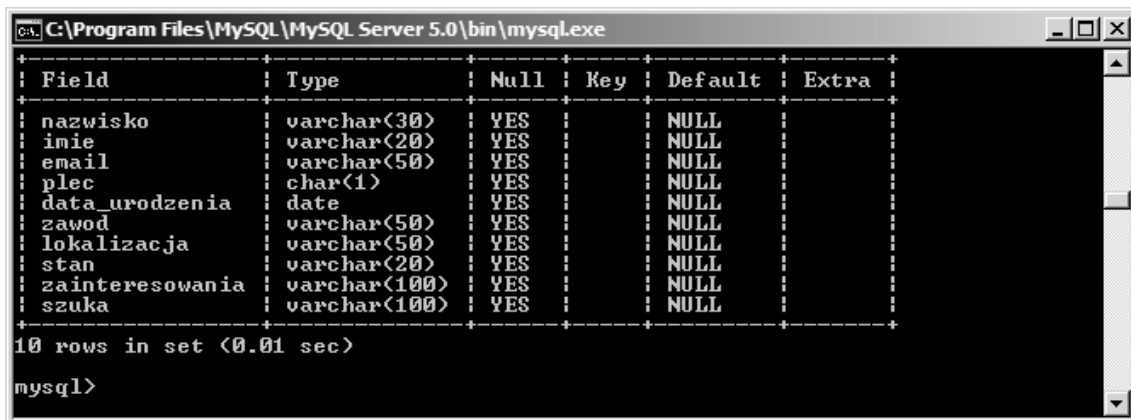


WYSIL SZARE KOMÓRKI

A co zrobić, gdybyśmy nie mieli nigdzie zanotowanego oryginalnego polecenia CREATE TABLE użytego do utworzenia tabeli? Czy istnieje jakiś sposób dotarcia do jego kodu?

Pokażcie mi moją ~~kasę~~ ^{tabełę}

A co by się stało, gdybyś zastosował polecenie DESCRIBE moje_kontakty, by podejrzeć kod użyty do utworzenia tej tabeli? Otóż gdybyś to zrobił, w oknie konsoli ujrzałbyś następujące wyniki:



```
C:\Program Files\MySQL\MySQL Server 5.0\bin\mysql.exe
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| nazwisko | varchar(30) | YES | | NULL | |
| imie | varchar(20) | YES | | NULL | |
| email | varchar(50) | YES | | NULL | |
| plec | char(1) | YES | | NULL | |
| data_urodzenia | date | YES | | NULL | |
| zawod | varchar(50) | YES | | NULL | |
| lokalizacja | varchar(50) | YES | | NULL | |
| stan | varchar(20) | YES | | NULL | |
| zainteresowania | varchar(100) | YES | | NULL | |
| szuka | varchar(100) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
10 rows in set (0.01 sec)

mysql>
```

Ale nam zależy na dotarciu do kodu polecenia CREATE, a nie do informacji o polach tabeli. W ten sposób będziemy bowiem mogli uniknąć konieczności ponownego wpisywania polecenia i określić, co powinniśmy zrobić już podczas tworzenia tabeli.

Polecenie SQL SHOW CREATE TABLE wyświetli kod polecenia CREATE TABLE, które pozwoli odtworzyć tabelę, oczywiście bez żadnych danych. Dzięki niemu w dowolnej chwili możemy się przekonać, w jaki sposób należy odtworzyć daną tabelę. Spróbuj wykonać następujące polecenie:

```
SHOW CREATE TABLE moje_kontakty;
```

Polecenie oszczędzające czas

Rzuć okiem na kod polecenia, którego użyliśmy do utworzenia tabeli, przedstawiony na stronie 217. Następnie porównaj go z zamieszczonym poniżej kodem zwróconym przez polecenie `SHOW CREATE TABLE moje_kontakty`. Nie są one identyczne, jednak gdybyś wykonał poniższe polecenie `CREATE TABLE`, to uzyskane wyniki byłyby takie same, jak w przypadku użycia oryginalnego polecenia. Nie musisz usuwać znaków lewego apostrofu ani ustawień dotyczących wartości domyślnych, jeśli jednak to zrobisz, to polecenie będzie bardziej przejrzyste i schludne.

Pomiędzy takimi znakami, określanymi jako lewy apostrof, są zapisywane nazwy kolumn oraz nazwa tabeli. Znaki lewego apostrofu są używane w wynikach generowanych przez polecenie `SHOW CREATE TABLE`.

```
CREATE TABLE `moje_kontakty`
(
  `nazwisko` varchar(30) default NULL,
  `imie` varchar(20) default NULL,
  `email` varchar(50) default NULL,
  `plec` char(1) default NULL,
  `data_urodzenia` date default NULL,
  `zawod` varchar(50) default NULL '',
  `lokalizacja` varchar(50) default NULL '',
  `stan` varchar(20) default NULL '',
  `zainteresowania` varchar(100) default NULL '',
  `szuka` varchar(100) default NULL ''
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

Jeśli jawnie nie zażądamy inaczej, to system zarządzania bazą danych przyjmuje, że wartości wszystkich kolumn przyjmują domyślnie wartość `NULL`.

Podczas tworzenia tabeli warto określić, czy w jej poszczególnych kolumnach mogą być zapisywane wartości `NULL`, czy też nie.

Nie musisz zwracać uwagi na ostatni wiersz kodu, umieszczony za nawiasem zamykającym. Określa on sposób przechowywania danych oraz używany zbiór znaków. Jak na razie w zupełności wystarczą nam ustawienia domyślne.

Aby wykonać to polecenie, będziesz musiał zmienić nazwę tabeli; no chyba że wcześniej usuniesz oryginalną tabelę.

Choć można by poprawić przejrzystość kodu (usuwając jego ostatni wiersz oraz wszystkie znaki lewego apostrofu), to jednak by utworzyć tabelę, nie trzeba wprowadzać w nim żadnych modyfikacji — wystarczy go skopiować i wkleić w przedstawionej postaci.

Tworzenie tabeli z KLUCZEM GŁÓWNYM

Ponizej przedstawiliśmy kod, który zwróciło polecenie `SHOW CREATE TABLE moje_kontakty`. Usunęliśmy z niego znaki lewego apostrofu oraz fragment ostatniego wiersza. Na samym początku listy kolumn dodaliśmy kolumnę `id_kontaktu`, której wartości, dzięki zastosowaniu wyrażenia `NOT NULL`, nie będą mogły przybierać wartości `NULL`. Z kolei na samym końcu listy dodaliśmy wyrażenie `PRIMARY KEY`, które określa, że kluczem głównym tabeli ma być nasza nowa kolumna `id_kontaktu`.

Pamiętaj, że zawartość kolumny klucza głównego musi być różna od `NULL` — `NOT NULL`! Gdyby w kolumnie klucza głównego pojawiła się wartość `NULL` lub gdyby można było pominąć wartość zapisywaną w tej kolumnie, to nie moglibyśmy zagwarantować, że wszystkie wiersze tabeli będą identyfikowane w jednoznaczny, unikalny sposób.

Utworzyliśmy nową kolumnę o nazwie `id_kontaktu`, w której będą zapisywane wartości liczbowe. Kolumna ta będzie pełnił funkcję klucza głównego naszej tabeli. Każda wartość w tej kolumnie będzie unikalna, dzięki czemu cała tabela stanie się tabelą atomową.

```
CREATE TABLE moje_kontakty
(
  id_kontaktu INT NOT NULL,
  nazwisko varchar(30) default NULL,
  imie varchar(20) default NULL,
  email varchar(50) default NULL,
  plec char(1) default NULL,
  data_urodzenia default NULL,
  zawod varchar(50) default NULL,
  lokalizacja varchar(50) default NULL,
  stan varchar(20) default NULL,
  zainteresowania varchar(100) default NULL,
  szuka varchar(100) default NULL,
  PRIMARY KEY (id_kontaktu)
)
```

To właśnie w tym miejscu określamy klucz główny tabeli. Składnia używana do tego celu jest naprawdę prosta — wystarczy zapisać słowa `PRIMARY KEY`, a następnie podać w nawiasach nazwę kolumny klucza głównego. W naszym przypadku kluczem głównym będzie kolumna `id_kontaktu`.

P: **Zatem mówicie, że KLUCZ GŁÓWNY nie może przybierać wartości NULL. Co jeszcze zapewnia jego unikalność?**

U: Najprościej rzecz ujmując — jedynie Ty. Zapisując w tabeli każdy nowy wiersz, w kolumnie `id_kontaktu` będziesz umieszczał nową, unikalną wartość. Na przykład w pierwszym wydawanym poleceniu `INSERT` w kolumnie `id_kontaktu` zapiszesz wartość 1, w kolejnym wierszu wartość 2 i tak dalej.

P: **Wyobrażam sobie, że takie określanie unikalnych wartości KLUCZA GŁÓWNEGO w każdym nowym rekordzie musi być naprawdę kłopotliwe. Czy nie ma żadnego prostszego rozwiązania?**

U: Owszem, są prostsze rozwiązania; i to nawet dwa. Pierwszym z nich jest zastosowanie jako klucza głównego jakiejś istniejącej kolumny danych, o której wiemy, że ma unikalne wartości. Wspominaliśmy jednak, że takie rozwiązanie może przysparzać pewnych problemów (na przykład w przypadku stosowania numerów PESEL użytkownicy mogą się obawiać przejęcia swoich poufnych informacji przez niepożądane osoby).

Prostym rozwiązaniem jest utworzenie zupełnie nowej kolumny, przeznaczonej do przechowywania unikalnych wartości klucza głównego, takiej jak kolumna `id_kontaktu` na poprzedniej stronie. W takim przypadku można nakazać systemowi obsługi baz danych, by automatycznie wypełniał wartość takiej kolumny w każdym rekordzie dodawanym do tabeli. Do tego celu służy specjalne słowo kluczowe. Zajrzyj na następną stronę — tam znajdziesz więcej informacji na ten temat.

P: **Czy polecenia SHOW mogą używać także do wyświetlania innych informacji, czy tylko do odtworzenia polecenia CREATE TABLE?**

U: Polecenia `SHOW` można także używać do wyświetlania informacji o konkretnych kolumnach:

```
SHOW COLUMNS FROM nazwa_tabeli;
```

To polecenie wyświetli nazwy wszystkich kolumn podanej tabeli wraz ze wszystkimi informacjami na ich temat.

```
SHOW CREATE DATABASE nazwa_bazy_danych;
```

Powyższe polecenie, podobnie jak polecenie `SHOW CREATE TABLE nazwa_tabela;`, zwraca kod polecenia `CREATE`, z tym że w tym przypadku będzie to polecenie służące do utworzenia całej bazy danych.

```
SHOW INDEX FROM nazwa_tabeli;
```

Z kolei to polecenie wyświetli wszystkie kolumny, dla których zdefiniowano indeksy, wraz z informacjami na temat typów tych indeksów. Jak na razie jedynym typem indeksów, z jakim się zetknęliśmy, był klucz główny, jednak w dalszej części książki to polecenie stanie się znacznie bardziej przydatne.

Istnieje jeszcze jedna wersja polecenia `SHOW`, i to wersja **BARDZO** użyteczna:

```
SHOW WARNINGS;
```

Jeśli w oknie konsoli pojawi się informacja, że podczas wykonywania polecenia `SQL` pojawiły się jakieś ostrzeżenia, to użyj tego polecenia, by wyświetlić ich treść.

Dostępne są jeszcze inne wersje polecenia `SHOW`, jednak te przedstawione powyżej dotyczą zagadnień, którymi się już zajmowaliśmy.

P: **No a co ze znakami lewego apostrofu, które zostały użyte w kodzie zwróconym przez polecenie SHOW CREATE TABLE? Czy jesteście pewni, że nie są potrzebne?**

U: Są one używane dlatego, że w niektórych przypadkach system zarządzania bazami danych może nie być w stanie określić, że nazwa ma być nazwą kolumny. Jeśli jednak zapiszesz nazwę kolumn pomiędzy znakami lewego apostrofu, to istnieje możliwość stosowania słów kluczowych języka `SQL` jako nazw kolumn; choć absolutnie nie polecamy takiego rozwiązania.

Na przykład wyobraźmy sobie, że z jakichś niewytłumaczalnych powodów chcielibyśmy nadać kolumnie nazwę `select`. Poniższa deklaracja kolumny jest jednak nieprawidłowa:

```
select varchar(50)
```

Zamiast niej musielibyśmy użyć następującej deklaracji:

```
`select` varchar(50)
```

P: **A niby czemu stosowanie słów kluczowych języka SQL jako nazw kolumn jest złym rozwiązaniem?**

U: Oczywiście wolno to robić, jednak nie jest to dobry pomysł. Pomyśl tylko, jak mylące i trudne do zrozumienia mogą się stać zapytania oraz jak męczące będzie wpisywanie tych wszystkich znaków lewego apostrofu. Poza tym `select` nie jest dobrą nazwą dla kolumny, gdyż nie mówi nic o danych, jakie są w niej zapisywane.

1, 2, 3... automatycznie inkrementowane

Dodanie do naszej kolumny `id_kontaktu` słowa kluczowego `AUTO_INCREMENT` nakaże, by w pierwszym wierszu tabeli przyjęła ona wartość 1, a w każdym kolejnym była powiększana o 1.

```
CREATE TABLE moje_kontakty
(
  id_kontaktu INT NOT NULL AUTO_INCREMENT,
  nazwisko varchar(30) default NULL,
  imie varchar(20) default NULL,
  email varchar(50) default NULL,
  plec char(1) default NULL,
  data_urodzenia default NULL,
  zawod varchar(50) default NULL,
  lokalizacja varchar(50) default NULL,
  stan varchar(20) default NULL,
  zainteresowania varchar(100) default NULL,
  szuka varchar(100) default NULL,
  PRIMARY KEY (id_kontaktu)
)
```

To jest to! W większości systemów zarządzania bazami danych wystarczy do deklaracji kolumny dodać słowo kluczowe `AUTO_INCREMENT`. (Użytkownicy MS SQL pamiętajcie: w Waszym przypadku będzie to słowo kluczowe `INDEX`, po którym należy podać wartość początkową oraz liczbę, o którą będą powiększane wartości w poszczególnych kolumnach. Bardziej szczegółowe informacje na ten temat można znaleźć w dokumentacji serwera MS SQL).

To słowo kluczowe robi dokładnie to, czego można by się spodziewać: sprawia, że w pierwszym wierszu tabeli, w danej kolumnie zostanie automatycznie zapisana wartość 1, która w kolejnych wierszach będzie automatycznie powiększana o 1.



No dobrze, to chyba faktycznie jest proste. Ale w jaki sposób zapisać w tabeli wiersz, w którym wartość takiej kolumny będzie już określona? Poza tym, czy można zmienić wartość zapisaną w takiej kolumnie?

A jak myślisz, co się w takich przypadkach stanie?

Najlepiej będzie, jeśli sam spróbujesz wykonać takie operacje, i przekonasz się, jakie będą ich skutki.



Ćwiczenie

1. Poniżej, w pustych wierszach, zapisz kod polecenia CREATE TABLE, które pozwoli utworzyć tabelę służącą do przechowywania imion i nazwisk osób. Powinna ona zawierać kolumnę klucza głównego, której wartości będą automatycznie inkrementowane, oraz dwie kolumny atomowe.

.....

.....

.....

2. Wykonaj powyższe polecenie w oknie konsoli bazy danych lub innym programie do zarządzania bazą.

3. Spróbuj wykonać każde z poniższych poleceń INSERT. Zakreśl te, które działają prawidłowo.

```
INSERT INTO twoja_tabela (id, imie, nazwisko)
VALUES (NULL, 'Marysia', 'Krawczyk');
```

```
INSERT INTO twoja_tabela (id, imie, nazwisko)
VALUES (1, 'Janka', 'Krawczyk');
```

```
INSERT INTO twoja_tabela
VALUES ('', 'Bartek', 'Krawczyk');
```

```
INSERT INTO twoja_tabela (imie, nazwisko)
VALUES ('Cecylia', 'Krawczyk');
```

```
INSERT INTO twoja_tabela (id, imie, nazwisko)
VALUES (99, 'Piotr', 'Krawczyk');
```

4. Czy udało się zapisać w tabeli informacje o wszystkich członkach rodziny Krawczyków? Zapisz poniżej zawartość tabeli po wykonaniu wszystkich poniższych poleceń.

twoja_tabela

id	imie	nazwisko



Rozwiązanie ćwiczenia

1. Poniżej, w pustych wierszach, zapisz kod polecenia CREATE TABLE, które pozwoli utworzyć tabelę służącą do przechowywania imion i nazwisk osób. Powinna ona zawierać kolumnę klucza głównego, której wartości będą automatycznie inkrementowane, oraz dwie kolumny atomowe.

```
CREATE TABLE twoja_tabela
(
  id INT NOT NULL AUTO_INCREMENT,
  imie VARCHAR(20),
  nazwisko VARCHAR(30),
  PRIMARY KEY (id)
);
```

2. Wykonaj powyższe polecenie w oknie konsoli bazy danych lub innym programie do zarządzania bazą.
3. Spróbuj wykonać każde z poniższych poleceń INSERT. Zakreśl te, które działają prawidłowo.

INSERT INTO twoja_tabela (id, imie, nazwisko)
VALUES (NULL, 'Marysia', 'Krawczyk');

INSERT INTO twoja_tabela (id, imie, nazwisko)
VALUES (1, 'Janka', 'Krawczyk');

INSERT INTO twoja_tabela
VALUES ('', 'Bartek', 'Krawczyk');

INSERT INTO twoja_tabela (imie, nazwisko)
VALUES ('Cecylia', 'Krawczyk');

INSERT INTO twoja_tabela (id, imie, nazwisko)
VALUES (99, 'Piotr', 'Krawczyk');

4. Czy udało się zapisać w tabeli informacje o wszystkich członkach rodziny Krawczyków? Zapisz poniżej zawartość tabeli po wykonaniu wszystkich poniższych poleceń.

To ostatnie polecenie „zadziała”, jednak zmodyfikuje wartość kolumny AUTO_INCREMENT.

twoja_tabela

id	imie	nazwisko
1	Marysia	Krawczyk
2	Bartek	Krawczyk
3	Cecylia	Krawczyk
<u>99</u>	Piotr	d Krawczyk

Wygląda na to, że zgubiliśmy Jankę; zapewne dlatego, że próbowaliśmy jej przypisać indeks, który został już użyty w rekordzie Marysi. Marysia, Marysia, Marysia!

Nie, istnieją
głupie pytania

P: Dlaczego pierwsze polecenie, to z wartością NULL w kolumnie id, zadziało? Przecież ta kolumna została zadeklarowana jako NOT NULL.

O: Faktycznie wygląda to tak, jakby to polecenie zadziało, choć nie powinno. Okazuje się jednak, że w przypadku użycia AUTO_INCREMENT wartości NULL są ignorowane. Oczywiście, gdybyśmy nie umieścili w deklaracji kolumny słowa kluczowego AUTO_INCREMENT, to próba wykonania takiego polecenia spowodowałaby zgłoszenie błędu, a rekord nie zostałby zapisany. Spróbuj sam, a się przekonasz.

Słuchajcie, nie mogę powiedzieć, żebyście mnie przekonali. Jak widać, postępując się poleceniem SHOW CREATE TABLE, mogę uzyskać i wykonać kod, który utworzy tabelę; jednak cały czas mam wrażenie, że będę musiał usunąć swoją oryginalną tabelę i ponownie, od nowa, wpisać całą jej zawartość tylko i wyłącznie po to, by dodać kolumnę klucza głównego.



Nie będziesz musiał wpisywać wszystkiego od nowa. Wystarczy, że użyjesz polecenia ALTER.

Nie trzeba kopiować danych zapisanych w istniejącej tabeli, a następnie jej usuwać i tworzyć od nowa. Można bowiem modyfikować strukturę istniejących tabel. Jednak żeby to zrobić, musimy sięgnąć po polecenie ALTER i kilka używanych przez niego słów kluczowych, które opisaliśmy w rozdziale 5.

Dodawanie KLUCZA GŁÓWNEGO do istniejącej tabeli

Poniżej przedstawiliśmy kod, który pozwoli dodać do tabeli `moje_kontakty` kolumnę klucza głównego, której wartości będą automatycznie inkrementowane. (Polecenie jest naprawdę długie, więc będziesz musiał obrócić książkę, żeby je przeczytać).

FIRST to słowo, które nakazuje, by nowa kolumna została dodana jako pierwsza kolumna tabeli. Choć nie ma to większego znaczenia, to jednak umieszczenie klucza głównego jako pierwszej kolumny tabeli jest dobrym pomysłem.

A to kod, który doda do tabeli nową kolumnę. Czy nie wygląda znajomo?

Oto nasze nowe polecenie SQL — ALTER.

```
ALTER TABLE moje_kontakty  
ADD COLUMN id_kontaktu INT NOT NULL AUTO_INCREMENT FIRST,  
ADD PRIMARY KEY (id_kontaktu);
```

*Działanie słów **ADD COLUMN** jest zgodne z ich znaczeniem w języku angielskim — dodają do tabeli nową kolumnę. W naszym przypadku kolumna ta będzie nosić nazwę `id_kontaktu`.*

Powinieneś już rozpoznać ten kod — wskazuje on kolumnę, która będzie pełnił rolę klucza głównego.



**WYSIL
SZARE KOMÓRKI**

Czy sądzisz, że powyższe polecenie doda wartości do kolumny `id_kontaktu` w już istniejących rekordach tabeli czy tylko w rekordach, które dopiero będą w niej zapisywane? Jak możesz to sprawdzić?

Modyfikacja tabeli i dodanie KLUCZA GŁÓWNEGO

Sprawdź wyniki działania polecenia ALTER przedstawionego na poprzedniej stronie. Wybierz bazę danych lista_grzesia poleceniem USE i wykonaj polecenie:

Ten komunikat informuje nas, że polecenie dodało nową kolumnę do 37 rekordów, które już były zapisane w bazie. Ty zapewne nie będziesz ich mieć aż tak wielu.

```
C:\Program Files\MySQL\MySQL Server 5.0\bin\mysql> ALTER TABLE moje_kontakty
-> ADD COLUMN id_kontaktu INT NOT NULL AUTO_INCREMENT FIRST,
-> ADD PRIMARY KEY (id_kontaktu);
Query OK, 37 rows affected (0.22 sec)
Records: 37 Duplicates: 0 Warnings: 0
mysql>
```

Zajefajnie...
Teraz już mam kolumnę klucza głównego wraz z odpowiednimi wartościami. Czy mogę użyć polecenia ALTER także do dodania kolumny do zapisywania numeru telefonu?

Aby sprawdzić, co się zmieniło w tabeli Grześka, wykonaj polecenie **SELECT * FROM moje_kontakty;**



Kolumna id_kontaktu została dodana jako pierwsza — przed wszystkimi pozostałymi kolumnami.

```
C:\Program Files\MySQL\MySQL Server 5.0\bin\mysql> SELECT * FROM moje_kontakty;
```

id_kontaktu	nazwisko	imie	email
1	Kubaski	Andrzej	Kubaski_Andrzej@pizza-nzk.com.pl
2	Nowak	Joanna	nowjn@op.pl
3	Symonowicz	Anna	anna_tymonowicz@pizza-nzk.com.pl
4	Oczkiewicz	Adam	oczam@zygyg.eu
5	Karczynska	Marta	karnar@interia.pl
6	Zefirek	Monika	moniaze@pizza-nzk.com.pl
7	Harmonia	Ania	a_harmonia@mojanuza.pl
8	Mikulski	Janek	mikunek@wp.pl
9	Tkacz	Martyna	mptkacz@o2.eu
10	Maliniak	Konrad	kon_malinka@op.pl
11	Piekarska	Kasia	mysia_kasia@o2.eu
12	Wawalski	Mina	wer_mina@op.pl

Ponieważ użyliśmy słowa kluczowego AUTO_INCREMENT, zawartość tej kolumny była uzupełniana podczas modyfikowania poszczególnych, istniejących już wierszy tabeli.

Kiedy w przyszłości będziemy dodawali nowy rekord, to w jego kolumnie id_kontaktu zostanie zapisana wartość o jeden większa od największej wartości zapisanej w całej tabeli w kolumnie id_kontaktu. A zatem, jeśli w ostatnim dodawanym rekordzie w kolumnie kontakt_id znajduje się wartość 23, to w kolejnym rekordzie znajdzie się wartość 24.

Pamiętaj, to nie jest koniec tabeli Grześka; tak naprawdę znajduje się w niej jeszcze wiele innych kontaktów.

Czy Grześkowi uda się dodać kolumnę do zapisywania numeru telefonu? Zajrzyj do rozdziału 5., żeby sprawdzić, czy to możliwe.



Przybornik SQL

A zatem opanowałeś materiał z czwartego rozdziału książki. Przyjrzyj się jeszcze raz wszystkim nowym narzędziom, jakie dodałeś do swojego SQL-owego przybornika. Kompletną listę porad znajdziesz w dodatku C, zamieszczonym na końcu książki.

DANE ATOMOWE — REGUŁA 1.

Jeśli dane mają być atomowe, to w tej samej kolumnie nie może się znajdować kilka elementów danych tego samego typu.

DANE ATOMOWE — REGUŁA 2.

Aby dane w tabeli były atomowe, nie może się w niej znajdować kilka kolumn zawierających informacje tego samego typu.

KLUCZ GŁÓWNY

Kolumna lub zbiór kolumn, które w unikalny sposób identyfikują wiersze tabeli.

AUTO_INCREMENT

Zastosowanie tego słowa kluczowego w deklaracji tabeli sprawi, że w momencie wykonywania poleceń INSERT w danej kolumnie będą się automatycznie i magicznie pojawiały unikalne liczby całkowite.

DANE ATOMOWE

Dane w tabeli są atomowe, jeśli zostały już podzielone na najmniejsze elementy, które są Ci potrzebne.

SHOW CREATE TABLE

Skorzystaj z tego polecenia, by poznać składnię polecenia SQL, które pozwoli Ci odtworzyć istniejącą tabelę.

PIERWSZA POSTAĆ NORMALNA (1NF)

Każdy wiersz danych musi zawierać dane atomowe oraz unikalny identyfikator.

Zaostrz ołówek



Rozwiązanie

Spróbujmy tak zmodyfikować tabelę kłowni_informacje, by stała się ona tabelą atomową. Spróbuj zaproponować inną strukturę tabeli, zakładając przy tym, że chcemy wyszukiwać w niej informacje na podstawie kolumn wyglad, aktywnosci oraz ostatnio_widziano.

W tym przypadku nie ma jednego, właściwego rozwiązania.

Najlepszą rzeczą, jaką możesz zrobić, jest wydzielenie takich informacji jak płeć, kolor koszuli, kolor spodni, typ kapelusza, instrument muzyczny, środek lokomocji, balony (pole logiczne — tak lub nie), śpiew (także pole logiczne z wartościami tak lub nie), taniec (także pole logiczne).

Aby zapewnić atomowość tabeli, musiałbyś wydzielić te wszystkie aktywności i umieścić je w osobnych kolumnach, i podobnie zrobić z informacjami o wyglądzie.

Dodatkowe punkty możesz sobie przyznać, jeśli chcesz rozbić adres i umieścić te informacje w takich kolumnach jak adres, miejscowość i województwo.