helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

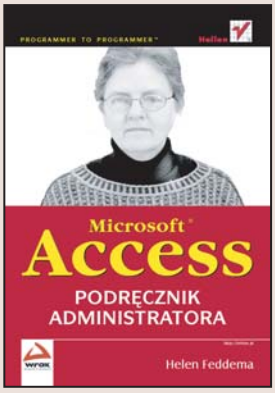
KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

Microsoft Access. Podręcznik administratora



Autor: Helen Feddema
Tłumaczenie: Rafał Jońca
ISBN: 83-246-0279-8
Tytuł oryginału: Expert One-on-One Microsoft Access Application Development
Format: B5, stron: 588

Program Microsoft Access cieszy się sporą popularnością wśród użytkowników. Nie wszyscy jednak wiedzą, że aby prawidłowo zarządzać bazami danych Accessa, należy używać specjalnych aplikacji. Ich napisane to skomplikowany proces – trzeba zaprojektować interfejs użytkownika, dobrać komponenty, stworzyć kwerendy, opracować procedury, a przede wszystkim umiejętnie połączyć to wszystko w całość.

W książce „Microsoft Access. Podręcznik administratora" znajdziesz kompletny zbiór informacji na temat budowania aplikacji w Accessie. Nauczysz się projektować i tworzyć tabele oraz wykorzystywać gotowe komponenty interfejsu użytkownika. Dowiesz się, jak napisać wydajne i szybkie kwerendy oraz przedstawić dane w postaci wykresów, tabel i raportów. Poznasz elementy języka VBA i możliwości wymiany danych między różnymi aplikacjami pakietu MS Office.

- Typy danych w tabelach MS Access
- Normalizacja danych
- Projektowanie formularzy
- Zastosowanie kwerend do przetwarzania danych
- Prezentacja danych na wykresach
- Drukowanie danych z zastosowaniem raportów
- Pisanie kodu modułów aplikacji
- Modyfikowanie i aktualizowanie aplikacji
- Konwersja baz danych ze starszych wersji Accessa
- Łączenie aplikacji z innymi składnikami pakietu MS Office

Jeśli chcesz stworzyć wydajną i sprawnie działającą aplikację, sięgnij po tę książkę – znajdziesz w niej wszystkie niezbędne informacje.

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

O autorce 11

Wprowadzenie 13

Część I Tworzenie aplikacji w Accessie 17

Rozdział 1. Tworzenie bazy danych dla aplikacji 19

Uzyskiwanie informacji 20
Rozpoznanie zadań i obiektów biznesowych 21
Określenie encji 22
Tworzenie tabel dla aplikacji 24
Sposoby tworzenia tabel 25
Tworzenie tabel 28
Typy danych pól tabeli 33
Normalizacja 46
Pierwsza postać normalna — eliminacja powtarzających się grup 46
Druga postać normalna — eliminacja redundancji danych 47
Trzecia postać normalna — eliminacja kolumn, które nie zależą od klucza 48
Czwarta postać normalna — izolacja niezależnych związków wielokrotnych 48
Piąta postać normalna — izolacja semantycznie powiązanych związków wielokrotnych 48
Określenie związków 49
Związek jeden do wielu 50
Związek jeden do jednego 52
Związek wiele do wielu 53
Podsumowanie 54

Rozdział 2. Zastosowanie formularzy do pracy z danymi 55

Typy formularzy 55
Tworzenie formularzy 60
Korzystanie z dodatku Design Schemes 60
Tworzenie standardowego formularza głównego 61
Tworzenie i osadzanie podformularzy typu pojedynczego i arkusza danych 74
Tworzenie formularzy ciągłych 91
Tworzenie formularza dla związku wiele do wielu 93
Podsumowanie 94

Rozdział 3. Wybór odpowiednich formantów dla formularza 97

Formanty standardowe 97
Przydatne formanty standardowe 97
Mniej użyteczne formanty standardowe 129

Formanty specjalne ... 130
Formant Karta (zakładki) ... 131
Formant Kalendarz ... 131
Formant TreeView (widok drzewa) ... 132
Formanty ActiveX ... 135
Formant DateTimePicker ... 136
Formant MonthView ... 136
Podsumowanie ... 137

Rozdział 4. Sortowanie i filtrowanie danych za pomocą kwerend ... 139

Kwerendy wybierające ... 139
Proste kwerendy wybierające ... 139
Kwerendy podsumowujące ... 145
Kwerendy krzyżowe ... 150
Kwerendy modyfikujące ... 155
Kwerenda dołączająca ... 155
Kwerenda aktualizująca ... 158
Kwerendy tworzące tabele ... 163
Kwerenda usuwająca ... 164
Kwerendy specyficzne dla języka SQL ... 166
Kwerendy składające ... 166
Kwerendy definicji danych ... 168
Podkwerendy ... 170
Podsumowanie ... 172

Rozdział 5. Zastosowanie tabel i wykresów przestawnych do interakcji z danymi ... 173

Utworzenie kwerendy będącej źródłem danych ... 174
Tabele przestawne ... 175
Pasek narzędziowy tabeli przestawnej ... 182
Okno właściwości tabeli przestawnej ... 185
Wykresy przestawne ... 186
Pasek narzędziowy wykresu przestawnego ... 190
Osadzony wykres przestawny ... 192
Podsumowanie ... 193

Rozdział 6. Drukowanie danych przy użyciu raportów ... 195

Zasady projektowania raportów ... 195
Kreator raportów ... 196
Korzystanie z szablonów raportów ... 200
Raporty tabelaryczne ... 201
Raporty grupujące ... 203
Raporty etykietowe ... 208
Raporty kolumnowe ... 209
Raporty z podraportami ... 214
Raporty filtrowane przez formularz ... 215
Raporty podsumowujące ... 218
Specjalne formatowanie raportów ... 218
Formatowanie warunkowe dla formantów i sekcji raportu ... 219
Umieszczenie na raporcie znaku wodnego ... 222

Tworzenie głównego menu za pomocą dodatku Menu Manager ... 226
Instalacja ... 226
Przygotowania ... 227
Uruchomienie dodatku ... 228
Podsumowanie ... 232

Rozdział 7. Pisanie kodu VBA w modułach ... 233

Eksplorator projektu ... 234
Okienko właściwości ... 234
Okno Immediate ... 235
Okno modułu ... 235
Uzyskiwanie pomocy ... 236
Typy modułów ... 236
Instrukcje używane w modułach ... 237
Instrukcja Call ... 237
Pętla Do While … Loop ... 238
Pętla Do Until … Loop ... 238
Pętla For … Next ... 239
Pętla For Each … Next ... 240
Instrukcja GoTo ... 240
Instrukcja If … Then … Else ... 240
Instrukcja Select Case … End Select ... 241
Instrukcja With … End With ... 243
Przykłady kodu z modułów standardowych ... 244
Wykorzystanie danych z tabeli tblInfo ... 244
Kod zakresu dat dla menu głównego ... 246
Przykładowy kod kryjący się za formularzami ... 248
Formularz nowego rekordu ... 251
Formularze sortowania i filtracji ... 256
Zdarzenie dwukrotnego kliknięcia dla podformularza arkusza danych ... 261
Podsumowanie ... 263

Część II Modyfikacja, uaktualnianie i konserwacja aplikacji w Accessie 265

Rozdział 8. Zarządzanie cyklem życia aplikacji ... 267

Modyfikacja aplikacji ... 267
Aktualizacja do nowej wersji pakietu Office ... 268
Aktualizacja aplikacji do nowego formatu bazy danych ... 270
Ustawienie osobnych partycji systemowych dla różnych wersji pakietu Office ... 270
Korzystanie z baz danych Accessa 2000 w nowszych wersjach Accessa ... 271
Praca z klientami ... 278
Relacje z klientami ... 279
Dokonywanie modyfikacji żądanych przez klienta ... 280
Zmiany związane z aplikacją ... 280
Modyfikacje standardowe ... 293
Dostosowanie komponentu listów Worda z głównego menu ... 296
Podsumowanie ... 305

Rozdział 9. Modyfikacja istniejącej aplikacji ... 307

Stosowanie konwencji nazewnictwa ... 307
Konwencja nazewnictwa Leszynskiego ... 309
Zastosowanie notacji LNC w bazie danych ... 315

Tworzenie menu ... 321
Normalizacja tabel i relacji ... 322
Modyfikacja formularzy, by było możliwe wyświetlanie i edytowanie danych ze związków wiele do wielu ... 328
Modyfikacja formularza fpriEBookNotes ... 329
Modyfikacja formularza frmTreeViewEBookNotes ... 333
Utworzenie nowego formularza frmAuthors ... 338
Modyfikacja raportów, by uwzględniały związek wiele do wielu ... 340
Podsumowanie ... 346

Rozdział 10. Przenoszenie starych danych do nowej bazy danych ... 347

Pobieranie starych danych ... 347
Wykorzystanie kwerend z wyliczanymi polami do przenoszenia danych z tabel nieznormalizowanych do znormalizowanych ... 350
Wykorzystanie kodu VBA do przenoszenia danych z nieznormalizowanej tabeli do powiązanych tabel znormalizowanych ... 358
Denormalizacja tabel ... 368
Podsumowanie ... 375

Część III Korzystanie z innych komponentów pakietu Office (i nie tylko) 377

Rozdział 11. Korzystanie z Worda ... 379

Pisanie kodu automatyzacji ... 380
Przydatne przyciski paska narzędziowego ... 381
Model obiektu Worda ... 382
Diagram modelu obiektu Worda ... 382
Uwaga na temat wyliczeń ... 384
Podstawowe polecenia automatyzacji ... 386
Skrótowe tworzenie kodu VBA dla Worda ... 387
Eksport danych Accessa do dokumentów Worda ... 388
Typy scalania ... 389
Przykłady wykonywania scalenia ... 390
Import danych z tabel Worda do Accessa ... 426
Problemy automatyzacji i sposoby ich unikania ... 432
Referencje ... 433
Pobranie katalogu dokumentów z Worda ... 434
Podsumowanie ... 438

Rozdział 12. Korzystanie z Outlooka ... 439

Tworzenie kodu automatyzacji ... 439
Model obiektu Outlooka ... 440
Diagram modelu obiektu Outlooka ... 440
Podstawowe polecenia automatyzacji ... 444
Obiekt NameSpace i inne obiekty o nietypowych nazwach ... 445
Składnia dotycząca referencji do obiektów Outlooka ... 445
Odniesienia do elementów Outlooka w kodzie i interfejsie ... 447
Kreator wymiany danych z Outlookiem ... 447
Eksport danych Accessa do elementów Outlooka ... 448
Tworzenie wiadomości e-mail z poziomu menu głównego ... 451
Formularz z listą umożliwiającą wybór wielu elementów ... 453
Formularz z arkuszem danych ... 464
Wysłanie e-maila z raportem ... 483

Import danych z elementów Outlooka do Accessa ... 495
Wykorzystanie biblioteki Redemption
w celu uniknięcia komunikatu ochrony modelu obiektu ... 504
Podsumowanie ... 506

Rozdział 13. Korzystanie z Excela ... 507

Pisanie kodu automatyzacji ... 510
Model obiektu Excela ... 511
Diagram modelu obiektu Excela ... 511
Podstawowe polecenia automatyzacji ... 513
Zeszyty, arkusze kalkulacyjne i zakresy ... 515
Skrótowe tworzenie kodu VBA dla Excela ... 516
Eksport danych z Accessa do Excela ... 516
Formularz z listą umożliwiającą wybór wielu elementów ... 517
Formularz z arkuszem danych ... 532
Import danych z Excela ... 537
Import z arkuszy i zakresów ... 538
Łącze do arkuszy i zakresów danych Excela ... 540
Podsumowanie ... 542

Rozdział 14. Korzystanie z elementów spoza pakietu Office ... 543

Wysyłanie faksów za pomocą WinFax ... 543
Wysłanie pojedynczego faksu za pomocą DDE ... 543
Faksowanie raportu za pomocą DDE ... 554
Wysyłanie wielu faksów przy użyciu kodu automatyzacji ... 561
Zapis danych do pliku tekstowego ... 566
Eksport kontaktów do Outlooka ... 567
Podsumowanie ... 572

Skorowidz ... 573

1

Tworzenie bazy danych dla aplikacji

Aplikacja to coś znacznie więcej niż tylko baza danych. Każda osoba znająca choć trochę Accessa może utworzyć bazę danych, ale baza danych ze zbiorem niepowiązanych ze sobą tabel, kwerend, formularzy i raportów to nie aplikacja. Aplikacja składa się z bazy danych — lub prawdopodobnie kilku baz danych — zawierających znormalizowane tabele z odpowiednimi związkami między nimi, kwerend filtrujących i sortujących dane, formularzy dodających i edytujących dane, raportów wyświetlających dane i prawdopodobnie tabeli lub wykresów przestawnych analizujących dane. Połączenie wszystkich tych elementów odbywa się dzięki inteligentnie napisanemu kodowi *VBA* (ang. *Visual Basic for Applications*). Niniejszy rozdział skupia się na przygotowaniach do wykonania aplikacji (pobranie i analiza informacji od klienta) i wykonaniu tabel przechowujących dane aplikacji.

Większość książek na temat Accessa przekazuje mnóstwo informacji na temat tabel bazy danych (lub innych obiektów bazodanowych), ale rzadko informuje o sprawach najważniejszych: w jaki sposób podzielić surowe dane uzyskane od klienta na odpowiednie tabele, z jakiego typu danych skorzystać dla poszczególnych pól i jakie związki między tabelami utworzyć, by powstała wydajna i dobrze zintegrowana aplikacja. Dzięki kilku sesjom z klientem — na zasadzie zadawania pytań i odpowiadania na nie — przedstawię, w jaki sposób wydobyć najistotniejsze informacje pomocne przy kreowaniu odpowiednich tabel i związków.

Zawsze łatwiej mi było zrozumieć pewien proces, gdy przyglądałam się, jak robi to ktoś inny. Próba wykonania zadania po przeczytaniu tylko i wyłącznie suchej dokumentacji technicznej jest często znacznie trudniejsza. Z tego powodu w tym rozdziale wyjaśnię, co robię, gdy przygotowuję się do utworzenia aplikacji i jej tabel. Kolejne rozdziały poświęcone będą wykonywaniu formularzy, kwerend i raportów. Oczywiście pewnych informacji technicznych nie da się uniknąć, ale zostaną one przedstawione naprzemiennie z opisami zadań do wykonania. Wyjaśnienia znajdą się na ogół po zadaniach do wykonania, a nie na odwrót. Czasem wystarczy zobaczyć, jak ktoś poprawnie wykonuje pewne zadanie, by móc samemu wykonywać je prawidłowo. Wywody czysto teoretyczne rzadko prowadzą do poprawnego wykonania zadań.

Nie będę w książce przedstawiać wielu zadań krok po kroku, by zilustrować proces tworzenia bazy danych. Nie pojawią się też długie listy właściwości lub innych atrybutów. Zakładam, iż Czytelnik jest zaznajomiony z zasadami tworzenia tabel, formularzy i innych obiektów bazodanowych (a jeśli nawet ich nie zna, to potrafi skorzystać z systemu pomocy programu), ale potrzebuje pomocy w podjęciu decyzji związanych z umieszczeniem w tabelach odpowiednich danych, wzajemnym ich powiązaniu i określeniu, jakie formularze, kwerendy i raporty najlepiej nadadzą się do poprawnego korzystania z bazy danych.

Choć kod jest niezbędnym elementem umożliwiającym połączenie elementów aplikacji w jedną, spójną całość, w tym rozdziale nie pojawi się nawet jeden fragment kodu. Wynika to z faktu, iż Access wykonuje kod z procedur obsługi zdarzeń, a tabele nie posiadają takich procedur. Przed napisaniem procedur trzeba utworzyć tabele przechowujące dane. Jest to podstawowy cel niniejszego rozdziału.

Uzyskiwanie informacji

Aby zacząć tworzyć aplikację w Accessie, trzeba spełnić dwa warunki: należy uświadomić sobie, jakie zadania ma wykonywać aplikacja i jakie wyniki zwracać; trzeba mieć też odpowiednią ilość rzeczywistych danych. Zamiast po prostu poprosić klienta o listę zadań, jakie aplikacja musi przeprowadzić, na ogół sama zadaję serie pytań, dzięki którym potrafię wydobyć najważniejsze dla mnie informacje. Typowa sesja pytań i odpowiedzi została przedstawiona w kolejnym podrozdziale. Jeśli jednak istnieje aktualnie wykorzystywana baza danych i dostępne są wydruki raportów oraz zrzuty ekranowe formularzy, znacznie łatwiej zorientować się, jakie zadania należy obecnie wykonać.

Aby móc osiągnąć najlepsze wyniki, trzeba mieć dostęp do dużych ilości rzeczywistych danych, na przykład informacji o elektronicznych książkach zawartych w bazie danych wykorzystywanej w rozdziale 9 „Modyfikacja istniejącej aplikacji”. Jeżeli ma się dostęp do wystarczająco dużej liczby reprezentatywnych danych i klienta, który chętnie odpowiada na pytania, określenie potrzebnych tabel i ich pól nie powinno nastręczać dużych problemów. Po uzyskaniu tych informacji wykonanie w Accessie tabel i związków między nimi, a także kwerend, formularzy, raportów i kodu VBA powinno doprowadzić do powstania aplikacji, której oczekiwał klient.

Co ciekawe, często jestem proszona o rozpoczęcie pracy nad aplikacją dla klienta, choć nie otrzymałam żadnych danych. Czasem trudno przekonać klienta do przedstawionej wcześniej koncepcji, ale uzyskanie rzeczywistych danych (w formie elektronicznej lub papierowej) jest ważne, by móc poprawnie wykonać tabele i ich pola. Jeśli sami (jako programiści) musimy wymyślić dane, by coś zaczęło działać po utworzeniu tabel i innych obiektów, prawdopodobnie na późniejszym etapie będziemy zmuszeni dokonać — z pewnością bardzo głębokich — zmian w tabelach. Aby tego wszystkiego uniknąć, najlepiej dysponować reprezentatywnymi danymi wprost od klienta.

Istnieją sytuacje, w których uzyskanie danych od klienta jest po prostu niemożliwe. Pierwsza sytuacja dotyczy firmy dopiero rozkręcającej biznes, która nie ma żadnych danych historycznych.

Druga dotyczy firm, których dane są poufne. W takich sytuacjach warto postarać się, by wymyślone dane testowe były zgodne z informacjami od klienta uzyskanymi w trakcie rozmów. Minimalizuje się wtedy ryzyko dokonywania poprawek w przyszłości.

Gdy uzyskało się dane w postaci elektronicznej lub papierowej, najlepiej użyć ich jako surowego materiału, który pomaga w określeniu pól potrzebnych w trakcie projektowania tabel i innych komponentów. Nie warto stosować tych danych jako gotowych komponentów i żywcem kopiować ich struktury do aplikacji. Przyglądając się rzeczywistym danym, nietrudno zauważyć, czy zawierają unikatowe identyfikatory produktów czy nie. Jeśli istnieje taki unikatowy numer produktu, powinien stać się on kluczem głównym tabeli produktów — w przeciwnym razie warto utworzyć pole z automatyczną numeracją. Jeśli klienci mają wiele adresów, trzeba wykonać powiązaną tabelę z danymi adresowymi (jedno z założeń normalizacji). Jeżeli klienci zawsze mają tylko jeden adres, jego dane można w sposób bezpośredni zapamiętać w tabeli klientów. W większości sytuacji, nawet jeśli otrzymało się bazę danych z tabelami Accessa, trzeba dokonać pewnych modyfikacji ze względu na normalizację, utworzyć tabele pomocnicze i słowniki wykorzystywane jako źródła danych dla list rozwijanych.

Rozpoznanie zadań i obiektów biznesowych

W trakcie projektowania aplikacji dla klienta (już po otrzymaniu reprezentatywnych danych) należy omówić modelowane przez nią procesy — nie tylko sposób, w jaki wykonuje się je obecnie, ale też ewentualne sposoby ich optymalizacji lub rozbudowania. Przykładowo — użytkownicy mogą wpisywać dane do pól tekstowych; jeśli dane ze swej natury są mało różnorodne (na przykład regiony sprzedaży lub rodzaje numerów telefonów), lista rozwijana z tabelą powiązaną (statycznie) jako źródłem danych powinna zapewnić zmniejszenie błędów i ułatwić sortowanie oraz filtrację danych.

Klient może przekazać całe sterty dokumentacji papierowej opisującej procesy biznesowe, która może być pomocna lub nie — w zależności od sposobu opisu tych procesów. Często praktyki biznesowe powstają przez lata w sposób przyrostowy wraz ze wzrostem doświadczenia — wtedy pewne nowe procedury mogą nie integrować się ze starymi tak dobrze, jak byśmy chcieli. W momencie projektowania aplikacji warto przejrzeć istniejące procedury i zastanowić się, czy należy je zmodyfikować w celu zwiększenia wydajności.

Nie warto próbować wiernie odwzorowywać procesów biznesowych w bazie danych — a przynajmniej nie bez ich wnikliwej analizy. Dokładne przyjrzenie się procedurom często prowadzi do zauważenia pewnych luk, którymi należy zająć się na poziomie bazy danych. To, że użytkownicy ręcznie pisali listy do klientów w programie Word i spisywali na kartkę ich adresy z innej bazy danych, nie oznacza od razu, iż nie należy przeanalizować możliwości automatycznego generowania listów dla Worda. (Więcej informacji na temat generowania listów dla Worda znajduje się w rozdziale 11, „Korzystanie z Worda”.).

Czasem zauważa się, że aplikacja powinna zapewniać procesy, które w ogóle nie są wykonywane, a byłyby bardzo wskazane, na przykład generowanie listów e-mail do klientów lub analiza danych za pomocą tabel i wykresów przestawnych. Więcej informacji na temat wysyłania listów e-mail do klientów z Accessa znajduje się w rozdziale 12, „Korzystanie z Outlooka”. Ponieważ aplikacja musi mieć gdzie przechowywać te wszystkie dane, jednym z pierwszych zadań projektanta jest określenie wymaganych tabel bazy danych.

Określenie encji

Pierwszym zadaniem w trakcie opracowywania bazy danych jest określenie jej elementów i zasad ich wzajemnej współpracy. (W literaturze bazodanowej często stosuje się termin techniczny — **encja**, ale my możemy pozostać przy mniej technicznym określeniu — **element**). Być może klient korzysta obecnie z innej bazy danych. W zależności od osoby, która tworzyła oryginalną bazę danych, może się to okazać większą przeszkodą niż pomocą.

Jako przykład sposobu określania elementów, z którymi aplikacja musi sobie radzić, prześledzimy spotkanie z hipotetycznym klientem żądającym dostarczenia aplikacji pomocnej w jego biznesie (firmie zabawkarskiej). Oto kilka podstawowych pytań zadawanych klientowi.

P: Czym się zajmujecie?

O: Sprzedajemy zabawki.

Potrzebujemy tabeli z zabawkami.

P: Czy dla każdej zabawki macie numer lub identyfikator produktu?

O: Tak, kombinację liter i cyfr.

Potrzebujemy pola tekstowego `ToyID` *jako klucza głównego dla tabeli* `tblToys`.

P: Czy produkujecie zabawki, czy może kupujecie je od dostawców, a następnie sprzedajecie?

O: Produkujemy własne i sprzedajemy zabawki innych producentów.

Potrzebujemy tabeli materiałów potrzebnych do produkcji zabawek. Być może konieczne okażą się dwie tabele — jedna dla zabawek zakupionych w celu odsprzedaży i jedna dla zabawek własnej produkcji. Trzeba określić, czy te dwa typy zabawek na tyle różnią się od siebie, by tworzyć różne tabele, czy też mogą znaleźć się w tej samej tabeli, ale z różnymi wartościami w kilku polach i z zastosowaniem osobnej tabeli materiałów dla zabawek własnej produkcji.

P: Jakie są różnice między zabawkami własnej produkcji a kupowanymi?

O: Dla zakupionych zabawek musimy przechowywać nazwę dostawcy, numer produktu stosowany przez dostawcę, cenę i datę zakupu. Dla produkowanych zabawek musimy wiedzieć, ile potrzeba materiału do jego produkcji, jakie są koszty produkcji i kiedy dochodzi do wykonania danego rodzaju zabawek.

Wygląda na to, że wystarczy jedna tabela zabawek z polem określającym, czy dana zabawka została zakupiona, czy może wyprodukowana. Pole służyłoby później do włączania lub wyłączania różnych formantów formularza. Potrzebujemy też tabeli dostawców używanej do określania zawartości pola nazwy dostawcy i tabeli materiałów w celu określania materiałów, z których powstaje zabawka.

P: Czy materiały kupujecie od innych dostawców niż zabawki do odsprzedaży, czy może jeden dostawca sprzedaje wam materiały i gotowe zabawki?

O: Większość naszych dostawców sprzedaje nam wyłącznie gotowe zabawki. Część sprzedaje tylko materiały. Jedynie kilku sprzedaje zarówno zabawki, jak i materiały.

Wszystkich dostawców można przechowywać w jednej tabeli z odpowiednimi polami typu Tak/Nie, wskazującymi, czy sprzedają gotowe zabawki, czy materiały, czy oba te elementy.

P: Własne zabawki tworzycie w warsztacie lub fabryce, czy też zlecacie ich wytwarzanie innej firmie?

O: Wykonujemy je we własnym warsztacie.

Nie jest potrzebna tabela zleceniobiorców.

P: Czy macie jeden, czy może kilka warsztatów?

O: Tylko jeden.

Nie jest potrzebna tabela powiązana dla warsztatów.

P: Czy wykonujecie inne prace poza sprzedażą zabawek?

O: Tak, naprawiamy zepsute zabawki.

Potrzebujemy tabeli napraw.

P: Naprawiacie tylko własne zabawki? Czy może reperujecie także te wyprodukowane przez firmy trzecie?

O: Naprawiamy własne zabawki i podobne innych producentów.

Nie możemy identyfikować naprawianych zabawek po prostu po polu `ToyID`. *Potrzebujemy automatycznie numerowanego pola, które jednoznacznie zidentyfikuje zabawki zakupione lub wyprodukowane w innym miejscu, a następnie oddane do naprawy.*

P: Czy naprawy wykonujecie we własnym zakresie, czy zlecacie je innym firmom?

O: Wykonujemy we własnym zakresie.

Potrzebujemy tabeli pracowników z możliwością określenia, którzy z nich zajmują się naprawami.

P: Czy wysyłacie katalogi lub inne materiały promocyjne?

O: Tak.

Potrzebujemy tabeli klientów oraz tabeli potencjalnych klientów.

P: Wysyłacie reklamy pocztą zwykłą czy elektroniczną, a może oboma sposobami?

O: Oboma.

Tabela listy mailingowej powinna zawierać adresy korespondencyjne i adresy e-mail.

P: Czy sprzedajecie zabawki w sklepie, w sposób wysyłkowy, czy przez internet?

O: W sklepie firmowym i w sposób wysyłkowy (zamówienia listowne lub telefoniczne). Na razie nie sprzedajemy przez internet, ale planujemy taką możliwość w przyszłości.

Potrzebujemy tabeli zamówień z polem określającym sposób sprzedaży. Klienci powinni być wybierani z tabeli klientów z możliwością wprowadzenia nowego klienta „w locie" w trakcie składania zamówienia. Ponieważ zamówienia listowne i telefoniczne będą wymagały adresu dostawy i kupującego, potrzebujemy osobnej tabeli z adresami dostawy.

Po tej prostej serii pytań i odpowiedzi wiemy już, że aplikacja potrzebuje tabel dla następujących elementów (są to encje aplikacji):

- zabawki,
- kategorie,
- dostawcy,
- klienci,
- adresy dostaw,
- lista mailingowa,
- materiały,
- naprawy,
- pracownicy,
- zamówienia.

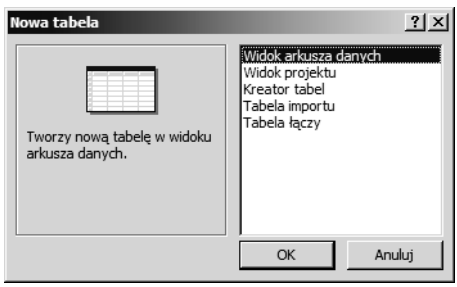
Dodatkowo potrzebnych będzie kilka powiązanych tabel przechowujących dane związane z głównymi tabelami. Co więcej, pojawią się tak zwane słowniki (tabele powiązane) wymagane do zapewnienia spójności danych. Stosuje się je do wypełniania rozwijanych list wyboru.

Tworzenie tabel dla aplikacji

Gdy ma się część informacji od klienta, można rozpocząć tworzenie tabel i określanie związków między nimi, zamieniając zbiór chaotycznych danych w znormalizowane tabele reprezentujące elementy bazy danych. Korzystając z listy tabel uzyskanej dzięki sesjom pytań i odpowiedzi od klienta, zacznijmy tworzyć aplikację dla sklepu z zabawkami. Zastosowanie jednej konwencji nazewnictwa od samego początku znacząco ułatwi późniejsze prace nad aplikacją. Używam konwencji nazewnictwa Leszynskiego (LNC), która została dokładniej opisana w rozdziale 9, „Modyfikacja istniejącej aplikacji". Dla tabel przedrostkiem LNC jest `tbl`, więc zaczynają się od niego wszystkie nazwy tabel.

Sposoby tworzenia tabel

Aby utworzyć nową tabelę, kliknij przycisk *Nowy* z okna *Baza danych*, gdy na pasku obiektów po lewej stronie wybrany jest obiekt *Tabele*. Okno dialogowe *Nowa tabela*, przedstawione na rysunku 1.1, zawiera kilka propozycji.



Rysunek 1.1.

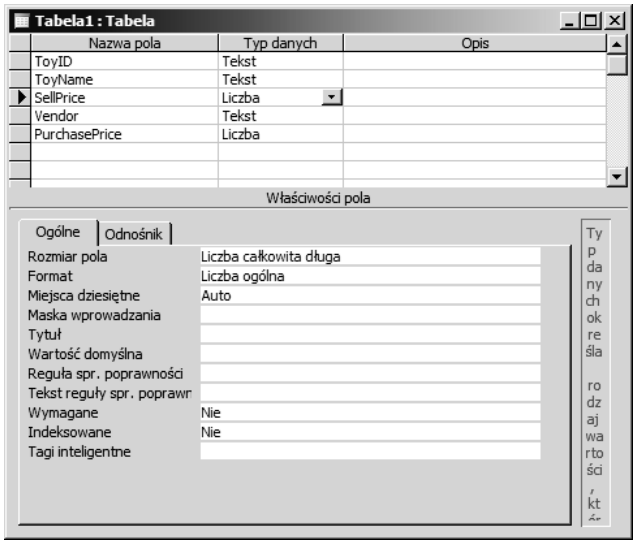
W większości przypadków najlepiej jest wybrać opcję *Widok projektu* i zacząć tworzyć pola tabeli. Pozostałe opcje przydają się w pewnych szczególnych sytuacjach. Warto rozważyć, czy tabela jest jedną z tabel standardowych (wtedy przydatnym skrótem staje się opcja *Kreator tabel*), czy raczej nie (wtedy najlepszy wybór to *Widok projektu*). Wszystkie opcje zostały dokładniej omówione w kolejnych podrozdziałach.

- Opcja *Widok arkusza danych* — ten wybór nie ma zbyt wiele do zaoferowania. Nowa tabela zostaje otwarta w widoku arkusza danych i można rozpocząć wpisywanie danych do pierwszego wiersza. Aby nadać nazwy polom w tym widoku, trzeba kliknąć je kilka razy (początkowo zawierają nazwy Pole*n*) aż do jego zaznaczenia, a następnie wpisać nową nazwę. Jest to znacznie mniej wygodne rozwiązanie od podawania nazw pól w widoku projektu. Access zgaduje typ danych na podstawie informacji wpisanych w pierwszym wierszu. Czasem się myli, więc i tak trzeba poprawiać typy pól w widoku projektu. Na przykładowej tabeli przedstawionej na rysunkach 1.2 i 1.3 po przejściu do widoku projektu nietrudno zauważyć, iż pole `ToyID` nie zostało rozpoznane jako pole klucza, natomiast pola przechowujące ceny są typu *Liczba* zamiast *Walutowy*.



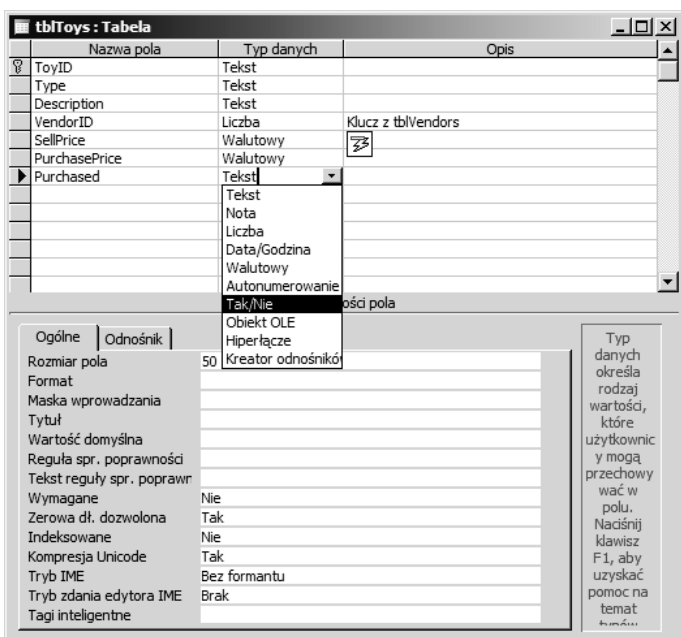
Rysunek 1.2.

Gdy tworzy się nową tabelę w widoku arkusza danych, wpisanie tekstu w polu powoduje utworzenie pola typu *Tekst*; wpisanie liczby powoduje utworzenie pola typu *Liczba*; wpisanie liczby ze znakiem dolara lub dopisanie tekstu „zł" (w zależności od ustawień regionalnych) powoduje powstanie pola typu *Walutowy*; wpisanie daty w rozpoznawalnym formacie spowoduje powstanie pola typu *Data/Godzina*. Jeśli trzeba utworzyć pole innego typu (na przykład *Tak/Nie*, *Nota* lub *Obiekt OLE*), trzeba sięgnąć do widoku projektu.



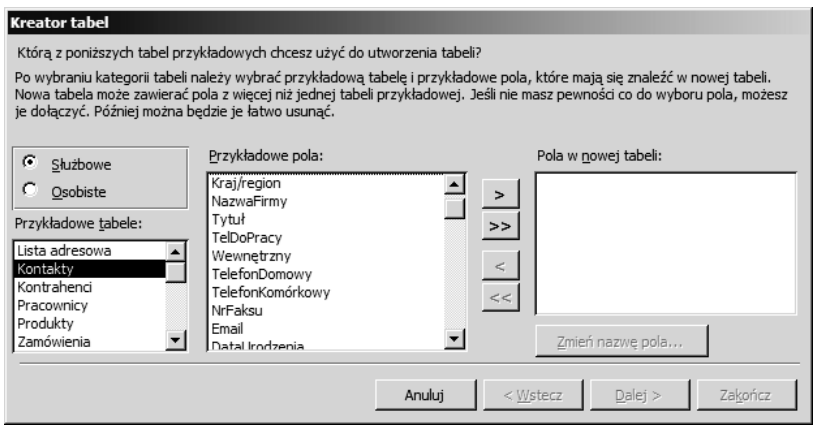
Rysunek 1.3.

- Opcja *Widok projektu* — najlepszy wybór dla niestandardowych tabel, które trzeba tworzyć od podstaw. Tabela zostaje otwarta w widoku projektu, co umożliwia wprowadzenie nazwy każdego pola w osobnym wierszu i wybranie odpowiedniego typu z rozwijanej listy *Typ danych* (patrz rysunek 1.4).



Rysunek 1.4.

- Opcja *Kreator tabel* — przydatna jako skrót przy tworzeniu standardowych tabel, na przykład tabeli nazw klientów i danych adresowych. Do tabel tych należy podchodzić z rezerwą, gdyż często nie są one znormalizowane. Przykładowo — tabela *Kontakty* przedstawiona na rysunku 1.5 zawiera kilka pól z numerami telefonów (prawdopodobnie w celu dopasowania jej do listy kontaktów programu Outlook), co w zależności od kontaktu może prowadzić do utworzenia zbyt wielu lub zbyt małej liczby pól z numerami telefonów. Poza pewnymi rzadkimi wyjątkami numery telefonów i identyfikatory powinny znaleźć się w powiązanej tabeli, która pozwala utworzyć dokładnie tyle elementów, ile jest aktualnie potrzebnych.



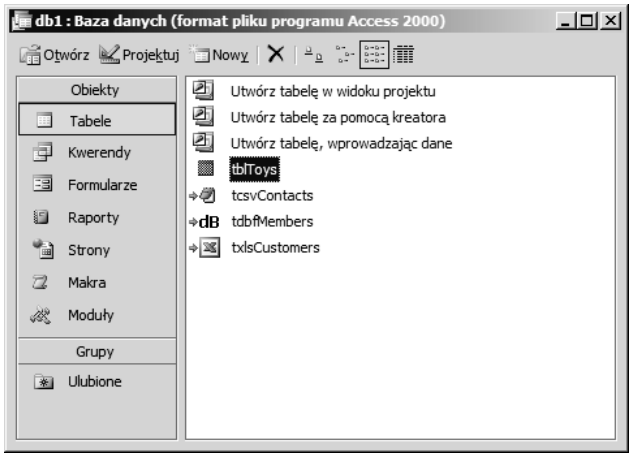
Rysunek 1.5.

- Opcja *Tabela importu* — umożliwia zaimportowane danych z zewnętrznego źródła do tabeli Accessa. Po dokonaniu importu warto sprawdzić pola pod kątem poprawności typów i, być może, podzielić dane na kilka powiązanych i znormalizowanych tabel.

- Opcja *Tabela łączy* — wiąże tabelę Accessa z danymi z innego programu, na przykład Excela. Tabele łączy nie są tak przydatne jak inne tabele, ponieważ nie można zmienić ich struktury. Warto je stosować tylko wtedy, gdy potrzebuje się szybkiego widoku aktualnych danych przechowywanych w innym programie.

  W oknie *Baza danych* tabele łączy mają strzałkę po lewej stronie nazwy tabeli i odpowiednią ikonę dla każdego typu danych. Rysunek 1.6 przedstawia trzy tabele łączy — jedną jest plik tekstowy z danymi rozdzielonymi przecinkami, drugą jest plik dBASE, a trzecią plik arkusza programu Excel. Użyłam znacznika `tcsv` dla pliku tekstowego, `tdbf` dla pliku dBASE i `txls` dla pliku Excela. Dzięki temu wiem, z jakim rodzajem pliku mam do czynienia, gdy nie widzę ikon z okna *Baza danych*.

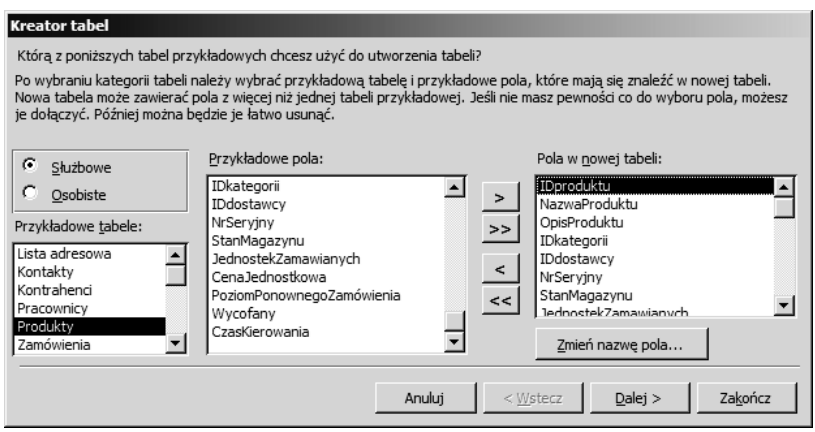
Tabela macierzysta to tabela zawierająca dane w Accessie; zdecydowana większość tabel, z którymi ma się do czynienia, to tabele macierzyste. Gdy tworzy się nową tabelę w Accessie, jest to tabela macierzysta. Podobnie dane importowane z zewnętrznego programu do Accessa również zostają umieszczone w tabeli macierzystej. Poza tymi tabelami istnieją również **tabele łączy**, które pozwalają pracować z danymi z innych programów, na przykład Excela lub dBASE.



Rysunek 1.6.

Tworzenie tabel

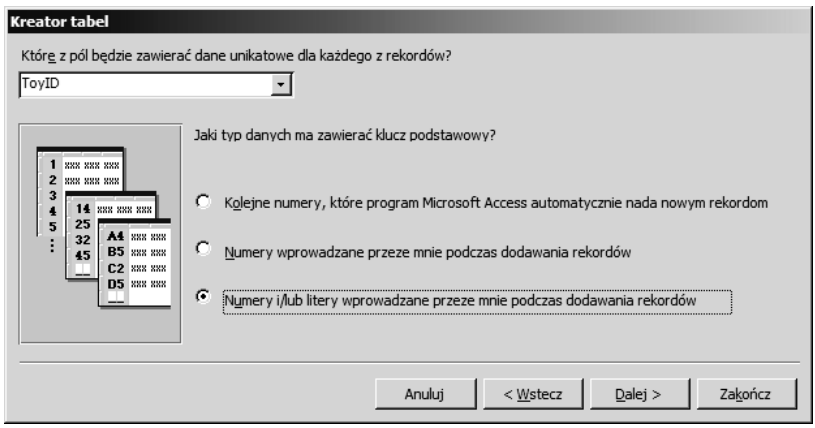
Zacznę od tabeli `tblToys`, która będzie główną tabelą przechowującą informacje o zabawkach sprzedawanych klientom (oraz dla nich produkowanych). Ponieważ kreator tabel oferuje tabelę *Produkty*, zacznijmy od niej i zmodyfikujmy ją według własnych potrzeb. Rysunek 1.7 przedstawia tabelę *Produkty* w kreatorze tabel. Wybrałam większość standardowych pól, aby ułatwić sobie wykonanie tabeli zabawek.



Rysunek 1.7.

Okno *Kreator tabel* zawiera przycisk umożliwiający zmianę nazwy pola. Zmianę nazwy można również przeprowadzić na późniejszym etapie, otwierając widok projektu i modyfikując zawarte tam pola. Po wybraniu pól (i ewentualnej zmianie ich nazw) kliknij przycisk *Dalej*, aby przejść do następnego etapu kreatora, w którym podaje się nazwę tabeli (w tym przypadku `tblToys`) i określa, czy klucz podstawowy ma zostać wybrany przez Accessa, czy sami go określimy. Wybierz opcję *Nie, samodzielnie ustawię klucz podstawowy*, ponieważ chcemy mieć pełną kontrolę nad wyborem pola klucza.

Ponowne kliknięcie przycisku *Dalej* powoduje wyświetlenie informacji, iż kreator poprawnie zakłada, że pole `ToyID` powinno być polem klucza. Dodatkowo daje do wyboru trzy możliwe rozwiązania dotyczące klucza (patrz rysunek 1.8). Należy zaznaczyć trzecią opcję, ponieważ pole identyfikatora zabawki będzie zawierać kombinację cyfr i liter. (Pierwsza opcja oznacza pole numerowane automatycznie, druga opcja dotyczy identyfikatora liczbowego).



Rysunek 1.8.

Ponowne kliknięcie przycisku *Dalej* powoduje wyświetlenie zapytania, czy chce się dołączyć nową tabelę do innych tabel bazy danych. Ponieważ jest to nowa tabela, po prostu jeszcze raz kliknij przycisk *Dalej*. Z ostatniego ekranu kreatora wybierz opcję *Modyfikuj projekt tabeli*, aby otworzyć nową tabelę w widoku projektu, dzięki któremu można uszczegółowić jej strukturę.

> *Jeśli nie ustawi się relacji między tabelami w kreatorze tabel, zawsze można zrobić to później w oknie relacji — w zasadzie wiele osób preferuje wykonywanie tego zadania właśnie w tym oknie, ponieważ dostarcza ono znacznie bardziej intuicyjny interfejs.*
>
> *Pierwszy krok polega na ustaleniu maski wprowadzania dla pola ToyID, aby zapewnić wpisywanie w nim danych zgodnych ze specyfikacją uzyskaną od klienta. Aby utworzyć maskę wprowadzania, można kliknąć przycisk z wielokropkiem po prawej stronie pola tekstowego maski (otworzy się wtedy kreator maski) lub też od razu wpisać szablon maski. Ponieważ kreator masek nie zawiera standardowo odpowiedniej opcji, musimy wpisać maskę w sposób bezpośredni. Poniższa tabela przedstawia znaki, z jakich można skorzystać w celu ograniczenia wartości możliwych do wpisania w polu tekstowym.*

Przypuśćmy, że dla przykładowej tabeli zabawek klient poinformował o stosowaniu dwóch wielkich liter i trzech cyfr do identyfikacji poszczególnych zabawek. Potrzebujemy więc znaku `>`, by zamienić wpisywane litery na wielkie, następnie dwa znaki `L` i trzy zera. Pole maski wprowadzania powinno zatem zawierać poniższy tekst:

```
>LL000
```

Znak maski	Dopuszczalne znaki wejściowe
0	Wymagana cyfra od 0 do 9; znaki plus i minus nie są dozwolone.
9	Opcjonalna cyfra od 0 do 9 lub spacja; znaki plus i minus nie są dozwolone.
#	Opcjonalna cyfra lub spacja; puste miejsca zostają skonwertowane na spacje; znaki plus i minus są dozwolone.
L	Wymagana litera od A do Z lub od a do z.
?	Opcjonalna litera od A do Z lub od a do z.
A	Wymagana litera lub cyfra.
a	Opcjonalna litera lub cyfra.
&	Wymagany znak lub spacja.
C	Opcjonalny znak lub spacja.
.,:;- /	Znaki ułamka, separatora tysięcy, wartości daty i czasu — używane znaki zależą od ustawień regionalnych z panelu sterowania.
<	Konwertuje następny znak na małą literę.
>	Konwertuje następny znak na wielką literę.
!	Powoduje wyświetlanie maski wprowadzania od prawej do lewej zamiast od lewej do prawej. W pewnych wersjach pakietu Office ten przełącznik nie działa poprawnie. Warto zajrzeć do bazy wiedzy firmy Microsoft (KB) do artykułu o numerze 209049 („Input Mask Character (!) Does Not Work as Expected"), który zawiera wyjaśnienie problemu dla Accessa 2000. Artykuł można przejrzeć, odwiedzając witrynę *http://support.microsoft.com*.
\	Informuje, że następny znak należy traktować dosłownie.
PASSWORD	Tworzy pole tekstowe do wpisywania hasła — wpisany tekst jest zapamiętywany poprawnie, ale w ramach bezpieczeństwa zamiast znaków zostają wyświetlone gwiazdki.

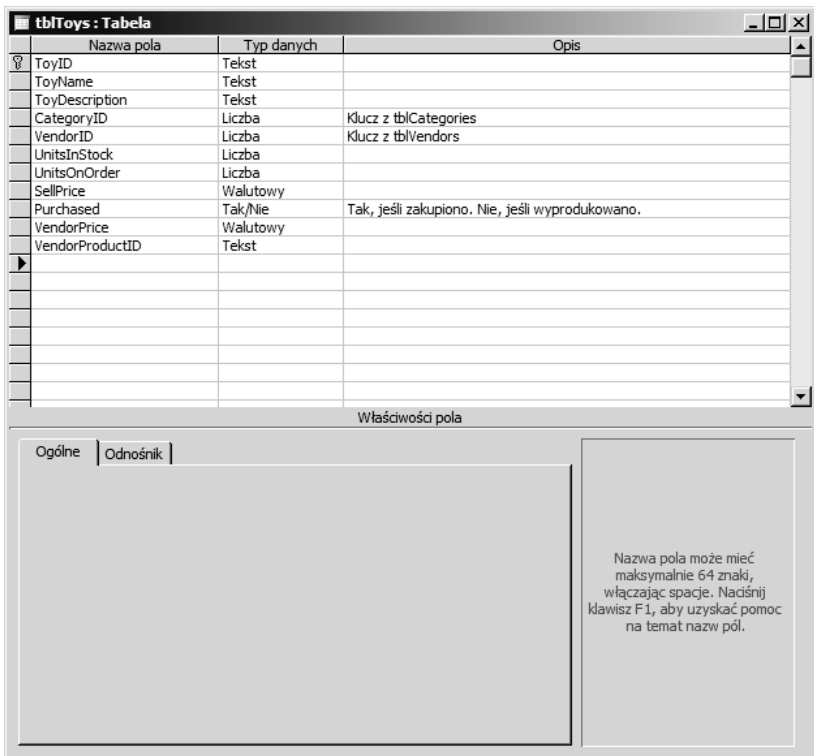
Poza standardowymi polami z kreatora tabel potrzebujemy kilku dodatkowych pól przechowujących dane związane z produkowanymi zabawkami. Rysunek 1.9 przedstawia tabelę z dodatkowymi polami. W tabeli pojawia się kolejny identyfikator, `VendorProductID`, ale nie stosujemy dla niego maski, gdyż różni dostawcy stosują odmienne formaty identyfikatorów.

Informacje na temat materiałów nie są przechowywane w tej tabeli, ale w innej, która dopiero zostanie utworzona.

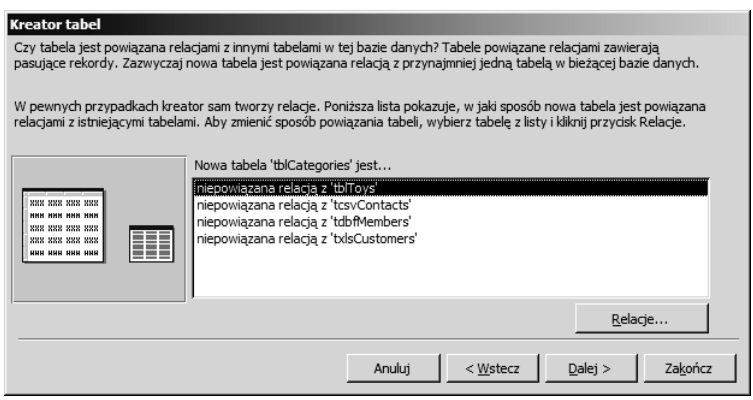
Kolejny krok to utworzenie tabel `tblVendors` i `tblCategories`, które zostaną powiązane z tabelą `tblToys`. Po wybraniu standardowej tabeli kategorii z kreatora tabel kliknij przycisk *Relacje*, aby powiązać tę tabelę z polem `CategoryID` tabeli `tblToys` (patrz rysunek 1.10).

Wybierz środkową opcję w oknie kolejnego kreatora (patrz rysunek 1.11), ponieważ jedna kategoria zabawek będzie dotyczyła wielu rekordów z tabeli `tblToys`.

Teraz okno kreatora tabeli informuje, że tabela `tblCategories` jest powiązana relacją z tabelą `tblToys` (patrz rysunek 1.12).



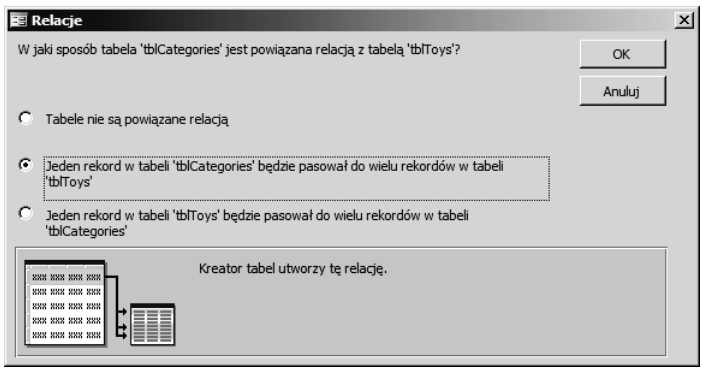
Rysunek 1.9.



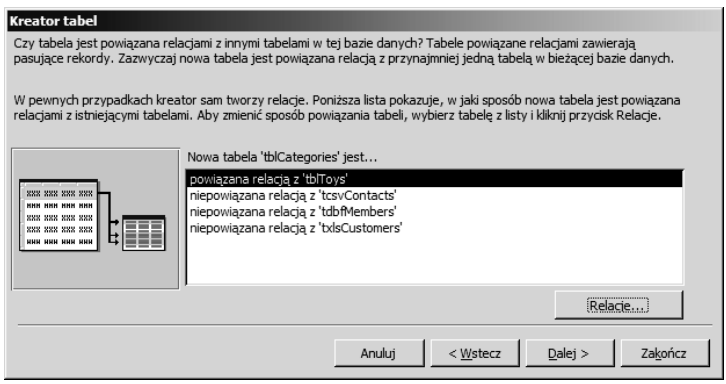
Rysunek 1.10.

Istnieje relacja między tabelami `tblCategories` i `tblToys`, którą można zobaczyć w oknie relacji (patrz rysunek 1.13).

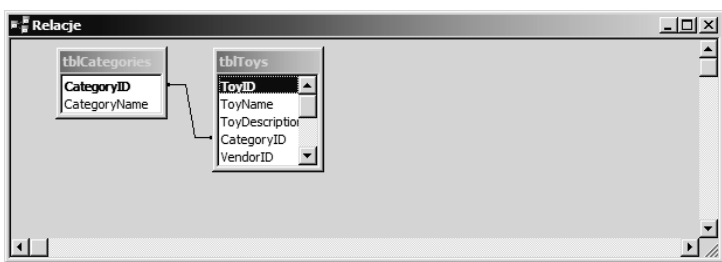
Choć w oknie relacji kreatora tabel została wybrana opcja *Jeden rekord w tabeli 'tblCategories' będzie pasował do wielu rekordów tabeli 'tblToys'*, tak naprawdę nie powstał związek jeden do wielu między tymi tabelami, mimo że tak właśnie powinno się stać. Jest to jeden



Rysunek 1.11.



Rysunek 1.12.



Rysunek 1.13.

z powodów, dla których relacje warto określać w oknie relacji, a nie w kreatorze. W dalszej części rozdziału dokładnie przedstawię, w jaki sposób zmodyfikować typy relacji ustawione przez kreatora tabel.

Powróćmy do tworzenia tabel. Tabelę `tblVendors` można utworzyć, korzystając z kreatora i predefiniowanej tabeli dostawcy. Następnie należy połączyć tę tabelę z polem `VendorID` tabeli `tblToys`. Kreator nie wykonuje wszystkich zadań wymaganych do poprawnego powiązania tabel. Jeśli chce się ustawić tabele `tblCategories` i `tblVendors` jako surowe źródła słownikowe, aby wartości były wybierane tylko z powiązanych tabel, trzeba to zrobić samemu. Więcej informacji na ten temat znajduje się w kolejnym podrozdziale.

Typy danych pól tabeli

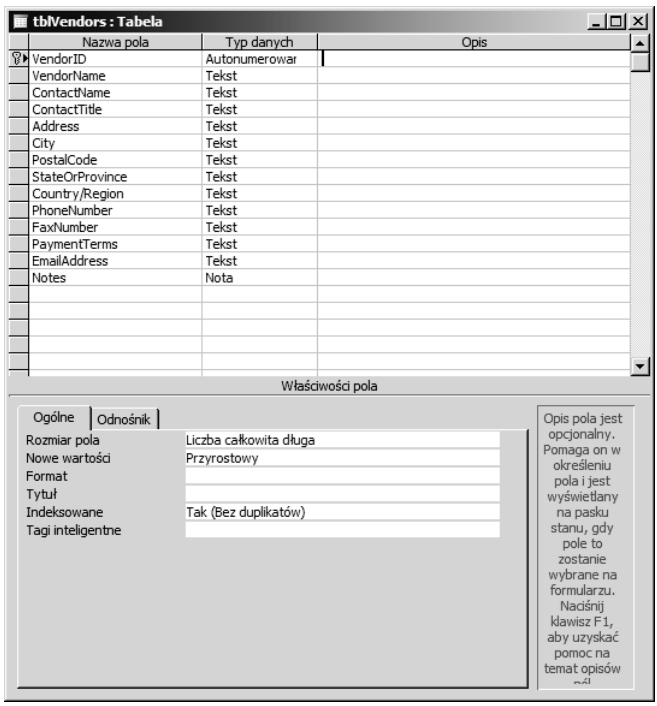
W trakcie tworzenia pól tabeli należy określić odpowiedni typ danych dla każdego pola, aby zapewnić wprowadzanie w danym polu jedynie poprawnych informacji (nie uda się wpisać tekstu do pola numerycznego!) i ułatwić sortowanie oraz filtrację. Poniższa tabela wymienia typy danych dostępne w Accessie wraz z krótkim komentarzem. Główny typ danych to typ, który widać na liście rozwijanej w trakcie tworzenia lub edycji pola; niektóre typy (na przykład *Liczba* i *Autonumerowanie*) mają podtypy, które ustawia się za pomocą właściwości *Rozmiar pola*.

Główny typ pola	Podtypy	Opis	Komentarz
Tekst		Dane tekstowe do 255 znaków	Używany dla danych tekstowych i liczb (na przykład identyfikatorów), które nie są używane w obliczeniach.
Nota		Blok tekstu zawierający do 65 535 znaków	Używany dla długich tekstów; pole to udostępnia jedynie ograniczone sortowanie (używanych jest w nim tylko 255 pierwszych znaków).
Liczba	Bajt	Liczba całkowita w zakresie od 0 do 255	Niewielkie liczby całkowite.
	Liczba całkowita	Liczba całkowita w zakresie od –32 768 do 32 767	Średniej wielkości liczby całkowite.
	Liczba całkowita długa	Liczba całkowita w zakresie od –2 147 483 648 do2 147 483 647	Długie liczby całkowite. Jest to domyślny typ. Odpowiada polu autonumerowania przy powiązanych tabelach.
	Pojedyncza precyzja	Wartości od –3,402823E38 do –1,401298E–45 dla liczb ujemnych i od 1,401298E–45 do 3,402823E38 dla liczb dodatnich	Dokładny do 7 miejsc po przecinku.
	Podwójna precyzja	Wartości od –1,79769313486231E308 do –4,94065645841247E–324 dla liczb ujemnych i od 4,94065645841247E–324 do 1,79769313486231E308 dla liczb dodatnich	Dokładny do 15 miejsc po przecinku.
	Identyfikator replikacji	Globalnie unikatowy identyfikator (GUID). Szesnastobajtowe pole używane jako unikatowy identyfikator przy replikacji baz danych	Używany jedynie w zreplikowanych bazach danych.

Główny typ pola	Podtypy	Opis	Komentarz
	Dziesiętne	Wartości od –10^28–1 do 10^28–1	Dokładny do 28 miejsc po przecinku.
Data/Godzina		Data lub czas	Daty zawsze warto przechowywać w polu tego typu, ponieważ umożliwia to wykonywanie na nich obliczeń.
Walutowy		Wartości walutowe lub liczby wymagające dużej dokładności w obliczeniach	Pole to pozwala pozbyć się efektu zaokrągleń w trakcie obliczeń. Pole jest dokładne do 15 miejsc przed przecinkiem i do 4 miejsc po przecinku.
Auto-numerowanie	Liczba całkowita długa	Automatycznie zwiększana wartość używana jako unikatowy identyfikator rekordu	Taki sam typ danych jak Liczba całkowita długa. Używany w tworzeniu relacji. Mogą pojawić się w dziury w numeracji, jeśli rekord został utworzony, a później usunięty.
	Identyfikator replikacji	Losowe wartości używane jako unikatowy identyfikator rekordu	Jest to bardzo długi i dziwnie wyglądający tekst.
Tak/Nie		Dane, które mogą przyjmować wartość „prawda" lub „fałsz"	Nie są dopuszczalne wartości puste.
Obiekt OLE		Dokumenty utworzone w programach wspierających mechanizm OLE (na przykład Word lub Excel)	Obiekt nie będzie widoczny w tabeli. Aby go wyświetlić, trzeba użyć formularza lub raportu. Po tym polu nie można sortować. Nie da się utworzyć dla niego indeksu.
Hiperłącze		Ścieżki URL lub UNC	Kliknięcie wartości tego pola powoduje otwarcie strony WWW (jeśli łącze jest poprawne).
Kreator odnośników		Nie jest to osobny typ danych, ale kreator umożliwiający określenie tabeli lub listy wartości, z których będzie pobierana wartość pola	Gdy pole zostanie ustawione na ten typ, w widoku arkusza danych nie będzie widoczna jego wartość.

Niezależnie, czy korzystam z kreatora, czy widoku projektu, nie stosuję pól stworzonych kreatorem odnośników. Zamiast tego używam tabeli powiązanej w listach rozwijanych, ustawiając odpowiednią właściwość dla tych formantów na formularzu. Powód takiego postępowania wynika z faktu, iż zastosowanie kreatora odnośników (na przykład dla `VendorID`) uniemożliwia wyświetlenie wartości tego pola w widoku arkusza danych — zostanie wyświetlona nazwa dostawcy z tabeli powiązanej. Aby zobaczyć identyfikator dostawcy, trzeba zmienić typ pola. Co więcej, typ danych kreatora odnośników zawsze powoduje zastosowanie listy rozwijanej na formularzu, choć czasem sytuacja wymaga zastosowania pola tekstowego, ponieważ wartość jest tylko odczytywana.

Rysunek 1.14 przedstawia tabelę `tblVendors` wykonaną przy użyciu kreatora tabel.



Rysunek 1.14.

> Można użyć klawisza *F6* jako skrótu do przechodzenia między polem tabeli w widoku projektu a zbiorem właściwości. Ten klawisz jest szczególnie przydatny, gdy ustawia się właściwość rozmiaru pola dla pól numerycznych.

Ponieważ klient powiedział, że dostawcy mogą sprzedawać zarówno gotowe zabawki, jak i materiały do ich produkcji, tabela potrzebuje dwóch pól typu *Tak/Nie*, aby móc wskazać, czy dostawca sprzedaje zabawki, materiały, czy obie te rzeczy naraz. Dodałam więc te pola, ustawiłam domyślną wartość pola `SellsToys` na *Tak* (ponieważ klient poinformował, iż większość dostawców sprzedaje zabawki) i domyślną wartość pola `SellsMaterials` na *Nie*.

Choć Access umożliwia stosowanie spacji (i większości znaków przestankowych) w nazwach pól (zauważ ukośnik w nazwie pola `Country/Region` *z tabeli* `tblVendors`*), osobiście staram się unikać spacji i innych nietypowych znaków (poza podkreśleniem), aby uniknąć problemów z określaniem pól w kodzie i poleceniach SQL, a także w celu zapewnienia zgodności w trakcie eksportu danych z tabel do innych programów, które mogą nie dopuszczać spacji w nazwach pól.*

Często w trakcie tworzenia tabel zdaję sobie sprawę, iż muszę klientom zadać dodatkowe pytania. Bardzo rzadko zna się wszystkie potrzebne informacje po pierwszym spotkaniu. Warto spotykać się z klientem od czasu do czasu, by rozwiać wątpliwości co do przyszłej struktury aplikacji. Na tym etapie pojawiło się pytanie, czy wszyscy dostawcy mają swoje siedziby w Stanach Zjednoczonych. Gdyby tak było, można by pozbyć się pola `Country/Region` *i nałożyć odpowiednie maski na skróty nazw stanów i kody pocztowe. Jeśli jednak dostawcy pochodziliby z wielu krajów, pole krajów musiałoby pozostać (ale usuwam z niego ukośnik). Poza tym*

pola `PostalCode` *i* `StateOrProvince` *pozostają bez masek wprowadzania lub też stosuję dla nich maski na poziomie formularzy (albo stosuję odpowiednie procedury zdarzeń, by uniemożliwić wpisanie błędnych danych).*

P: Czy wszyscy dostawcy znajdują się w Stanach Zjednoczonych? Czy niektórzy znajdują się w innych krajach?

O: Niektórzy dostawcy pochodzą spoza USA.

Należy pozostawić pola w oryginalnej postaci.

Pojawiają się również kolejne pytania.

P: Czy potrzebujecie tylko jednego numeru telefonu i jednego numeru faksu? Czy dostawcy mogą mieć więcej niż jeden numer telefonu? Jak wygląda sprawa z adresami e-mail?

O: Niektórzy dostawcy mają numery telefonów komórkowych i wiele adresów e-mail.

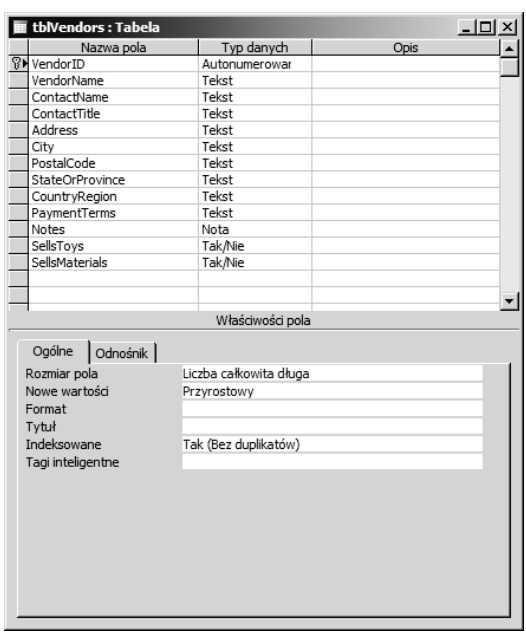
Trzeba usunąć pola telefonu i adresu e-mail z tabeli `tblVendors` *i utworzyć osobne tabele powiązane przechowujące te informacje.*

P: Czy wysyłacie listy w formacie Worda do przedstawicieli dostawców?

O: Nie. Nazwiska przedstawicieli przechowujemy tylko po to, by wiedzieć, o kogo prosić, gdy dzwonimy do dostawcy.

Nie musimy więc umieszczać imienia i nazwiska przedstawiciela w osobnych polach, co byłoby konieczne, gdyby wysyłało się do niego listy.

Rysunek 1.15 przedstawia ostateczną wersję tabeli `tblVendors`.



Rysunek 1.15.

Powiązane tabele wymagają jedynie pola `VendorID` oraz odpowiednio pól opisu telefonu i numeru telefonu oraz adresu e-mail. Pole `VendorID` powinno być typu *Liczba całkowita długa*, aby pasowało do typu pola `VendorID` z tabeli `tblVendors`. W momencie zapisu nie należy tworzyć pola klucza; `VendorID` jest kluczem obcym w tabelach `tblVendorPhones` i `tblVendorEMails` (definicja **klucza obcego** znajduje się w dalszej części rozdziału). Ponieważ dostawcy mogą znajdować się poza Stanami Zjednoczonymi, nie trzeba tworzyć maski wprowadzania dla numerów telefonów. Rysunek 1.16 przedstawia widok projektu obu tabel. Zostaną one powiązane relacją z tabelą `tblVendors` w oknie *Relacje* w dalszej części rozdziału. Tabela `tblVendorPhones` zawiera pole numeru telefonu i dodatkowe pole określające typ numeru (służbowy, domowy, faks itp.). Dla każdego dostawcy będzie można określić dowolną liczbę numerów telefonów.



Rysunek 1.16.

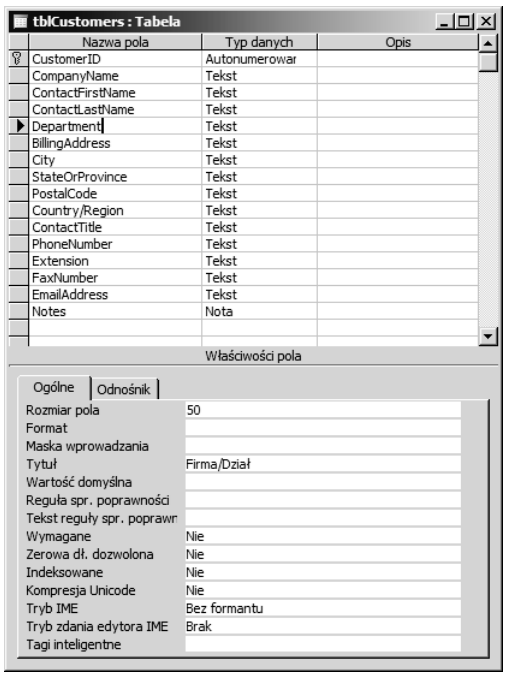
Kontynuujemy prace nad tabelami aplikacji dla firmy zabawkarskiej. Tabelę `tblCustomers` można utworzyć na podstawie przykładowej tabeli *Kontrahenci* z kreatora tabel. Wystarczy jedynie zmienić nazwy poszczególnych pól w sposób przedstawiony na rysunku 1.17.

Podobnie jak w przypadku tabeli dostawców, także teraz mamy kilka dodatkowych pytań do klienta.

P: Czy potrzebujecie tylko jednego numeru telefonu i jednego numeru faksu? Czy dostawcy mogą mieć więcej niż jeden numer telefonu? Jak wygląda sprawa z adresami e-mail?

O: Niektórzy dostawcy mają numery telefonów komórkowych i wiele adresów e-mail. W zasadzie niektórzy klienci mają nawet własne witryny internetowe.

Trzeba usunąć pola telefonu i adresu e-mail z tabeli `tblVendors` *i utworzyć osobne powiązane tabele przechowujące te informacje. Musimy też dodać pole* `WebSite` *typu Hiperłącze.*



Rysunek 1.17.

P: Czy wszyscy klienci są z USA, czy może klientami są również obywatele innych państw?

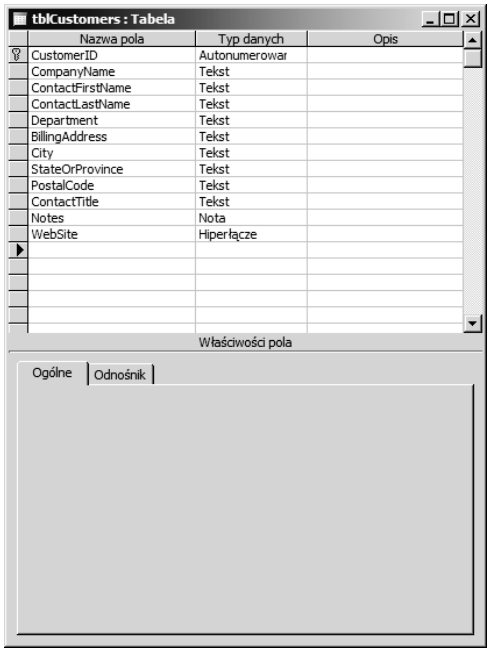
O: Wszyscy klienci są obywatelami Stanów Zjednoczonych.

Możemy usunąć pole `Country/Region`*, a także zastosować odpowiednie maski dla pól* `StateOrProvince` *i* `PostalCode`.

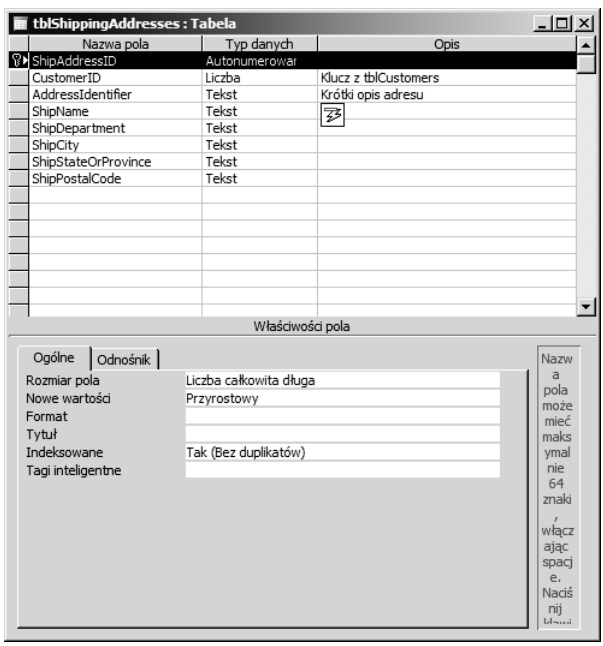
Końcową wersję tabeli `tblCustomers` przedstawia rysunek 1.18.

Tabele `tblCustomerPhones` i `tblCustomerEMails` są bardzo podobne do tabel `tblVendorPhones` i `tblVendorEMails`. Potrzebujemy jeszcze jednej tabeli, która będzie przechowywała adresy dostaw (potrzeba takiej tabeli została wskazana już na pierwszym spotkaniu z klientem). Adres kupującego można przechowywać bezpośrednio w tabeli `tblCustomers`, ponieważ istnieje tylko jeden taki adres dla każdego klienta. Z drugiej strony może istnieć wiele adresów dostaw. Tabela `tblShippingAddresses` zawiera automatycznie numerowane pole `ShipAddressID`, pole łączące `CustomerID`, pole identyfikatora adresu (w celu ułatwienia wyboru adresu z listy rozwijanej na formularzu) i zbiór pól adresowych. Choć pola adresowe mogłyby mieć dokładnie takie same nazwy jak podobne pola z tabeli `tblCustomers`, stosuję przed nimi przedrostek „Ship”, aby móc je w łatwy sposób odróżnić w poleceniach SQL. Uzyskaną tabelę przedstawia rysunek 1.19.

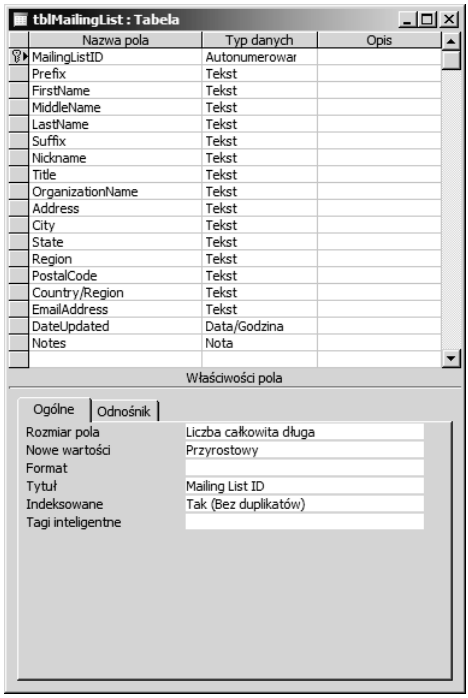
Kolejną tabelą do wykonania jest `tblMailingList` (lista adresowa i lista mailingowa), którą najlepiej utworzyć na podstawie szablonu *Lista adresowa* z kreatora tabel. Należy pominąć wszystkie pola poza nazwiskiem, adresem, adresem e-mail, datą aktualizacji i notatkami. Rysunek 1.20 przedstawia utworzoną w ten sposób tabelę.



Rysunek 1.18.



Rysunek 1.19.

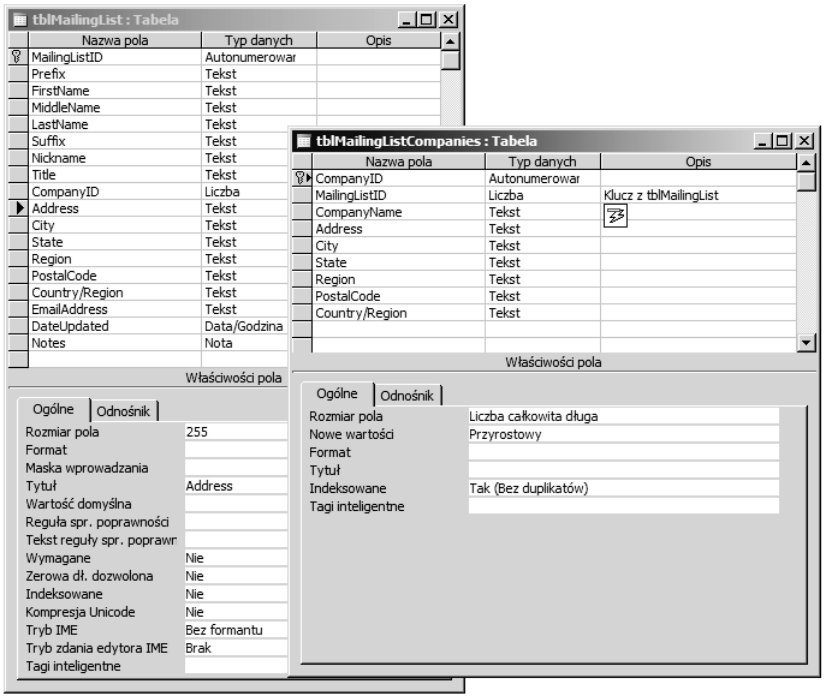


Rysunek 1.20.

Po przyjrzeniu się początkowej wersji tabeli doszłam do wniosku, że na liście może się znajdować kilka osób z tej samej firmy, więc informacje o firmie warto umieścić w osobnej tabeli i połączyć obie tabele za pomocą pola `CompanyID`. Z drugiej strony pewne osoby na liście adresowej mogą nie być powiązane z żadną firmą, więc warto pozostawić dane adresowe i dodać pole `CompanyID` wskazujące w razie potrzeby na odpowiedni rekord tabeli `tblMailingListCompanies`. Rysunek 1.21 przedstawia zmodyfikowaną tabelę `tblMailingList` oraz nową tabelę `tblMailingListCompanies`. W trakcie wprowadzania nowego wpisu do listy adresowej pola adresowe z tabeli `tblMailingList` będą dostępne tylko wtedy, gdy nie zostanie wybrana żadna firma z pola `CompanyID`. W przypadku wybrania firmy użyjemy jej adresu do wysyłania reklam.

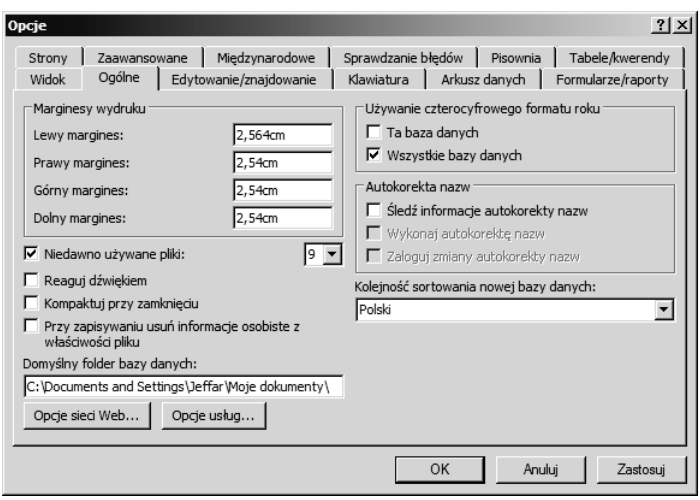
Klienci zapewne też będą otrzymywać katalogi i reklamy, ale nie oznacza to, iż cała lista adresowa musi znajdować się w tabeli `tblCustomers` — możemy zastosować unię, by połączyć dane z tabel `tblCustomers` i `tblMailingList` w momencie wysyłania listów. (Więcej informacji na temat unii znajduje się w rozdziale 4, „Sortowanie i filtrowanie danych za pomocą kwerend”.).

Dla pól daty zalecam wybranie takiego formatu, który wyświetla wszystkie cztery cyfry roku, aby uniknąć problemów z określaniem, czy dany rok dotyczy XX lub XXI wieku. Dla poszczególnych pól można wybrać jeden ze standardowych formatów, używając właściwości formatu lub wpisać format bezpośrednio, na przykład `d/m/rrrr`. Więcej informacji na temat formatu daty można znaleźć w pomocy programu Access. Ewentualnie można włączyć wyświetlanie czterocyfrowego roku w sposób globalny (przesłania to wszystkie ustawienia z właściwości formatu), używając okna dialogowego *Opcje* wywoływanego z menu *Narzędzia*.



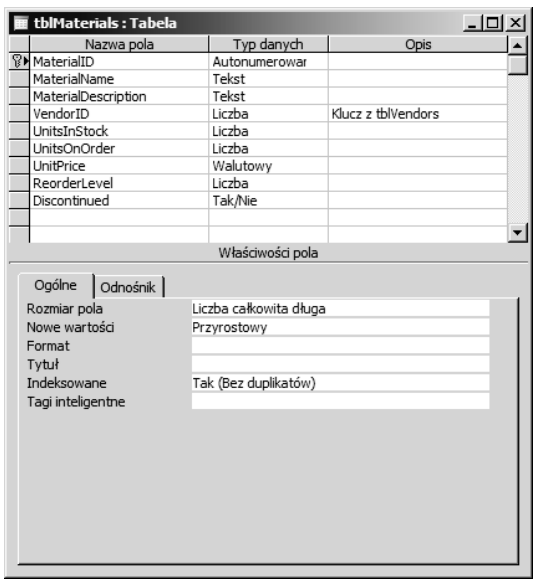
Rysunek 1.21.

Należy przejść do zakładki *Ogólne* i włączyć odpowiednią opcję z sekcji *Używanie czterocyfrowego formatu roku* (patrz rysunek 1.22). Gdy już znajdujesz się na tej stronie, wyłącz opcje związane z autokorektą nazw. Autokorekta to nic więcej jak tylko problemy, ponieważ nie dokonuje wszystkich potrzebnych modyfikacji, a czasem zmienia coś wtedy, gdy nie powinna tego robić. W rozdziale 9, „Modyfikacja istniejącej aplikacji". przedstawiam lepszy sposób zmiany nazw obiektów bazy danych, który korzysta z mojego dodatku LNC Rename.



Rysunek 1.22.

Wykonajmy kolejną tabelę o nazwie `tblMaterials`, która zawiera listę materiałów używanych do produkcji zabawek. Warto skorzystać z szablonu *Produkty* kreatora tabel, pomijając zbyteczne pola i modyfikując nazwy pól. Tabela jest powiązana z tabelą `tblVendors` dzięki polu `VendorID`. Rysunek 1.23 przedstawia okno projektu tabeli `tblMaterials`.



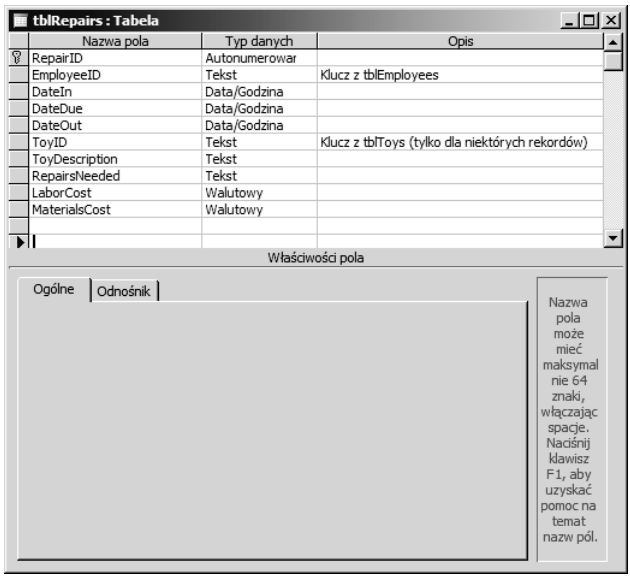
Rysunek 1.23.

Ponieważ materiał może być wykorzystywany do produkcji wielu zabawek, zaś zabawka najczęściej składa się z wielu materiałów, potrzebujemy związku wiele do wielu między tabelami `tblToys` i `tblMaterials`. Oznacza to konieczność wprowadzenia tabeli łączącej, która zawiera tylko pola kluczy. Rysunek 1.24 przedstawia tabelę o nazwie `tblToyMaterials`.



Rysunek 1.24.

Access nie zawiera żadnego szablonu dla tabeli napraw (`tblRepairs`), więc utworzyłam nową tabelę bezpośrednio w widoku projektu, dodając pola przedstawione na rysunku 1.25.



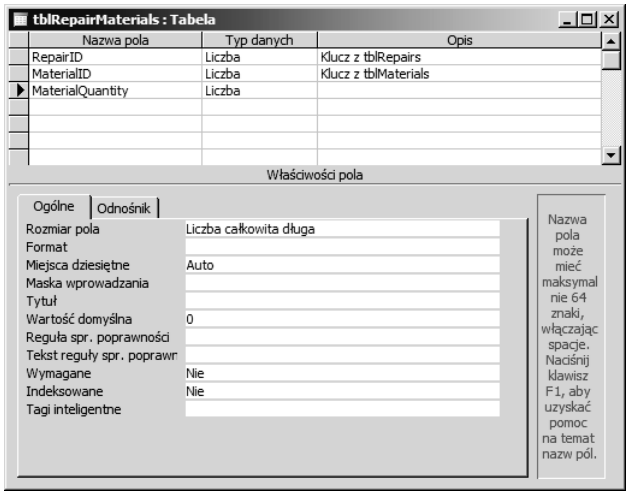
Rysunek 1.25.

> Jest kilka wyrazów, których nie warto stosować jako nazwy pól. Są to m.in.: Data, Rok, Liczba, Numer itp. Należy bowiem unikać wszelkich wyrazów, które są wbudowanymi funkcjami Accessa, właściwościami lub słowami kluczowymi, aby uchronić się przed potencjalnymi problemami w kodzie VBA lub innych miejscach. Warto wtedy rozszerzyć nazwę pola, pisząc na przykład `RokWydania` lub `DataZamówienia`.

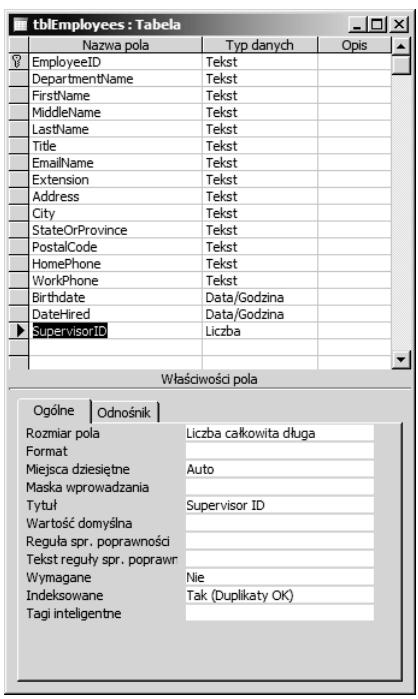
Ponieważ w trakcie napraw zużywa się materiały, potrzebujemy kolejnej tabeli (`tblRepairMaterials`), która zawiera listę materiałów używanych w trakcie napraw wraz z ilością każdego materiału. Tabelę tę przedstawia rysunek 1.26.

Tabela `tblEmployees` bazuje na domyślnej tabeli *Pracownicy* z kreatora tabel, z której zostały usunięte zbyteczne pola. Firma klienta stosuje numeryczną identyfikację pracowników. Ponieważ pracownicy mają już przydzielone identyfikatory, nie możemy zastosować pola z automatycznym numerowaniem. Pole `EmployeeID` będzie więc polem tekstowym wypełnianym istniejącymi numerami pracowników. Identyfikatory dla nowych pracowników będzie wyliczała odpowiednia procedura formularza. Rysunek 1.27 przedstawia tabelę `tblEmployees`.

Ostatnią z głównych tabel jest `tblOrders`; ona także bazuje na szablonie (*Zamówienia*) z kreatora tabel. Zostały z niej jednak usunięte dane adresowe. Zamiast nich pojawiło się pole `ShipAddressID` wskazujące odpowiedni adres z tabeli `tblShippingAddresses`. Potrzebujemy też pola `ToyID`, aby zidentyfikować zakupioną zabawkę, i pola `ToyQuantity` określającego liczbę sztuk (co ciekawe, tego elementu brakuje w oryginalnym szablonie). Tworzoną tabelę przedstawia rysunek 1.28.



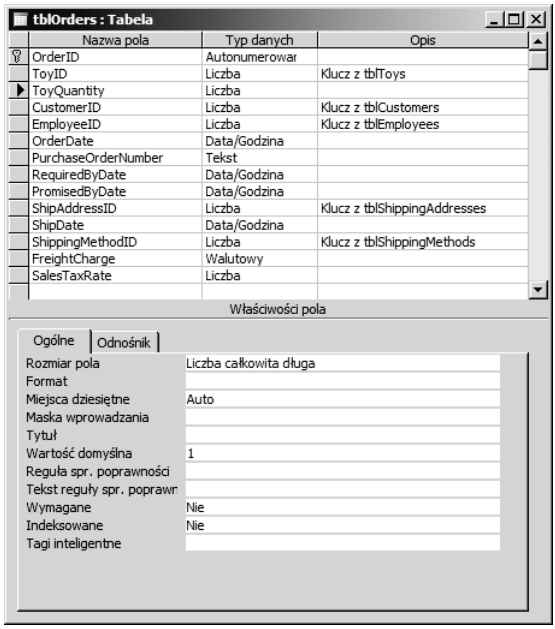
Rysunek 1.26.



Rysunek 1.27.

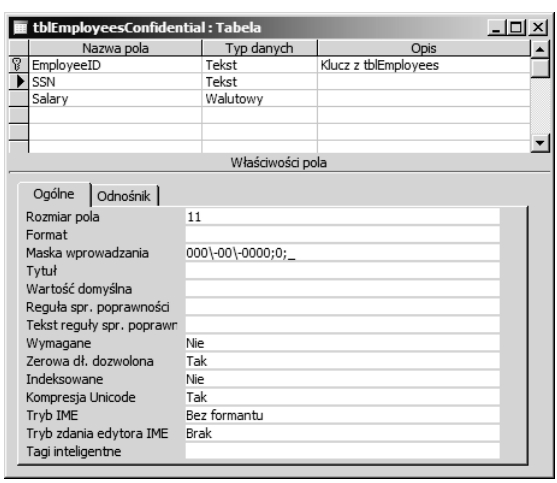
Pole `SupervisorID` (określające przełożonego) przyjmuje wartość `EmployeeID` innego rekordu, wybieraną z listy rozwijanej na formularzu.

Niektóre informacje na temat pracowników powinny być poufne, więc informacje takie jak numer SSN (numer ubezpieczenia społecznego stosowany w USA) i kwota zarobków powinny znaleźć się w osobnej tabeli (`tblEmployeesConfidential`), która została przedstawiona



Rysunek 1.28.

na rysunku 1.29. (Dla numeru SSN została określona maska wprowadzania). Umieszczając te informacje w osobnej tabeli, można zapewnić dostęp do nich jedynie wybranym pracownikom, używając uprawnień na poziomie obiektu dla zabezpieczonej bazy danych. Nawet jeśli nie chce się zabezpieczać bazy danych, warto poufne informacje umieścić w innej tabeli, by nie były widoczne w trakcie typowych prac nad główną tabelą pracowników.



Rysunek 1.29.

Jedno z pól tabeli zamówień (pole `ShippingMethodID`) wymaga tabeli powiązanej odnoszącej się do sposobów przesyłki zamówionego towaru. Tabelę `tblShippingMethods` utworzyłam ręcznie. Jej strukturę przedstawia rysunek 1.30.



Rysunek 1.30.

Zastosowanie typu *Autonumerowanie* dla pola `ShippingMethodID` umożliwia wybór sposobu dostawy z grupy opcji formularza (w Accessie przyciski grupy opcji stosują wartości całkowite). Wybrana wartość jest ściśle powiązana z nazwą sposobu dostawy z tabeli `tblShippingMethods`.

Normalizacja

Aż do tego miejsca zajmowaliśmy się **normalizacją** tabel, choć w ogóle nie wspomniałam o tym terminie. Normalizację baz danych można (i często tak właśnie się robi) opisywać w bardzo złożony i przewrotny sposób, posługując się wieloma fachowymi terminami, choć nie jest to konieczne. W trakcie tworzenia baz danych w Accessie normalizacja sprowadza się do eliminacji duplikacji danych w różnych tabelach i do stosowania pól kluczy do tworzenia połączeń między tabelami. Istnieje pięć poziomów normalizacji baz danych (od **pierwszej postaci normalnej** do **piątej postaci normalnej**). W Accessie najczęściej stosuje się tylko trzy pierwsze. Poniżej definiuję wszystkie pięć postaci, najpierw w żargonie technicznym, a następnie w sposób bardziej przystępny.

Pierwsza postać normalna — eliminacja powtarzających się grup

Oznacza to, że tabela nie powinna zawierać wielu pól dotyczących podobnych informacji, na przykład wielu numerów telefonów lub wielu adresów. W pewnych sytuacjach (na ogół po zaimportowaniu tabel z plików tekstowych) powtarzające się dane mogą znajdować się w jednym polu i być oddzielone przecinkami (na przykład wszystkie stopnie naukowe). Problemy pojawiające się po umieszczeniu różnych informacji w jednym polu są raczej oczywiste: gdy wszystkie stopnie przechowuje się w jednym polu, a chce się wyświetlić tylko osoby ze stopniem doktora, trzeba się będzie sporo natrudzić, pisząc zapytanie wydobywające odpowiednie informacje. Co więcej, możliwe jest, że zwrócone dane nie będą tymi, których się oczekiwało, z powodu źle postawionych znaków przestankowych w danych źródłowych.

Kilka szablonów tabel z kreatora tabel łamie pierwszą postać normalną, na przykład tabela *Kontrahenci* zawiera wiele pól z numerami telefonów. Zamiast przechowywać wiele pól dotyczących telefonów w tabelach `tblVendors` i `tblCustomers`, utworzyłam osobne tabele zawierające numery telefonów i adresy e-mail dostawców i klientów: `tblCustomerPhones`, `tblCustomerEMails`, `tblVendorPhones`, `tblVendorEMails`. Rozbicie informacji na kilka tabel ma dwa aspekty praktyczne: gwarantuje możliwość dodania kolejnego numeru telefonu lub adresu e-mail (jeśli istniałyby tylko pola numeru telefonu stacjonarnego i faksu, gdzie wpisałoby się numer telefonu komórkowego?) i ułatwia korzystanie z zebranych informacji w innych miejscach bazy danych (aby pobrać wszystkie numery wybranego klienta, wystarczy ograniczyć pobieranie danych z tabeli `tblCustomerPhones` do konkretnej wartości pola `CustomerID`).

Powtarzające się dane z jednego pola (jak we wcześniejszym przykładzie ze stopniami naukowymi) powinny zostać przeniesione do osobnej tabeli, by zwiększyć dokładność wprowadzania danych (użytkownicy powinni wybierać stopnie naukowe z tabeli poglądowej, zamiast je wpisywać) i zapewnić możliwość określenia dowolnie dużej liczby stopni naukowych dla jednej osoby.

Druga postać normalna — eliminacja redundancji danych

Istnieją dwa sposoby pojawiania się redundancji danych w bazie danych: pierwszy polega na wpisaniu tych samych danych w różnych rekordach tabeli. Taka sytuacja wystąpiłaby, gdyby użyć oryginalnego szablonu tabeli zamówień z adresami dostaw i wpisać kilka zamówień dotyczących tego samego użytkownika. Jeśli dane tego samego klienta wpisze się w trzech różnych rekordach, zachodzi duplikacja danych. Uniknęłam takiej sytuacji, umieszczając adresy dostaw w osobnej tabeli (`tblShippingAddresses`) i stosując pole `ShipAddressID` w tabeli `tblOrders`. Pole to powiązane jest z polem o takiej samej nazwie z tabeli `tblShippingAddresses`, więc unika się wielokrotnego wpisywania tych samych danych do wielu rekordów. Co więcej, daje to gwarancję, że zmiana adresu dostaw przez klienta będzie wymagała modyfikacji bazy danych tylko w jednym miejscu zamiast we wszystkich rekordach zamówień zawierających adres klienta.

Redundancja danych pojawia się też wtedy, gdy te same informacje znajdują się w dwóch różnych tabelach. Na przykład dla tabel klientów i zamówień informacje na temat adresu osoby kupującej nie powinny znaleźć się w obu tych tabelach. Należy albo umieścić dane adresowe kupującego w osobnej tabeli i połączyć je związkiem jeden do jednego z tabelą `tblCustomers` za pomocą pola `CustomerID`, albo umieścić je w tabeli klientów i usunąć z tabeli zamówień. Podobna sytuacja dotyczy adresów dostaw (nie powinna nastąpić ich duplikacja w obu tabelach). W przedstawionej aplikacji umieszczenie danych dostaw w osobnej tabeli jest szczególnie istotne, gdyż jeden klient może stosować wiele takich adresów.

> *W pewnych sytuacjach w celach archiwalnych można zapamiętać rekord z danymi dostawy używanymi w trakcie składania zamówienia — nawet jeśli ten adres ulegnie potem zmianie. W takiej sytuacji tabela* `tblOrders` *powinna zawierać dane adresowe dostawy oraz pole* `ShipAddressID`*. Gdy adres dostawy zostaje wybrany na formularzu, dotyczące go szczegółowe dane adresowe należy pobrać z tabeli* `tblShippingAddresses` *i zapisać do pól adresowych tabeli* `tblOrders`*. Metoda ta eliminuje potrzebę wpisywania adresu dostaw w każdym rekordzie, ale zachowuje stary adres nawet wtedy, gdy klient zmienił go w terminie późniejszym.*

Trzecia postać normalna — eliminacja kolumn, które nie zależą od klucza

W skrócie można powiedzieć, iż oznacza to przeniesienie do osobnej tabeli wszystkich pól, które w sposób ścisły nie należą do rekordu. Przykładowo — początkowa wersja tabeli `tblMailingList` utworzona z szablonu kreatora tabeli zawiera informacje na temat osoby otrzymującej listy (imię, nazwisko, stanowisko itp.) oraz informacje na temat firmy (nazwa firmy i jej adres). Ponieważ dane firmy nie należą do osoby, utworzyłam osobną tabelę `tblMailingListCompanies` połączoną polem `CompanyID` z listą adresową klientów. W tabeli `tblMailingList` pozostawiłam jednak pola adresowe, by używać ich dla osób, które preferują otrzymywać listy na adres domowy zamiast na adres firmowy.

Czwarta postać normalna — izolacja niezależnych związków wielokrotnych

W bazie danych ze związkami wiele do wielu nie należy dodawać nieistotnych pól do tabel, które łączą dwie tabele „wielu". Przykładowo — baza danych studentów może zawierać związek wiele do wielu między studentami i zajęciami, który wykorzystuje tabelę `tblStudentClasses` do połączeniu obu wspomnianych tabel. Tabela łącząca może zawierać pola semestru i roku, które wskazują, kiedy student uczestniczył w danych zajęciach. Takie rozwiązanie jest poprawne. Gdyby jednak dodać do tabeli łączącej numer telefonu, zostałaby złamana zasada czwartej postaci normalnej, ponieważ pole to nie należy do połączenia rekordów studenta z zajęciami, ale do rekordu studenta, a zatem powinno znaleźć się w tabeli studentów.

Przykładowa baza danych dla firmy zabawkarskiej zawiera związek wiele do wielu między zabawkami i materiałami. Zabawka wyprodukowana jest z wielu materiałów, a jeden materiał często służy do produkcji wielu zabawek. Tabelą łączącą dla tego związku jest `tblToyMaterials`. Jako typowy przedstawiciel tego rodzaju tabel zawiera tylko pola kluczy obcych dwóch łączonych ze sobą tabel. Aby nie złamać zasady czwartej postaci normalnej, do tej tabeli można by dodać tylko pola powiązane z kombinacją dwóch łączonych rekordów.

> *Jest mało prawdopodobne, że trzeba się będzie martwić o złamanie tej postaci normalnej (w odróżnieniu od pierwszych trzech). Rzadko zdarzają się tabele łączące łamiące tę zasadę, a jeszcze rzadziej pojawia się konieczność ich modyfikacji.*

Piąta postać normalna — izolacja semantycznie powiązanych związków wielokrotnych

Złamanie tej postaci wymaga bardzo złożonego i mało prawdopodobnego scenariusza, więc prawdopodobnie nigdy nie trzeba będzie przejmować się tą zasadą. W pewnych sytuacjach ta postać normalna wymaga podzielenia nawet powiązanych ze sobą pól na osobne tabele, na przykład dla związku wiele do wielu między studentami i zajęciami. Choć informacje na temat roku studiów i semestru mogłyby zostać dodane do tabeli łączącej, w pewnych sytuacjach trzeba je umieścić w osobnej tabeli zapewniającej uzyskanie odpowiednich kombinacji semestr-rok.

Określenie związków

Kreator tabel umożliwia wstępne określenie związków (relacji) między tabelami, ale nie wykonuje wszystkich wymaganych zadań. Nawet jeśli określi się, że jeden rekord z pierwszej tabeli może dopasowywać się do wielu rekordów drugiej tabeli, nie jest to jeszcze związek typu jeden do wielu. Określenia odpowiedniego związku trzeba dokonać ręcznie w oknie *Relacje*. Omówię trzy rodzaje związków lub relacji, jakie mogą zostać wykonane w bazie danych Access, a następnie przedstawię sposoby ich ustawiania w oknie *Relacje*.

Zacznijmy od zdefiniowania kilku terminów używanych przy określaniu związków między tabelami.

- **Klucz główny (podstawowy)** — pole (czasem zbiór pól) z różnymi wartościami (lub kombinacją wartości) dla każdego rekordu tabeli. Pole klucza musi być unikatowe i nie może być puste.
- **Klucz obcy** — nieunikatowe pole w tabeli, która jest powiązana z kluczem głównym innej tabeli. W związkach jeden do wielu klucz główny znajduje się w tabeli „jeden", natomiast klucz obcy — w tabeli „wielu".
- **Aktualizacja kaskadowa** — gdy wymusza się integralność referencyjną, zmiana wartości klucza głównego rekordu z tabeli podstawowej (na przykład zmiana zawartości pola `EmployeeID` z tabeli `tblEmployees`) spowoduje przeniesienie tej zmiany na wszystkie pola kluczy obcych powiązanych z tym rekordem. Na ogół warto stosować to rozwiązanie.
- **Usuwanie kaskadowe** — gdy wymusza się integralność referencyjną, usunięcie rekordu z tabeli podstawowej powoduje usunięcie wszystkich rekordów powiązanych z rekordem głównym, a należących do innych tabel. Nie jest to bezpieczne rozwiązanie i w większości sytuacji lepiej go unikać.
- **Złączenie wewnętrzne** — musi istnieć wspólna wartość w łączonych polach z obu tabel. Przykładowo — złączenie wewnętrzne tabel `tblCustomers` i `tblOrders` spowoduje wyświetlenie jedynie rekordów związanych z klientami, którzy złożyli zamówienia.
- **Lewostronne złączenie zewnętrzne** — wszystkie rekordy z lewej strony operacji `LEFT JOIN` polecenia SQL zostają dołączone do wyniku, nawet jeśli nie udało się znaleźć dla nich dopasowania w drugiej tabeli. Lewostronne złączenie zewnętrzne tabel `tblCustomers` i `tblOrders` spowoduje wyświetlenie wszystkich klientów, także tych, którzy nie złożyli żadnego zamówienia.
- **Prawostronne złączenie zewnętrzne** — wszystkie rekordy z prawej strony operacji `RIGHT JOIN` polecenia SQL zostają dołączone do wyniku, nawet jeśli nie udało się znaleźć dla nich dopasowania w drugiej tabeli. Prawostronne złączenie zewnętrzne tabel `tblMailingListCompanies` i `tblMailingList` spowoduje wyświetlenie wszystkich odbiorców listów, także tych, którzy nie mają przypisanej firmy (nie istnieje dla nich dopasowanie w tabeli `tblMailingListCompanies`).

- **Integralność referencyjna** — zbiór zasad zapewniających poprawność relacji między rekordami powiązanych tabel i zabezpieczających przed niewłaściwą zmianą lub usunięciem danych. Włączenie integralności referencyjnej dla powiązania między tabelami `tblCustomers` i `tblOrders` (na podstawie pola `CustomerID`) będzie oznaczało, że nie uda się wstawić nowego zamówienia bez wybrania klienta. Co więcej, dla domyślnej integralności referencyjnej nie uda się usunąć rekordu z tabeli podstawowej, do którego odwołują się rekordy w innych tabelach (usunięcie nastąpi, jeśli włączy się usuwanie kaskadowe), ani zmienić wartości klucza głównego, gdy pojawia się on jako klucz obcy w innych tabelach (chyba że włączy się aktualizację kaskadową).

Związek jeden do wielu

> *Choć Access nie wymaga, by powiązane pola miały identyczne nazwy (muszą mieć jedynie ten sam typ), zalecam stosowanie tych samych nazw dla klucza głównego (podstawowego) i klucza obcego. Ułatwia to odnalezienie odpowiednich pól w trakcie tworzenia relacji.*

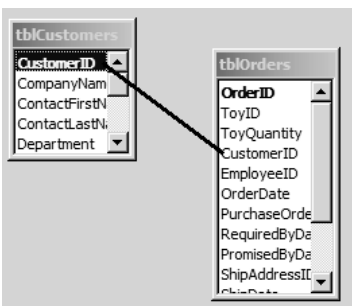
Związku jeden do wielu (jest to najpopularniejszy typ związku) potrzebujemy wtedy, gdy jeden rekord z jednej tabeli może dopasować się do wielu rekordów innej tabeli. Baza danych dla firmy zabawkarskiej potrzebuje wielu tego typu związków. Zostały one wymienione poniżej. Po lewej stronie znajduje się tabela **podstawowa** („jeden"), a po prawej stronie — tabela **powiązana** („wielu"). Niektóre z tych związków stanowią część związków wiele do wielu omówionych w dalszej części rozdziału.

- `tblCategories` — `tblToys`
- `tblCustomers` — `tblCustomerEMails`
- `tblCustomers` — `tblCustomerPhones`
- `tblCustomers` — `tblOrders`
- `tblEmployees` — `tblRepairs`
- `tblMailingListCompanies` — `tblMailingList`
- `tblMaterials` — `tblRepairMaterials`
- `tblMaterials` — `tblToyMaterials`
- `tblRepairs` — `tblRepairMaterials`
- `tblShippingAddresses` — `tblOrders`
- `tblShippingMethods` — `tblOrders`
- `tblToys` — `tblToyMaterials`
- `tblVendors` — `tblMaterials`
- `tblVendors` — `tblToys`
- `tblVendors` — `tblVendorEMails`
- `tblVendors` — `tblVendorPhones`

Jeśli w trakcie próby utworzenia relacji pojawi się komunikat błędu o treści: „Microsoft Access nie może utworzyć tej relacji i wymusić więzów integralności", oznacza to, że dane w jednej z tabel łamią zasady integralności referencyjnej (na przykład rekord z tabeli `tblOrders` może nie zawierać wartości w polu `CustomerID`). Należy wtedy poprawić dane i ponownie spróbować utworzyć relację.

Podobnie komunikat o błędzie: „Relacja musi dotyczyć takiej samej liczby pól o takich samych typach danych" wskazuje, że zapewne doszło do próby powiązania pól różnych typów. Należy więc zmienić typ danych jednego z pól (warto pamiętać, iż automatyczne numerowanie wymaga w drugim polu typu długiej liczby całkowitej) i ponownie spróbować utworzyć relację.

Jako przykład sposobu tworzenia związku jeden do wielu w oknie *Relacje* (pozostałe typy związków tworzy się bardzo podobnie) dokonajmy powiązania tabel `tblCustomers` i `tblOrders`. Zacznijmy od otwarcia okna *Relacje* i przeciągnięcia do niego tabel `tblCustomers` i `tblOrders` z okna bazy danych (ewentualnie można je wybrać, używając okna dialogowego *Pokazywanie tabeli* wywoływanego z menu podręcznego). Zwróć uwagę na to, że pole `CustomerID` z tabeli `tblCustomers` zostało pogrubione, ponieważ jest kluczem głównym tej tabeli. Odpowiadające mu pole `CustomerID` z tabeli `tblOrders` nie zostało pogrubione, ponieważ jest jedynie kluczem obcym. Aby utworzyć związek, przeciągnij pole `CustomerID` z tabeli `tblCustomers` do pola o tej samej nazwie z tabeli `tblOrders` (patrz rysunek 1.31).

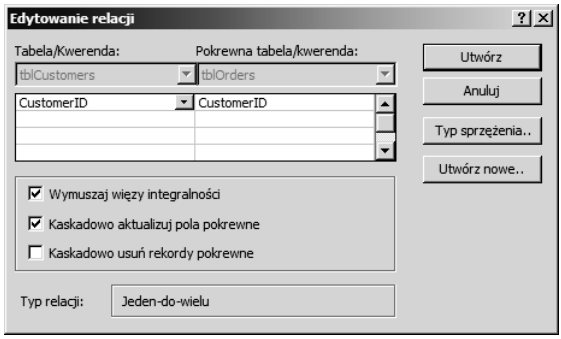


Rysunek 1.31.

Po zwolnieniu przycisku myszy pojawi się okno dialogowe *Edytowanie relacji*. Na dole okna znajduje się ramka, w której Access wyświetla typ relacji; na ogół wykrywa ją poprawnie.

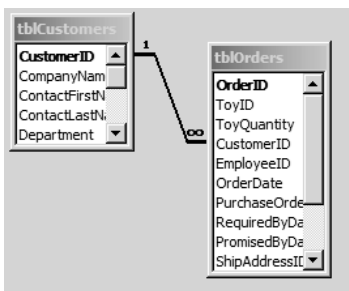
> *Jeśli w ramce na dole okna nie pojawił się poprawny typ relacji — na przykład zamiast jeden do wielu pojawił się inny tekst (np. Nieokreślona) — prawdopodobnie próbowało się połączyć złe pola lub połączyło się poprawne pola, ale o nieodpowiednich typach. Po naprawieniu błędu Access powinien wyświetlić poprawny typ związku.*

Gdy relacja została wykryta poprawnie jako jeden do wielu, wystarczy tylko włączyć opcje *Wymuszaj więzy integralności* i *Kaskadowo aktualizuj pola pokrewne* i kliknąć przycisk *Utwórz* (patrz rysunek 1.32).



Rysunek 1.32.

Pojawiła się linia łącząca pole `CustomerID` z tabeli `tblCustomers` z polem klucza obcego z tabeli `tblOrders` (patrz rysunek 1.33). Warto zauważyć, że po lewej stronie linii znajduje się liczba 1 (wskazująca, iż tabela `tblCustomers` jest tabelą podstawową), a po prawej stronie znak ∞ (oznaczający tabelę powiązaną).



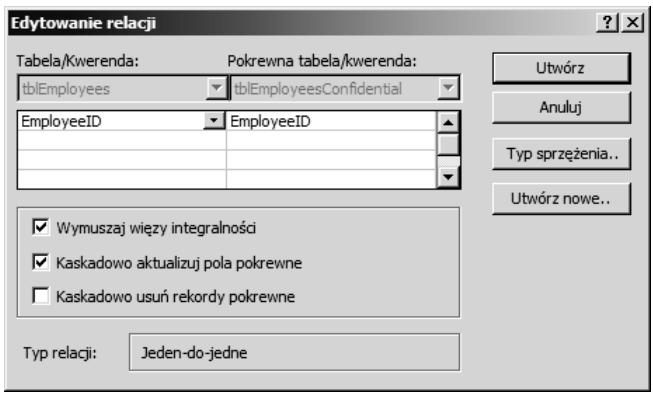
Rysunek 1.33.

Związek jeden do jednego

Związek jeden do jednego (występuje stosunkowo rzadko) wymagany jest tylko wtedy, gdy rekord z jednej tabeli może dopasować się tylko do jednego rekordu z innej tabeli. Pole łączące jest kluczem głównym obu tabel. Na ogół związek tego typu służy do ograniczenia dostępu do pewnych danych, na przykład poufnych danych pracowników. W przykładowej bazie danych występuje tylko jeden tego rodzaju związek — między tabelami `tblEmployees` i `tblEmployeesConfidential`. Aby utworzyć związek, przeciągnij pole `EmployeeID` z tabeli `tblEmployees` nad to samo pole tabeli `tblEmployeesConfidential`. Na dole okna dialogowego jako typ związku pojawi się *Jeden-do-jednego* (patrz rysunek 1.34).

Jeśli na dole okna dialogowego znajduje się informacja o związku jeden do wielu, oznacza to, że połączyło się pola, które nie są kluczami głównymi tabel. Należy ustawić oba pola na klucze główne i ponownie spróbować utworzyć związek.

W oknie *Relacje* linia reprezentująca związek jeden do jednego ma na obu końcach liczbę 1.



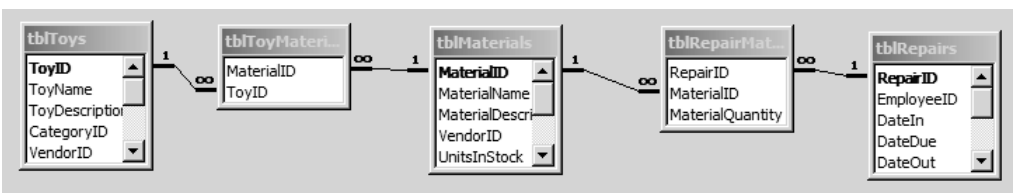
Rysunek 1.34.

Związek wiele do wielu

Związek wiele do wielu to tak naprawdę połączenie dwóch związków jeden do wielu. Istnieją dwie tabele podstawowe i tabela łącząca. Tabela łącząca zawiera dwa pola kluczy obcych — po jednym dla każdego pola klucza głównego tabel podstawowych. Może również zawierać (choć nie zdarza się to często) kilka innych pól przechowujących informacje związane z konkretną kombinacją rekordów tabel podstawowych. W przykładowej bazie danych występują dwa tego typu związki (tabela łącząca znajduje się w środku):

- `tblToys` — `tblToyMaterials` — `tblMaterials`,
- `tblRepairs` — `tblRepairMaterials` — `tblMaterials`.

Gdy utworzy się dwa związki typu jeden do wielu, uzyska się jeden związek wiele do wielu. Rysunek 1.35 przedstawia oba wymienione powyżej związki wiele do wielu w oknie *Relacje*. Łatwo zauważyć dwa zestawy tabel podstawowych, między którymi znajduje się tabela łącząca; tabela `tblMaterials` jest tabelą podstawową w obu związkach wiele do wielu.



Rysunek 1.35.

Jeśli stosuje się konwencję, w której nazwy pól kluczy głównych i obcych zawierają przyrostek ID, bardzo łatwo rozpoznać, które pola należy ze sobą połączyć w oknie Relacje. Pewne pola kluczy nie muszą być powiązane z innymi tabelami — na przykład pole `MailingListID` *z tabeli* `tblMailingList` *nie potrzebuje żadnych połączeń, ponieważ nie istnieje tabela, której wiele rekordów pasowałoby do jednego rekordu z* `tblMailingList`.

Podsumowanie

Skoro powstały odpowiednie tabele i związki między nimi dla bazy danych sklepu z zabawkami, możemy przystąpić do tworzenia formularzy służących do wpisywania i edycji danych oraz kwerend do sortowania i filtracji.