

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

Access 2002. Projektowanie baz danych. Księga eksperta

Autorzy: Stephen Forte, Thomas Howe, Kurt Wall,
Paul Kimmel, Russ Mullen

Tłumaczenie: Paweł Gonera

ISBN: 83-7197-669-0

Tytuł oryginału: [Access 2002 Development Unleashed](#)

Format: B5, stron: 666

Zawiera CD-ROM



Książka ta jest napisana przez programistów Accessa oraz dla programistów Accessa. Obecnie wiele książek pisanych jest przez zawodowych instruktorów i pisarzy. Autorzy tej książki zajmują się tym samym co Ty – tworzą w Accessie aplikacje dla swoich użytkowników. W książce tej dzielimy się naszym doświadczeniem. Skupiamy się na tych właściwościach Accessa, których najprawdopodobniej będziesz używał – nie opisujemy w nieskończoność każdego przycisku i każdej zakładki okna dialogowego. Niezależnie od tego, czy piszesz zawodowo aplikacje w Accessie, zmieniasz narzędzie z Visual Basic'a lub innego języka programowania, myślisz o zastosowaniu Accessa jako interfejsu użytkownika dla serwera bazy danych SQL Server, Oracle, Informix lub Sybase, bądź też jesteś po prostu niedoświadczonym użytkownikiem Accessa, chcącym poznać jego szczegóły techniczne – jest to książka dla Ciebie. Jest to jedyna książka o Accessie, która tak dokładnie opisuje pracę z bazą Oracle. Za pierwszym razem, gdy musiałem stworzyć aplikację Accessa korzystającą z Oracle, byłem bardzo zawiedzony brakiem literatury na ten temat. Doświadczenie Jamesa Ralstona w łączeniu Accessa z bazą Oracle zostało wykorzystane w jednym z rozdziałów. Ponadto znajdziesz tu dwa pełne rozdziały na temat współdziałania Accessa z SQL Serverem oraz wiele fragmentów innych rozdziałów, opisujących ten sam problem.

Najlepiej opisuje wykorzystanie Accessa w Internecie. Rozwój popularności Internetu wpłynął nie tylko na wiele głównych firm przemysłu komputerowego – w tym Microsoftu – ale dzięki niemu niespodziewanie zmienił się także sposób tworzenia oprogramowania. Wiele książek o Accessie nadal traktuje Internet jako problem do przemyślenia; ta książka opisuje Sieć obok innych środowisk programowania. Istotnie, Internet jest kolejną platformą programowania, a Access jest bazą danych, pozwalającą na tworzenie aplikacji ukierunkowanych na dane. Opisałiśmy również sposoby stworzenia od podstaw serwera WWW na komputerze z Windows 2000/NT i korzystanie z Windows 95 z zainstalowanym Personal Web Server. Zagadnienia opisane w tej książce obejmują wiele problemów związanych z zastosowaniami Accessa w Internecie: począwszy od XML, ASP, DAP i komponentów sieciowych Office, na najnowszych trendach w programowaniu skończywszy.



Spis treści

O Autorach	21
Wstęp	23
Część I Projektowanie bazy danych	27
Rozdział 1. Co nowego w Accessie 2000.....	29
Zmiany interfejsu użytkownika	30
Zmiany w VBE.....	30
Zgodność z bazami danych Accessa 2000	31
Rejestrowanie błędów konwersji bazy danych.....	32
Strony dostępu do danych w trybie offline.....	32
Ulepszona integracja z SQL Server 2000.....	33
Rozszerzona obsługa właściwości	33
Modyfikacja wsadowa.....	34
Ochrona projektów Accessa za pomocą hasła	34
Inne własności Accessa 2002	34
Widoki danych: Wykres przestawny i Tabela przestawna.....	34
Obsługa XML.....	34
Kreator tabel połączonych.....	35
Tryb kwerend ANSI-92 bez ADO	35
Rozdział 2. Planowanie procesu rozwoju	37
Określenie wymagań	38
Dlaczego faza określenia wymagań jest tak ważna?.....	38
Odnalezienie rzeczywistego problemu.....	39
Śledztwo	39
Diagramy procesu	42
Identyfikacja zakresu projektu	43
Spisywanie wymagań	43
Architektura	44
Planowanie rozwoju	44
Strategia dostarczania.....	44
Standardy.....	45
Konstrukcja.....	47
Dziel i rządź: działania w fazie Konstrukcja.....	47
Edycje i kompilacje.....	49
Szczegółowy projekt	51

Kontrola projektu	51
Tworzenie aplikacji	53
Kontrola aplikacji	53
Testowanie	53
Odnajdywanie usterek	55
Kontrola wersji	58
Rozdział 3. Projekt bazy danych i normalizacja.....	61
Teoria projektowania relacyjnego	61
Korzyści z używania modelu relacyjnego	63
Tabele i niepowtarzalność	63
Klucze obce i domeny	64
Relacje	65
Normalizacja danych	67
Zasady integralności danych	69
Rozdział 4. Zaawansowane kwerendy	71
Kwerendy w Accesie 2002	71
Właściwości kwerend	73
Okno projektowania kwerendy	73
Panel tabel	76
Siatka kwerendy	82
Tworzenie zaawansowanych kwerend	84
Kwerendy podsumowujące	84
Zliczanie wierszy za pomocą funkcji Policz	85
Obliczanie średnich za pomocą funkcji Średnia	86
Funkcje agregujące Minimum i Maksimum	86
Funkcje Pierwszy i Ostatni	87
Kontrola poprawności danych przy pomocy funkcji Odchylenie standardowe i Wariancja	87
Tworzenie własnych funkcji agregujących przy pomocy funkcji Wyrażenie	88
Warunek Gdzie	88
Użycie kwerend krzyżowych	91
Użycie kwerend parametrycznych	94
Tworzenie parametrów dla kwerendy w siatce QBE	94
Tworzenie parametrów w kwerendach korzystających z programu	96
Wykonywanie kwerendy poprzez zbiór parametrów	97
Kwerendy przekazujące	98
Kwerendy definiujące dane	99
Tworzenie nowej tabeli	99
Określanie właściwości pól	100
Modyfikowanie tabeli	101
Tworzenie indeksów	102
Usuwanie tabeli	102
Rozdział 5. Jet 4.0 — silnik baz danych Microsoft	103
Historia Microsoft Jet	103
Jet 1.0	103
Jet 1.1	104
Jet 2.0	104
Jet 2.5	104

Jet 3.0	104
Jet 3.5	104
Jet 3.51	105
Jet 4.0	105
Przyszłe wersje Microsoft Jet.....	105
Praktyczne zastosowanie nowych opcji silnika Jet 4.0	106
Wbudowany dostawca baz danych OLE.....	106
Blokowanie na poziomie rekordu	106
Pełna obsługa Unicode	106
Typy danych Jet	109
Ulepszenia opcji Autonumerowanie	111
Przeszukiwalne pola Memo	111
Kontrola połączeń i zamknięcie bierne	111
Nowa składnia SQL	112
Część II Dostęp do danych.....	117
Rozdział 6. Wprowadzenie do obiektów danych ActiveX	119
Historia dostępu do danych	119
Firmowe interfejsy API.....	119
Open Database Connectivity (ODBC).....	120
Microsoft Jet/Obiekty Data Access (DAO)	120
Zdalne obiekty danych (RDO) i ODBCDirect.....	120
Inicjatywa Universal Data Access.....	120
Obiekty ActiveX Data (ADO).....	121
ADO 1.0	121
ADO 1.5	122
ADO 2.0	122
ADO 2.1	122
ADO 2.5	123
ADO 2.6 — Stan obecny.....	123
Model obiektowy ADO	124
Obiekt Connection.....	125
Wykonywanie wyrażeń SQL w obiekcie Connection.....	127
Obiekty ADO: Recordset	128
Użycie Recordset z obiektami Command i Parameter	130
Wykorzystywanie obiektu Record	133
Wykonywanie kwerendy funkcjonalnej poprzez obiekt Command.....	134
Obiekty Field i Property.....	135
Wykorzystanie obiektu Stream	136
Zarządzanie błędami za pomocą obiektu Error.....	136
Przejsię z obiektów DAO do ADO.....	137
Konwersja z poprzednich wersji Accessa	138
Czy warto przejść na obiekty ADO?.....	138
Porównanie modelu obiektowego ADO i DAO	140
Rozdział 7. Zaawansowane ADO	143
Użycie dostawcy OLE DB dla Jet w Accessie 2002	143
CurrentProject.Connection	146
Wykorzystanie Microsoft Data Links do podłączenia się do bazy danych.....	146

Uzyskanie poprzez ADO dostępu do danych w bazach nierelacyjnych	151
Lista użytkowników Jet.....	151
Tworzone zestawy rekordów	152
Kształtowanie danych	153
Zaawansowana obróbka danych przy użyciu ADO	155
Modyfikowanie danych w zestawie rekordów	156
Trwałe zestawy rekordów	157
Definiowanie danych przy użyciu ADOX.....	159
Obiekt Catalog.....	160
Tworzenie bazy danych.....	160
Tworzenie tabel i pól.....	161
Tworzenie relacji w ADOX	164
Tworzenie kwerend w ADOX.....	165

Część III Interfejs użytkownika..... 169

Rozdział 8. Projektowanie formularza..... 171

Właściwości formularza	172
Zakładka Dane.....	172
Zakładka Format	173
Zakładka Inne	175
Użyteczne właściwości nie znajdujące się na arkuszu właściwości	177
Zdarzenia.....	178
Formanty formularza w Accessie	179
Formant Lista rozwijana.....	180
Pole listy.....	183
Pole listy wielokrotnego wyboru.....	183
Podformularze	184
Wbudowany formant Karta.....	185
Grupa opcji.....	185
Podręczne menu	185
Hiperłącza.....	186
Tworzenie widoków tabeli przestawnej i wykresu przestawnego.....	186
Tworzenie widoku tabeli przestawnej.....	187
Konwersja pomiędzy wykresem przestawnym i tabelą przestawną	191

Rozdział 9. Rozbudowa formularzy przy użyciu formantów ActiveX.... 193

Jak korzystać z formantów ActiveX.....	193
Typy formantów ActiveX	194
Gdzie znaleźć formanty ActiveX?	194
Czy formanty ActiveX są bezpieczne?	195
Czy mogę korzystać z formantów ActiveX i rozpowszechniać je w moich aplikacjach?	195
Użycie formantów ActiveX.....	195
Instalowanie formantu ActiveX	196
Rejestrowanie formantów ActiveX.....	196
Dodawanie formantu ActiveX do formularza.....	197
Ustawianie właściwości formantu ActiveX	198
Pisanie kodu umożliwiającego wykonywanie metod i reagowanie na zdarzenia	198

21 formantów ActiveX	198
Formant Animation	200
Formant Calendar	200
Formant Common Dialog.....	201
Formant DateTimePicker	202
Formant FlatScrollBar.....	203
Formant ImageCombo.....	203
Formant ImageList	205
Formant ListView	206
Formant MAPISession	208
Formant MAPIMessages.....	208
Formant MonthView	209
Formant ProgressBar	209
Formant RichText	211
Formant Slider.....	212
Formant StatusBar.....	213
Formant SysInfo	214
Formant TabStrip	214
Formant Toolbar.....	215
Formant TreeView	216
Formant UpDown.....	217
Formant WebBrowser	218
Dystrybucja formantów ActiveX.....	220
Rozdział 10. Tworzenie raportów	221
Czym są raporty?	221
Struktura raportów w Accessie.....	222
Tworzenie prostych raportów przy użyciu kreatora	222
Dostosowywanie raportów	225
Zmiana źródła rekordów w raporcie	226
Zmiana struktury grupowania w raporcie	227
Umieszczenie grupowania w raporcie.....	228
Użycie funkcji w raporcie	228
Praca z podraportami	234
Tworzenie prostego podraportu	234
Tworzenie prostych etykiet adresowych	236
Publikowanie raportu.....	238
Metody publikowania raportów	239
Modyfikowanie raportu podczas jego działania.....	241
Filtrowanie i sortowanie.....	241
Zdarzenia raportu (podczas jego działania).....	242
Właściwości sekcji związane z procesem projektowania	245
Właściwości sekcji (podczas działania raportu).....	246
Programowe tworzenie raportów.....	248
Tworzenie źródła rekordów	249
Tworzenie obiektu Report.....	250
Tworzenie sekcji	252

Wskazówki	252
Tworzenie grupowania dwutygodniowego	252
Ukryj powtarzające się dane	253
Alfabetyczne grupowanie danych	253
Tworzenie numerowanych list	253
Tworzenie pustych linii co n znaków	254
Zerowanie numeru strony dla nowych grup	254
Rysowanie pionowych linii	255
Przesuwanie numerów stron parzystych i nieparzystych	255
Identyfikacja użytkownika drukującego raport	255
Wyrównanie stron do oprawy	256
Obliczanie podsumowań strony	256
Precyzyjne przesuwanie formantów	257

Część IV Tajniki VBA 259

Rozdział 11. Tworzenie obiektów przy użyciu modułów klas 261

Korzyści z używania obiektów	262
Ukrywanie złożoności	262
Użycie technologii Microsoft IntelliSense	263
Organizowanie kodu	263
Dopuszczenie przeglądania obiektów w Object Browser	264
Tworzenie wielu egzemplarzy obiektów	264
Tworzenie kodu łatwego do aktualizacji i utrzymania	264
Ograniczenie dostępu do kodu	264
Tworzenie przenośnego kodu	264
Przegląd obiektów, właściwości i metod	264
Tworzenie klas	265
Wstawianie modułu klasowego	265
Tworzenie właściwości	266
Użycie zmiennych publicznych	266
Użycie procedur właściwości	266
Zmienna publiczna czy procedury właściwości	269
Tworzenie wyliczeniowych typów danych	269
Tworzenie metod	271
Użycie metod	271
Tworzenie zdarzeń	271
Użycie zdarzeń	272
Uruchamianie zdarzeń „Przy inicjacji” i „Przy zakończeniu”	272
Użycie obiektów	273
Tworzenie zmiennej obiektu	274
Przypisywanie zmiennej obiektu do obiektu	274
Użycie obiektu	274
Zwalnianie obiektu	275
Tworzenie wielu egzemplarzy obiektów	275
Przegląd innych obiektów	276
Obiekt TextFile	276
Klasa MyTimer	277

Obiekt Sound	278
Obiekt Letter	279
Obiekt Outlook	280
Implementacja obiektu obsługi błędów	281
Użycie obiektów w połączeniu z kolekcjami VBA	282
Tworzenie kolekcji VBA	283
Właściwości i metody kolekcji VBA	283
Rozdział 12. Usuwanie błędów w aplikacjach Accessa	287
Usuwanie błędów logicznych	287
Praca z Visual Basic Integrated Development Environment (IDE)	288
Project Explorer	288
Okno Code	289
Okno Properties	289
Okno Immediate	290
Okno Locals	290
Okno Watch	291
Object Browser	291
Okno Call Stack	291
Obiekt Debug	292
Debug.Print	292
Debug.Assert	293
Użycie okna Immediate	293
Podglądanie zmiennych	293
Zmiana wartości zmiennych	293
Wyświetlanie wartości funkcji wbudowanych	293
Uruchamianie funkcji własnych	293
Uruchamianie własnych procedur	293
Wskazówki pomocne w korzystaniu z okna Immediate	294
Użycie programu uruchomieniowego	294
Ustawianie punktów zatrzymania	295
Przechodzenie przez kod	295
Użycie podczas usuwania błędów technologii Microsoft IntelliSense	297
Użycie okna Locals	298
Użycie okna Watch	298
Użycie okna Call Stack	299
Użycie kompilacji warunkowej	299
Pisanie solidnego kodu	300
Testowanie aplikacji	304
Ćwiczenie technik usuwania błędów	304
Rozdział 13. Profesjonalna obsługa błędów	305
Usuwanie błędów składni	306
Usuwanie błędów logicznych	308
Usuwanie błędów wykrytych w trakcie użytkowania	308
Proste narzędzie do obsługi błędów	308
Obiekt Err	311
Reagowanie na błędy	313

Instrukcje Resume	314
Uzyskiwanie dodatkowych informacji o błędzie	314
Zaawansowane narzędzie do obsługi błędów	316
Moduł klasowy Error (Obiekt)	316
Błędy w różnych aplikacjach	327
Obsługa błędów w procedurach zagnieżdżonych	327
Zaawansowane zagadnienia związane z błędami	328
Procedury zdarzeń związane z błędami	328
On Error GoTo 0	328
On Error Resume Next	329
Metoda AccessError	329
Inne funkcje związane z błędami	329
Ustawienie opcji wyłapujących błędy	329
Rozdział 14. Optymalizacja aplikacji	331
Ulepszanie podstaw: optymalizacja sprzętu i systemu Windows	332
Instalowanie aplikacji w celu uzyskania optymalnej wydajności	334
Optymalizacja silnika bazy danych Jet	335
Bezpieczne modyfikowanie ustawień silnika Jet	339
Narzędzia służące do pomiaru wydajności	340
Spojrzenie za kulisy	341
Optymalizacja bazy danych od podstaw	343
Projektowanie tabel w celu osiągnięcia poprawy wydajności	343
Normalizacja danych w celu osiągnięcia poprawy wydajności	344
Tworzenie indeksów w celu przyspieszenia pracy kwerend	344
Wcześniejsze tworzenie relacji jako sposób na poprawę wydajności	345
Poprawa wydajności kwerend	345
Wybór typu zestawu wyników zapewniającego optymalną wydajność	347
Przyspieszenie funkcjonowania formularzy	353
Zacznijmy od początku	353
Szybsze pobieranie rysunków	354
Podstawowy, szybki formularz	354
Szybsze drukowanie raportów	357
Pisanie szybkiego kodu	358
Użycie pamięci przez kod	359
Praca z modułami	359
Kompilowanie i dekompilowanie kodu	360
Porady na temat kodowania	360
Użycie Option Explicit	360
Precyzyjne wybieranie rozmiaru zmiennych	361
Oszczędzanie przestrzeni stosu przy użyciu zmiennych typu string	361
Dokładne określanie typu obiektów	361
Umieszczenie kodu we wnętrzu procedury zamiast odwoływania się do innych funkcji	362
Zmiana True i False	362
Użycie Len() zamiast pustego ciągu	362
Użycie True i False zamiast zera	362
Szybkie odwołania do obiektów	363
Użycie szybkich tablic	363

Używaj stałych, gdy tylko jest to możliwe	364
Właściwe użycie zakładek (Bookmarks)	364
Zamykaj i niszczy.....	365
Używaj SQL zamiast DAO	365
Indeksowanie kolekcji.....	365
Tworzenie szybszych pętli	366
Usuń z kodu IIF()	366
Porządkowanie Select Case.....	366
Używaj Execute zamiast RunSQL	367
Używaj QueryTimer.....	367
Testuj wydajność transakcji	367
Kontroluj odświeżanie.....	367
Używaj wczesnego wiązania i zwracaj uwagę na odniesienia ActiveX	367
Przejdźcie do architektury klient-serwer	367
Chleba i igrzysk.....	368

Część V Access i architektura klient-serwer.....369

Rozdział 15. Wprowadzenie do projektów programu

Microsoft Access oraz narzędzi wizualnych 371

Wprowadzenie do projektów programu Microsoft Access	371
Wady i zalety ADP.....	372
Użycie ADP	372
Tworzenie ADP.....	373
Nowe okno bazy danych	375
Praca z ADP i istniejącymi bazami danych serwera SQL.....	375
Praca z tabelami	375
Kwerendy w serwerze SQL.....	377
Procedury przechowywane	378
Diagramy bazy danych.....	379
Formularze, strony, raporty i moduły.....	379
Zarządzanie serwerem SQL poprzez ADP.....	380
Powtórne przyłączenie do bazy serwera SQL.....	382
Tworzenie projektu opartego na nowej bazie danych	383
Tworzenie tabel	383
Tworzenie relacji na diagramie bazy danych.....	385
Tworzenie widoków	386
Tworzenie procedur przechowywanych.....	387
Tworzenie aplikacji w Accessie	388

Rozdział 16. Tworzenie interfejsu użytkownika

dla Microsoft SQL Server 389

Architektura klient-serwer: OLE DB kontra ODBC	389
Tworzenie połączenia z serwerem SQL	390
Tworzenie źródła danych ODBC (DSN)	390
Łączenie tabel.....	391

Procedury przechowywane i kwerendy przekazujące	393
Tworzenie raportów opartych na procedurach przechowywanych poprzez zapytania przekazujące	394
Raportowanie z serwera SQL w Accessie	396
Zaawansowane możliwości: przekazywanie parametrów do procedury przechowywanej w czasie działania programu	397
Dodatkowe filtrowanie danych raportu	398
Formularze w aplikacji	400
Formularze związane	400
Formularze niezwiązane	400
Zaawansowane właściwości: dostawca OLE DB dla serwera SQL	402
Następny zestaw wyników	402
Wykonywanie poleceń z parametrami	403
Długa droga	404
Użycie metody CreateParameter	405
Użycie Refresh	406
Obsługa zwracanych wartości	407
Wykonanie procedury bez obiektu Command	409
Użycie klasy Connection	409
Użycie klasy Connection w aplikacji	410
Rozdział 17. Interfejs Accessa 2002 do Oracle'a	411
Dostęp do danych Oracle'a przez Accessa	412
Tabele połączone	412
Kwerendy przekazujące (SPT)	414
Użycie znaków specjalnych	417
Funkcje w Oracle'u i Accessie	419
Ciągi	419
Obliczenia w Oracle'u	425
Obliczenia na datach	427
Poznajemy widoki i procedury przechowywane	430
Tworzenie widoków	431
Połączenie z Oracle'em poprzez ADO	432
Tworzenie procedur przechowywanych	435
Uruchamianie procedury	436
Tworzenie niezwiązanego interfejsu do Oracle'a	437
Tworzenie niezwiązanego interfejsu	438
Część VI Współoperatywność	447
Rozdział 18. Użycie automatyzacji ActiveX	449
Czym jest automatyzacja ActiveX	450
Dlaczego używamy automatyzacji	450
Różnice między serwerem automatyzacji a klientem automatyzacji	450
Określanie zasobów wymaganych przez automatyzację	450
„Wielka trójka”	451
Tworzenie i ustanowienie odwołania do innej aplikacji	451
Ustanawianie odwołania do innej aplikacji	451
Przegląd obiektów, właściwości i metod	452

Poznajemy strukturę obiektów	452
Użycie narzędzia Object Browser	453
Tworzenie zmiennej obiektowej	454
Odwołanie do pracującej aplikacji	455
Przypisywanie zmiennej obiektowej do aplikacji	455
Tworzenie egzemplarza aplikacji	456
Jednoczesne użycie funkcji GetObject i New	456
Użycie wczesnego i późnego łączenia typów	457
Użycie funkcji CreateObject	457
Użycie metod i właściwości obiektów automatyzacji	458
Ustawianie właściwości obiektu	459
Wykonywanie metod obiektów	459
Zwalnianie obiektów automatyzacji	459
Łączymy wszystko razem	459
Zamykanie aplikacji serwera automatyzacji	460
Użycie właściwości UserControl do sprawdzenia, w jaki sposób została otwarta aplikacja	461
Użycie WithEvents w celu udostępnienia zdarzeń serwera automatyzacji	461
Uruchamiamy WithEvents	463
Techniki i wskazówki do automatyzacji	464
Ustanowienie odwołania, użycie wczesnego wiązania typów i słowa kluczowego New	464
Użycie istniejącego egzemplarza aplikacji, jeżeli jest ona uruchomiona	464
Wyłącz odświeżanie ekranu	465
Informacja o przetwarzaniu	465
Wykonywanie programu przez serwer automatyzacji	466
Użycie konstrukcji With/End With	466
Zwalnianie zmiennych obiektowych	466
Nie wyświetlaj okien dialogowych i komunikatów	467
Używaj obsługi błędów	467
Rozdział 19. Integracja z Microsoft Office	469
Powody integracji z Microsoft Office	470
Użycie Worda	471
Użycie Excela	471
Użycie PowerPointa	471
Użycie Outlooka	471
Użycie Grapha	471
Wybór właściwego narzędzia	472
Użycie rejestratora makr do pisania kodu	472
Użycie makr automatycznych	474
Microsoft Forms	474
Object Browser	475
Nazwy klas aplikacji Office	475
Przykład automatyzacji	476
Automatyzacja Worda	477
Model obiektów Worda	478
Użycie szablonów Worda	479
Wstawianie danych do dokumentu Worda	480
Przykłady kodu automatyzacji Worda	483
Pola	488

Informacje o dokumencie.....	489
Inne możliwości Worda	490
Automatyzacja Excela	490
Model obiektów Excela.....	491
Przykłady automatyzacji Excela	491
Automatyzacja PowerPointa.....	494
Model obiektów PowerPointa	494
Przykłady automatyzacji PowerPointa.....	494
Automatyzacja Outlooka	497
Model obiektów Outlooka.....	497
Dodawanie i wyświetlanie folderów	498
Dodawanie nowego zadania i wyświetlenie zadań	498
Tworzenie wiadomości e-mail z załącznikiem	498
Tworzenie elementów Outlooka	499
Wyświetlanie domyślnych folderów	500
Wyświetlenie foldera publicznego.....	500
Szukanie elementu w Outlooku.....	500
Filtrowanie elementów w Outlooku.....	500
Automatyzacja Graph.....	501
Model obiektów Graph.....	501
Tworzenie wykresu	501
Rozdział 20. Użycie Visual Basic z Accessem.....	503
Tworzenie komponentów ActiveX.....	503
Czym są komponenty ActiveX	504
Różnice między ActiveX EXE i ActiveX DLL	504
Tworzenie komponentu ActiveX	505
Kompilowanie biblioteki DLL	508
Użycie komponentu ActiveX cSound.....	509
Dystrybucja komponentów ActiveX.....	510
Komponent obsługi błędów	519
Dane modułu obsługi błędów.....	519
Komponent cError.....	520
Tworzenie formantów ActiveX.....	520
Rodzaje formantów ActiveX.....	520
Atrybuty formantów ActiveX	521
Tworzenie programowych formantów ActiveX	521
Tworzenie formantów ActiveX interfejsu użytkownika.....	534
Część VII Zagadnienia wielodostępu	541
Rozdział 21. Zagadnienia wielodostępu, serwer plików, blokowanie	543
Zagadnienia wielodostępu.....	543
Przegląd wieloużytkownikowej architektury Jet.....	544
Omówienie blokad Jet	545
Tryby bazy danych.....	545
Schemat blokowania	547
Szczegółowość blokad	548

Wybór właściwej architektury	548
Operacje na blokadach.....	549
Określanie stanu blokad	549
Obsługa błędów blokowania	551
Transakcje	553
Optymalizacja aplikacji wielodostępnych	555
Indeksowanie tabel.....	555
Partycjonowanie aplikacji	555
Blokady Oracle/SQL Server.....	555
Rozdział 22. Bezpieczeństwo	557
Elementy bezpieczeństwa	558
Zabezpieczenie bazy danych hasłem.....	558
System bezpieczeństwa grupy roboczej	558
Tworzenie grupy roboczej	559
Użytkownicy i grupy	562
Omówienie domyślnych ustawień użytkowników i grup	562
Tworzenie użytkowników	563
Ustawianie i zmiana hasła użytkownika	563
Tworzenie grup	564
Przypisywanie użytkowników do grup	565
Rozróżnianie między domyślnymi i specjalnymi grupami i użytkownikami	565
Poznajemy uprawnienia	566
Tworzenie systemu bezpieczeństwa przy użyciu opcji startowych.....	568
Zagadnienia bezpieczeństwa przy użyciu replikacji	569
Ochrona podzielonych baz danych.....	570
Opcja With OwnerAccess	571
Bezpieczeństwo systemu klient-serwer	572
Zarządzanie użytkownikami	572
Wylizanie grup i użytkowników oraz wyświetlanie przynależności.....	574
Identyfikacja bieżących użytkowników za pomocą ADOX	575
Wyszukiwanie użytkowników z pustym hasłem	576
Ustawianie i usuwanie hasła	577
Zarządzanie prawami własności obiektów	578
Zarządzanie wieloma aplikacjami	578
Używanie SQL	579
Zabezpieczanie bazy danych krok po kroku.....	581
Częste błędy bezpieczeństwa.....	581
Część VIII Publikowanie w sieci za pomocą Accessa 2002.....	583
Rozdział 23. Konfiguracja serwera WWW do publikowania	
w sieci WWW	585
Środowisko programistyczne a środowisko produkcyjne	585
Wybór platformy	586
Personal Web Server i Peer Web Services.....	587
Internet Information Server.....	587
Co to jest Option Pack	588

Uruchomienie serwera WWW.....	589
Instalacja.....	589
NT Option Pack dla Windows 95/98	590
Microsoft Transaction Server 2.0.....	593
Zarządzanie i konfiguracja serwera WWW.....	595
Personal Web Manager	595
Microsoft Management Console.....	597
Zabezpieczanie aplikacji WWW	603
Różnice pomiędzy witryną a katalogiem wirtualnym	604
Co to jest IUSER ?	606
Typ systemu plików	606
Struktura katalogów i wymagane uprawnienia	607
ASP/HTML — położenie i uprawnienia.....	607
Bazy danych — położenie i uprawnienia.....	607
Rozdział 24. Przenoszenie Accessa 2002 do sieci WWW za pomocą komponentów sieciowych Office.....	611
Czym są komponenty sieciowe Office	611
Co potrafią komponenty sieciowe Office.....	612
Wymagane licencje na użycie komponentów sieciowych	612
Użycie formantu Office Arkusz	613
Rozpoczynamy	613
Użycie formantu w Accessie.....	613
Użycie formantu Office Wykres.....	616
Rozpoczynamy	616
Użycie formantu w Accessie.....	616
Użycie formantu Office Tabela przestawna	619
Rozpoczynamy	619
Rozdział 25. Użycie stron dostępu do danych.....	621
Czym są strony dostępu do danych?	621
Architektura oraz wymagania stron dostępu do danych	622
Tworzenie Twojej pierwszej strony dostępu do danych	622
Oglądanie strony dostępu do danych	625
Tworzenie interaktywnych odnośników.....	626
Łączenie komponentów sieciowych Office z DAP.....	627
Dodanie komponentu sieciowego Arkusz Excel.....	628
Dodanie komponentu sieciowego Wykres.....	629
Skrypty w stronach dostępu do danych	630
Zmiana źródła danych w trakcie działania strony	631
Rozdział 26. Publikowanie w sieci przy użyciu Accessa 2002 i Active Server Pages	633
Użycie Active Server Pages	633
Architektura Active Server Pages.....	634
Active Server Pages kontra CGI	635
Uruchomienie stron ASP.....	636

Rozpoczynamy pracę z Active Server Pages	636
Konstrukcja kodu ASP	639
Ograniczenia eksportu stron ASP	639
Active Server Pages	640
Silnik ASP	640
Skrypty wykonywane na serwerze	640
Obiekty aplikacji i sesji	646
Użycie obiektów żądań i odpowiedzi	647
Plik global.asa	649
Przykłady użycia obiektów ASP	650
Użycie ADO w aplikacjach ASP	651
Przykład: tworzenie strony WWW dostępnej dla członków grupy	652
Publikacja w sieci z Accessa 2002 przy użyciu XML	659
Podstawy XML	659
Programowe tworzenie pliku XML	662
Tworzenie wykresów przy użyciu formantu Wykres	664
Skorowidz	667

Rozdział 7.

Zaawansowane ADO

W tym rozdziale:

- ◆ Użycie dostawcy OLE DB dla Accessa 2002
- ◆ Uzyskanie dostępu do danych w bazach nierelacyjnych poprzez ADO
- ◆ Zaawansowana obróbka danych przy użyciu ADO
- ◆ Definiowanie danych przy użyciu ADOX

Teraz, kiedy już umiesz używać obiektów ADO i porównałeś je z DAO, przyjrzyj się tym cechom ADO, które pomogą Ci tworzyć aplikacje w Accessie 2002.

Użycie dostawcy OLE DB dla Jet w Accessie 2002

Gdy używasz ADO w aplikacjach Accessa, najprawdopodobniej będziesz korzystał z dostawcy OLE DB dla Jet (użycie dostawcy OLE DB dla SQL Server omówiono w rozdziale 15.). Dostawca ten pozwala na bezpośredni dostęp do plików typu MDB. Zdolność bezpośredniej komunikacji ze źródłem danych to wielki krok naprzód dla ADO. Aby użyć dostawcy OLE DB dla Jet, musisz podać jego nazwę i pełną ścieżkę dostępu do bazy danych jako część ciągu połączeniowego. Aby zachować przejrzystość, umieszczono w ADO właściwość `Provider`, której odpowiednie ustawienie umożliwi użycie dostawcy dla Jet. Jego unikatowym CLASSID w ciągu połączenia jest *Microsoft.Jet.OLEDB.4.0*. Przedstawiony tu przykład pokazuje, jak tworzyć połączenie do bazy Accessa przy użyciu dostawcy OLE DB dla Jet:

```
Dim Connection As ADODB.Connection
Set Connection = New ADODB.Connection

Connection.Provider = "Microsoft.Jet.OLE.DB.4.0"
Connection.ConnectionString = "data source= C:\databases\sample.mdb"
Connection.Open
```

Jeśli zachodzi konieczność określenia innych informacji dotyczących połączenia (np. hasła bazy danych czy pliku grupy roboczej), dołącz je do ciągu połączeniowego, poprzedzając średnikiem. W tabeli 7.1 znajdziesz charakterystyczne dla dostawcy wła-

ściwości połączenia, których możesz użyć w ciągu połączeniowym. Inne charakterystyczne dla dostawcy właściwości, znajdujące się w zbiorze właściwości połączenia, zostaną zignorowane.

Tabela 7.1. Właściwości ciągów połączenia charakterystyczne dla dostawcy OLE DB dla Microsoft Jet

Nazwa	Opis
Compact ReclaimedSpace Amount	Średnia ilość miejsca odzyskiwana podczas kompaktowania bazy danych
Connection Control	Kontroluje dostęp użytkownika do bazy danych
Create System Database	Wykorzystuje się ją do tworzenia systemowej bazy danych podczas tworzenia nowego źródła danych
Database Locking Mode	Tryb używany podczas blokowania bazy danych; więcej informacji na ten temat znajdziesz w rozdziale 21., dotyczącym obsługi wielu użytkowników. Zauważ, że baza może być otwarta jednocześnie tylko w jednym trybie. Użytkownik, który jako pierwszy otwiera bazę, określa jej tryb blokowania
Database Password	Hasło używane do otwarcia bazy danych. Od hasła użytkownika różni się tym, że hasło bazy danych dotyczy pliku, a nie konkretnego użytkownika. Więcej informacji na ten temat znajdziesz w rozdziale 21.
Don't Copy Locale on Compact	Właściwość wykorzystywana do sterowania kopiowaniem informacji narodowych podczas kompaktowania bazy
Encrypt Database	Wykorzystuje się ją do sterowania szyfrowaniem bazy w trakcie kompaktowania
Engine Type	Określenie silnika, który aktualnie jest wykorzystywany w celu uzyskania dostępu do danych
Exclusive Asynch Delay	Określa maksymalną długość przerwy w czasie zapisu asynchronicznego
Flush Transaction Timeout	Określa czas oczekiwania, po którym dane asynchroniczne są zapisywane na dysk
Global Bulk Transactions	Określa, czy masowe operacje SQL są wykonywane w formie transakcji. Właściwość ta wskazuje ustawienie domyślne dla wszystkich operacji w aktualnym połączeniu
Global Partial Bulk Ops	Właściwość ta określa zachowanie Jet w przypadku błędu w masowych operacjach SQL DML. Może zostać unieważniona przez ustawienie właściwości Jet na OLEDB:Partial Bulk Ops
Implicit Commit Synch	Definiuje wewnętrzny sposób zapisu transakcji — synchroniczny lub asynchroniczny
Lock Delay	Określa opóźnienie (w milisekundach), po którym nastąpi ponowna próba wykonania blokady
Lock Retry	Określa ilość prób nałożenia blokady
Max Buffer Size	Podaje maksymalną ilość pamięci dla bufora Jet, przeznaczoną na zapisywane dane
Max Locks Per File	Definiuje maksymalną ilość blokad w bazie danych; wartością domyślną jest 9500

Tabela 7.1. Właściwości ciągów połączenia charakterystyczne dla dostawcy OLE DB dla Microsoft Jet (ciąg dalszy)

Nazwa	Opis
New Database Password	Określa nowe hasło bazy danych
ODBC Command Time Out	Podaje czas (w milisekundach), po którym wykonanie zdalnego polecenia ODBC nie powiedzie się
Page Locks to Table Lock	Określa ilość blokad strony, zanim blokady te zostaną zamienione na blokadę tabeli
Page Timeout	Ilość milisekund po których baza danych uważa dane z bufora za nieaktualne.
Recycle Long-Valued Pages	Określa, czy strony BLOB będą „agresywnie” odzyskiwane
Registry Path	Ścieżka dostępu do klucza rejestru dla Jet. Nie zawiera znacznika <i>HKEY_LOCAL_MACHINE</i> . Wartość ta może zostać zamieniona na drugorzędną lokalizację w celu przechowywania wartości rejestru aplikacji, które nie są współdzielone z innymi aplikacjami korzystającymi z Jet na tym komputerze. Przykładowo, ustawienie dla Accessa 2002 to: <i>SOFTWARE\Microsoft\Office\9.0\Access\Jet\4.0\Engines</i>
Reset ISAM Stats	Wskazuje, czy liczniki wydajności powinny być zerowane po każdym pobraniu danych
Shared Asynch Delay	Określa opóźnienie (w milisekundach) zapisu asynchronicznego w przypadku otwarcia bazy danych w trybie wielodostępu
System database	Podaje lokalizację systemowej bazy Jet, która ma być używana przez określonych użytkowników. Powoduje to unieważnienie wartości ustalonej w rejestrze lub klucza rejestru <i>systemdb</i> , używanego wraz z Jet — OLEDB:Registry Path. Właściwość ta może zawierać ścieżkę dostępu do pliku
Transaction Commit Mode	Określa, czy zapis podczas zatwierdzania transakcji jest asynchroniczny, czy synchroniczny
User Commit Sync	Określa, czy zmiany w transakcji są zapisywane w sposób synchroniczny, czy asynchroniczny

Oprócz właściwości umieszczonych w tabeli 7.1 silnik bazy danych Microsoft Jet posiada kilka opcji, których ustawienie określa zachowanie silnika. Opcje te często mają bezpośredni wpływ na jego wydajność. Domyślnie, po uruchomieniu Jet używane są wartości znajdujące się w rejestrze, w kluczu *\HKEY_LOCAL_MACHINES\Software\Microsoft\Jet*. Istnieje jednak możliwość tymczasowego unieważnienia tych ustawień. W przypadku ADO wartości te stanowią część ciągu połączenia. Stałe ciągu połączenia znajdują się w tabeli 7.2.

Następujący fragment kodu powoduje otwarcie zabezpieczonej bazy, określając jej hasło i plik grupy roboczej:

```
Dim Connection As ADODB.Connection
Set Connection = New ADODB.Connection

Connection.Provider = "Microsoft.Jet.OLE.DB.4.0"
Connection.ConnectionString = "data source= C:\databases\sample.mdb"&_
";Jet OLEDB:Database Password=supersecret " &_
"; OLEDB:System database=" & "c:\windows\system\system.mdw"
Connection.Open
```

Tabela 7.2. Opcje ciągu połączenia ADO

DAO	ADO
dbPageTimeout	Jet OLEDB:Page Timeout
dbSharedAsyncDelay	Jet OLEDB:Shared Async Delay
dbExclusiveAsyncDelay	Jet OLEDB:Exclusive Async Delay
dbLockRetry	Jet OLEDB:Lock Retry
dbUserCommitSync	Jet OLEDB:User Commit Sync
dbImplicitCommitSync	Jet OLEDB:Implicit Commit Sync
dbMaxBufferSize	Jet OLEDB:Max Buffer Size
dbMaxLocksPerFile	Jet OLEDB:Max Locks Per File
dbLockDelay	Jet OLEDB:Lock Delay
dbRecycleLVs	Jet OLEDB:Recycle Long-Valued Pages
dbFlushTransactionTimeout	Jet OLEDB:Flush transaction Timeout

CurrentProject.Connection

Gdy chcesz uzyskać dostęp do aktualnej bazy poprzez kod ADO, Access umożliwia Ci „pójście na skróty”, oddając do Twojej dyspozycji odnośnik do obiektu, Connection — CurrentProject.Connection. Używając tej składni, możesz z łatwością przydzielić aktualnej bazie obiekt Connection. Odnośnik ten pozwala zaoszczędzić czas, jeśli chcesz utworzyć zestaw rekordów.

```
Dim Recordset As ADODB.Recordset
Set Recordset = New ADODB.Recordset
Recordset.Open "Customers", CurrentProject.Connection
```



Zalecamy ostrożność podczas korzystania z CurrentProject.Connection. Pamiętaj, że polecenie to nie może być używane w innym środowisku niż Access 2002. Jeśli zamierzasz wykorzystać swój kod w innej aplikacji niż Access (np. Visual Basic), do utworzenia połączenia musisz użyć dłuższej składni.

Wykorzystanie Microsoft Data Links do podłączenia się do bazy danych

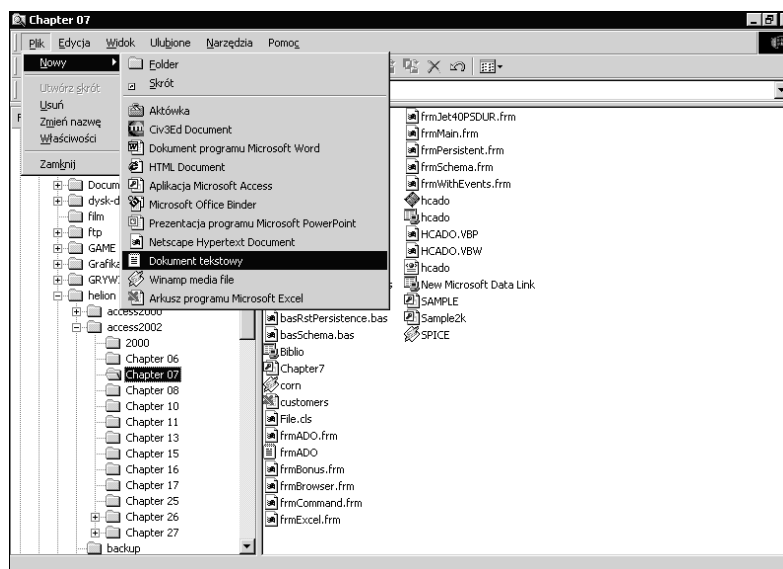
Dziś, gdy obiekty ADO są podstawowym sposobem uzyskiwania dostępu do danych, twórcy Accessa uznali, że programiści potrzebują standardowego sposobu obsługiwania, ładowania, otwierania i zarządzania informacjami o połączeniu OLE DB, przypominającego dawne narzędzie do zarządzania i administrowania sterownikami ODBC. Możliwość tę dają Microsoft Universal Data Links (UDL). Mechanizm UDL umożliwia zapisywanie informacji o połączeniu w pliku UDL, a następnie otwarcie obiektu Connection w ADO, na podstawie informacji zapisanych w tym pliku. Możliwość ta będzie bardzo przydatna, gdy będziesz chciał przetestować podłączenie bazy danych do różnych komputerów bez zmiany kodu.

Mechanizm Microsoft Universal Data Links składa się z:

- ♦ graficznego interfejsu użytkownika, służącego do tworzenia połączeń OLE DB;
- ♦ interfejsu automatyzacji.

Zanim zaczniesz używać UDL w aplikacjach VB lub VBA, musisz najpierw je utworzyć. Tworzenie UDL nie jest skomplikowane. Uruchom Eksploratora Windows, z menu *Plik* wybierz *Nowy*, a następnie *Dokument tekstowy* (rysunek 7.1).

Rysunek 7.1.
Tworzenie nowego pliku UDL

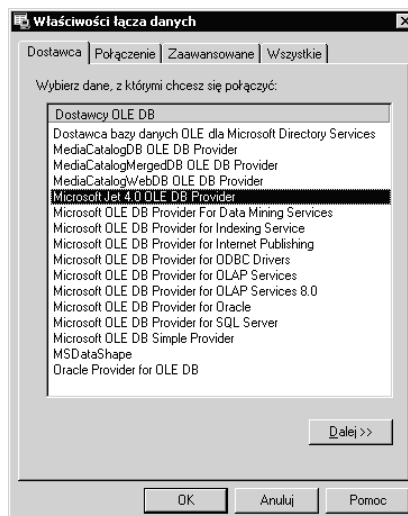


Po utworzeniu pliku UDL kliknij go dwukrotnie, aby rozpocząć edycję. Przedstawiony na rysunku 7.2 pierwszy ekran zawiera listę wszystkich zainstalowanych na Twoim komputerze dostawców OLE DB. Wybierz jednego z nich i kliknij *Dalej*, aby przejść do zakładki *Połączenie*. Teraz możesz określić wszystkie, charakterystyczne dla dostawcy informacje dotyczące połączenia (rysunek 7.3). W naszym przykładzie użyliśmy dostawcy OLE DB dla Jet, więc musieliśmy określić ścieżkę dostępu, identyfikator użytkownika i hasło (gdybyśmy wybrali SQL Server, musieliśmy podać nazwę serwera i bazy danych). Następną zakładką, *Zaawansowane*, służy do określenia właściwości dostawcy, ustalanych zwykle przez takie właściwości, jak *CommandTimeout*. Ostatnia zakładka — *Wszystkie* — zawiera podsumowanie wszystkich wybranych właściwości.

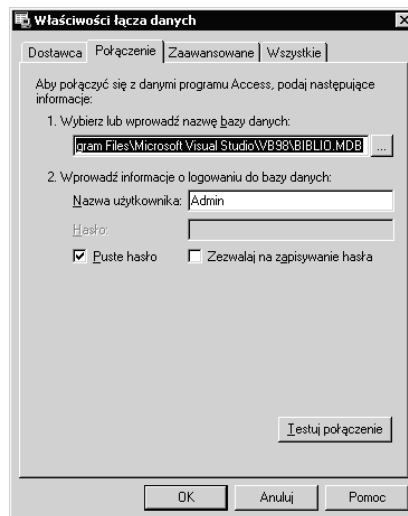
Teraz, gdy utworzyłeś już plik UDL, możesz umieścić go w kodzie VBA, aby utworzyć w ADO obiekt *Connection*. Służąca do tego składnia jest prosta. Utwórz obiekt *Connection* — tak, jak to zwykle robisz — i użyj metody *Open*. Następnie wskaż plik UDL za pomocą identyfikatora *File Name*:

```
Dim conn As ADODB.Connection
Set conn = New ADODB.Connection
Conn.Open "File Name=c:\chapter7.udl;"
```

Rysunek 7.2.
Wybór dostawcy
OLE DB



Rysunek 7.3.
Wprowadzanie
informacji
o połączeniu



Fragment kodu na wydruku 7.1 znajduje się na dołączonej do książki płycie CD. Pokazuje on, jak utworzyć obiekt Connection oparty na pliku UDL, umieszczonym w folderze bazy danych Northwind. Łączysz się z bazą i wypełniasz okno zawartością tabeli *Customers*. Bardzo przydatną cechą tego kodu jest to, że jeśli będziesz chciał użyć bazy Northwind w wersji SQL, wystarczy zmienić właściwości pliku UDL i uruchomić kod ponownie.

Wydruk 7.1. Użycie UDL w kodzie VBA

```
Private Sub OpenViaLink()

On Error GoTo Except

Dim Connection As ADODB.Connection
Set Connection = New ADODB.Connection
```

```

Connection.Open "File Name=c:\temp\biblio.udl;"

Dim Recordset As ADODB.Recordset
Set Recordset = New ADODB.Recordset
Recordset.Open "Select * From publishers", Connection

Do Until Recordset.EOF
    Debug.Print Recordset.Fields(1)
    Recordset.MoveNext
Loop

Recordset.Close
Set Recordset = Nothing

Connection.Close
Set Connection = Nothing

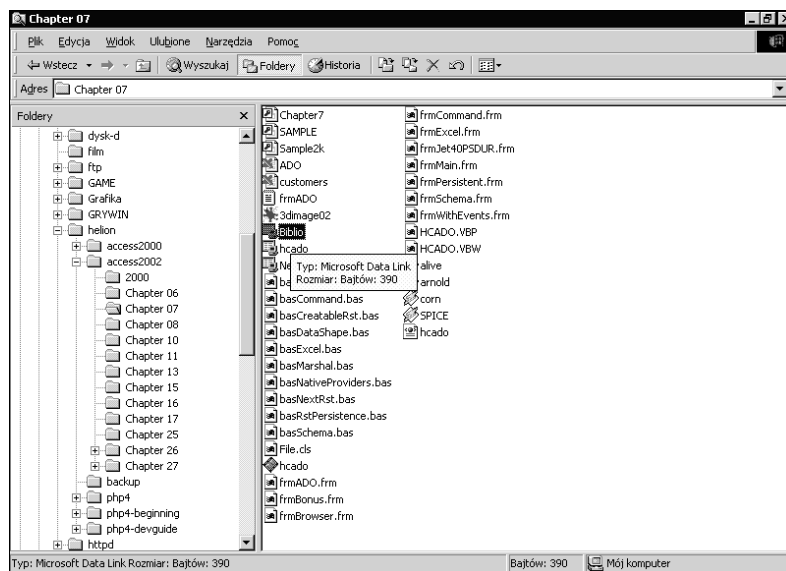
Except:
    MsgBox Err.Description
End Sub

```

Zarządzanie plikami UDL

Aby zmienić plik UDL, należy w programie Windows Explorer kliknąć dwukrotnie jego ikonę lub wpisać w oknie *Uruchom*, wywoływanym z menu *Start*, pełną ścieżkę do pliku UDL (rysunek 7.4).

Rysunek 7.4.
Dwukrotne kliknięcie pliku UDL uruchamia okno właściwości łącza danych

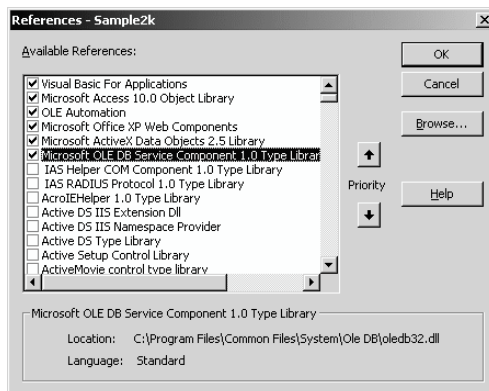


Programowe zarządzanie plikami UDL

Ponieważ pliki UDL posiadają swój własny interfejs automatyzacji, możesz nimi zarządzać i kontrolować je programowo. Aby użyć UDL w kodzie VBA, ustaw odwołanie na

typ biblioteki o nazwie Microsoft OLE DB Service Component 1.0 Type Library (rysunek 7.5).

Rysunek 7.5.
Ustawianie
odnośnika
na Microsoft
OLE DB Service
Component 1.0
Type Library



Po ustawienie odwołania możesz rozpocząć zarządzanie plikami UDL. Dodatkowo możesz poprosić użytkownika o podanie informacji o połączeniu OLE DB poprzez okna dialogowe UDL (możliwość ta jest przydatna w prototypowych aplikacjach lub wersjach demonstracyjnych produktów przeznaczonych dla użytkowników zaznajomionych z funkcjonowaniem Microsoft Data Links). Możesz tego dokonać, wykorzystując metodę PromptNew obiektu DataLinks. Kod na wydruku 7.2 prosi użytkownika o podanie informacji o połączeniu OLE DB poprzez okno dialogowe UDL, a następnie łączy się z bazą danych i otwiera zestaw rekordów.

Wydruk 7.2. Programowe UDL

```
Private Sub OpenUDLDialog()

    On Error GoTo Except

    Dim UDL As MSDASC.DataLinks
    Set UDL = New MSDASC.DataLinks

    Dim Connect As String
    Connect = UDL.PromptNew

    Dim Recordset As ADODB.Recordset
    Set Recordset = New ADODB.Recordset

    ' Zakładamy, że użytkownik wybrał bazę danych Biblio.mdb
    Recordset.Open "Select * From Publishers", Connect
    MsgBox Recordset.Fields(1)
    Exit Sub

Except:
    MsgBox Err.Description
End Sub
```


Uzyskanie poprzez ADO dostępu do danych w bazach nierelacyjnych

Nawiązując do omówionego w rozdziale 6. wykorzystania Accessa do komunikowania się z danymi nierelacyjnymi (np. danymi arkusza Excel) trzeba powiedzieć w tym miejscu, że ADO posiada kilka zaawansowanych opcji, które ułatwią Ci pracę z tym typem danych. Przyjrzymy się tym opcjom. Są nimi:

- ♦ lista użytkowników Jet;
- ♦ tworzenie zestawów rekordów;
- ♦ kształtowanie danych.

Lista użytkowników Jet

Dostawca OLE DB dla Jet umożliwi Ci otwarcie zestawu rekordów, zawierającego nazwy użytkowników aktualnie zalogowanych do bazy, której nazwę określisz w połączeniu. Dzięki temu nie będziesz już musiał korzystać z dostarczanego z poprzednimi wersjami Jet narzędzia LDBView. Kod na wydruku 7.3 przedstawia sposób utworzenia zestawów rekordów z aktualnymi użytkownikami bazy danych. Użyliśmy w nim metody `OpenSchema` obiektu `Connection` z parametrem charakterystycznym dla aparatu JET/JOLT. Zauważ, że ADO nie podpowiada Ci żadnych stałych. Będziesz musiał ręcznie wprowadzić wartość GUID (`{947bb102-5d43-11d1-bdbf-00c04fb92675}`). Pozostaje nam mieć nadzieję, że niedogodność ta zostanie poprawiona w kolejnej wersji obiektów ADO.

Wydruk 7.3. Użycie listy użytkowników

```
Sub UserRoster()

    On Error GoTo Except

    Dim Connection As ADODB.Connection
    Set Connection = New ADODB.Connection
    Connection.Provider = "Microsoft.Jet.OLEDB.4.0"
    Connection.ConnectionString = "file name=c:\temp\biblio.udl"
    Connection.Open

    Dim Recordset As ADODB.Recordset
    Set Recordset = Connection.OpenSchema(adSchemaProviderSpecific, , _
    "{947bb102-5d43-11d1-bdbf-00c04fb92675}")

    Do Until Recordset.EOF
        Debug.Print Recordset!COMPUTER_NAME
        Debug.Print Recordset!LOGIN_NAME
        Debug.Print Recordset!CONNECTED
        Debug.Print Recordset!SUSPECTED STATE
    Loop
End Sub
```

```
Recordset.MoveNext  
Loop  
  
Exit Sub  
Except:  
MsgBox Err.Description  
End Sub
```

Tworzone zestawy rekordów

Obiekty ADO dają Ci możliwość tworzenia zestawów rekordów „z niczego”. Możesz teraz korzystać z metod `Fields.Append` i `AddNew`, aby dodawać pola i dane do zestawu rekordów. Technika ta nadaje się świetnie do tworzenia procedur pobierających dane z nierelacyjnych źródeł (np. folderu z plikami) i zwracających zestaw rekordów. Dzięki ADO aplikacja może przez cały czas współpracować z zestawami rekordów, a jej kod jest dużo prostszy. Poniższy przykład przedstawia sposób utworzenia zestawu rekordów, zawierającego nazwy plików znajdujących się w określonym folderze. Jeśli umieścisz ten kod w module, będziesz go mógł użyć w wielu aplikacjach. My na przykład wykorzystaliśmy go do sporządzenia listy wszystkich plików typu ZIP w katalogu serwera FTP, a następnie przejrzania ich i utworzenia strony, z której można te pliki pobrać. Przewagą tej techniki nad dostawcą NT Active OLE DB jest fakt, iż działa ona w systemie Windows 95. Wydruk 7.4 pokazuje, w jaki sposób można otworzyć zestaw rekordów oparty na liście nazw plików w folderze.

Wydruk 7.4. Tworzenie zestawu rekordów z nierelacyjnych danych

```
Sub CreateTableRst_Files(Optional Path As String = "c:\temp")  
  
On Error GoTo Except  
  
Dim Recordset As ADODB.Recordset  
Set Recordset = New ADODB.Recordset  
  
Recordset.CursorLocation = adUseClient  
Recordset.Fields.Append "FileName", adVarChar, 255, adFldRowID  
Recordset.Fields.Append "Extention", adChar, 3, adFldFixed  
Recordset.Open , , adOpenStatic, adLockBatchOptimistic  
  
If Right(Path, 1) <> "\" Then Path = Path & "\"  
Path = Dir(Path & "*.*", vbNormal)  
  
Do While strPath > ""  
Recordset.AddNew Array("FileName", "Extention"), _  
Array(strPath, Right(strPath, 3))  
  
Path = Dir  
Loop  
  
Recordset.MoveFirst  
  
Do Until Recordset.EOF  
Debug.Print Recordset!FileName  
Recordset.MoveNext
```

```

Loop
Exit Sub

Except:
    MsgBox Err.Description
End Sub

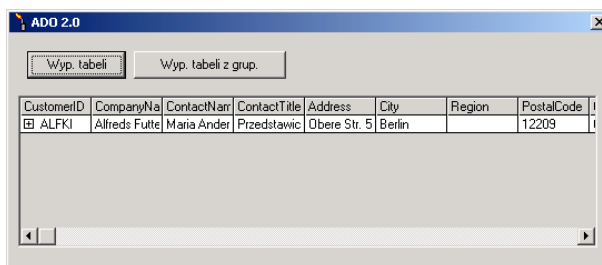
```

Kształtowanie danych

Umożliwienie przeglądania danych w zhierarchizowany sposób jest w obiektach ADO bardzo proste. Możesz użyć nowej zdolności ADO, o nazwie *Data Shaping* (ang. kształtowanie danych), aby przeglądać nadrzędne i podrzędne rekordy w jednym zestawie. Możesz użyć dostawcy MSDataShape OLE DB do tworzenia hierarchii opartych na relacjach i hierarchii grupowania. Tworzenie tego typu hierarchicznych zestawów rekordów może ułatwić dość pracochłonną obróbkę hierarchicznych danych. Przykładowo, na rysunku 7.6 znajduje się grupowa hierarchia wszystkich zamówień klientów z bazy Northwind, przeglądana w nowej siatce, o nazwie Microsoft Hierarchical FlexGrid (wersja OLAP popularnej FlexGrid). Formant ten dołączony jest do pakietu Office 2002 w wersji Developer Edition oraz do Visual Basic 6 (nasze rysunki i listingi zostały stworzone w VB 6 i używają bazy danych Accessa jako źródła danych). Zwróć uwagę na znak plus, znajdujący się przy każdym z rekordów. Umożliwia on rozszerzanie ich i przeglądanie ich szczegółów, co pokazuje rysunek 7.7. Zanim powstał dostawca MSDataShape OLE DB, utworzenie takiej siatki wymagało napisania wielu linijek kodu.

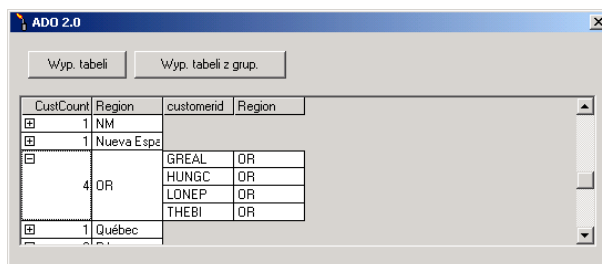
Rysunek 7.6.

Zestaw rekordów
uzyskany poprzez
dostawcę
MSDataShape OLE
DB w Visual Basic



Rysunek 7.7.

Zestaw rekordów
GrupujWedług
w Visual Basic



Wydruk 7.5. Tworzenie rekordu z ukształtowanymi danymi i wypełnianie siatki MSHFlexGrid w VB

```

Sub CubeGroupHierarchy()

    Dim ConnectionString As String

```

```

ConnectionString = "Provider=MSDataShape" & _
";data provider=Microsoft.Jet.OLEDB.4.0" & _
";data source=" & App.Path & "\sample.mdb"

Dim Recordset As ADODB.Recordset
Set Recordset = New ADODB.Recordset

Recordset.Source = "shape {Select customerid, " & _
"Region from Customers} rst1 " & _
"COMPUTE COUNT (rst1.customerid) AS CustCount, rst1 By Region "

Recordset.Open , ConnectionString
Set frmADO.MSHFlexGrid1.Recordset = Recordset

End Sub

```

Jak to widać w tym fragmencie kodu, musisz użyć specjalnej, powiązanej z *Data Shaping* składni SQL. Kompletną listę zastosowań tej składni znajdziesz w dostarczanym wraz z Visual Basic 6 i pakietem Office Developer 2000 dokumentacji MSDN oraz na stronie WWW o adresie <http://www.microsoft.com/data>. Możesz również użyć tego kodu do operacji *Data Shaping* i przeniesienia danych do programu Microsoft Excel. Kod na wydruku 7.6 przedstawia sposób, w jaki — za pomocą poprzedniego przykładu — można przenieść dane do arkusza programu Excel.

Wydruk 7.6. Użycie *Data Shaping* w programie Microsoft Excel

```

Function GetDatabase() As String
' Zmień tę funkcję tak, aby zwracała ścieżkę do przykładowej bazy danych
Getatabase = "c:\temp\sample.mdb"
End Function

Sub CUBRelation()

Dim Recordset As ADODB.Recordset
Set Recordset = New ADODB.Recordset

Dim ConnectionString As String
ConnectionString = "Provider=MSDataShape;" & _
";data provider=Microsoft.Jet.OLEDB.4.0" & _
";data source=" & GetDatabase()

Recordset.Source = _
"shape {Select * from customers where Customerid='ALFKI'}"&_
" Append ({Select * From Orders} As rsOrders " & _
"RELATE customerid to customerid)"

Recordset.ActiveConnection = ConnectionString

Recordset.Open , , adOpenStatic, adLockBatchOptimistic

ShowRecordset Recordset

End Sub

Sub ShowRecordset(ByVal Recordset As ADODB.Recordset, _
Optional SkipClear As Boolean = False)

```

```
Dim Child As ADO.Recordset
Dim Col As ADO.Field
Dim CurrentColumn As Integer
Dim CurrentRow As Integer

If Not SkipClear Then
    Cells.Select
    Selection.ClearContents
End If

Do Until Recordset.EOF
    CurrentColumn = CurrentColumn + 1
    If Not SkipClear Then
        CurrentRow = 1
    Else
        CurrentRow = 2
    End If

    For Each Col In Recordset.Fields
        If col.Type <> adChapter Then
            Cells(intCurrentColumn, intCurrentRow) = _
                col.Name & ": " & col.Value
        Else
            Set Child = col.Value
            ShowRecordset Child, True
        End If

        CurrentRow = CurrentRow + 1
    Next

    Recordset.MoveNext
Loop

Cells(1, 1).Select

End Sub
```

Zaawansowana obróbka danych przy użyciu ADO

W ostatnim rozdziale pokazaliśmy, jak tworzyć zestawy rekordów zawierające dane i jak korzystać z obiektów Command. Teraz pokażemy kilka dodatkowych możliwości obiektów ADO. Przyjrzymy się kolejno:

- ♦ modyfikowaniu danych w zestawie rekordów;
- ♦ trwałym zestawom rekordów.

Modyfikowanie danych w zestawie rekordów

Aby modyfikować dane w zestawie rekordów, musisz jedynie przejść do niego i rozpocząć edycję. Po przejściu do kolejnego rekordu lub użyciu metody Update dokonane przez Ciebie zmiany zostaną zapisane. Jest to duży krok naprzód w porównaniu z obiektami DAO, gdzie musiałeś najpierw użyć polecenia Edit, a następnie obowiązkowo użyć polecenia Update. Kod na wydruku 7.7 pokazuje, w jaki sposób można dokonać edycji danych.

Wydruk 7.7. Edycja rekordu

```
Sub EditRecords()

    On Error GoTo Except

    Dim Recordset As ADODB.Recordset
    Set Recordset = New ADODB.Recordset

    Dim SQL As String
    SQL = "Select * From Customers Where CustomerID=1"

    Recordset.Open SQL, CurrentProject.Connection, _
        adOpenKeyset, adLockOptimistic

    Recordset!CompanyName = "Nowa nazwa"
    Recordset.Update

Exit Sub

Except:
    MsgBox Err.Description
End Sub
```



Możesz otworzyć zestaw rekordów bazujący na wyrażeniu SQL wymagającym klucza podstawowego; zajmie to znacznie mniej czasu, niż otwieranie całej tabeli i w niej dopiero wyszukiwanie żadanego rekordu.



Ponieważ zmiany dokonywane w zestawie rekordów uwzględniane są bez użycia polecenia Update (wystarczy do tego przejście do kolejnego rekordu), zalecamy ostrożność podczas edycji danych.

Dodawanie rekordu

Dodawanie rekordu przebiega w prawie identyczny sposób, jak jego edycja. Musisz otworzyć zestaw rekordów i użyć metody AddNew. W przypadku operacji Edit zastosowanie mają te same zasady, co w przypadku Update. Po dodaniu rekordu znajdziesz się w aktualnym rekordzie. Wydruk 7.8 zawiera przykład dodawania rekordu do tabeli:

Wydruk 7.8. Dodawanie rekordu do tabeli

```
Sub AddRecords()

    On Error GoTo Except
```

```

Dim Recordset As ADODB.Recordset
Set Recordset = New ADODB.Recordset

Recordset.Open "Customers", CurrentProject.Connection, _
    adOpenDynamic, adLockOptimistic, adCmdTableDirect

Recordset.AddNew
' Należy zdefiniować algorytm generowania unikatowego identyfikatora
Recordset!CustomerID = 3
Recordset!Name_Pre = "Pan"
Recordset!Name_First = "Jan"
Recordset!Name_Last = "Nowak"
Recordset.Update
MsgBox "ID nowego autora: " & Recordset!CustomerID

Recordset.Close
Set Recordset = Nothing
Exit Sub
Except:
    MsgBox Err.Description
End Sub

```

Być może zauważyłeś, że otworzyłeś tabelę ze stałą `adCmdTableDirect`. Jeśli użyjesz tej stałej z typem kursora `adOpenDynaset` i blokowaniem optymistycznym, osiągniesz taki sam efekt, jak przy użyciu polecenia `dbOpenTable` w DAO. Tabela 7.3 zawiera wszystkie opcje kursorów i blokowania ADO oraz ich odpowiedniki w DAO:

Tabela 7.3. *Kombinacje obiektów ADO dla Accessa*

Typ kursora	Opcje i typy blokowania	Odpowiednik w DAO
<code>adOpenForwardOnly</code>	<code>adLockReadOnly</code>	<code>dbOpenSnapshot</code> <code>dbForwardOnly</code>
<code>AdOpenKeySet</code>	<code>adLockReadOnly</code>	Brak odpowiednika
<code>AdOpenKeySet</code>	<code>adLockPessimistic</code>	<code>dnOpenDynaset</code>
<code>AdOpenKeySet</code>	<code>adLockOptimistic</code>	<code>dnOpenDynaset</code>
<code>AdOpenKeySet</code>	<code>adLockBatchOptimistic</code>	<code>dnOpenDynaset</code>
<code>AdOpenStatic</code>	<code>adLockReadOnly</code>	<code>dnOpenDynaset</code>
<code>adOpenDynamic</code>	<code>adLockOptimistic</code> , <code>adCmdTableDirect</code>	<code>dbOpenTable</code>

Trwałe zestawy rekordów

Jeśli musisz zadbać również o obsługę użytkowników nie podłączonych do sieci, pojęcie trwałości rekordów stanowić będzie dla Ciebie fantastyczny podarunek od firmy z Redmond. Teraz masz bowiem możliwość zapisania zestawu rekordów na dysku i obrabiania go w aplikacji VB lub VBA. Później będziesz mógł ponownie włączyć ten zestaw do bazy. Aby tego dokonać, otwórz zestaw rekordów ADO z typem blokowania `adLockBatchOptimistic`, używając kursora `client`. Następnie użyj metody `Save` obiektu `Recordset`. Musisz określić nazwę pliku i format, w jakim ma być zapisany. ADO oferuje następujące możliwości:

- ◆ adPersistADTG;
- ◆ adPersistXML.

Jeśli wybierzesz format adPersistADTG lub adPersistXML, będziesz mógł później z łatwością otworzyć ten plik w ADO. Wydruk 7.9 pokazuje, jak otworzyć zestaw rekordów, zmienić rekord, zapisać zmiany do pliku, otworzyć rekord ponownie i włączyć go z powrotem do bazy.

Wydruk 7.9. *Zapisywanie zestawu rekordów na dysku, jego ponowne otwarcie oraz włączenie do bazy danych*

```
Sub Persist()  
  
    On Error GoTo Except  
  
    Dim SQL As String  
    Dim Msg As String  
    Dim NewValue As String  
  
    SQL = "Select * From Customers Where CustomerID='ALFKI'"  
  
    Dim Connection As ADODB.Connection  
    Set Connection = New ADODB.Connection  
    Connection.Provider = "Microsoft.Jet.OLEDB.3.51"  
    Connection.ConnectionString = "data source=" & _  
        App.Path & "\sample.mdb"  
  
    Connection.Open  
  
    Dim Recordset As ADODB.Recordset  
    Set Recordset = New ADODB.Recordset  
  
    Recordset.CursorLocation = adUseClient  
    Recordset.Open SQL, _  
        Connection, adOpenStatic, adLockBatchOptimistic  
  
    Msg = "Proszę podać nową nazwę firmy dla: " & _  
        Recordset!CompanyName & vbNewLine & _  
        "Identyfikator klienta=" & Recordset!CustomerID  
  
    NewValue = InputBox(Msg)  
  
    Recordset.Update "CompanyName", NewValue  
  
    On Error Resume Next  
    Kill App.Path & "\hcado.dat"  
  
    On Error GoTo Except  
    Recordset.Save App.Path & "\hcado.dat", adPersistADTG  
  
    Recordset.Close  
    Set Recordset = Nothing  
  
    MsgBox "Zapisane do pliku: " & App.Path & "\hcado.dat", vbInformation
```



```
Exit Sub

Except:
    MsgBox Err.Description
End Sub

Sub Resynch()

    On Error GoTo Except

    Dim SQL As String
    SQL = "Select * From Customers Where CustomerID='ALFKI'"

    Dim Recordset As ADODB.Recordset
    Set Recordset = New ADODB.Recordset

    ' Otwarcie recordsetu zapisanego w pliku
    Recordset.Open App.Path & "\hcado.dat", , _
        adOpenStatic, adLockBatchOptimistic, adCmdFile

    Dim Msg As String
    Msg = "Poprzednie wartości: " & _
        Recordset!CompanyName.OriginalValue

    Msg = Msg & vbCrLf & _
        "Wartości w pliku      : " & Recordset!CompanyName

    MsgBox Msg, vbInformation, "Poprzednie i zapisane wartości"

    Dim Connection As ADODB.Connection
    Set Connection = New ADODB.Connection
    Connection.Provider = "Microsoft.Jet.OLEDB.3.51"
    Connection.ConnectionString = "data source=" & _
        App.Path & "\sample.mdb"
    Connection.Open

    Recordset.ActiveConnection = Connection
    Recordset.UpdateBatch

    MsgBox "Baza danych została zsynchronizowana!", vbInformation

Exit Sub

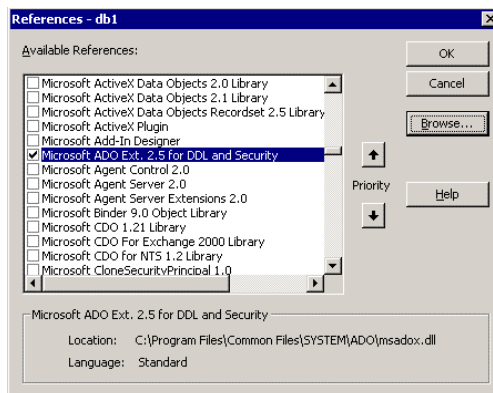
Except:
    MsgBox Err.Description
End Sub
```

Definiowanie danych przy użyciu ADOX

Korzystając z obiektów ADO, możesz z łatwością dokonywać definiowania danych. Aby tworzyć i modyfikować tabele, kwerendy i inne obiekty związane z danymi, na przykład indeksy, musisz skorzystać ze specjalnej biblioteki ADO o nazwie ADOX.

Aby użyć ADOX, musisz ustawić odnośnik na *Microsoft ADO Ext. 2.5 For DDL and Security*, wybierając z menu *Tools* pozycję *References* (rysunek 7.8).

Rysunek 7.8.
*Microsoft ADO
Ext. 2.5 For DDL
and Security*



Gdy ustawiłeś już ten odnośnik, jesteś gotów do korzystania z ADOX.

Obiekt Catalog

Jeśli korzystasz z ADOX, wszystko, co robisz, związane jest z obiektem *Catalog*. Obiekt ten oznacza bazę danych. Pracując w Accessie, musisz jedynie ustawić jej właściwość — *ActiveConnection* — na prawidłowy obiekt *Command*.

```
Dim Catalog As ADOX.Catalog
Set Catalog = New ADOX.Catalog
'Wskaż obiektowi Catalog
'aktualną bazę danych
Catalog.ActiveConnection = CurrentProject.Connection
```

Tworzenie bazy danych

Aby utworzyć nową, pustą bazę danych przy użyciu ADOX, musisz użyć metody *Create* obiektu *Catalog* i podać ścieżkę dostępu do nowej bazy (wydruk 7.10).

Wydruk 7.10. Tworzenie bazy danych przy użyciu ADOX

```
Sub CreateDataBase()
    On Error GoTo Except
    Dim Catalog As ADOX.Catalog
    Set Catalog = New ADOX.Catalog
    Catalog.Create "provider=Microsoft.JET.OLEDB.4.0;" & _
        "data source=C:\newdb.mdb"
    MsgBox "Baza danych utworzona!", vbInformation
    set Catalog = Nothing
Exit Sub
```

```

Except:
    MsgBox Err.Description
End Sub

```

Tworzenie tabel i pól

Aby utworzyć tabelę i jej pola, możesz utworzyć obiekt ADOX o nazwie table i dołączyć do niego kolumny (wydruk 7.11).

Wydruk 7.11. Tworzenie tabel i pól

```

Sub CreateTable_JOLT40()

    On Error GoTo Except

    Dim Catalog As ADOX.Catalog
    Set Catalog = New ADOX.Catalog
    Catalog.ActiveConnection = CurrentProject.Connection

    Dim Table As ADOX.Table
    Set Table = New ADOX.Table

    Table.Name = "Customer_ADO"
    Table.Columns.Append "CustomerID", adInteger
    Table.Columns.Append "Name_Prefix", adWChar, 25.
    Table.Columns.Append "Name_First", adWChar, 2
    Table.Columns.Append "Name_Last", adWChar, 50

    Catalog.Tables.Append Table

    Set Table = Nothing
    Set Catalog = Nothing

    Exit Sub

Except:
    MsgBox Err.Description
End Sub

```

Tworzenie dołączonej tabeli

Aby utworzyć w Accessie dołączoną tabelę, musisz utworzyć tabelę nie dołączając do niej żadnych pól, a następnie ustawić dwie właściwości:

- ♦ Jet OLE DB:Link Datasource;
- ♦ JET OLE DB:Remote Table Name.

Jako Link Datasource podaj ścieżkę dostępu do bazy danych, do której chcesz się dołączyć. Jako Remote Table Name ustaw nazwę tej bazy. Cały przykład znajduje się na wydruku 7.12.

Wydruk 7.12. Tworzenie dołączonej tabeli

```

Sub CreateAttachedJetTable()
    On Error GoTo Except

    Dim Catalog As ADOX.Catalog
    Set Catalog = New ADOX.Catalog
    Catalog.ActiveConnection = CurrentProject.Connection

    Dim Table As ADOX.Table
    Set Table = New ADOX.Table

    Table.Name = "Customers_Linked"
    Set Table.ParentCatalog = Catalog

    Table.Properties("Jet OLEDB:Link Datasource") = "c:\temp\newdb.mdb"
    Table.Properties("Jet OLEDB:Remote Table Name") = "Customers"

    Catalog.Tables.Append Table
    Set Catalog = Nothing

    Exit Sub
Except:
    MsgBox Err.Description
End Sub

```

Ustawianie właściwości pola

Kiedy utworzysz pole, możesz użyć w ADOX obiektu `Column`, aby ustawić właściwości charakterystyczne dla tego pola. Na przykład możesz ustawić typ pola na *Autonumerowanie*, nadając właściwości `AutoIncrement` wartość `True`:

```

Dim Column As ADOX.Column
Set Column = Table.Columns("CustomerID")
Column.Properties("AutoIncrement") = True

```

Istnieją także inne właściwości, które możesz ustawić dla obiektu `Column`. Umieściliśmy je w tabeli 7.4, wraz z ich odpowiednikami w DAO.

Tabela 7.4. Właściwości obiektu `Field`

Pole DAO		Obiekt <code>Column</code> w ADOX	
Właściwość	Wartość	Właściwość	Wartość
Attributes	dbAutoIncrField	AutoIncrement	True
Attributes	dbFixedField	ColumnAttributes	adColFixed
Attributes	dbHyperlinkField	Jet OLEDB:Hyperlink	True
Attributes	dbSystemField		
Attributes	dbVariableField	ColumnAttributes	inna niż adColFixed

Tworzenie indeksu

Tworząc indeks w ADOX, będziesz musiał użyć obiektu klucza. Po jego utworzeniu musisz dołączyć do niego pole, a następnie określić, czy ma to być klucz podstawowy, czy unikatowy oraz jaka ma być kolejność sortowania:

```
Set Index = New ADOX.Index
Index.Name = 'Primary Key'
Index.Columns.Append "CustomerID"
Index.Columns("CustomerID").SortOrder = adSortDescending
Index.IndexNulls = adIndexNullsDisallow
Index.PrimaryKey = True
Table.Indexes.Append Index
```

Właściwość `IndexNulls` ustawiona jest domyślnie na `adIndexNullsDisallow`, co oznacza, że wartości `Null` nie są w indeksie dopuszczalne i w przypadku, gdy pole w indeksie będzie taką wartość zawierać, nie zostanie dokonany żaden wpis do indeksu. W tabeli 7.5 znajdują się wartości akceptowane przez właściwość `IndexNulls`, wraz z ich odpowiednikami w DAO.

Tabela 7.5. Wartości dla właściwości `IndexNulls`

DAO	ADOX	Opis	
Wymagane Ignore Nulls	Index Nulls		
True	False	AdIndexNullsDisallow	Wartość Null nie jest akceptowana w polu indeks; wpis do indeksu nie zostanie dokonany
False	True	AdIndexNullsIgnore	Wartość Null dopuszczona w polu indeks; wpis do indeksu nie zostanie dokonany
False	False	AdIndexNullsIgnoreAny	Wartość Null dopuszczona w polu indeks; wpis do indeksu zostanie dokonany

Kod na wydruku 7.13 łączy te wszystkie przykłady, tworząc tabelę pola *Autonumerowanie* i indeks klucza głównego w tej tabeli.

Wydruk 7.13. Tworzenie tabeli z polem Autonumerowanie i indeksem klucza głównego

```
Sub CreateAutoNumberPK()
    On Error GoTo Except

    Dim Catalog As ADOX.Catalog
    Set Catalog = New ADOX.Catalog
    Catalog.ActiveConnection = CurrentProject.Connection

    Dim Table As New ADOX.Table
    Set Table = New ADOX.Table

    Table.Name = "Customer_ADO"
    Table.ParentCatalog = Catalog

    Table.Columns.Append "CustomerID", adInteger
    Table.Columns("CustomerID").Properties("Autoincrement") = True
```

```
Table.Columns.Append "Name_Prefix", adVarChar, 25
Table.Columns.Append "Name_First", adVarChar, 25
Table.Columns.Append "Name_Last", adVarChar, 50

Catalog.Tables.Append Table

Dim Index As ADOX.Index
Set Index = New ADOX.Index

Index.Name = "PrimaryKey"
Index.Columns.Append "CustomerID"
Index.Columns("CustomerID").SortOrder = adSortDescending

Index.IndexNulls = adIndexNullsDisallow
Index.PrimaryKey = True

Table.Indexes.Append Index

Set Table = Nothing
Set Index = Nothing
Set Catalog = Nothing

Exit Sub

Except:
    MsgBox Err.Description
End Sub
```

Tworzenie relacji w ADOX

Aby utworzyć relację w ADOX, musisz utworzyć obiekt Key, określić jego właściwości, a następnie dołączyć ten klucz do obiektu Table. Użyj następującej składni:

```
Catalog.Tables("Products").Keys.Append Key
```

Kod na wydruku 7.14 wykonuje wszystkie te czynności oraz tworzy relacje.

Wydruk 7.14. Tworzenie relacji

```
Sub CreateForeignKey()

    On Error GoTo Except

    Dim Catalog As ADOX.Catalog
    Set Catalog = New ADOX.Catalog
    Catalog.ActiveConnection = CurrentProject.Connection

    Dim Key As ADOX.Key
    Set Key = New ADOX.Key

    Key.Name = "CategoriesProducts"
    Key.Type = adKeyForeign
    Key.RelatedTable = "Categories"
    Key.Columns.Append "CategoryID"
    Key.Columns("CategoryID").RelatedColumn = "CategoryID"
    Key.UpdateRule = adRICascade
```

```
Catalog.Tables("Products").Keys.Append Key
Exit Sub
```

```
Except:
    MsgBox Err.Description
End Sub
```

Tworzenie kwerend w ADOX

Aby utworzyć kwerendę w ADOX, musisz użyć pustego obiektu `Command` i ustawić jego właściwość `CommandText`, a następnie dołączyć go do aktualnego obiektu `catalog`:

```
Command.CommandText = "Select * FROM Categories"
Catalog.Views.Append "qryCategories", Command
```

Gdy odnosisz się do kwerendy w Accessie, musisz pamiętać, że Microsoft Jet 4.0 zarządza wszystkimi tabelami, kwerendami, polami i indeksami za Ciebie. Jet definiuje kwerendy Accessa na dwa sposoby — jako widoki lub procedury.

Przez *widok* rozumiemy zwracającą wiersze, nieparametryczną kwerendę, podczas gdy *procedura* oznacza wszystkie inne kwerendy.

Tworzenie widoku

Microsoft Jet zajmie się za Ciebie definiowaniem kwerendy. Aby utworzyć widok, należy utworzyć kwerendę poprzez obiekt `Command` (wydruk 7.15).

Wydruk 7.15. Tworzenie widoku w ADOX

```
Sub CreateView()
    On Error GoTo Except

    Dim Command As New ADODB.Command
    Command.CommandText = "Select * FROM Categories"
    Dim Catalog As New ADOX.Catalog
    Catalog.ActiveConnection = CurrentProject.Connection

    Catalog.Views.Append "qryCategories", Command
    Set Catalog = Nothing
    Exit Sub
Except:
    MsgBox Err.Description
End Sub
```

Tworzenie procedury

Tworzenie procedury jest równie proste, jak tworzenie widoku. Wystarczy zdefiniować w obiekcie `Command` właściwość `CommandText` i w takiej postaci dołączyć go do obiektu `catalog` (wydruk 7.16).

Wydruk 7.16. Tworzenie procedury

```
Sub CreateProcedure()
    On Error GoTo Except

    Dim Catalog As New ADOX.Catalog
    Catalog.ActiveConnection = CurrentProject.Connection

    Dim Command As New ADODB.Command
    Command.CommandText = "Parameters [@Region] Text;" & _
        "Select * from Employees where Region = [@Region]"

    Catalog.Procedures.Append "qryEmployeesRegion", Command

    Set Command = Nothing
    Set Catalog = Nothing

    Exit Sub
Except:
    MsgBox Err.Description
End Sub
```

Modyfikowanie wyrażenia SQL kwerendy

Aby zmodyfikować wyrażenie SQL kwerendy, musisz najpierw utworzyć obiekt `command`, oparty na aktualnej kwerendzie w bazie danych, a następnie zmodyfikować właściwość `commandtext` i zresetować obiekt `command` (wydruk 7.17).

Wydruk 7.17. Zmiana wyrażenia SQL

```
Sub ModifyQuerySQL()
    On Error GoTo Except

    Dim Catalog As New ADOX.Catalog
    Catalog.ActiveConnection = CurrentProject.Connection

    Dim Command As New ADODB.Command
    Set Command = Catalog.Procedures("qryEmployeesRegion").Command

    Command.CommandText = _
        "Parameters [forms]![frmOrder]![txtRegion] Text;" & _
        "Select * from Employees " & _
        "Where Region = [forms]![frmOrder]![txtRegion] "

    Set Catalog.Procedures("qryEmployeesRegion").Command = Command

    Set Command = Nothing
    Set Catalog = Nothing
    Exit Sub

Except:
    MsgBox Err.Description
End Sub
```

Zauważyłeś zapewne, że w kodzie ADO znajdującym się na wydruku 7.17 ustawienie właściwości `Command` obiektu `Procedure` na zmodyfikowany obiekt `Command` powoduje zapisanie zmian. Gdybyśmy nie umieścili tego ostatniego kroku, zmiany nie zostałyby umieszczone na stałe w bazie danych. Różnica ta wynika z tego, że obiekty `Command` zostały zaprojektowane jako tymczasowe. Fakt ten ma duże znaczenie, gdyż w DAO obiekty `Querydefs` uważane były za trwałe. Może Ci się wydawać, że przedstawione tutaj fragmenty kodu ADO są równoznaczne. Oto pierwszy z nich:

```
Set Command = Catalog.Procedures("qryEmployeesRegion").Command
Comman.CommandText = "Parameters [@Region] Text;" & _
    "Select * from Employees where Region = [@Region]"
Set Catalog.Procedures("qryEmployeesRegion").Command = Command
```

a to drugi:

```
Catalog.Procedures("qryEmployeesRegion").CommandText = _
    "Parameters [@Region] text;" & _
    "Select * from Employees where Region = [@Region] "
```

Mimo tego, że można odnieść wrażenie, iż działają one podobnie, drugi fragment kodu nie spowoduje rzeczywistej aktualizacji kwerendy w bazie. W drugim przykładzie możesz użyć obiektu `Command` z jego nowym wyrażeniem SQL, zapisanym w kodzie VBA. Jednak po zakończeniu pracy zmiany w SQL nie zostaną zapisane w bazie. W pierwszym przypadku natomiast — ponieważ nakazujesz zapisanie `Command` — zmiany zostaną uwzględnione w bazie danych.