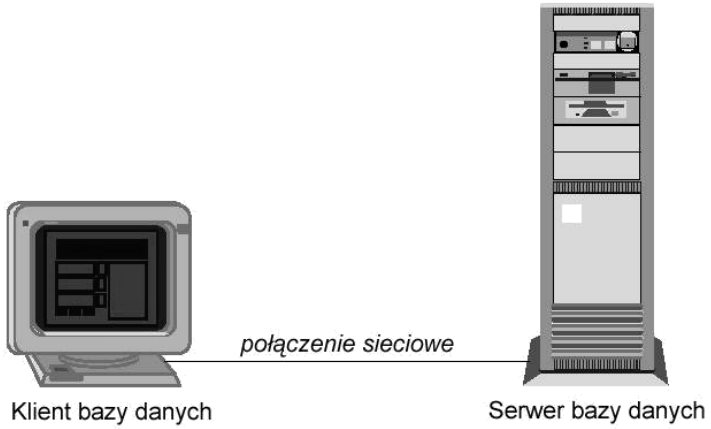
1. Administrowanie bazą danych MS SQL Serwer 2005

Opracowali: Sławomir Samolej, Andrzej Bożek
Politechnika Rzeszowska,
Katedra Informatyki i Automatyki,
Rzeszów, 2008.

1.1. Wprowadzenie

Duże bazy danych pracują zwykle w konfiguracji klient-serwer. W środowisku sieciowym oznacza to, że komputer pracujący jako serwer bazy danych odpowiada na zapytania kierowane ze stacji użytkowników – klientów bazy (Rys. 1.1). Zapytania te są najczęściej generowane automatycznie przez oprogramowanie użytkowe klienta.



Rys. 1.1. Architektura typu klient-serwer w środowisku sieciowym

Oprogramowanie zainstalowane na stacji klienta umożliwia formułowanie i wysyłanie do bazy zapytań języka SQL. Na komputerze pracującym jako serwer znajdują się pliki bazy danych oraz oprogramowanie umożliwiające dostęp do zapisanych tam informacji. Ponadto, serwer bazy danych przy pomocy systemu zarządzania bazą danych (SZBD) dba o spójność i bezpieczeństwo bazy danych.

1.2. Narzędzia

Z punktu widzenia klienta, SQL Serwer to przede wszystkim zbiór narzędzi i bibliotek służących do łączenia się z bazą. Bezpośrednio po zainstalowaniu bazy danych, wszystkie najważniejsze narzędzia są dostępne z pozycji „Start -> Wszystkie Programy -> Microsoft SQL Server 2005”. Z pełną wersją instalacji SQL Serwera dostępne są następujące programy narzędziowe:

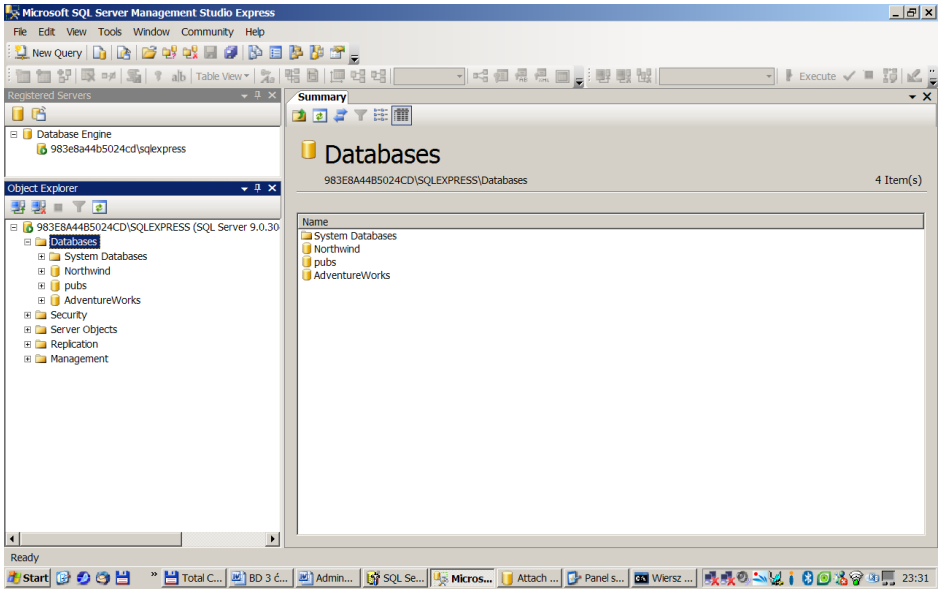
- **BI Developmnet Studio** – zintegrowane środowisko do pisania aplikacji dla SQL Serwera 2005, będące odpowiednio sprofilowaną wersją Visual Studio 2005. W przypadku, gdy na komputerze

jest zainstalowane Visual Studio 2005, BI Development Workbench jest jego częścią, a projekty przez nie tworzone pojawiają się z innymi projektami Visual Studio.

- **SQL Server Management Studio** – podstawowe narzędzie do zarządzania bazą danych. Umożliwia dołączanie lub odłączanie baz danych do/z serwera, tworzenie struktury bazy danych i wypełnienie jej, usuwanie bazy danych, wykonywanie poleceń SQL i przygotowywanie funkcji wbudowanych SQL.
- **Books Online** – dokumentacja SQL serwera wraz z podręcznikami wprowadzającymi wybrane zagadnienia.
- **SQL Server Configuration Manager** – narzędzie pozwalające zarządzać serwisami w skład instalacji SQL Servera 2005, zatrzymywać je i uruchamiać, a także konfigurować protokoły sieciowe.
- **SQL Sever Area Configuration** – narzędzie, które obsługuje włączenie domyślnie wyłączonych serwisów i opcji.
- **SQL Server Profiler** – narzędzie pozwalające śledzić aktywność wewnątrz serwera SQL w celach diagnostycznych i audytowych.
- **Database Engine Tubing Advisor** – narzędzie pozwalające na automatyczne dostrajanie bazy danych pod kątem potrzebnych indeksów.
- **Narzędzia wiersza poleceń** – część narzędzi działajaca w trybie tekstowym, np. sqlcmd.
- **Aplikacje** – programy odwołujące się do SQL Servera.

1.3. MS SQL Server Management Studio

MS SQL Server Management Studio jest podstawowym narzędziem do zarządzania bazą danych. Zawiera w sobie mechanizmy służące do zarządzania całym środowiskiem SQL Server z serwerami analitycznymi, raportowymi czy mobilnymi włącznie. Pozwala również na wykonywanie zapytań do baz danych oraz serwisów analitycznych. Na rysunku 1.2 pokazano widok programu po podłączeniu do serwera.



Rys. 1.2 MS SQL Management Studio

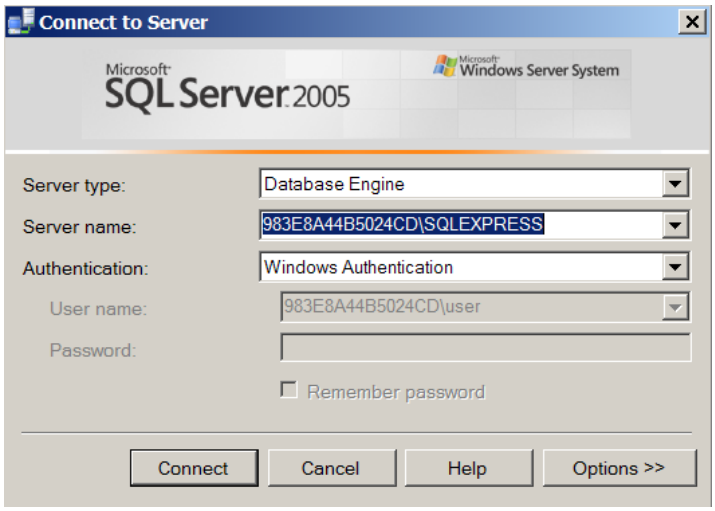
Najważniejszymi oknami programu są:

- **Registered Servers** – lista zarejestrowanych w narzędziu serwerów MS SQL.
- **Object Explorer** – wyświetla wszystkie obiekty, które znajdują się na serwerze.
- **Template Explorer** – okno z dostępnymi szablonami skryptów .
- **Sumary** – okno zawierające podsumowanie aktualnego obiektu. Widok okna zmienia się w zależności od tego, jaki obiekt zostanie zaznaczony w oknie Object Explorer.
- **Properties** – okno umożliwiające wyświetlenie i zmianę właściwości wybranego obiektu.
- **Solution Explorer** – okno bieżącego rozwiązania. Zawiera wszystkie projekty i pliki wchodzące w skład rozwiązania.
- **Bookmarks** – okno z zakładkami.
- **Web Browser** – okno przeglądarki, które zwykle służy do pokazywania plików pomocy, ale może być użyte równie dobrze do przeglądania dowolnych stron w Internecie.

Wymienione okna można włączać i wyłączać z zastosowaniem menu (opcja View) oraz korzystając z paska narzędzi, grupującego najważniejsze dostępne w MS SQL Server Management Studio podprogramy.

1.3.1 Połączenie z bazą danych

Aby pracować z bazą danych, środowisko musi wcześniej wykonać do niej podłączenie. Kiedy wymagane jest połączenie z bazą danych, pojawia się okno logowania (por. rys. 1.3)

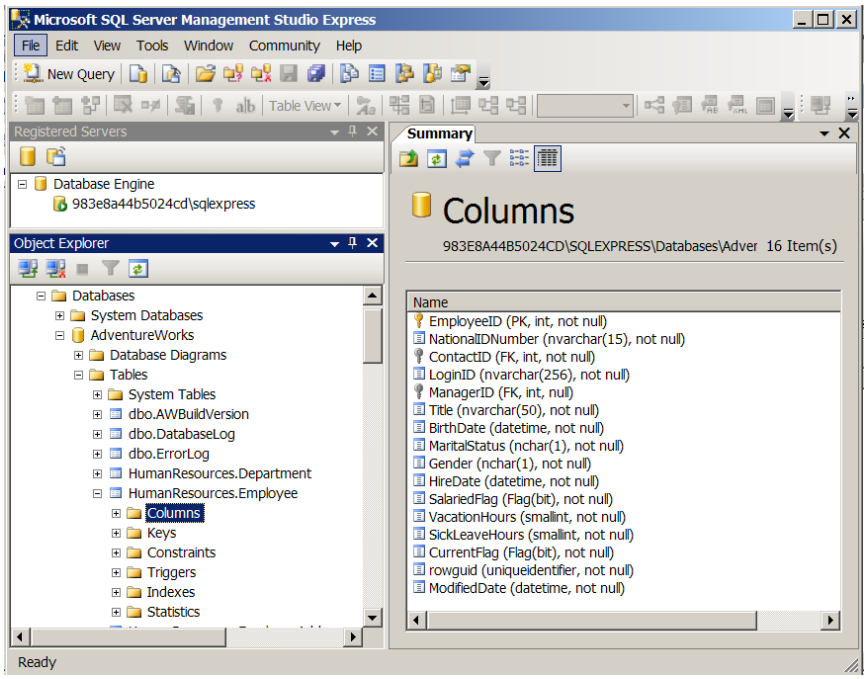


Rys. 1.3 Okno służące do logowania

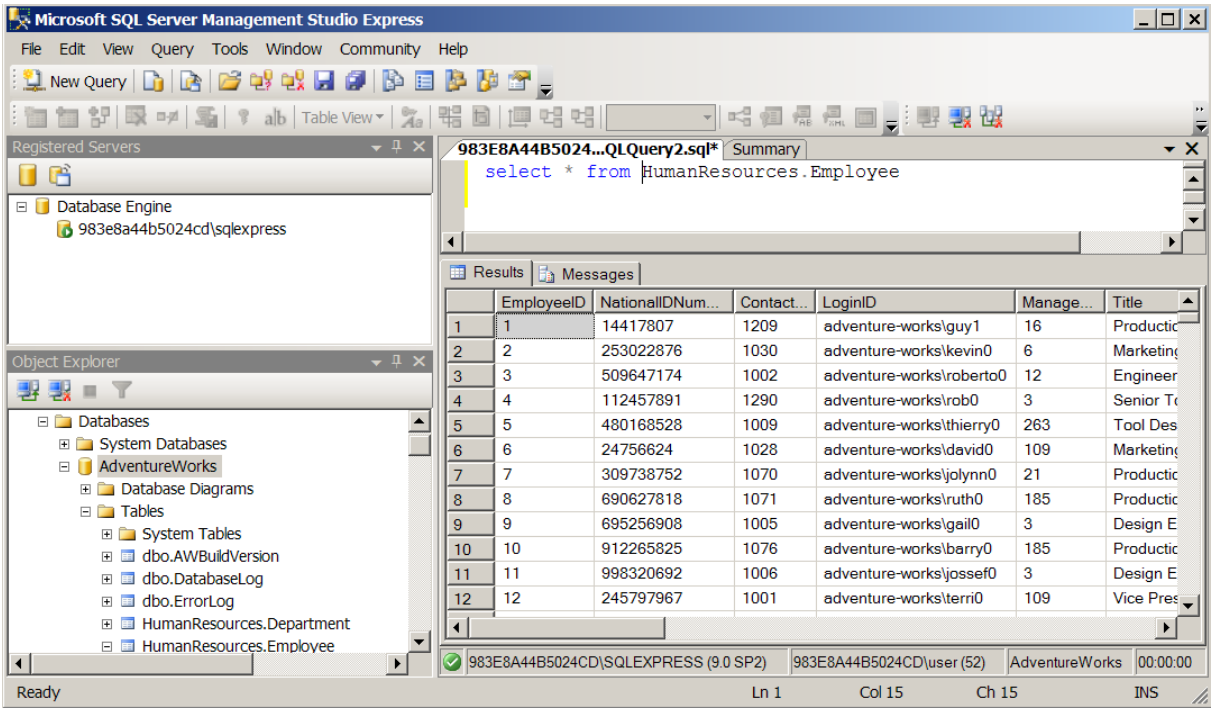
W oknie należy wskazać typ, nazwę oraz sposób autentykacji do serwera. Do MS SQL Server Management Studio może zostać podłączonych, w celu zarządzania, wiele serwerów.

1.3.2 Eksploracja bazy danych

Kiedy serwer zostanie zarejestrowany, można przystąpić do jego przeglądania za pomocą okna **Object Explorer**. W wymienionym oknie pojawi się struktura drzewa, pokazująca wszystkie obiekty wewnątrz danego serwera bazy danych. Podstawowa funkcjonalność programu pozwala na przejrzenie struktury tabel wskazanej bazy danych, por. rys. 1.4. Możliwe jest zdefiniowanie nowej tabeli lub bazy danych, uzupełnienie pól tabeli lub wypełnienie tabeli danymi. Możliwe także jest przygotowanie i uruchomienie poleceń języka SQL, wykonywanych na danej bazie danych czy tabeli, por. rys. 1.5.



Rys. 1.4 Widok definicji kolumn wybranej tabeli



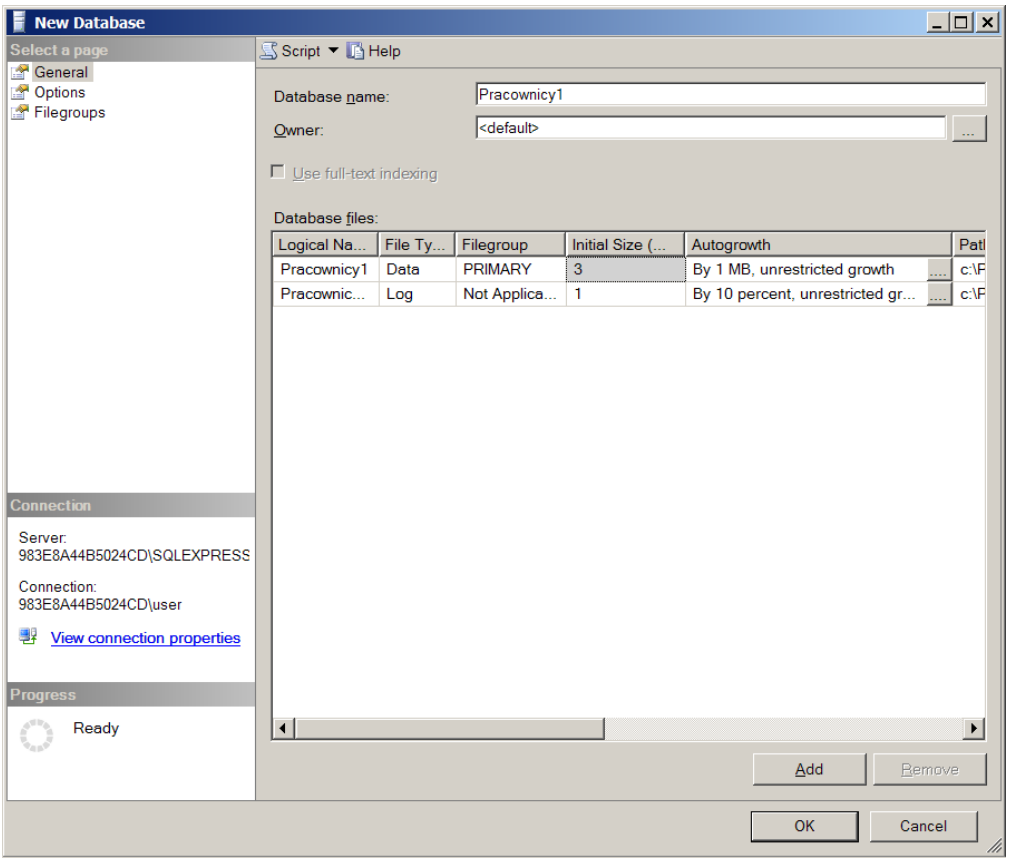
Rys. 1.5 Efekt wywołania polecenia SQL dla wskazanej bazy danych.

1.3.3 Tworzenie, modyfikacja i usuwane baz danych

Tworzyć, modyfikować i usuwać bazę danych można zarówno z poziomu interfejsu SQL Server Management Studio, jak i za pośrednictwem języka T-SQL (rozszerzenia języka SQL zastosowanego w produktach Microsoft i Sysbase). Przedmiotem dalszych rozważań będzie zastosowanie SQL Server Management Studio do tworzenia bazy.

Aby utworzyć nową bazę danych, z menu kontekstowego węzła **Databases** w eksploratorze obiektów należy wybrać polecenie **New Database**. Ukaże się widok, taki jak na rys. 1.6. Na pierwszej

zakładce (**General**) należy wpisać przede wszystkim wpisać nazwę bazy danych (**Database name**) i wskazać, który użytkownik będzie właścicielem utworzonej bazy danych. Pozostałe zakładki służą do ustalenia szczegółowych parametrów pracy bazy oraz zasad dostępu do niej i wykraczają poza zakres ćwiczenia.



Rys. 1.6 Podstawowe ustawienia dla tworzonej bazy danych

Po utworzeniu bazy danych, nowa baza jest widoczna w drzewie baz danych (gałąź **Databases**) i w dalszej kolejności mogą dla niej zostać zdefiniowane tabele. Bazę danych można odłączyć od serwera (polecenie menu kontekstowego **Tasks** -> **Detach**), co pozwala na przeniesienie pliku bazy danych bez narażania go na modyfikacje podczas przenoszenia. Plik bazy danych można również zwyczajnie usunąć (polecenie menu kontekstowego **Delete**). Odłączoną wcześniej bazę danych lub otrzymaną z zewnątrz bazę danych można przyłączyć do serwera (polecenie **Attach** z menu kontekstowego). Na tym poziomie zarządzania bazą można również dostroić inne ogólne parametry bazy, takie jak rozmiar, nazwa pliku, maksymalny rozmiar, itp.

1.4. Logiczna struktura bazy danych

Fizyczna struktura bazy danych określa sposób, w jaki jest ona zbudowana, sposób jej umieszczania na dyskach oraz konfigurację. Logiczna struktura bazy danych określa strukturę informacji zawartej w bazie danych – logikę bazy. Poniżej zostanie omówiona logiczna struktura danych oraz podstawowe własności tabel, widoków indeksów i synonimów, a także operacje na nich wykonywane.

Do logicznej struktury bazy danych zalicza się:

- tabele,
- widoki,

- indeksy,
- procedury składowane,
- funkcje użytkownika,
- wyzwalacze,
- synonimy.

Tabele są podstawowymi strukturami do przechowywania danych. Definicja tabeli składa się przede wszystkim z definicji kolumn, które ją tworzą. Definicja kolumny obejmuje określenie typu zmiennych, informację o domyślnej wartości pól kolumny i o tym, czy mogą one przyjmować wartości nieokreślone (NULL), ograniczenia nałożone na dane oraz klucze podstawowe i obce. Klucze podstawowe służą do jednoznacznego rozróżnienia wierszy wchodzących w skład tabeli. Klucze obce odwołują się do wartości kluczy podstawowych w innych tabelach, tworząc w ten sposób relację pomiędzy nimi.

Indeksy są pomocniczymi strukturami, usprawniającymi wyszukiwanie danych w tabeli. Przechowują one wskazania do wierszy w tabeli, zawierających konkretną wartość. Ponieważ indeksy mają strukturę drzewa, ich przeszukiwanie jest znacznie szybsze niż przeszukiwanie całej tablicy. Przy braku indeksów, każde przeszukiwanie wiązałoby się z koniecznością przeszukiwania całej tabeli danych.

Rozróżnia się 2 typy indeksów. Indeksy zwykłe, które są strukturami danych istniejącymi obok tabeli, oraz indeksy klastrowe, które sortują całą tabelę. Użycie indeksu klastrowego przypomina ułożenie haseł w encyklopedii – wiersze są ułożone w tabeli według porządku zdefiniowanego w indeksie.

Widoki są strukturami podobnymi do tabel, z tą jednak różnicą, że nie przechowują same danych, a jedynie odwołują się do danych zapisanych w innych tabelach. Jeżeli dla przykładu dane o klientach są zapisane w kilku tabelach (np. osoba, adres, miasto, ulica, firma) w postaci znormalizowanej, jeden widok może złączyć wszystkie dane, pokazując je jako jedną tabelę. Odczytywanie danych z widoku odbywa się w ten sam sposób, jak odczytywanie danych z tabeli.

Procedury składowane są to fragmenty wykonywalnego kodu SQL, przechowywanego na serwerze. Do procedur można przekazywać parametry, a procedura może zwracać parametry na zewnątrz.

Funkcje użytkownika są podobne do procedur, z jedną różnicą – muszą bezpośrednio zwracać wartość i mogą być bezpośrednio używane w przypisaniach.

Wyzwalacze są to fragmenty kodu SQL, które są wykonywane, gdy na serwerze wystąpi określona akcja. Zwykle tymi akcjami są operacje na danych zawartych w tabelach i widokach. Dobrym przykładem użycia wyzwalacza jest zapisywanie starych wartości danych do tabeli archiwalnej.

Synonimy są to inne, zastępcze nazwy obiektów znajdujących się w bazie danych.

1.5. Tabele

Wszystkie dane zawarte w bazie są zapisane w tabelach. Definicja tabeli zawiera definicje kolumn, które ją tworzą. Podstawową informacją o kolumnie jest typ przechowywanych przez nią danych. Typy danych w SQL Server 2005 można podzielić na następujące kategorie:

- **Dane numeryczne** reprezentujące wszelkiego rodzaju liczby. W ramach danych numerycznych rozróżniamy **dane dokładne** – przechowywane bez żadnych zaokrągleń, jak również **dane przybliżone** – przechowujące liczbę z zadaną precyzją.
  Dane dokładne podzielono na całkowite i pieniężne.

Tabela 1.1 Całkowite typy danych

Typ	**Zakres**	**Liczba bajtów**
Tinyint	0..255	1
Smallint	-32768..32767	2
Int	-2 147 483 648.. 2 147 483 647	4
bigi nt	-9 223 372 036 854 775 808 .. 9 223 372 036 854 775 807	8

Tabela 1.2 Typy pieniężne

Typ	**Zakres**	**Liczba bajtów**
smallmoney	-214 478.3648.. 214 748.3647	4
money	-922 337 203 685 477.5808 .. 922 337 203 685 477.5807	8

Obliczenia na danych przybliżonych dają przybliżony wynik mieszczący się w zakresie danych.

Tabela 1.3 Typy przybliżone

Typ	**Zakres**	**Liczba bajtów**
float	Od -1.79E+38 do -2.23E-38, 0, od 2.23E-38 do 1.79E+38	Zależna od precyzji
real	Od -3.40E+38 do -1.18E-38, 0, od 1.18E-38 do 3.40E+38	8

- **Data i czas** są to wszystkie typy danych pozwalające na reprezentację daty i czasu.

Tabela 1.4 Typy reprezentujące czas i datę

Typ	**Zakres**	**Dokładność**
datetime	1 stycznia 1753 do 31 grudnia 9999	3,33 ms
smalldatetime	1 stycznia 1900 do 6 czerwca 2079	1 min

- **Dane łańcuchowe** służą do przechowywania łańcuchów wartości. Trzy podkategorie tych danych to *dane znakowe*, *dane znakowe w formacie UNICODE* i *dane binarne*.

Tabela 1.5 typy łańcuchowe

	Znakowe	Znakowe w formacie UNICODE	Binarne
Stała długość	char	nchar	binary
Zmienna długość	varchar	nvarchar	varbinary
Duże obiektu	text lub vartext(max)	ntext lub nvartex(max)	image lub varbinary(max)

Deklarując typy o stałej lub zmiennej długości, podajemy w nawiasach maksymalną długość łańcucha w znakach lub w bajtach. Maksymalna długość dla danych znakowych i binarnych wynosi 8000, natomiast dla łańcuchów w formacie UNICODE – 4000. Typy text, ntext i image są to tak zwane duże obiekty LOB (ang. Large OBject). Maksymalnie mogą zajmować 2 147 483 647 bajtów (1 073 741 823 znaków w postaci UNICODE).

- **Inne typy danych**, których nie daje się przyporządkować do żadnej z powyższych kategorii, np. sql_variant, timestamp, typ tabelaryczny, uniqueidentifier, kursor, XML
- **Typy zdefiniowane przez użytkownika**.

Z kolumną są związane ograniczenia (ang. constraint). Ograniczenia te mogą być następujące:

- NULL/NOT NULL
- CHECK
- UNIQUE
- PRIMARY KEY
- FOREGIN KEY

NOT NULL – to ograniczenie określa, że kolumna nie może przyjmować wartości nieokreślonej (NULL). Wartość nieokreślona (NULL) oznacza, że polu nie została przypisana konkretna wartość danego typu. Wartości nieokreślone, zależnie od sytuacji, mogą stać się zarówno błogosławieństwem jak i przekleństwem bazy danych. Z jednej strony wartości nieokreślone pokazują, że dana informacja jest nieznana (np. adres klienta nie został jeszcze podany). Z drugiej zaś strony, wartość NULL wymaga dodatkowej obsługi, zarówno po stronie bazy danych, jak i po stronie aplikacji. W bazie danych w każdym wierszu dla każdej kolumny istnieje znacznik określający, czy wartość tej kolumny jest nieokreślona. Natomiast w aplikacji, jeśli dana wartość może być nieokreślona, wymagane są dodatkowe sprawdzania i ewentualna obsługa błędów (jeśli np. chcemy wysłać paczkę, a adres nie został podany).

Z reguły, jeśli nie ma poważnych powodów, aby dana kolumna mogła przyjmować wartości nieokreślone, należy definiować kolumnę jako NOT NULL. Jeżeli nie znamy danej wartości, lepiej stosować wartości domyślne (pusty łańcuch lub 0), które możemy obsługiwać w łatwiejszy sposób.

CHECK – jest to wyrażenie logiczne związane z kolumną. Dane we wszystkich wierszach muszą pasować do wyrażenia tak, aby zawsze było ono prawdziwe. Jeżeli jakaś zmiana w bazie danych doprowadza do pogwałcenia ograniczenia narzuconego przez CHECK, jest wywoływany automatycznie błąd i bieżąca transakcja jest cofana.

UNIQUE – ograniczenie to oznacza, że wszystkie wartości w kolumnie muszą być różne. Istnieją dwa sposoby zapewnienia niepowtarzalności. Można ją osiągnąć poprzez użycie klucza głównego albo niepowtarzalnego indeksu. Klucz główny nie pozwala jednak na użycie wartości nieokreślonych – wartość NULL nie podlega ograniczeniom stawianym przez ograniczenie UNIQUE. Ponadto może istnieć tylko jeden klucz główny tabeli, a ograniczeń UNIQUE może być wiele. Różnica pomiędzy ograniczeniem a indeksem dotyczy natury poprawności konstrukcji logicznej. Indeksy służą przede wszystkim do przyspieszenia wyszukiwania danych. Logika bazy powinna być zawarta w ograniczeniach, a nie w indeksach. W praktyce ograniczenie UNIQUE jest implementowane przez unikatowy indeks.

PRIMARY KEY – klucz główny, pola kolumn posiadających ten atrybut jednoznacznie identyfikują wiersze. Jako klucz główny może być użyta zarówno jedna kolumna jak i zbór kolumn.

FOREGIN KEY – klucz obcy. Odwołuje się do wartości klucza głównego w innej tabeli, definiując relację pomiędzy tabelami. Kiedy relacja zostanie ustanowiona, wszystkie wartości w tej tabeli muszą mieć albo wartości nieokreślone, albo znajdujące się w kolumnie klucza głównego tabeli, do której się odwołujemy.

Z kluczem tym łączą się również możliwości wyboru opcji związanych z usunięciem rekordu nadrzędnego. Możliwe są 4 rozwiązania:

- NO ACTION – próba usunięcia rekordu nadrzędnego, który posiada rekordy podrzędne, kończy się błędem.
- CASCADE – próba usunięcia rekordu nadrzędnego kończy się usunięciem wszystkich rekordów podrzędnych.
- SET NULL – przy usuwaniu rekordu nadrzędnego, wartości kluczy obcych w rekordach podrzędnych zostają przestawione na nieokreślone.

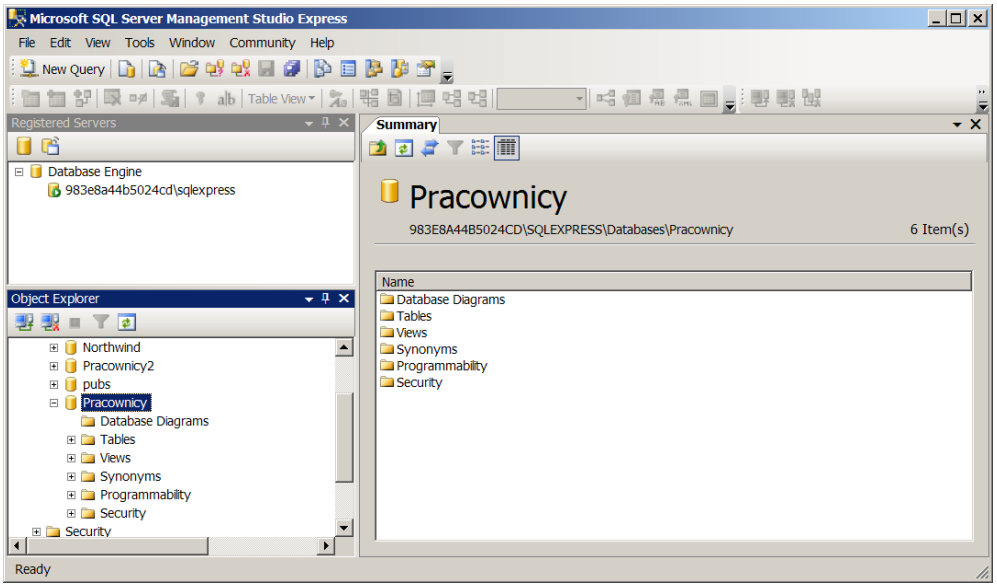
- SET DEFAULT – przy usuwaniu rekordu nadrzędnego, wartości kluczy obcych w rekordach podrzędnych zostają przestawione na wartość domyślną.

1.5.1 Tworzenie tabel

Po utworzeniu bazy danych (por. punkt 1.3.3, rys. 1.6) jest ona widoczna w oknie Object Explorer. Rozwinięcie gałęzi drzewa, opisującego utworzoną bazę, umożliwia dostęp do kolejnych podgałęzi (rys. 1.7):

- **Database Diagrams** – w tej gałęzi mogą się znajdować diagramy bazy danych, jeśli zostaną utworzone.
- **Tables** – w tej gałęzi znajdują się tabele bazy danych.
- **Views** – w tej gałęzi znajdują się zdefiniowane ew. widoki dla bazy danych.
- **Synonyms** – w tej gałęzi umieszcza się zdefiniowane synonimy w bazie danych.
- **Programmability** – w tej gałęzi umieszcza się procedury składowane, funkcje użytkownika i wyzwalacze, operujące na bazie danych.
- **Security** – w tej gałęzi definiuje się użytkowników bazy danych i ich uprawnienia.

W dalszej części opracowania szczególna uwaga zostanie poświęcona podgałęzi **Tables**.



Rys. 1.7 Podgałęzie opisu bazy danych.

Po rozwinięciu podgałęzi **Tables,** dostępne są tylko tzw. tabele systemowe wspomagające zarządzaniem lokalnej bazy. Tworzenie tabel bazy danych z zastosowaniem MS SQL Server Management Studio rozpoczyna się od definicji tabel. W kolejnych tabelach 1.6, 1.7 i 1.8 zawarto zalecenia dotyczące definicji poszczególnych kolumn (relacji) przykładowej bazy danych.

Aby utworzyć nową tabelę, należy wskazać podgałąź **Tables** i z menu kontekstowego wybrać opcję **New Table…**. W głównym oknie MS SQL Server Management Studio pojawi się nowa zakładka z pustą tabelą, w której każdy wiersz jest definicją kolumny danej tabeli bazy danych. Rysunek 1.8 zawiera rezultat wstępnej definicji tabeli.

Tworzenie bazy danych można usystematyzować w następujący sposób:

1. Pierwszym krokiem definiowania tabel powinno być zdefiniowanie nazw kolumn, typu danych, które będą przechowywać poszczególne kolumny oraz wskazanie, czy zezwala się, że dane mogą być „puste” (NULL) dla wszystkich tabel (por. rys. 1.8). Jeśli podczas opracowywania definicji tabel bazy danych (por. tabele 1.6,1.7,1.8) wskazano, że dany element relacji jest obowiązkowy,

nie może on być nigdy przy wypełnianiu tabeli pusty. Stąd musi mieć ograniczenie NOT NULL.

Uwaga:

Podczas definiowania nowych tabel, każda z definicji będzie się pojawiać w nowej zakładce głównego okna MS SQL Server Management Studio. Podczas próby zamknięcia danej zakładki system bazy danych będzie informował, że na podstawie aktualnego stanu definicji tabeli będzie ona wprowadzana do bazy danych. Utworzona tabela jest od tej chwili widoczna jako podgałąź gałęzi **Tables** dla danej bazy danych. Zamkniętą wcześniej definicję tabeli można ponownie otworzyć w trzech trybach pracy:

a. Jeśli z menu kontekstowego wybierze się opcję **Edit**, to w oknie głównym wyświetlony zostanie kod SQL tworzący tabelę automatycznie wygenerowany na podstawie zdefiniowanej struktury tabeli,
b. Jeśli z menu kontekstowego wybierze się opcję **Design**, to w oknie głównym dostępny będzie wcześniejszy widok umożliwiający graficzną definicję tabeli,
c. Jeśli z menu kontekstowego wybierze się opcję **Open Table**, to w oknie głównym dostępny będzie widok tabeli przeznaczony do wprowadzania danych.

2. Drugim krokiem jest wskazanie kluczy głównych tabel. Wykonać to można, wskazując wiersz definicji kolumny i wybranie z menu kontekstowego polecenia **Set Primary Key** (rys. 1.9).

Tab. 1.6. Schemat relacji *pracownik*

Atrybut	Typ (rozmiar)	Ograniczenia integralnościowe
NUMER	int	klucz podstawowy
NAZWISKO	nvarchar(15)	atrybut obowiązkowy
ETAT	nvarchar(20)	referencja do atrybutu NAZWA relacji ETAT
SZEF	int	referencja do atrybutu NUMER relacji PRACOWNIK
PRACUJE_OD	smalldatetime	domyślna wartość bieżącej daty
PŁACA_POD	smallmoney	
PŁACA_DOD	smallmoney	domyślna wartość 0
ID_ZESP	smallint	referencja do atrybutu ID_ZESP relacji ZESPÓŁ

Tab. 1.7. Schemat relacji *zespół*

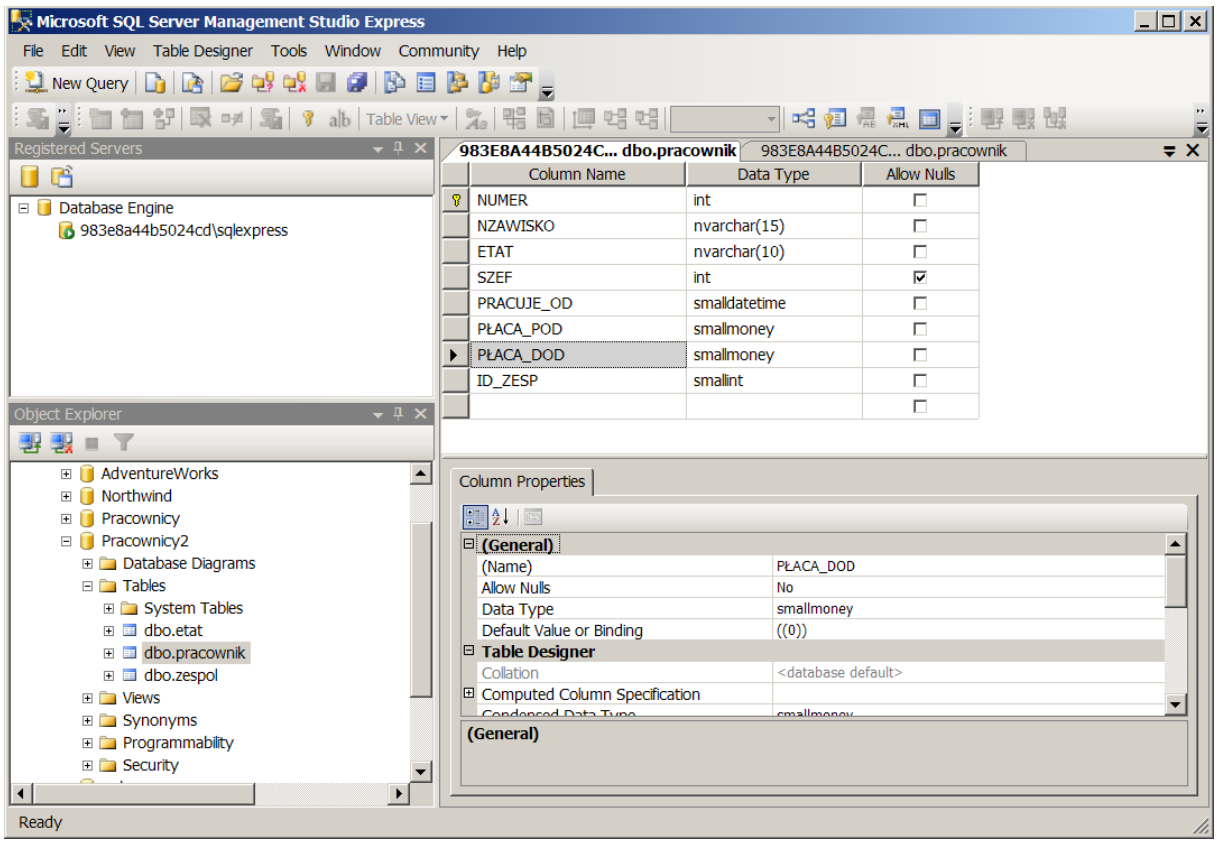
Atrybut	Typ (rozmiar)	Ograniczenia integralnościowe
ID_ZESP	smallint	klucz podstawowy
NAZWA	nvarchar (20)	atrybut obowiązkowy
ADRES	nvarchar (20)	

Tab. 1.8. Schemat relacji *etat*

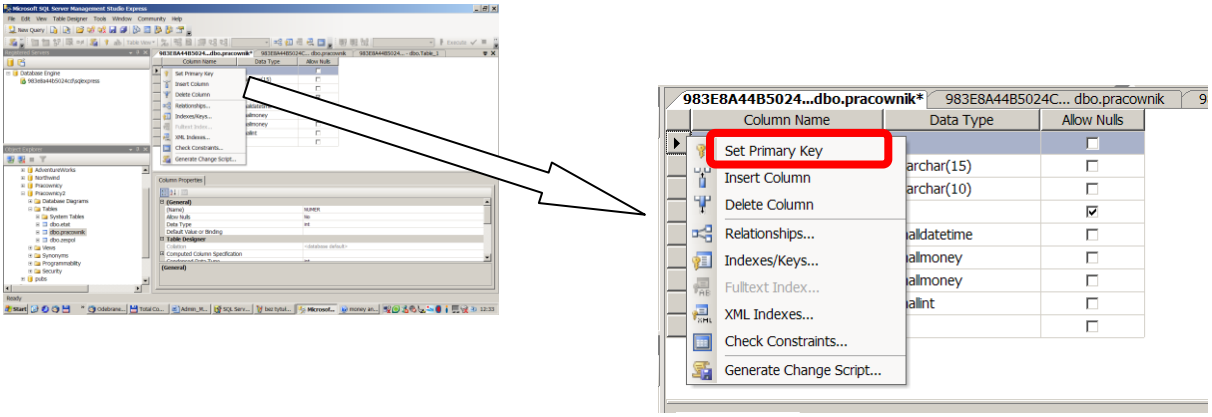
Atrybut	Typ (rozmiar)	Ograniczenia integralnościowe
NAZWA	nvarchar (20)	klucz podstawowy
PLACA_MIN	smallmoney	atrybut obowiązkowy, wartość > 0
PLACA_MAX	smallmoney	atrybut obowiązkowy, wartość <= 5000

3. Trzecim krokiem może być zdefiniowanie w wybranych kolumnach założonych na etapie projektowania wartości domyślnych. W przykładowej bazie danych wprowadzono 2 wartości domyślne w relacji pracownik. Atrybut PRACUJE_OD, w razie nie wprowadzenia tej danej, ma

przyjąć wartość dzisiejszej daty oraz atrybut PŁACA_DOD, w razie nie wprowadzenia tej danej, ma mieć wartość 0. Aby uzupełnić wybrany atrybut o wartość domyślną, należy podświetlić dany wiersz tabeli definiującej relację. Spowoduje to równocześnie wyświetlenie, poniżej pola definiującego relację, dodatkowego okna opisującego własności danej kolumny (Column Properties) (rys. 1.10). Wśród dostępnych właściwości do modyfikowania należy wskazać **Defalult Value or Binding**. Możliwe jest wtedy zaproponowanie domyślnej wartości dla danej kolumny. Dla atrybutu PRACUJE_OD zaproponowano standardową funkcję MS SQL Server – getdate(), która w chwili wypełniania tabeli nowymi wartościami zwraca bieżącą datę (por. rys. 1.10). Dla atrybutu PŁACA_DOD w odpowiadające pole wstawiono wartość 0 (por. rys. 1.8).

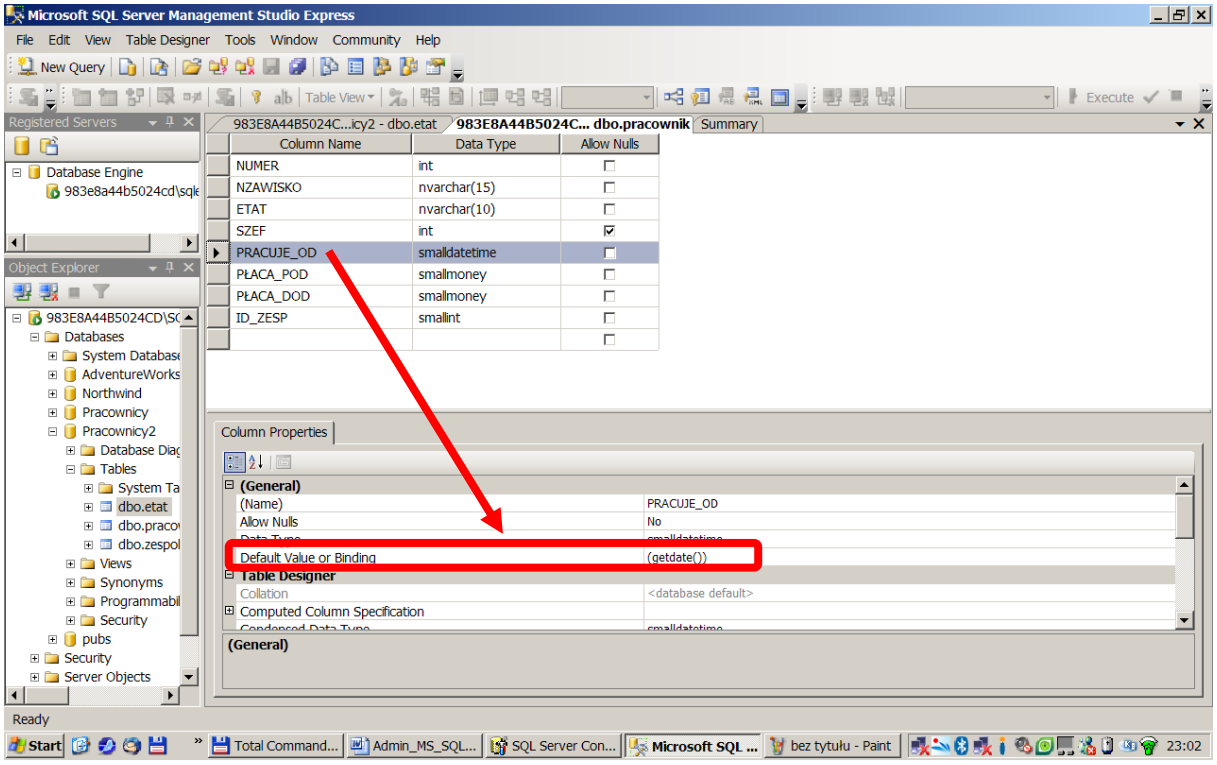


Rys. 1.8 Wstępna definicja relacji pracownicy.

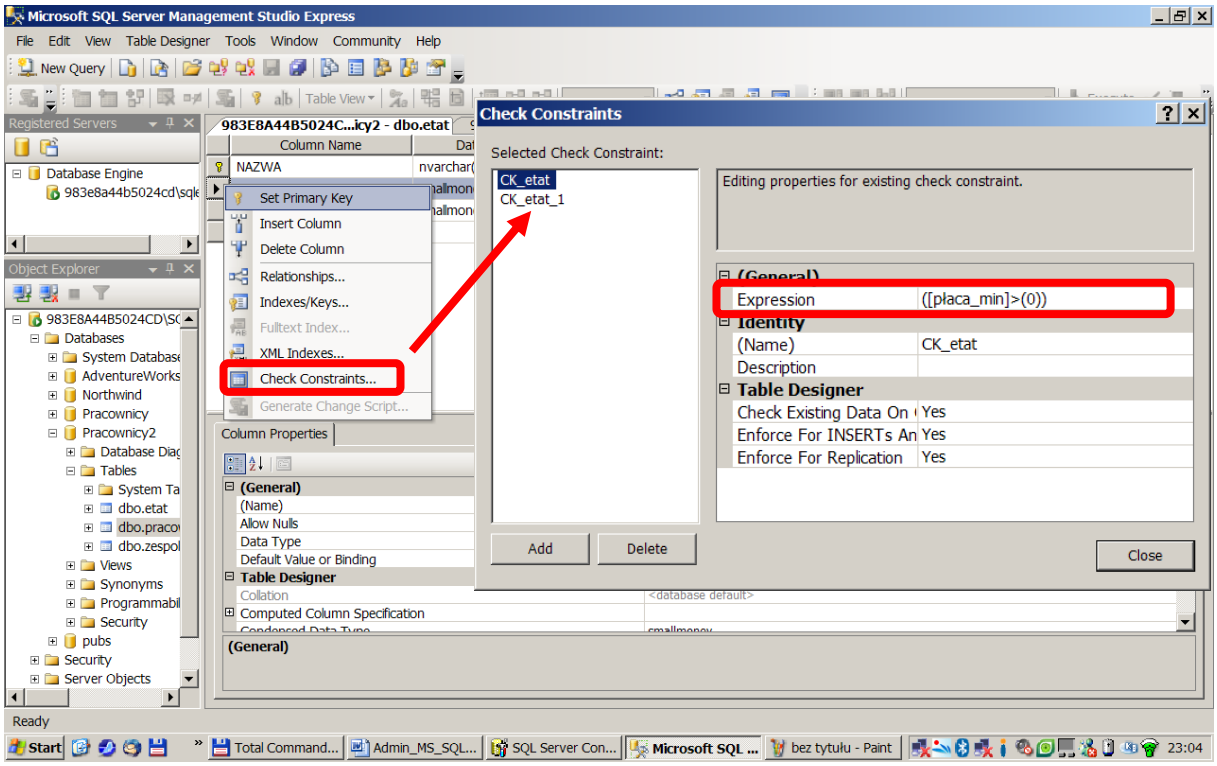


Rys. 1.9 Ustalanie klucza głównego relacji.

4. Czwartym krokiem może być uzupełnienie ograniczeń dla poszczególnych kolumn. W przykładowej bazie założono 2 ograniczenia w relacji etat (por. tabela 1.8). Atrybut PLACA_MIN musi być większy od 0, a atrybut PLACA_MAX nie może być większy od 5000.



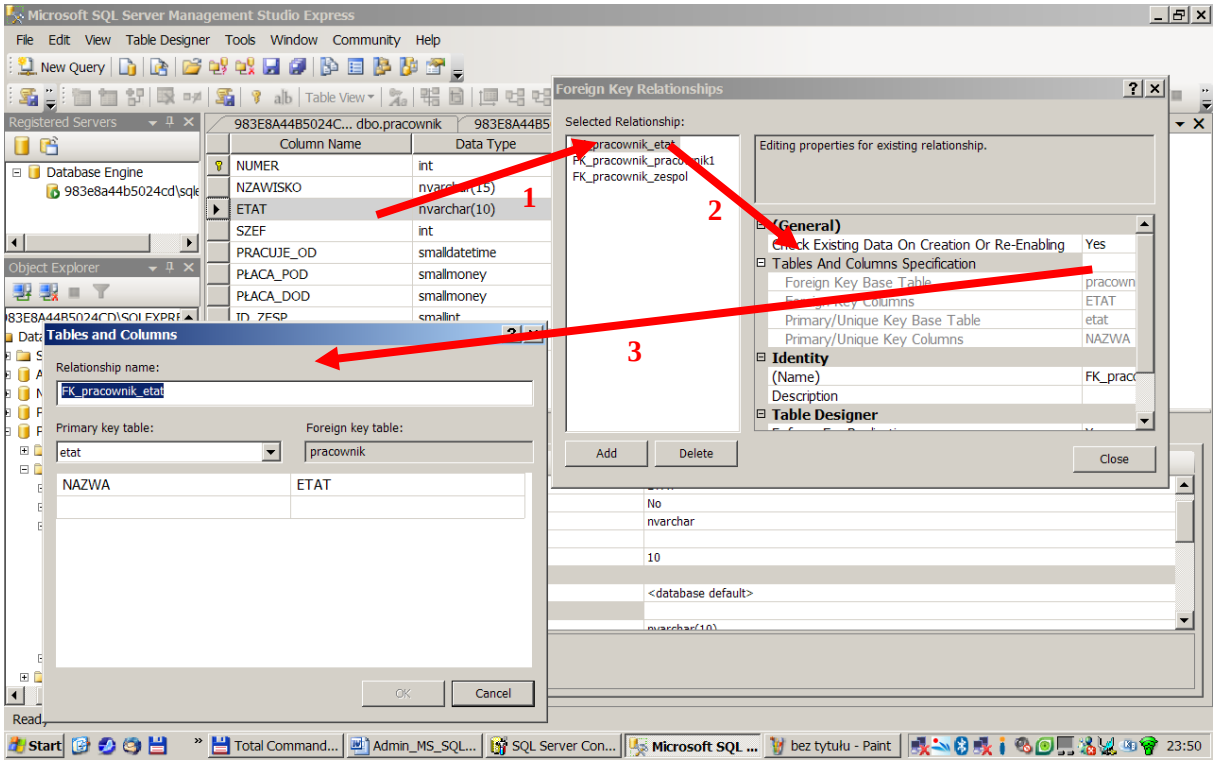
Rys 1.10 Definiowanie wartości domyślnych atrybutów.



Rys 1.11 Definiowanie ograniczeń.

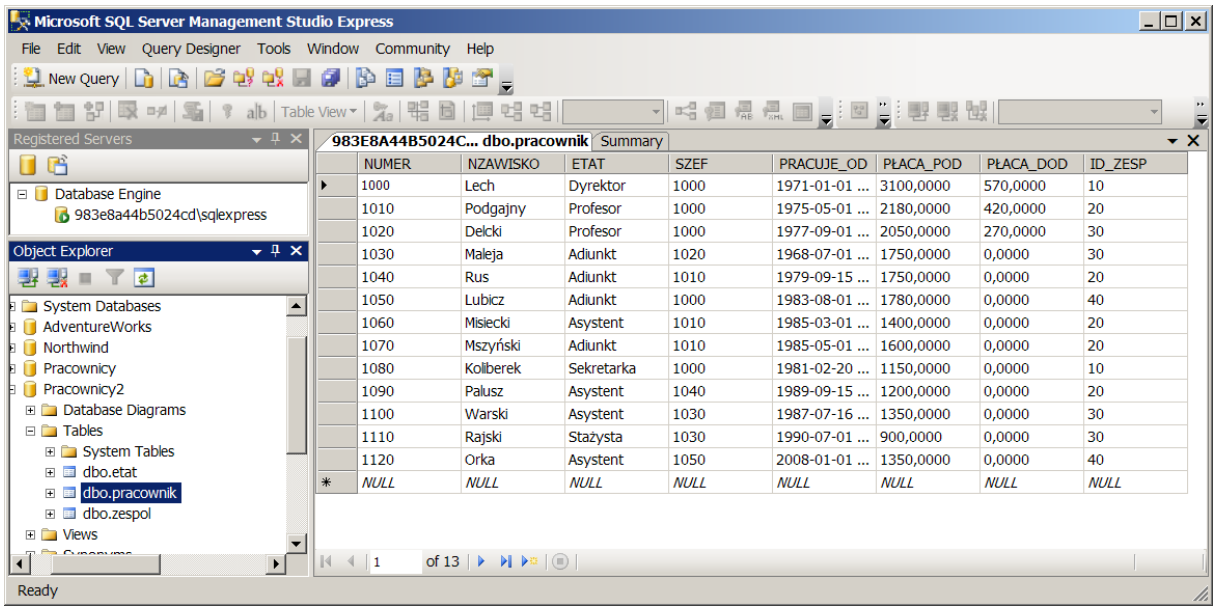
Aby dodać ograniczenie do atrybutu, należy go wskazać, a następnie wybrać z menu kontekstowego opcję **Check Constraints…** (rys. 1.11). Spowoduje to pojawienie się dodatkowego okna dialogowego (por. rys. 1.11), w którym do tabeli można dodać ograniczenie CHECK, identyfikowane przez indywidualny literał (por. nazwy CK_etat, CK_etat1, rys. 1.11). Przykładowo literał CK_etat powiązany został z wyrażeniem „płaca_min > 0”.

5. Piątym krokiem może być zdefiniowanie powiązań pomiędzy relacjami – wskazanie kluczy obcych w danej tabeli, wskazujących klucze główne w innych tabelach. Warunkiem powodzenia operacji powiązania tabel jest istnienie definicji wszystkich tabel ze wskazaniami głównych kluczy relacji. Zdefiniowanie kluczy obcych w danej relacji rozpoczyna się od wskazania danego atrybutu, a następnie wybrania z menu kontekstowego opcji **Relationships…**. Powoduje to wyświetlenie okna **Foregin Key Relationships** (por. rys. 1.12), w którym można dodać dla danej definicji tabeli nowe odniesienie do atrybutu w innej relacji lub odniesienie do atrybutu we własnej relacji. Każde odniesienie otrzymuje indywidualny literał tekstowy. Istnieje możliwość powiązania poszczególnych atrybutów pomiędzy relacjami przez wywołanie pomocniczego okna **Tables and Colums** (otwarcie okna następuje po wciśnięciu klawisza „…” dostępnego przy opcji **Tables and Colums Specification** – por. rys. 1.12). Ostatecznie wybiera się, który z atrybutów w relacji ma być rozumiany jako klucz obcy (np. na rys. 1.12 wskazano, że kluczem obcym jest atrybut ETAT z relacji pracownik) i z jakim kluczem głównym jakiej relacji ma być powiązany (np. na rys. 1.12 wskazano, że atrybut ETAT z relacji pracownik będzie powiązany z atrybutem NAZWA z relacji etat). Jak już wspomniano, można dokonywać powiązań wewnątrz tabeli. Przykładowo, w relacji pracownik przewidziano wprowadzenie wewnętrznego odwołania pomiędzy atrybutem SZEF a NUMER.



Rys.1.12 Definiowanie kluczy obcych.

6. W celu zatwierdzenia wszystkich ustawień w definicji bazy danych, należy pozamykać okna z definicjami poszczególnych tabel/relacji. Spowoduje to przeniesienie ostatecznej definicji bazy danych do odpowiednich plików dyskowych.
7. Po zdefiniowaniu struktury bazy danych, można przystąpić do wprowadzania kolejnych wierszy z danymi. Wystarczy, jak już wspomniano, otworzyć poszczególne tabele w trybie **Open Table**. Rysunek 1.13 zawiera widok relacji pracownik z uzupełnionymi danymi.



Rys. 1.13 Wypełniona tabela pracownik

8. Klikając na daną tabelę w oknie Object Explorer i wybierając z menu kontekstowego opcję Edit, można uzyskać kod SQL definiujący daną relację, który został automatycznie wygenerowany przez MS SQL Management Studio. Przykładowo, ostateczna definicja tabeli pracownicy wygląda następująco:

```
USE [Pracownicy2]
GO
/****** Object:  Table [dbo].[pracownik]    Script Date: 12/12/2008
00:09:15 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[pracownik](
      [NUMER] [int] NOT NULL,
      [NZAWISKO] [nvarchar](15) NOT NULL,
      [ETAT] [nvarchar](10) NOT NULL,
      [SZEF] [int] NULL,
      [PRACUJE_OD] [smalldatetime] NOT NULL CONSTRAINT
[DF_pracownik_PRACUJE_OD]  DEFAULT (getdate()),
      [P£ACA_POD] [smallmoney] NOT NULL,
      [P£ACA_DOD] [smallmoney] NOT NULL CONSTRAINT [DF_pracownik_P£ACA_DOD]
DEFAULT ((0)),
      [ID_ZESP] [smallint] NOT NULL,
 CONSTRAINT [PK_pracownik] PRIMARY KEY CLUSTERED
(
      [NUMER] ASC
```

```
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
ALTER TABLE [dbo].[pracownik]  WITH CHECK ADD  CONSTRAINT
[FK_pracownik_etat] FOREIGN KEY([ETAT])
REFERENCES [dbo].[etat] ([NAZWA])
GO
ALTER TABLE [dbo].[pracownik] CHECK CONSTRAINT [FK_pracownik_etat]
GO
ALTER TABLE [dbo].[pracownik]  WITH CHECK ADD  CONSTRAINT
[FK_pracownik_pracownik1] FOREIGN KEY([SZEF])
REFERENCES [dbo].[pracownik] ([NUMER])
GO
ALTER TABLE [dbo].[pracownik] CHECK CONSTRAINT [FK_pracownik_pracownik1]
GO
ALTER TABLE [dbo].[pracownik]  WITH CHECK ADD  CONSTRAINT
[FK_pracownik_zespol] FOREIGN KEY([ID_ZESP])
REFERENCES [dbo].[zespol] ([ID_ZESP])
GO
   ALTER TABLE [dbo].[pracownik] CHECK CONSTRAINT [FK_pracownik_zespol]
```

1.6. Proponowany przebieg ćwiczenia

1. Zapoznanie się z materiałem wprowadzającym i odpowiadającymi wiadomościami z wykładu, dotyczącymi projektowania i tworzenia baz danych.
2. Zaimplementowanie przykładowego schematu bazy danych, omówionego w materiale wprowadzającym.
3. Wypełnienie utworzonego schematu bazy danych danymi (uzupełnienie tabel). Przykładowe dane dla poszczególnych tabel przedstawiono w tabelach 1.9, 1.10, 1.11.

Tab. 1.9 Relacja **pracownik**.

NUMER	NAZWISKO	ETAT	SZEF	PRACUJE_OD	PŁACA_POD	PŁACA_DOD	ID_ZESP
1000	Lech	dyrektor		01-JAN-71	3160	570	10
1080	Koliberek	sekretarka	1000	20-FEB-83	1150		10
1010	Podgajny	profesor	1000	01-MAY-75	2180	420	20
1040	Rus	adiunkt	1010	15-SEP-79	1750		20
1070	Muszyński	adiunkt	1010	01-MAY-85	1600		20
1060	Misiecki	asystent	1010	01-MAR-85	1400		20
1090	Palusz	asystent	1040	15-SEP-89	1200		20
1020	Delcki	profesor	1000	01-SEP-77	2050	270	30
1030	Maleja	adiunkt	1020	01-JUL-68	1750		30
1100	Warski	asystent	1030	15-JUL-87	1350		30
1110	Rajski	stażysta	1030	01-JUL-90	900		30
1050	Lubicz	adiunkt	1000	01-SEP-83	1780		40
1120	Orka	asystent	1050	01-APR-88	1350		40
1130	Kolski	stażysta	1050	01-SEP-91	900		40

Tab. 1.10 Relacja **zespol**.

ID_ZESP	NAZWA	ADRES
10	administracja	Piotrowo 3a
20	bazy danych	Wieżowa 75

30	sieci komputerowe	Garbary 3
40	systemy operacyjne	Piotrowo 3a
50	translatory	Mansfelda 4

Tab. 1.11 Relacja **etat**.

ETAT	PŁACA_MIN	PŁACA_MAX
stażysta	800	1000
sekretarka	900	1200
asystent	1000	1600
adiunkt	1600	2000
profesor	2000	2500
dyrektor	2500	3200

4. Przejrzenie kodów języka T-SQL, wygenerowanych podczas tworzenia bazy danych.

1.7. Literatura

[1] Zawadzki M., SQL SERVER 2005, PWN, Warszawa 2007.

[2] Świder K., Dec G., Trybus B., Inżynieria systemów informatycznych. Teoria i praktyka budowy systemów oprogramowania. Wydawnictwo Politechniki Rzeszowskiej, 2005.