Projektowanie baz danych – diagramy ERD, relacje między tabelami, związki, rekordy

Autor: Paweł Wilkosz

Opublikowano: 2011-04-21

Wprowadzenie

Aby system zarządzania bazą danych mógł prawidłowo operować na zbiorze informacji oraz dbać o ich integralność oraz bezpieczeństwo, należy go odpowiednio zaprojektować. Dobrze zaprojektowana struktura bazy danych to taka, w której nie występuje redundancja (nadmiarowość) oraz niespójności związane z chaotycznym rozmieszczaniem informacji w tabelach. Żyjący w latach 1923–2003 r. teoretyk baz danych, twórca koncepcji OLAP oraz laureat Nagrody Turinga za wybitne osiągnięcia w dziedzinie informatyki – Edgar Frank Codd w swojej pracy pt. „A Relational Model of Data for Large Shared Data Banks" zaprezentował sposób tworzenia relacyjnej bazy danych oraz jej klasyfikację w formie trzech postaci normalnych. Współcześnie każdy system do zarządzania bazą danych, w tym **SQL Server**, daje gwarancję zachowania integralności informacji pod warunkiem, iż baza, którą zarządza, znajduje się co najmniej w postaci 3.

W dalszej części artykułu omówię postulaty Codda.

Normalizacja bazy danych

Pierwsza postać normalna (1NF)

Treść: *Baza danych znajduje się w pierwszej postaci normalnej, gdy każda składowa krotki jest wartością atomową.*

Pierwsza zasada postaci normalnej bazy danych określa w sposób jednoznaczny, iż w tabelach mogą znajdować się wyłącznie informacje lub atrybuty, które z natury są niepodzielne. W tabeli 1 została przedstawiona przykładowa tabela łamiąca zasadę 1. Składa się ona z dwóch kolumn: „Kobieta" i „Mężczyzna", i zawiera informacje o imieniu danej osoby. Dane składowane w ten sposób w kolumnie można określić jako kolekcję stringów (imion), dopasowanych do konkretnej płci. Nietrudno zauważyć, że takie operacje jak: wyszukiwanie, indeksowanie, porównywanie wzorców itp., stanowią w tym wypadku poważny problem. Najwłaściwszym rozwiązaniem jest przechowywanie imion w osobnych rekordach oraz przypisanie do nich identyfikatora płci (tabela 2). Na tej podstawie w kolumnach przechowywane będą informacje niepodzielne, wyłącznie jednego typu.

Tab. 1. Tabela łamiąca zasadę pierwszej postaci normalnej.

Kobieta	Mężczyzna
Joanna, Paulina, Lucyna	Marcin, Krzysztof, Michał

Tab. 2. Tabela zgodna z zasadą pierwszej postaci normalnej.

Płeć	Imię
K	Joanna
K	Paulina
K	Lucyna
M	Marcin
M	Krzysztof
M	Michał

Druga postać normalna (2NF)

Treść: *Baza danych znajduje się w drugiej postaci normalnej, kiedy spełnia warunki pierwszej postaci normalnej, oraz gdy każda kolumna zależy funkcyjnie od całego klucza głównego.*

Załóżmy, iż projektujemy bazę danych uczelni wyższej. W jednej z tabel (tabela 3) składowane będą informacje o studentach. Jako klucz główny zostały wybrane kolumny: „PESEL" oraz „Wydział". Tak zaprojektowana tabela łamie zasadę drugiej postaci normalnej, gdyż co prawda Jan Kowalski jest zależny od swojego numeru PESEL (wszędzie tam, gdzie pojawi się numer 123456, będzie identyfikowany Jan Kowalski i odwrotnie, nawet gdy zdarzą się powtórzenia w nazwisku), lecz Wydział pozostaje bez uzasadnionej relacji funkcyjnej ze studentem (tam, gdzie pojawi się student, powinien być dopasowany Wydział, ale odwrotnie już niekoniecznie).

Tab. 3. Tabela łamiąca zasadę drugiej postaci normalnej.

[PK] PESEL	[PK] Wydział	Imię	Nazwisko
123456	Informatyka	Jan	Kowalski
654321	Automatyka	Michał	Nowak

Trzecia postać normalna (3NF)

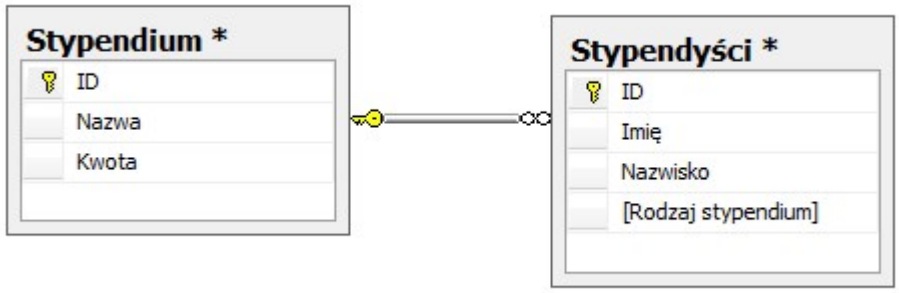
Treść: *Baza danych znajduje się w trzeciej postaci normalnej, kiedy spełnia warunki drugiej postaci normalnej oraz gdy każda kolumna zależy funkcyjnie wyłącznie od klucza głównego.*

Treść trzeciej postaci normalnej wymaga usunięcia nadmiarowych informacji, które w żaden sposób nie są związane z kluczem głównym. W projekcie bazy danych dla uczelni chcielibyśmy przechowywać informacje na temat przyznawanego stypendium. Wysokość wypłaty zależy od rodzaju pomocy materialnej (naukowa, socjalna, motywacyjna, brak). Projektując tabelę jak w tabeli 4, można łatwo popaść w kłopoty, gdy rozporządzeniem rektora stawka np. stypendium naukowego wzrośnie o 10 zł. Należałoby wtedy przejrzeć wszystkie rekordy i dokonać odpowiednich korekt, co łatwo może spowodować błędy. Sprowadzenie tabeli zgodnej z trzecią formą normalną wiąże się z przeniesieniem wartości stypendium do osobnej tabeli oraz powiązanie jej z tabelą „Studenci" relacją klucza obcego (rysunek 1).

Tab. 4. Tabela łamiąca zasadę trzeciej postaci normalnej.

Imię	Nazwisko	Rodzaj stypendium	Kwota
Joanna	Kowalska	Naukowe	10
Lucyna	Nowak	Naukowe	10
Michał	Kowalski-Nowak	Socjalne	20
Justyna	Nowak-Kowalska	Naukowe	10

Mateusz	Kowalski	Brak	0
Ewa	Nowakowska	Socjalne	20



Rysunek 1. Tabele zgodne z zasadą trzeciej postaci normalnej.

Oprócz trzech podstawowych zasad omówionych powyżej, definiuje się dwie kolejne. Nie są one zbyt często stosowane w praktyce, lecz wykorzystanie ich na pewno wpłynie na jakość projektowanych baz danych. Tymi zasadami są: postać Boyce'a-Codda (czasami nazywana w literaturze jako postać czwarta) oraz piąta postać normalna.

Postać Boyce'a-Codda (BCNF):

Treść: *Baza danych znajduje się w postaci normalnej BCNF, kiedy spełnia warunki trzeciej postaci normalnej oraz gdy zachodzi zależność: X→A i atrybut A nie zawiera się w X, to X jest kluczem bądź zawiera klucz.*

Ogólnie rzecz ujmując, postać BCNF jest sumą pierwszej, drugiej i trzeciej postaci normalnej. W praktyce oznacza to, iż wartością klucza obcego może być NULL albo klucz główny z innej tabeli. Należy mieć na uwadze, iż postać BCNF nie daje gwarancji, że redundacje poza zależnościami funkcyjnymi nie wystąpią.

Piąta postać normalna

Treść: *Baza danych znajduje się w piątej postaci normalnej, kiedy spełnia warunki postaci BCNF oraz usunięto z niej zależności funkcjonalne, które nie wynikają z zależności od atrybutów klucza.*

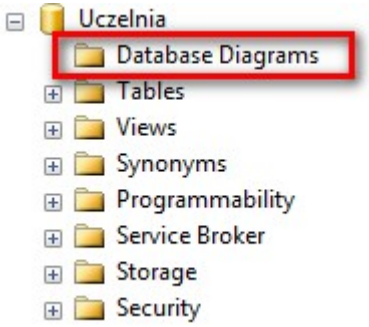
Zasada ta mówi o fakcie nie istnienia rozkładu odwracalnego na zbiór mniejszych tabel, czyli tabela została podzielona na najmniejsze możliwe kawałki w celu eliminacji redundacji.

Diagramy ERD

W drugiej części artykułu omówię sposoby modelowania struktury baz danych za pomocą diagramów ERD (Entity Relationship Diagrams). Skoncentruję się przede wszystkim na wyjaśnieniu podstawowych pojęć, których znajomość jest konieczna, aby zrozumieć dalszy ciąg tekstu.

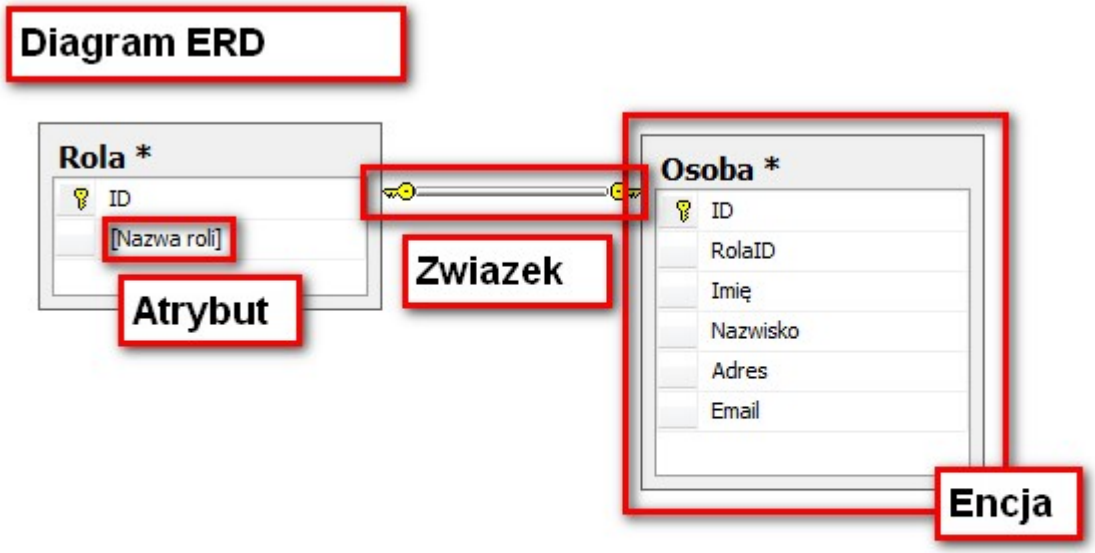
1. Encja – istniejący obiekt, rozróżnialny od innych bytów tego samego typu (np. student, wydział, katedra itp.).
2. ERD – *Entity Relationship Diagrams* – diagramy związków encji, graficzna prezentacja związków pomiędzy zbiorami encji.
3. Atrybut – informacja charakteryzująca encję (np. wiek, wydział).
4. Rekord – zestaw danych opisanych przez atrybut (np. wiek = 25 i wydział = Informatyka).
5. Tabela – wydzielony logicznie zbiór danych, również reprezentacja encji.
6. Związek – nazwane powiązanie pomiędzy dwoma encjami (np. student -> wydział). Każdy związek składa się z nazwy, liczebności (krotności) oraz opcjonalności.

Diagramy Encji, podobnie jak diagramy UML dla modelowania obiektowego, wykorzystywane są do graficznej prezentacji bazy danych. Dobrze przygotowany diagram ERD pozwala na zrozumienie struktury danych, przygotowania późniejszej strategii optymalizacji bazy oraz stanowi podstawową dokumentację systemu przechowywania informacji. ERD jest częścią wielu popularnych narzędzi CASE (Computer Aided Software Engineering), pozwalających na wygenerowanie fizycznej struktury bazodanowej na podstawie zaprojektowanego diagramu. Z podobnego kreatora można skorzystać w codziennej pracy z Microsoft SQL Server Management Studio, wybierając strukturę Database Diagram poniżej bazy danych, dla której wykonuje się projekt (rysunek 2).



Rysunek 2. Diagram bazy danych w SQL Server Management Studio.

Bardzo prosty diagram ERD wraz z opisem podstawowych właściwości został zaprezentowany na rysunku 3.

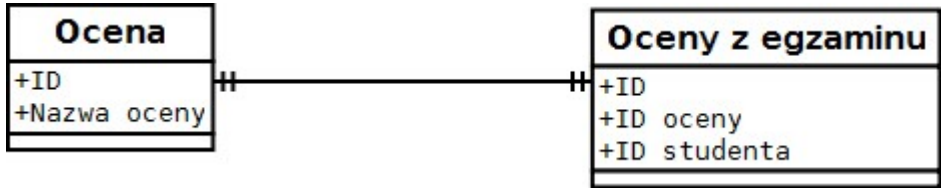


Rysunek 3. Diagram ERD wraz z opisem składowych.

Relacje pomiędzy tabelami

Rozróżnia się trzy relacje, jakie mogą zachodzić pomiędzy dwoma encjami:

1. Relacja *jeden-do-jeden* oznacza, iż jeden rekord w pierwszej tabeli odpowiada dokładnie jednemu rekordowi w tabeli drugiej. W praktyce relację *jeden-do-jeden* stosuje się w przypadku tzw. tabel słownikowych (rysunek 4).

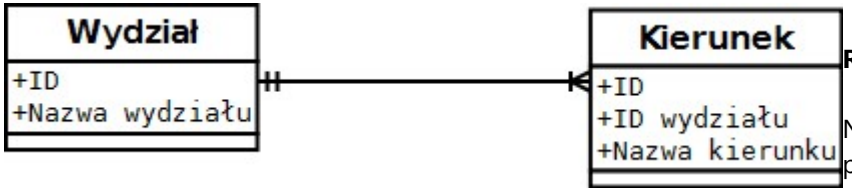


Rysunek 4. Relacja jeden-do-jeden.

Relacja ta ułatwia późniejszą modyfikację np. nazwy oceny tylko w jednym miejscu, zamiast przeglądania i zamieniania całości rekordów w encji Oceny_z_egzaminu. W klasycznej notacji ERD relację tę zapisuje się za pomocą symbolu przedstawionego na rysunku 5.

Rysunek 5. Symbol relacji jeden-do-jeden.

2. Relacja *jeden-do-wielu* oznacza, iż jeden rekord w pierwszej tabeli odpowiada wielu rekordom w tabeli drugiej i jest to relacja najczęściej stosowana. Jako przykład można użyć strukturę Wydziału składającą się z wielu kierunków (rysunek 6).

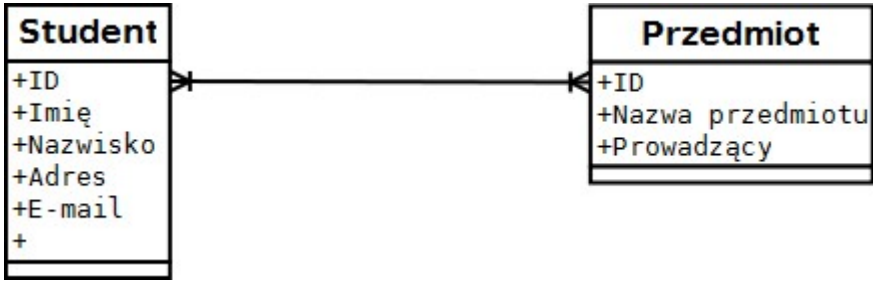


Rysunek 6. Relacja jeden-do-wielu.

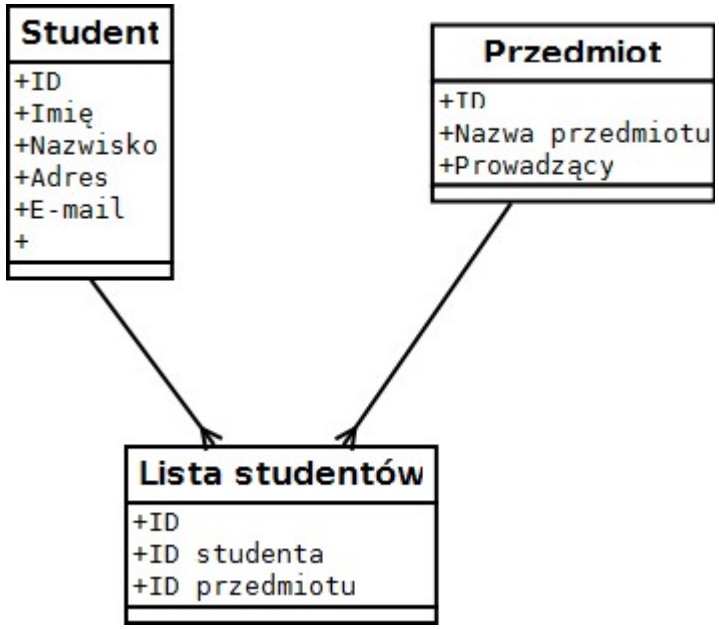
Na diagramach ERD relacja *jeden-do-wielu* prezentowana jest za pomocą symbolu przedstawionego na rysunku 7.

Rysunek 7. Symbol relacji jeden-do-wielu.

3. Relacja *wiele-do-wielu* oznacza, że kilka rekordów z pierwszej tabeli odpowiada wielu rekordom z tabeli drugiej. W praktyce stosowanie tejże relacji jest wielce niewskazane ze względu na redundancję danych, zatem rozwiązaniem tego problemu będzie utworzenie tzw. tabeli łącznikowej, scalającej obie tabele relacją *jeden-do-wielu*. Przykładowa relacja *wiele-do-wielu* została zaprezentowana na rysunku 8, natomiast prawidłowa prezentacja – za pomocą tabeli łącznikowej na rysunku 9.



Rysunek 8. Relacja wiele-do-wielu.



Rysunek 9. Relacja wiele-do-wielu zrealizowana za pomocą tabeli łącznikowej.

Relacja *wiele-do-wielu* prezentowana jest za pomocą symbolu przedstawionego na rysunku 10.

Rysunek 10. Symbol relacji wiele-do-wielu.

Do opisu powyższych relacji można dodatkowo wprowadzić parametry charakteryzujące opcjonalne bądź obligatoryjne wykorzystanie danego atrybutu. Opcjonalność zapisuje się, dodając puste koło obok symbolu relacji (rysunek 11).

Rysunek 11. Symbol relacji jeden do wielu z relacją opcjonalności rekordów.

Podsumowanie

W niniejszym artykule zaprezentowane zostały podstawy modelowania baz danych przy wykorzystaniu zasad normalizacji oraz diagramów ERD. W kolejnej części uwagę Czytelnika

skoncentrujemy na języku SQL (ze szczególnym naciskiem na Transact-SQL).

Paweł Wilkosz

Paweł Wilkosz pełni funkcję lidera PLSSUG Kraków od stycznia 2009 r. Na co dzień pracuje jako inżynier oprogramowania w firmie Motorola Solutions, Inc. Paweł pasjonuje się relacyjnymi bazami danych oraz rozwiązaniami klasy Business Inteligence ze szczególnym naciskiem na Microsoft SQL Server. W styczniu 2011 r. laureat nagrody Microsoft MVP – Most Valuable Professional (SQL Server). Chętnie zagłębia się w wewnętrzne mechanizmy serwera SQL oraz możliwości integracji i adaptacji danych do innych, nie tylko komercyjnych rozwiązań. Swoje spostrzeżenia publikuje na łamach magazynu Software Developer's Journal oraz aktywnie uczestnicząc w rozwoju produktów Open-Source (głównie w zakresie tworzenia rozszerzeń i komponentów BI).