

Systemy informatyczne
Studia I stopnia, tryb stacjonarny i niestacjonarny
Instrukcja laboratoryjna cz. 6 w7'6

1. Wybrane ogólne aspekty modelowania danych.

1.1 Związki (powiązania) encji.

Związki (powiązania) encji to pojęcie stosowane w kontekście baz danych, które odnosi się do powiązań między dwoma lub więcej encjami. Encja to zbiór informacji, który reprezentuje jednostkę lub obiekt, np. klienta, produkt, zamówienie itp.

Związki encji opisują, jakie powiązania występują między encjami w bazie danych. Przykładowo, encja "klient" może być powiązana z encją "zamówienie" poprzez relację "jeden do wielu", co oznacza, że jeden klient może mieć wiele zamówień, ale jedno zamówienie może być przypisane tylko do jednego klienta. Inne przykłady związków encji to relacja "jeden do jednego" (jedna encja jest powiązana tylko z jedną inną encją) i relacja "wiele do wielu" (wiele encji jest powiązanych z wieloma innymi encjami).

Dobrze zaprojektowane związki encji są kluczowe dla utrzymania spójności i integralności danych w bazie danych. W praktyce oznacza to, że dane są przechowywane i aktualizowane w sposób spójny, a zmiany w jednej encji automatycznie wpływają na inne powiązane z nią encje.

1.2 Związki (powiązania) 1:N identyfikujące i nieidentyfikujące.

a) Powiązania identyfikujące (ang. identifying relationships) - w takich relacjach klucz obcy w encji "wiele" zawiera część lub całość klucza głównego encji "jeden". Innymi słowy, klucz obcy w encji "wiele" jednoznacznie identyfikuje powiązaną encję w encji "jeden". Przykładem powiązania identyfikującego jest relacja między zamówieniem a jego pozycjami - klucz obcy w encji "pozycje zamówienia" zawiera numer zamówienia, co umożliwia jednoznaczne powiązanie każdej pozycji zamówienia z konkretnym zamówieniem.

b) Powiązania nieidentyfikujące (ang. non-identifying relationships) - w takich relacjach klucz obcy w encji "wiele" nie zawiera żadnej części klucza głównego encji "jeden". Klucz obcy umożliwia jedynie powiązanie rekordów z encji "wiele" z konkretną encją w encji "jeden", ale nie identyfikuje jej w sposób jednoznaczny. Przykładem powiązania nieidentyfikującego jest relacja między klientem a zamówieniem - każde zamówienie jest przypisane do jednego klienta, ale nie jest identyfikowane za pomocą klucza obcego zawierającego część klucza głównego encji "klient".

Dobrze zaprojektowane powiązania identyfikujące i nieidentyfikujące są ważne dla utrzymania spójności i integralności danych w bazie danych. Powiązania identyfikujące są szczególnie przydatne, gdy potrzebne są operacje zmiany, aktualizacji lub usuwania danych, ponieważ umożliwiają jednoznaczne identyfikowanie rekordów w bazie danych. Powiązania nieidentyfikujące są przydatne, gdy potrzebne są tylko informacje o powiązaniu rekordów bez potrzeby jednoznacznego ich identyfikowania.

1.3 Przykład dla powiązań identyfikujących i nieidentyfikujących.

Powiązania identyfikujące:

Encja 1: "Order" (zamówienie)

Encja 2: "OrderItem" (pozycja zamówienia)

Powiązanie: "Order" 1:N "OrderItem"

Komentarz: każda pozycja zamówienia jest powiązana z konkretnym zamówieniem, a zamówienie może mieć wiele pozycji. W tym przypadku klucz obcy encji "OrderItem" jest kluczem częściowym encji "Order", ponieważ składa się z klucza głównego encji "Order" oraz innego atrybutu (na przykład "OrderItemNumber"), który identyfikuje konkretną pozycję w zamówieniu.

Powiązania nieidentyfikujące:

Encja 1: "Student" (student)

Encja 2: "Course" (kurs)

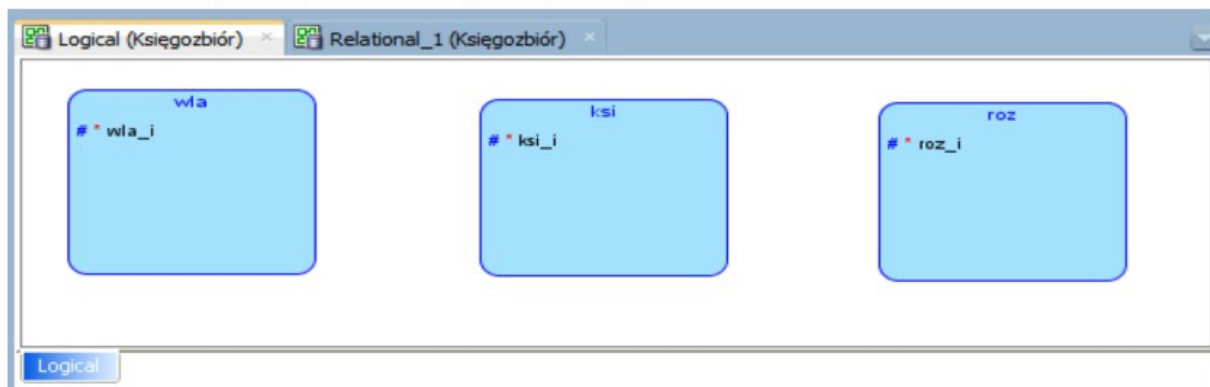
Powiązanie: "Student" N:M "Course"

Komentarz: studenci mogą być zapisani na wiele kursów, a kursy mogą mieć wielu studentów. W tym przypadku powiązanie jest nieidentyfikujące, ponieważ nie ma jednoznacznie określonego klucza głównego ani klucza obcego. Zamiast tego, w bazie danych jest utworzona trzecia tabela (na przykład "Enrollment"), która łączy klucze główne encji "Student" i "Course" za pomocą dwóch kluczy obcych.

1.4 Przykładowy wzornik "Księgozbiór" w SDDM.

Modelowane są trzy encje: wla (Właściciele) – nazwa opisowa wlas, ksi (Książki) – nazwa opisowa ksia, i roz (Rozdziały) – nazwa opisowa rozd, dla każdej jest definiowany wprost jeden atrybut kluczowy z typem Integer, pozostałe zgodnie z typem powiązania 1:N są dodawanie automatycznie przez program. Wzornik jest zapisywany pod nazwą „Księgozbiór”.

Model logiczny ERD początkowy, bez powiązań



Definicje poszczególnych encji – okno właściwości.

Encja wla

Entity Properties - wlas

General

Name: wlas

Short Name:

Synonyms:

Synonym to display: wla

Preferred Abbreviation: wla

Long Name: wlas

Based on Structured Type:

Super Type:

Source:

Allow Type Substitution: ☒

Create Surrogate Key: ☐

Deprecated: ☐

OK Apply Naming Rules Cancel Help

Entity Properties - wlas

Attributes

Details Overview UDP

Attributes:

Name	Data type
1 wla_j	Integer

Attribute Properties

Name: wla_j

Data Type: ☐ Domain ☒ Logical ☐ Distinct

☐ Structured ☐ Collection

Source Type: Integer Preferred ☐

☒ Primary UID ☐ Relation UID ☒ Mandatory ☐ Deprecated

Comments Comments in RDBMS Notes

OK Apply Naming Rules Cancel Help

Entity Properties - wlas

Unique Identifiers

Name	PUID	Deprecated
wlas PK	<input checked="" type="checkbox"/>	<input type="checkbox"/>

OK Apply Naming Rules Cancel Help

Encja ksi

Entity Properties - ksia

General

Name: ksia

Short Name:

Synonyms:

Synonym to display: ksi

Preferred Abbreviation: ksi

Long Name: ksia

Based on Structured Type:

Super Type:

Source:

Allow Type Substitution: ☒

Create Surrogate Key: ☐

Deprecated: ☐

OK Apply Naming Rules Cancel Help

Entity Properties - ksia

Attributes

Details Overview UDP

Attributes:

Name	Data type
1 ksi_	Integer

Attribute Properties

Name: ksi_

Data Type: ☐ Domain ☒ Logical ☐ Distinct

☐ Structured ☐ Collection

Source Type: Integer Preferred ☐

☒ Primary UID ☐ Relation UID ☒ Mandatory ☐ Deprecated

Comments Comments in RDBMS Notes

Entity Properties - ksia

Unique Identifiers

Name	PUID	Deprecated
ksia PK	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Encja roz

Entity Properties - roz

General

Name: roz

Short Name:

Synonyms:

Synonym to display: roz

Preferred Abbreviation: roz

Long Name: roz

Based on Structured Type:

Super Type:

Source:

Allow Type Substitution: ☒

Create Surrogate Key: ☐

Entity Properties - roz

Attributes

Details Overview UDP

Attributes:

Name	Data type
roz_j	Integer

Attribute Properties

Name: roz_j

Data Type: ☐ Domain ☒ Logical ☐ Distinct

☐ Structured ☐ Collection

Source Type: Integer Preferred ☐

☒ Primary UID ☐ Relation UID ☒ Mandatory ☐ Deprecated

Comments Comments in RDBMS Notes

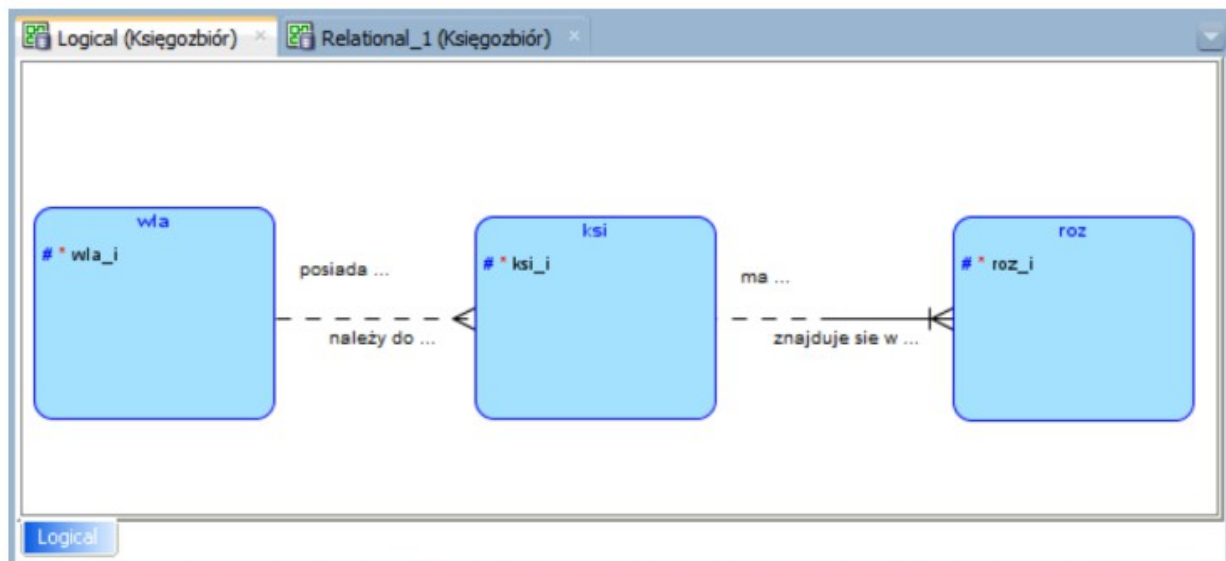
OK Apply Naming Rules Cancel Help

Entity Properties - roz

Unique Identifiers

Name	PUID	Deprecated
roz PK	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Model logiczny ERD po zdefiniowaniu powiązania 1:N

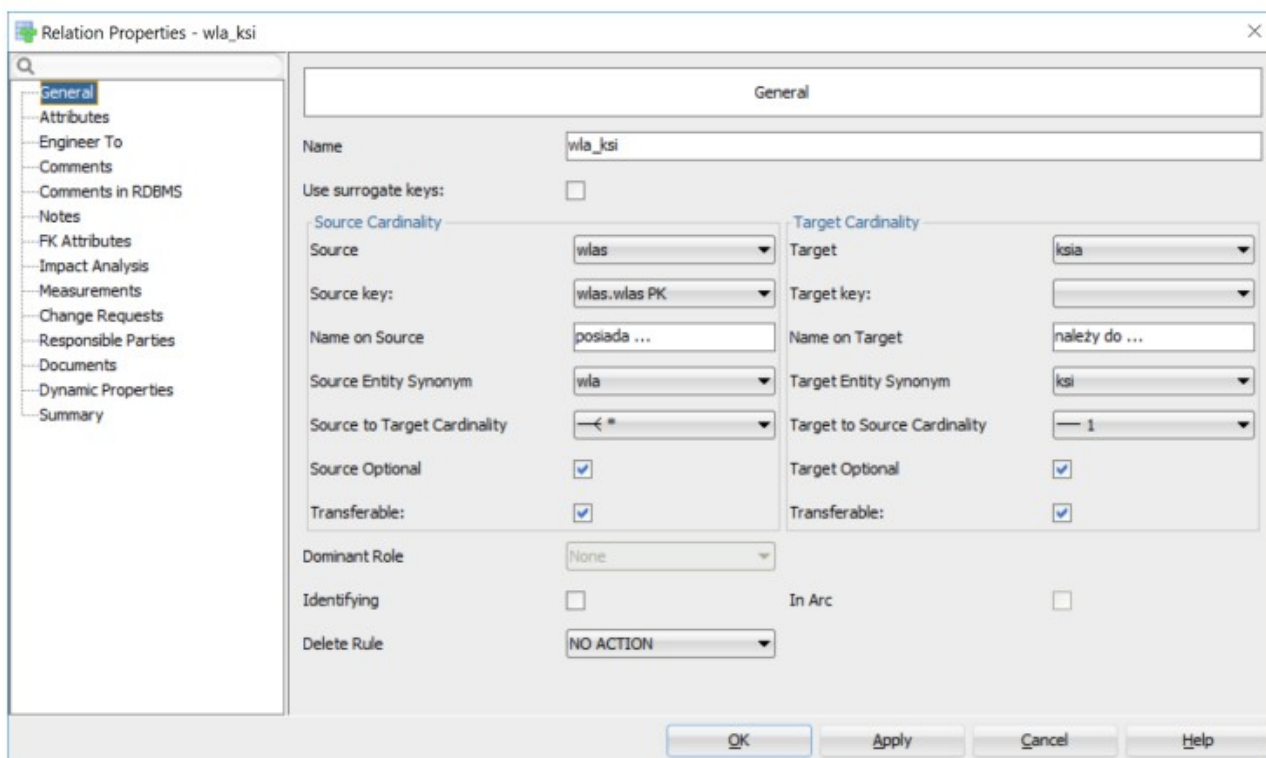


Kształty powiązań różnią się tylko dodanym paskiem (pionową kreską) obok symbolu „wiele” dla powiązania identyfikującego, natomiast zobrazowanie encji nie zmieniło się.

Definicje atrybutów encji ksi i roz po utworzeniu powiązania Wykaz atrybutów dla encji wla nie zmienił się, natomiast w przypadku pozostałych encji SDDM automatycznie utworzył (dodał) „miejsce” dla atrybutu, który będzie zawierał wartości atrybutu klucza głównego z encji źródłowej, czyli w danej encji będzie atrybutem-kluczem obcym. Należy w przypadku tego dodanego atrybutu zwrócić uwagę na wyszarzone pola edycyjne, wyboru i radiowe: „Name”, „Logical”, „Source Type”, „Mandatory”, i zaznaczone pola wyboru „Primary UID” oraz „Relational UID”.

Własności powiązania nieidentyfikującego wla_ksi

Encja wla – źródłowa, ksi – docelowa



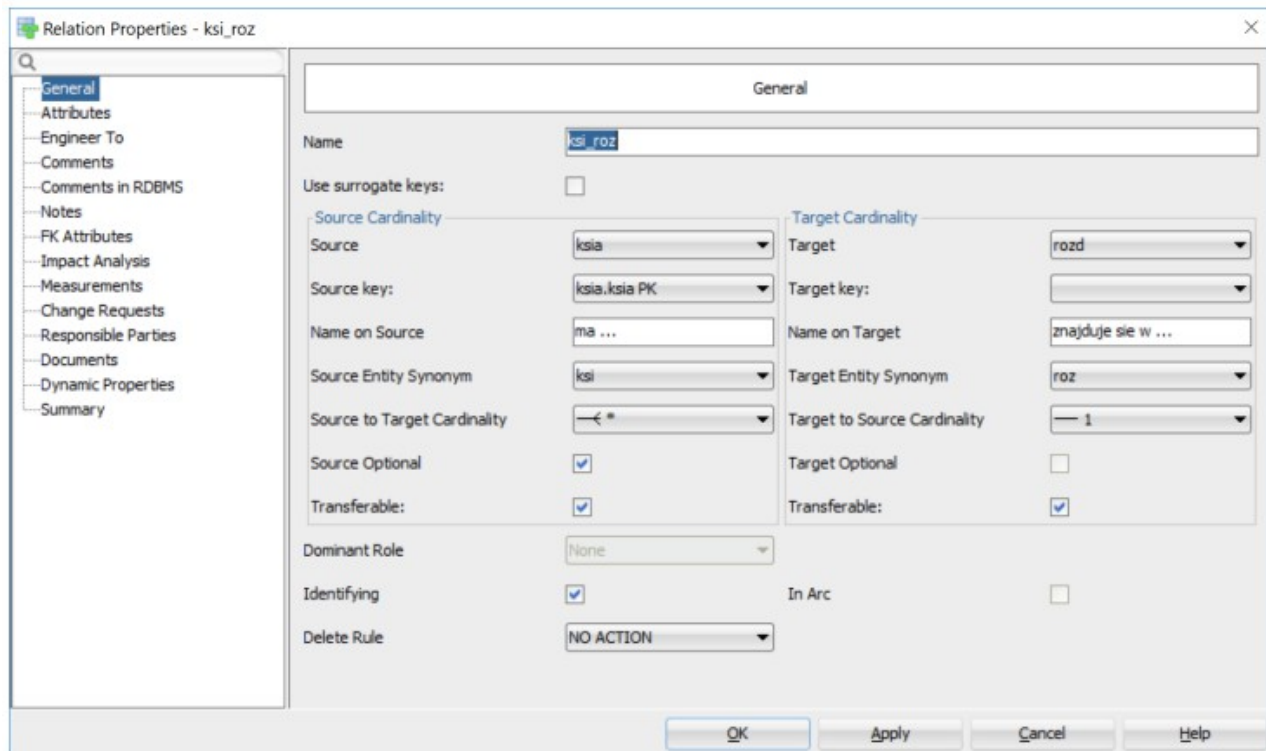
The dialog box 'Relation Properties - wla_ksi' shows the configuration for a non-identifying relationship. The 'General' tab is active. The 'Name' field is 'wla_ksi'. 'Use surrogate keys' is unchecked. Under 'Source Cardinality', 'Source' is 'wlas', 'Source key' is 'wlas.wlas PK', 'Name on Source' is 'posiada ...', 'Source Entity Synonym' is 'wla', 'Source to Target Cardinality' is '← *', 'Source Optional' is checked, and 'Transferable' is checked. Under 'Target Cardinality', 'Target' is 'ksia', 'Target key' is empty, 'Name on Target' is 'należy do ...', 'Target Entity Synonym' is 'ksi', 'Target to Source Cardinality' is '— 1', 'Target Optional' is checked, and 'Transferable' is checked. 'Dominant Role' is 'None', 'Identifying' is unchecked, 'In Arc' is unchecked, and 'Delete Rule' is 'NO ACTION'. Buttons at the bottom are 'OK', 'Apply', 'Cancel', and 'Help'.

General	
Name	wla_ksi
Use surrogate keys:	<input type="checkbox"/>
Source Cardinality	
Source	wlas
Source key:	wlas.wlas PK
Name on Source	posiada ...
Source Entity Synonym	wla
Source to Target Cardinality	← *
Source Optional	<input checked="" type="checkbox"/>
Transferable:	<input checked="" type="checkbox"/>
Target Cardinality	
Target	ksia
Target key:	
Name on Target	należy do ...
Target Entity Synonym	ksi
Target to Source Cardinality	— 1
Target Optional	<input checked="" type="checkbox"/>
Transferable:	<input checked="" type="checkbox"/>
Dominant Role	None
Identifying	<input type="checkbox"/>
In Arc	<input type="checkbox"/>
Delete Rule	NO ACTION

Własności powiązania identyfikującego ksi_roz

Encja ksi – źródłowa, roz – docelowa

Pole wyboru „Identifying” zaznaczone, pole wyboru „Target Optional” wyszarzone.

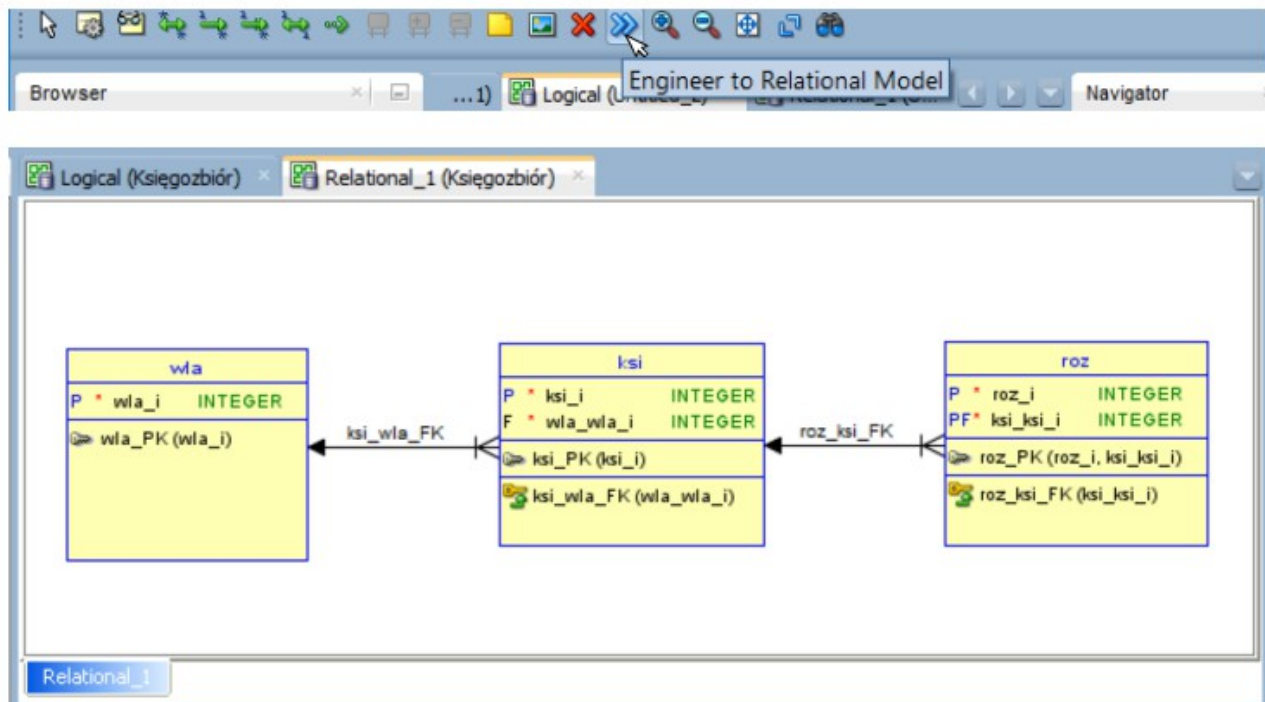


The dialog box 'Relation Properties - ksi_roz' shows the configuration for an identifying relationship. The 'General' tab is active. The 'Name' field is 'ksi_roz'. 'Use surrogate keys' is unchecked. Under 'Source Cardinality', 'Source' is 'ksia', 'Source key' is 'ksia.ksia PK', 'Name on Source' is 'ma ...', 'Source Entity Synonym' is 'ksi', 'Source to Target Cardinality' is '← *', 'Source Optional' is checked, and 'Transferable' is checked. Under 'Target Cardinality', 'Target' is 'roz', 'Target key' is empty, 'Name on Target' is 'znajduje sie w ...', 'Target Entity Synonym' is 'roz', 'Target to Source Cardinality' is '— 1', 'Target Optional' is disabled (greyed out), and 'Transferable' is checked. 'Dominant Role' is 'None', 'Identifying' is checked, 'In Arc' is unchecked, and 'Delete Rule' is 'NO ACTION'. Buttons at the bottom are 'OK', 'Apply', 'Cancel', and 'Help'.

General	
Name	ksi_roz
Use surrogate keys:	<input type="checkbox"/>
Source Cardinality	
Source	ksia
Source key:	ksia.ksia PK
Name on Source	ma ...
Source Entity Synonym	ksi
Source to Target Cardinality	← *
Source Optional	<input checked="" type="checkbox"/>
Transferable:	<input checked="" type="checkbox"/>
Target Cardinality	
Target	roz
Target key:	
Name on Target	znajduje sie w ...
Target Entity Synonym	roz
Target to Source Cardinality	— 1
Target Optional	<input type="checkbox"/>
Transferable:	<input checked="" type="checkbox"/>
Dominant Role	None
Identifying	<input checked="" type="checkbox"/>
In Arc	<input type="checkbox"/>
Delete Rule	NO ACTION

Diagram relacyjny

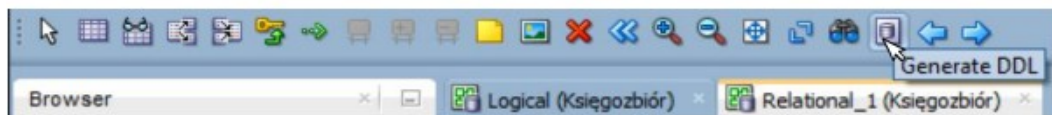
Tworzony jest po kliknięciu dla okna „Logical ...” ikony „Engineer ...”, a następnie przycisku „Engineer” w kolejnym oknie.



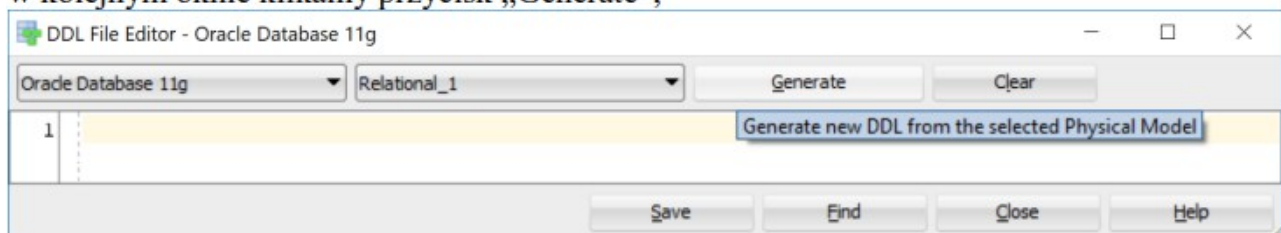
Na tym diagramie kształty powiązań nieidentyfikującego i identyfikującego są identyczne (oba mają „pasek UID”, czyli „UID Bar”), a różnica jest widoczna w określeniu klucza głównego tabel.

Utworzenie modelu fizycznego (skryptu budowania tabel w bazie),

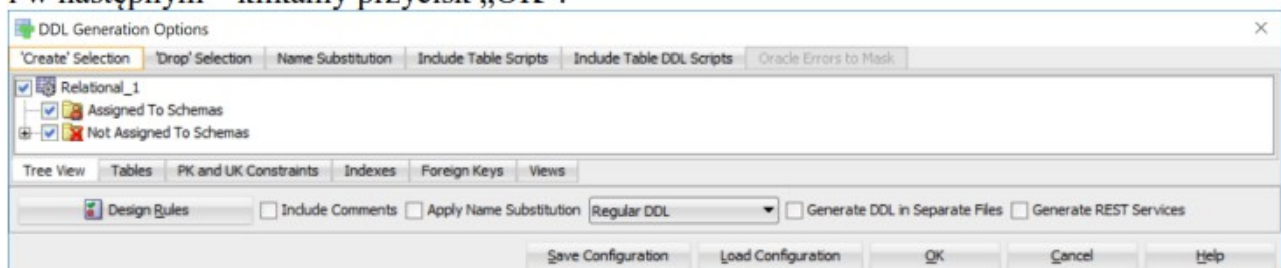
Po kliknięciu z okna modelu relacyjnego ikony „Generate DDL”



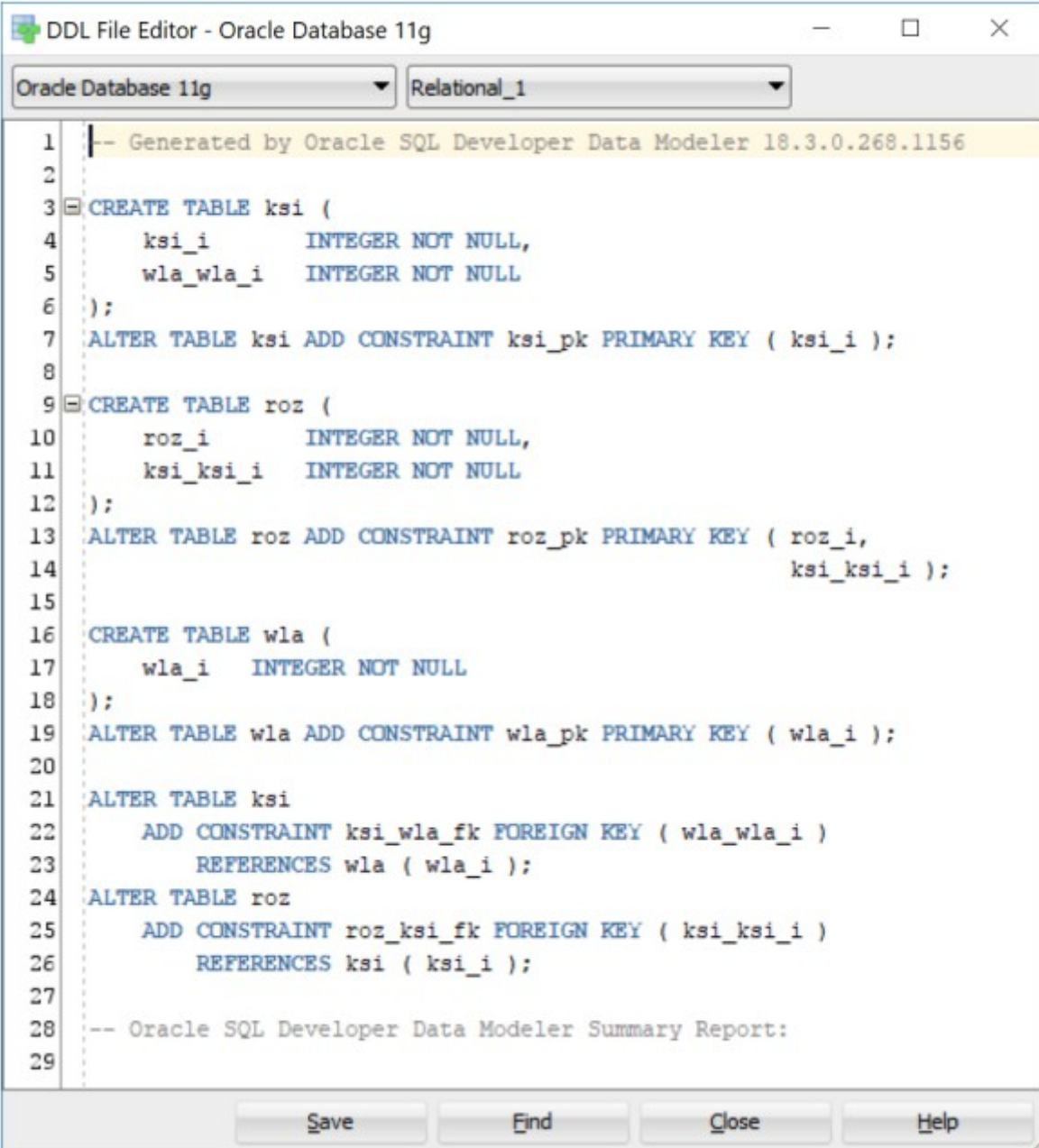
w kolejnym oknie klikamy przycisk „Generate”,



i w następnym – klikamy przycisk „OK”.



Wygeneruje to skrypt gotowy do wczytania do bazy:



```
-- Generated by Oracle SQL Developer Data Modeler 18.3.0.268.1156

1
2
3 CREATE TABLE ksi (
4     ksi_i          INTEGER NOT NULL,
5     wla_wla_i      INTEGER NOT NULL
6 );
7 ALTER TABLE ksi ADD CONSTRAINT ksi_pk PRIMARY KEY ( ksi_i );
8
9 CREATE TABLE roz (
10     roz_i          INTEGER NOT NULL,
11     ksi_ksi_i       INTEGER NOT NULL
12 );
13 ALTER TABLE roz ADD CONSTRAINT roz_pk PRIMARY KEY ( roz_i,
14                                                    ksi_ksi_i );
15
16 CREATE TABLE wla (
17     wla_i          INTEGER NOT NULL
18 );
19 ALTER TABLE wla ADD CONSTRAINT wla_pk PRIMARY KEY ( wla_i );
20
21 ALTER TABLE ksi
22     ADD CONSTRAINT ksi_wla_fk FOREIGN KEY ( wla_wla_i )
23     REFERENCES wla ( wla_i );
24 ALTER TABLE roz
25     ADD CONSTRAINT roz_ksi_fk FOREIGN KEY ( ksi_ksi_i )
26     REFERENCES ksi ( ksi_i );
27
28 -- Oracle SQL Developer Data Modeler Summary Report:
29
```

Analiza skryptu tworzenia tabel w bazie.

Ten kod tworzy trzy tabele: ksi, roz i wla, oraz dodaje ograniczenia klucza głównego i obcego między nimi.

W tabeli ksi definiowane są dwa atrybuty kluczowe: ksi_i i wla_wla_i, z których oba są wymagane (NOT NULL). W tabeli roz definiowany jest atrybut kluczowy roz_i i atrybut obcy ksi_ksi_i, który wskazuje na klucz główny tabeli ksi. W tabeli wla definiowany jest atrybut kluczowy wla_i.

Następnie są dodawane ograniczenia klucza głównego dla każdej z tabel. W tabeli ksi klucz główny składa się z atrybutu ksi_i, a w tabeli roz z atrybutów roz_i i ksi_ksi_i.

Kolejne dwa polecenia dodają ograniczenia klucza obcego. Pierwsze ograniczenie łączy kolumnę wla_wla_i w tabeli ksi z kolumną wla_i w tabeli wla. Drugie ograniczenie łączy kolumnę ksi_ksi_i w tabeli roz z kolumną ksi_i w tabeli ksi.

W powyższym kodzie SQL, istnieją powiązania identyfikujące oraz nieidentyfikujące między tabelami.

Powiązanie identyfikujące występuje między tabelą ksi a tabelą wla, gdzie klucz obcy wla_wla_i w tabeli ksi odnosi się do klucza głównego wla_i w tabeli wla. Dzięki temu powiązaniu, możemy zidentyfikować poszczególne wiersze w tabeli ksi na podstawie klucza głównego w tabeli wla.

Powiązanie identyfikujące występuje również między tabelą roz a tabelą ksi, gdzie klucz obcy ksi_ksi_i w tabeli roz odnosi się do klucza głównego ksi_i w tabeli ksi. Dzięki temu powiązaniu, możemy zidentyfikować poszczególne wiersze w tabeli roz na podstawie klucza głównego w tabeli ksi.

Natomiast powiązania nieidentyfikujące występują poprzez dodanie klucza składającego się z dwóch kolumn w tabeli roz, czyli roz_i oraz ksi_ksi_i, który odnosi się do klucza głównego w tabeli roz. Dzięki temu powiązaniu, możemy zidentyfikować poszczególne wiersze w tabeli roz, ale tylko w kontekście konkretnych wierszy w tabeli ksi.

1.5 Encje zależne (słabe) i niezależne (mocne).

W kontekście modelowania bazy danych encje można podzielić na dwie kategorie: encje zależne i niezależne.

Encja niezależna to encja, która istnieje niezależnie od innych encji w bazie danych. Oznacza to, że posiada swoje własne atrybuty, które nie zależą od innych encji, a także nie zawiera klucza obcego, który wskazywałby na powiązanie z inną encją.

Encja zależna z kolei to encja, która zależy od innej encji w bazie danych. Oznacza to, że posiada atrybuty, które są połączone z atrybutami innej encji za pomocą klucza obcego. Powiązanie między encjami zazwyczaj reprezentuje relację jeden do wielu, czyli jeden rekord w encji nadrzędnej może być powiązany z wieloma rekordami w encji zależnej.

Różnicą między encją zależną a niezależną jest to, że encja zależna zawsze wymaga istnienia encji nadrzędnej, z którą jest powiązana za pomocą klucza obcego. Natomiast encja niezależna może istnieć samodzielnie.

W praktyce często zdarza się, że encja zależna staje się nadrzędną dla innej encji, tworząc w ten sposób hierarchię powiązań między encjami.

2. Przykładowe zadanie do wykonania w SDDM.

Przykładowe zadanie, które można wykonać w SQL Developer Data Modeler, to stworzenie diagramu ERD dla systemu rezerwacji hotelowej.

Wymagania:

- Modelujemy trzy encje: Klienci, Hotele, Rezerwacje
- Dla każdej encji definiujemy wprost jeden atrybut kluczowy z typem Integer
- Powiązanie między Klientami a Rezerwacjami jest typu 1:N, a między Hotelami a Rezerwacjami jest typu N:M
- Dodajemy atrybuty dla każdej encji, takie jak: imię i nazwisko dla Klientów, nazwa i lokalizacja dla Hotelów, data rozpoczęcia i zakończenia dla Rezerwacji

Aby rozwiązać to zadanie, należy uruchomić SQL Developer Data Modeler i wykonać następujące kroki:

1. Utworzyć nowy projekt i nazwać go "System rezerwacji hotelowej".
2. Dodać trzy encje: Klienci, Hotele, Rezerwacje. Dodać dla każdej encji atrybut kluczowy o nazwie "id" i typie Integer.
3. Dodać atrybuty dla każdej encji: "imie" i "nazwisko" dla Klientów, "nazwa" i "lokalizacja" dla Hotelów, "data_rozpoczecia" i "data_zakonczenia" dla Rezerwacji.
4. Dodać powiązanie między Klientami a Rezerwacjami. Wybrać Klientów jako stronę 1 i Rezerwacje jako stronę N. Ustawić kardynalność na 1:N.
5. Dodać powiązanie między Hotelami a Rezerwacjami. Wybrać Hotele jako stronę N i Rezerwacje jako stronę M. Ustawić kardynalność na N:M.
6. Połączyć atrybut "id" dla encji Klienci i Hotele z odpowiednimi atrybutami w encji Rezerwacje, aby stworzyć powiązania identyfikujące.
7. Ustawić atrybuty kluczowe i powiązania identyfikujące jako klucze główne dla każdej encji.
8. Uruchomić generowanie kodu SQL i przetestować diagram ERD za pomocą polecenia "SELECT" w bazie danych.

Po wykonaniu tych kroków powinien powstać poprawny diagram ERD dla systemu rezerwacji hotelowej, który można wykorzystać do tworzenia bazy danych.

Skrypt tworzący powyższe tabele w języku Oracle:

```
CREATE TABLE klienci (  
    id INTEGER NOT NULL PRIMARY KEY,  
    imie VARCHAR2(50),  
    nazwisko VARCHAR2(50)  
);  
CREATE TABLE hotele (  
    id INTEGER NOT NULL PRIMARY KEY,  
    nazwa VARCHAR2(50),  
    lokalizacja VARCHAR2(100)  
);  
CREATE TABLE rezerwacje (  
    id INTEGER NOT NULL PRIMARY KEY,  
    data_rozpoczecia DATE,  
    data_zakonczenia DATE,  
    klient_id INTEGER NOT NULL,  
    hotel_id INTEGER NOT NULL,  
    FOREIGN KEY (klient_id) REFERENCES klienci(id),  
    FOREIGN KEY (hotel_id) REFERENCES hotele(id)  
);
```

Przykładowa baza danych, którą utworzyliśmy, składa się z trzech tabel: "Klienci", "Hotele" i "Rezerwacje".

1. Powiązanie między Klientami a Rezerwacjami to relacja jeden-do-wielu (1:N). Oznacza to, że jeden klient może dokonać wielu rezerwacji, ale każda rezerwacja dotyczy tylko jednego klienta. W tym przypadku, atrybut "id" w tabeli Klienci jest kluczem głównym, a atrybut "klient_id" w tabeli Rezerwacje jest kluczem obcym, który odnosi się do klucza głównego w tabeli Klienci.
2. Powiązanie między Hotelami a Rezerwacjami to relacja wiele-do-wielu (N:M). Oznacza to, że wiele hoteli może mieć wiele rezerwacji, a jedna rezerwacja może dotyczyć wielu hoteli. Aby zrealizować to połączenie, została dodana trzecia tabela pośrednicząca "Rezerwacje_Hotele", która zawiera klucze obce odwołujące się do tabeli Rezerwacje i Hotele.
3. Atrybut "id" w tabelach Klienci i Hotele jest kluczem głównym, a atrybuty "klient_id" i "hotel_id" w tabeli Rezerwacje_Hotele są kluczami obcymi, które odnoszą się odpowiednio do kluczy głównych w tabelach Klienci i Hotele.

Wszystkie klucze główne i klucze obce zostały zdefiniowane podczas projektowania bazy danych za pomocą SQL Developer Data Modeler, co umożliwiło łatwe i precyzyjne tworzenie tabel oraz ich powiązań.

W tej bazie danych można wykorzystać encje identyfikujące i nieidentyfikujące do określenia relacji między tabelami.

W przypadku tabeli "Rezerwacje", pola "klient_id" i "hotel_id" stanowią klucze obce, które odnoszą się do encji "Klienci" i "Hotele". Te klucze obce stanowią powiązania identyfikujące, ponieważ wskazują na rekordy w tabelach "Klienci" i "Hotele" na podstawie ich kluczy głównych ("id"). Dzięki temu możemy określić, kto dokonał rezerwacji i w którym hotelu.

Z drugiej strony, atrybuty "imie" i "nazwisko" w tabeli "Klienci" oraz "nazwa" i "lokalizacja" w tabeli "Hotele" to atrybuty nieidentyfikujące. Oznacza to, że są one zależne od klucza głównego tabeli, ale nie odnoszą się bezpośrednio do kluczy obcych w tabeli "Rezerwacje". Atrybuty te służą do przechowywania danych związanych z klientami i hotelami, które nie są bezpośrednio związane z rezerwacjami.

Mamy trzy encje: "Klienci", "Hotele" i "Rezerwacje". Każda z tych encji posiada atrybut kluczowy o nazwie "id" i typie Integer, który jest również kluczem głównym.

Encja "Klienci" posiada dwa dodatkowe atrybuty: "imie" i "nazwisko". Encja "Hotele" posiada również dwa dodatkowe atrybuty: "nazwa" i "lokalizacja". Encja "Rezerwacje" posiada dwa atrybuty opisujące daty rozpoczęcia i zakończenia rezerwacji oraz dwa atrybuty kluczowe: "klient_id" i "hotel_id". Te atrybuty są powiązaniami identyfikującymi z encjami "Klienci" i "Hotele" odpowiednio.

Mamy dwa powiązania: między "Klienci" i "Rezerwacje" oraz między "Hotele" i "Rezerwacje". Pierwsze powiązanie jest oznaczone jako 1:N, co oznacza, że jeden klient może mieć wiele rezerwacji, ale każda rezerwacja może być powiązana tylko z jednym klientem. Drugie powiązanie jest oznaczone jako N:M, co oznacza, że jeden hotel może być powiązany z wieloma rezerwacjami, a jedna rezerwacja może być powiązana z wieloma hotelami.

W obu powiązaniach powiązania identyfikujące są wykorzystywane jako klucze obce w tabeli "Rezerwacje", co zapewnia poprawność relacji między tabelami.

Takie diagramy ERD są często wykorzystywane do projektowania baz danych, ponieważ pozwalają na łatwe zrozumienie relacji między tabelami, a także na wizualne przedstawienie struktury bazy danych.

3. Krótkie podsumowanie wiadomości.

Encje to abstrakcyjne pojęcia reprezentujące obiekty lub zdarzenia w rzeczywistości. W kontekście modelowania bazy danych, encja to zbiór atrybutów opisujących dane obiektu oraz klucz identyfikujący (np. identyfikator klienta).

Encje mogą być identyfikowane lub nieidentyfikowane. Encja identyfikowana posiada klucz identyfikujący, który jednoznacznie identyfikuje rekord w tabeli, natomiast encja nieidentyfikowana nie posiada klucza identyfikującego.

Encje mogą być zależne lub niezależne. Encja zależna zależy od innej encji i jest z nią powiązana relacją, natomiast encja niezależna nie zależy od innej encji.

Diagram relacyjny to graficzne przedstawienie struktury bazy danych, w którym encje są reprezentowane przez prostokąty, a relacje między nimi są reprezentowane przez linie.

SQL Developer Data Modeler to narzędzie do projektowania baz danych, które pozwala na tworzenie diagramów relacyjnych oraz generowanie kodu SQL na podstawie tych diagramów.

W SQL Developer Data Modeler, encje są reprezentowane przez obiekty typu Entity, a relacje między encjami są reprezentowane przez obiekty typu Relationship.

Aby utworzyć diagram relacyjny w SQL Developer Data Modeler, należy wybrać opcję "New Diagram" z menu "File" i wybrać typ diagramu. Następnie należy dodać encje i relacje do diagramu za pomocą odpowiednich narzędzi.

Aby stworzyć klucz identyfikujący dla encji w SQL Developer Data Modeler, należy wybrać encję i wybrać opcję "Create Primary Key" z menu kontekstowego. Następnie należy wybrać atrybuty, które mają tworzyć klucz identyfikujący.

Zastosowanie encji i diagramów relacyjnych w SQL Developer Data Modeler polega na projektowaniu struktury bazy danych, której celem jest umożliwienie przechowywania i zarządzania danymi w sposób spójny i efektywny. Diagramy relacyjne pozwalają na łatwe wizualizowanie relacji między encjami, co ułatwia projektowanie i późniejsze zarządzanie bazą danych.