Bazy Danych

Park Rozrywki

Autorzy:

Justyna Budzyńska
Maria Kujawa

Spis treści

1. Zakres i cel projektu (opis założeń funkcjonalnych projektowanej bazy danych)

Celem projektu było zaprojektowanie relacyjnej bazy danych na poziomie konceptualnym jak i logicznym oraz jej fizyczna implementacja.

Oprogramowanie wykorzystane podczas realizacji projektu:

- ➔ lokalna baza danych Oracle Database w wersji 19c,
- ➔ narzędzie do projektowania i modelowania bazy danych Toad Data Modeler w wersji 7.3,
- ➔ narzędzie potrzebne do zarządzania bazą danych SQL Developer w wersji 21.4.3,
- ➔ Java w wersji 8.

Założenia funkcjonalne

Park rozrywki jest przedsiębiorstwem udostępniającym różnego rodzaju usługi klientom. Do tych usług zaliczamy: korzystanie z atrakcji oraz punktów gastronomicznych dostępnych na terenie parku, a także opcjonalnie w ofertę parku może wchodzić nocleg w hotelu, jeżeli dany park rozrywki takowy posiada, bądź jest podjęta współpraca z hotelem, który znajduje się w pobliżu. Klient może skorzystać z usług parku rozrywki po zakupieniu biletu w kasie. Istnieją różnorodne warianty czasowe oraz cenowe biletów (zależne od czasu wstępu jak i od ewentualnej zniżki). Park rozrywki zatrudnia pracowników na różnego rodzaju stanowiskach tj. pracownik kasy czy pracownik atrakcji, które mają wyróżnione dodatkowe atrybuty, ale istnieje również wiele innych stanowisk. Zatrudnieni pracownicy otrzymują miesięczne wynagrodzenie, które opcjonalnie może zawierać premię. Park rozrywki podzielony jest na strefy i każda z usług oferowana przez park (czyli każda atrakcja i punkt gastronomiczny) musi mieć zdefiniowaną strefę, w której się znajduje.

2. Definicja systemu

Definicja systemu zakłada możliwość:

- ➔ podglądu danych dotyczących parku rozrywki
- ➔ dodawania, usuwania i modyfikacji już istniejących danych dotyczących parku rozrywki
- ➔ podgląd danych dotyczących właścicieli
- ➔ dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących właścicieli
- ➔ podgląd danych dotyczących adresów
- ➔ dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących adresów
- ➔ podglądu danych dotyczących pracowników

- ➔ dodawania, usuwania i modyfikacji już istniejących danych dotyczących pracowników
- ➔ podgląd danych dotyczących specjalizacji pracownika - pracowników kasy
- ➔ dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących specjalizacji pracownika - pracowników kasy
- ➔ podgląd danych dotyczących specjalizacji pracownika - pracowników atrakcji
- ➔ dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących specjalizacji pracownika - pracowników atrakcji
- ➔ podgląd danych dotyczących stanowisk
- ➔ dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących stanowisk
- ➔ podgląd danych dotyczących wynagrodzeń
- ➔ dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących wynagrodzeń
- ➔ podglądu danych dotyczących punktów gastronomicznych znajdujących się na terenie parku rozrywki
- ➔ dodawania, usuwania i modyfikacji już istniejących danych dotyczących punktów gastronomicznych znajdujących się na terenie parku rozrywki
- ➔ podgląd danych dotyczących zatrudnień w punktach gastronomicznych
- ➔ dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących zatrudnień w punktach gastronomicznych
- ➔ podgląd danych dotyczących hoteli (należących do/współpracujących z parkiem rozrywki)
- ➔ dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących hoteli
- ➔ podgląd danych dotyczących zatrudnień w hotelu
- ➔ dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących zatrudnień w hotelu
- ➔ podgląd danych dotyczących kas
- ➔ dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących kas
- ➔ podgląd danych dotyczących kas, które są obsługiwane przez pracowników
- ➔ dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących kas, które są obsługiwane przez pracowników
- ➔ podgląd danych dotyczących biletów
- ➔ dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących biletów
- ➔ podgląd danych dotyczących klientów
- ➔ dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących klientów
- ➔ podgląd danych dotyczących atrakcji
- ➔ dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących atrakcji
- ➔ podgląd danych dotyczących atrakcji, które są obsługiwane przez pracowników
- ➔ dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących atrakcji, które są obsługiwane przez pracowników

- ➔ podgląd danych dotyczących stref
- ➔ dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących stref

2.1 Perspektywy użytkowników

Użytkownicy, którzy mogą korzystać z opisywanego przez nas systemu to:

- ➔ **Właściciel/Administrator**
  Może dodawać, usuwać i modyfikować wszystkie dane zapisane w bazie.

- ➔ **Pracownik**
  Posiada dostęp swoich do danych osobowych, swojego wynagrodzenia, danych miejsca, w którym jest zatrudniony (dotyczy wszystkich pracowników), przydzielonych kas (dotyczy pracowników kas) oraz przydzielonych atrakcji (dotyczy pracowników atrakcji),

- ➔ **Klient**
  Posiada dostęp do ogólnych danych dotyczących parku rozrywki, atrakcji znajdujących się w nim, danych o dostępnych, bądź zakupionych przez siebie biletach wstępu do parku oraz informacji o hotelu.

3. Model konceptualny

3.1 Definicja zbiorów encji określonych w projekcie (decyzje projektowe)

W niniejszym projekcie na poziomie konceptualnym wyróżniamy następujące encje:

- ➔ **Park_rozrywki** - encja główna, która zawiera najważniejsze informacje dotyczące parku rozrywki (posiada 8 atrybutów),
- ➔ **Atrakcja -** encja zawierająca informacje dotyczące atrakcji na terenie parku rozrywki (posiada 7 atrybutów),
- ➔ **Bilet** - encja zawierająca informacje dotyczące biletu wstępu do parku rozrywki (posiada 6 atrybutów),
- ➔ **Hotel** - encja zawierająca informacje dotyczące hotelu współpracującego z parkiem (posiada 6 atrybutów),
- ➔ **Kasa** - encja zawierająca informacje o kasie biletowej parku rozrywki (posiada 2 atrybuty),
- ➔ **Klient** - encja zawierająca informacje o kliencie parku rozrywki (posiada 7 atrybutów)
- ➔ **Pracownik** - encja zawierająca informacje o pracowniku parku rozrywki (posiada 15 atrybutów),

- ➔ **Pracownik_atrakcji** - encja reprezentująca specjalizacje pracownika parku rozrywki jako pracownika atrakcji (posiada 2 atrybuty),
- ➔ **Pracownik_kasy** - encja reprezentująca specjalizacje pracownika parku rozrywki jako pracownika kasy (posiada 3 atrybuty),
- ➔ **Punkt_gastronomiczny** - encja zawierająca informacje o punkcie gastronomicznym znajdującym się na terenie parku rozrywki (posiada 5 atrybutów).

3.2 Ustalenie związków między encjami i ich typów

Nazwa związku	Krotność	Typ uczestnictwa	Opis
Park_rozrywki - Pracownik	jeden do wielu	Park rozrywki jest obowiązkowy dla Pracownika	Pracownik musi pracować w jednym Parku rozrywki
		Pracownik jest opcjonalny dla Parku rozrywki	Park rozrywki może zatrudniać wielu Pracowników, ale zatrudnianie co najmniej jednego nie jest obowiązkowe
Park_rozrywki - Kasa	jeden do wielu	Park rozrywki jest obowiązkowy dla Kasy	Kasa musi być posiadana przez jeden Park rozrywki
		Kasa jest opcjonalna dla Parku rozrywki	Park rozrywki może posiadać wiele Kas, ale nie jest obowiązkowe posiadanie co najmniej jednej Kasy
Park_rozrywki - Atrakcja	jeden do wielu	Park rozrywki jest obowiązkowy dla Atrakcji	Atrakcja musi być posiadana przez jeden Park rozrywki
		Atrakcja jest opcjonalna dla Parku rozrywki	Park rozrywki może posiadać wiele Atrakcji, ale posiadanie co najmniej jednej nie jest obowiązkowe

Park_rozrywki - Hotel	jeden do wielu	Park rozrywki jest obowiązkowy dla Hotelu	Hotel musi być posiadany przez Park rozrywki.
		Hotel jest opcjonalny dla Parku rozrywki	Park rozrywki może posiadać wiele Hoteli, ale posiadanie co najmniej jednego nie jest obowiązkowe
Park_rozrywki - Punkt_gastronomiczny	jeden do wielu	Park rozrywki jest obowiązkowy dla Punktu gastronomicznego	Punkt gastronomiczny musi być posiadany przez jeden Park rozrywki
		Punkt gastronomiczny jest opcjonalny dla Parku rozrywki	Park rozrywki może posiadać wiele Punktów gastronomicznych, ale posiadanie co najmniej jednego nie jest obowiązkowe
Pracownik - Atrakcja	wiele do wielu	Pracownik jest opcjonalny dla Atrakcji	Atrakcja może być obsługiwana przez wielu pracowników, ale obsługiwanie jej przez co najmniej jednego pracownika nie jest obowiązkowe
		Atrakcja jest opcjonalna dla Pracownika	Pracownik może obsługiwać wiele atrakcji, ale obsługiwanie co najmniej jednej nie jest obowiązkowe
Pracownik - Kasa	wiele do wielu	Pracownik jest opcjonalny dla Kasy	Kasa może być obsługiwana przez wielu pracowników, ale obsługiwanie jej przez co najmniej jednego nie jest obowiązkowe
		Kasa jest opcjonalna dla Pracownika	Pracownik może pracować przy wielu kasach, ale obsługiwanie co najmniej jednej nie jest obowiązkowe

Pracownik - Punkt_gastronomiczny	wiele do wielu	Pracownik jest opcjonalny dla Punktu gastronomicznego	Punkt gastronomiczny może zatrudniać wielu Pracowników, ale zatrudnianie co najmniej jednego nie jest obowiązkowe
		Punkt gastronomiczny jest opcjonalny dla Pracownika	Pracownik może pracować w wielu Punktach gastronomicznych, ale pracowanie w co najmniej jednym nie jest obowiązkowe
Pracownik - Hotel	wiele do wielu	Pracownik jest opcjonalny dla Hotelu	Hotel może zatrudniać wielu Pracowników, ale zatrudnianie co najmniej jednego Pracownika nie jest obowiązkowe
		Hotel jest opcjonalny dla Pracownika	Pracownik może pracować w wielu Hotelach, ale pracowanie w co najmniej jednym nie jest obowiązkowe
Kasa - Bilet	jeden do wielu	Kasa jest obowiązkowa dla Biletu	Bilet musi sprzedawany przez jedną kasę
		Bilet jest opcjonalny dla Kasy	Kasa może sprzedawać wiele biletów, ale może też nie sprzedawać żadnego
Bilet - Klient	jeden do wielu	Bilet jest opcjonalny dla Klienta	Klient może kupić wiele Biletów, ale kupienie co najmniej jednego nie jest obowiązkowe
		Klient jest opcjonalny dla Biletu	Bilet może być kupiony przez jednego Klienta, ale kupienie go przez jakiegoś Klienta nie jest obowiązkowe

3.3 Określenie atrybutów i ich dziedzin

Nazwa encji:	Park_rozrywki		
Atrybut	**Typ i dziedzina**	**Czy obowiązkowy?**	**Opis**
ID_parku	Integer	TAK	Unikatowy identyfikator (ID) parku rozrywki.
Nazwa_parku	VarChar(30)	TAK	Nazwa parku rozrywki.
Adres_parku	VarChar(400)	TAK	Adres parku rozrywki. Pole segmentowe (ulica, numer budynku, numer lokalu, kod pocztowy, miejscowość, kraj)
Nr_telefonu	VarChar(15)	TAK	Numer telefonu parku rozrywki.
Adres_email	VarChar(30)	TAK	Adres email parku rozrywki.
Wlasciciel	VarChar(400)	TAK	Właściciel parku rozrywki. Pole segmentowe (imię, drugie imię, nazwisko, data urodzenia, adres zamieszkania, pesel, numer telefonu, adres email).
Liczba_atrakcji	Integer	TAK	Liczba atrakcji w parku rozrywki.
Czy_posiada_hotel	Czy_posiada_hotelD ('Tak', 'Nie')	TAK	Flaga mówiąca o tym, czy park rozrywki posiada swój hotel.

Nazwa encji:	Atrakcja		
Atrybut	**Typ i dziedzina**	**Czy obowiązkowy?**	**Opis**
ID_atrakcji	Integer	TAK	Unikatowy identyfikator (ID) atrakcji.
Nazwa_atrakcji	VarChar(30)	TAK	Nazwa atrakcji.
Liczba_miejsc	Integer	TAK	Liczba miejsc na atrakcji.
Minimalny_wiek	Integer	NIE	Minimalny wiek uczestnika atrakcji.
Minimalny_wzrost	Integer	NIE	Minimalny wzrost uczestnika atrakcji.
Poziom	PoziomD (‘Dla każdego’, ‘Familijny’,‘Normalny, ‘Średniozaawansowany’, ‘Ekstremalny’)	TAK	Poziom trudności/zaawansowania atrakcji (Dla każdego, Familijny, Normalny, Średniozaawansowany, Ekstremalny)
Strefa	VarChar(20)	TAK	Strefa, w której znajduje się atrakcja.

Nazwa encji:	Bilet		
Atrybut	**Typ i dziedzina**	**Czy obowiązkowy?**	**Opis**
ID_biletu	Integer	TAK	Unikatowy identyfikator (ID) biletu.
Typ_biletu	Typ_biletuD (‘Jednodniowy’, ‘Trzydniowy’, ‘Tygodniowy’)	TAK	Typ biletu (Jednodniowy, Trzydniowy, Tygodniowy).
Data_zakupu	Date	NIE	Data zakupu biletu.
Data_wstepu	Date	TAK	Data wstępu do parku rozrywki.
Cena_biletu	Money	TAK	Cena biletu.
Rodzaj_znizki	Rodzaj_znizkiD (‘Ulgowy’, ‘Studencki’, ‘Dla seniora’)	NIE	Rodzaj zniżki (Ulgowy, Studencki, Dla seniora).

Nazwa encji:	Hotel		
Atrybut	**Typ i dziedzina**	**Czy obowiązkowy?**	**Opis**
ID_hotelu	Integer	TAK	Unikatowy identyfikator (ID) hotelu.
Nazwa_hotelu	VarChar(30)	TAK	Nazwa hotelu.
Adres_hotelu	VarChar(400)	TAK	Adres hotelu. Pole segmentowe (ulica, numer budynku, numer lokalu, kod pocztowy, miejscowość, kraj).
Nr_telefonu	VarChar(15)	TAK	Numer telefonu hotelu.
Adres_email	VarChar(30)	TAK	Adres email hotelu.
Liczba_osob	Integer	TAK	Liczba miejsc w hotelu.

Nazwa encji:	Kasa		
Atrybut	**Typ i dziedzina**	**Czy obowiązkowy?**	**Opis**
ID_kasy	Integer	TAK	Unikatowy identyfikator (ID) kasy.
Nr_kasy	Integer	TAK	Numer kasy biletowej.

Nazwa encji:	Klient		
Atrybut	**Typ i dziedzina**	**Czy obowiązkowy?**	**Opis**
ID_klienta	Integer	TAK	Unikatowy identyfikator (ID) klienta.
Imie	VarChar(20)	TAK	Imię klienta.
Drugie_imie	VarChar(20)	NIE	Drugie imię klienta.
Nazwisko	VarChar(30)	TAK	Nazwisko klienta.
Data_urodzenia	Date	TAK	Data urodzenia klienta.
Nr_telefonu	VarChar(15)	TAK	Numer telefonu klienta.
Adres_email	VarChar(30)	NIE	Adres email klienta.

Nazwa encji:	Pracownik		
Atrybut	**Typ i dziedzina**	**Czy obowiązkowy?**	**Opis**
ID_pracownika	Integer	TAK	Unikatowy identyfikator (ID) pracownika.
Imie	VarChar(20)	TAK	Imię pracownika.
Drugie_imie	VarChar(20)	NIE	Drugie imię pracownika.
Nazwisko	VarChar(30)	TAK	Nazwisko pracownika.
Data_urodzenia	Date	TAK	Data urodzenia pracownika.
PESEL	Character(11)	NIE	PESEL pracownika.
Plec	PlecD (‘K’, ‘M’)	TAK	Flaga mówiąca o płci pracownika.
Nr_telefonu	VarChar(15)	TAK	Numer telefonu pracownika.
Adres_email	VarChar(30)	TAK	Adres email pracownika.
Adres_ zamieszkania	VarChar(400)	TAK	Adres zamieszkania pracownika. Pole segmentowe (ulica, numer budynku, numer lokalu, kod pocztowy, miejscowość, kraj)
Data_zatrudnienia	Date	TAK	Data zatrudnienia pracownika.
Data_zwolnienia	Date	NIE	Data zwolnienia pracownika.
Nr_konta	Character(26)	TAK	Numer konta pracownika.

Wynagrodzenie	Money	NIE	Wynagrodzenie miesięczne pracownika. Pole wielowartościowe, Pole segmentowe (data wynagrodzenia, kwota wynagrodzenia, kwota premii).
Stanowisko	VarChar(30)	TAK	Stanowisko pracownika.

Nazwa encji:	**Pracownik_atrakcji**		
Atrybut	**Typ i dziedzina**	**Czy obowiązkowy?**	**Opis**
Nr_licencji	Character(6)	TAK	Numer licencji pozwalającej na obsługę atrakcji.
Data_szkolenia	Date	TAK	Data ostatniego szkolenia (pracownik atrakcji)

Nazwa encji:	**Pracownik_kasy**		
Atrybut	**Typ i dziedzina**	**Czy obowiązkowy?**	**Opis**
Nr_dostepu	Character(16)	TAK	Numer karty pozwalającej na dostęp do kasy.
Data_szkolenia	Date	TAK	Data ostatniego szkolenia (pracownik kasy).
Czy_angielski	Czy_angielskiD (‘Tak’, ‘Nie’)	TAK	Flaga mówiąca o tym, czy dany pracownik kasy mówi też w języku angielskim.

Nazwa encji:	**Punkt_gastronomiczny**		
Atrybut	**Typ i dziedzina**	**Czy obowiązkowy?**	**Opis**
ID_punktu_ gastronomicznego	Integer	TAK	Unikatowy identyfikator (ID) punktu gastronomicznego.
Nazwa_punktu_ gastronomicznego	VarChar(30)	TAK	Nazwa punktu gastronomicznego.
Nr_budynku	Integer	TAK	Numer budynku, w którym znajduje się punkt.
Strefa	VarChar(30)	TAK	Strefa, w której znajduje się punkt gastronomiczny.

Typ	TypD (‘Foodtruck’, ‘Restauracja’, ‘Kawiarnia’, ‘Automat’)	TAK	Typ punktu gastronomicznego (Foodtruck, Restauracja, Kawiarnia Automat).

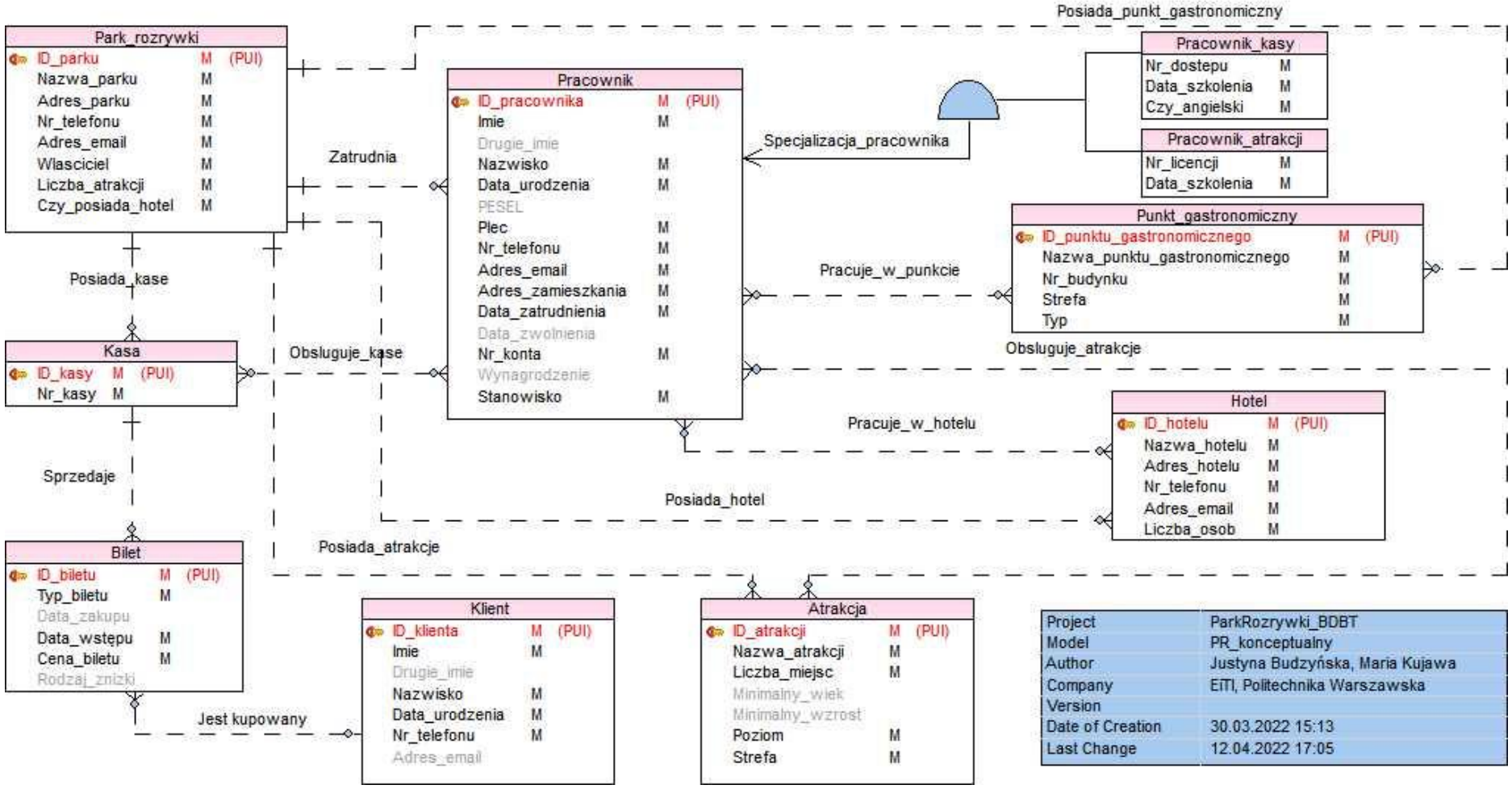
3.4 Dodatkowe reguły integralnościowe (reguły biznesowe)

- ➔ każdy z punktów gastronomicznych jest identyfikowany jako oddzielna usługa parku rozrywki pomimo występowania na tym samym obszarze co atrakcja (np. w przypadku automatów z jedzeniem i piciem, które mogą znajdować się w wielu miejscach, niekoniecznie w strefie gastronomicznej),
- ➔ park rozrywki nie musi posiadać, ani nie musi współpracować z żadnym hotelem (jest to opcjonalne). Sytuacja, kiedy go posiada/współpracuje z nim nie oznacza jednak, że hotel ten znajduje się pod tym samym adresem co park (może się on znajdować w pobliżu, w promieniu kilku kilometrów od parku),
- ➔ pracownik atrakcji musi posiadać numer licencji pozwalającej mu obsługiwać daną atrakcję w parku rozrywki, ma to na celu ograniczenie możliwości obsługi atrakcji dla innych pracowników parku rozrywki,
- ➔ pracownik kasy musi posiadać kartę z numerem dostępu (umożliwiającą obsługę kasy) przypisanym do konkretnego pracownika tak, aby można było łatwo zidentyfikować pracownika, który przeprowadził daną transakcję (np. w przypadku jakiegoś błędu)

3.5 Klucze kandydujące i główne (decyzje projektowe)

Nazwa encji	Klucz główny	Klucz kandydujący
Park_rozrywki	ID_parku	Nazwa_parku, Adres_email
Atrakcja	ID_atrakcji	Nazwa_atrakcji
Bilet	ID_biletu	-
Hotel	ID_hotelu	Nazwa_hotelu, Adres_email
Kasa	ID_kasy	Nr_kasy
Klient	ID_klienta	Nr_telefonu, Adres_email
Pracownik	ID_pracownika	PESEL, Nr_telefonu, Adres_email
Pracownik_atrakcji	ID_pracownika	Nr_licencji
Pracownik_kasy	ID_pracownika	Nr_dostepu
Punkt_gastronomiczny	ID_punktu_gastronomicznego	Nazwa_punktu_gastronomicznego

3.6 Schemat ER na poziomie konceptualnym



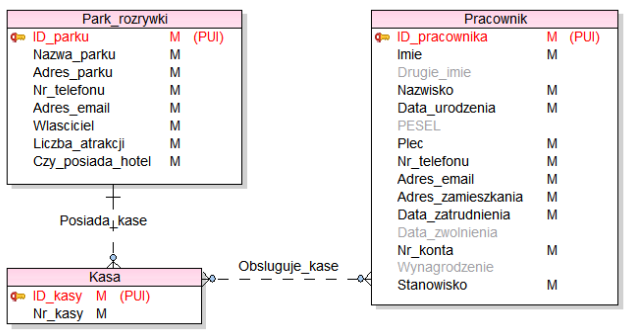
3.7 Problem pułapek szczelinowych i wachlarzowych – analiza i przykłady

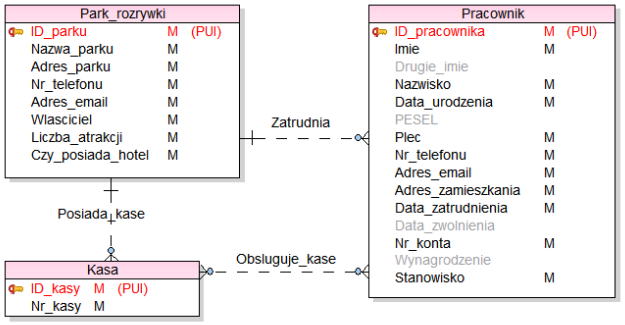
Pułapki szczelinowe

➔ Park rozrywki musi zatrudniać pracowników, gdyż nie każdy pracownik musi być pracownikiem kasy, atrakcji, hotelu czy punktu gastronomicznego. Pracownik może zajmować zdefiniowane stanowisko w parku rozrywki, nie będąc przy tym pracownikiem żadnego z powyższych miejsc (np. księgowy lub projektant atrakcji).

Przedstawienie przykładu pułapki szczelinowej i jej rozwiązania

PUŁAPKA SZCZELINOWA	ROZWIĄZANIE



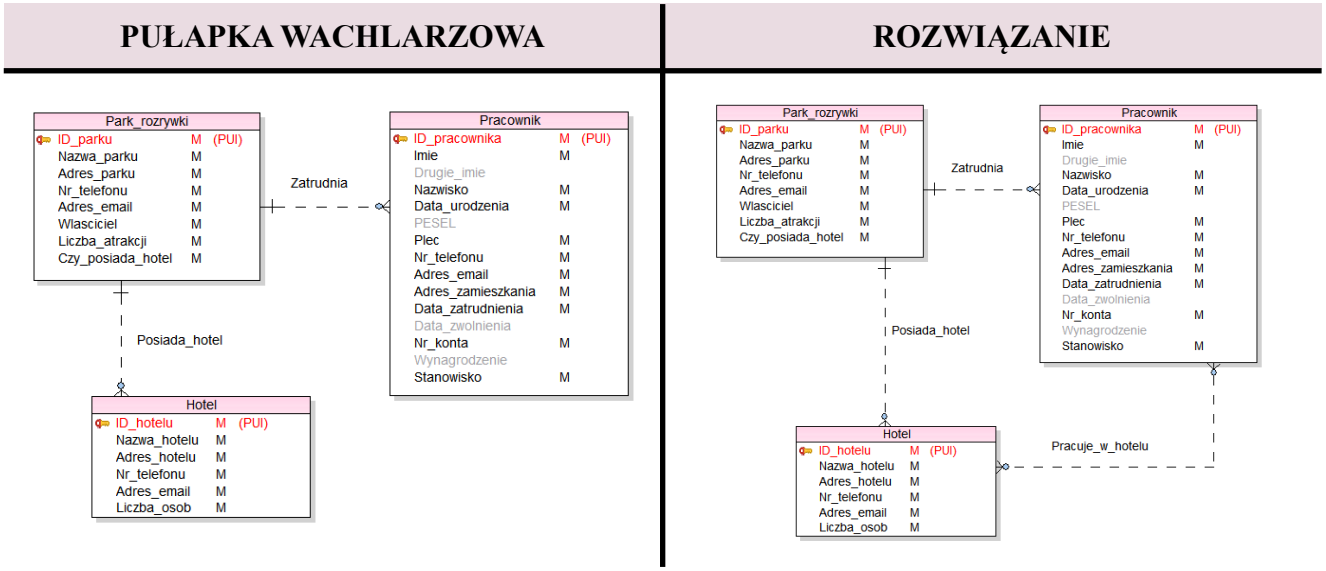


W sytuacji, w której nie ma powiązania pomiędzy pracownikiem a parkiem rozrywki istnieje zależność, że każdy pracownik musi być przypisany do kasy (lub punktu gastronomicznego, hotelu czy atrakcji). Rozwiązaniem takiej sytuacji jest wprowadzenie powiązania *zatrudnia* pomiędzy parkiem rozrywki i pracownikiem. Wtedy Pracownik jest zatrudniony przez park rozrywki, ale dzięki zachowaniu powiązania pomiędzy konkretną kasą (lub punktem gastronomicznym, hotelem czy atrakcją) wiemy w jakim miejscu pracuje. W przypadku pracowników, którzy nie pracują w żadnym z powyższych miejsc parku rozrywki, również nie ma problemu, gdyż są oni zatrudnieni po prostu przez park rozrywki.

Pułapki wachlarzowe

➔ Encja określająca miejsce pracy musi być jasno powiązana z pracownikiem, gdyż w przypadku braku tego powiązania pojawia się tzw. wachlarz możliwości. Polega on na tym, że nie jesteśmy w stanie jednoznacznie określić miejsca, w którym dany pracownik zajmuje zdefiniowane stanowisko.

Przedstawienie przykładu pułapki wachlarzowej i jej rozwiązania



W sytuacji widocznej powyżej po lewej stronie, kiedy nie istnieje powiązanie pomiędzy pracownikiem a hotelem (bądź punktem gastronomicznym, kasą czy atrakcją) nie mamy jednoznacznego powiązania pracownika z miejscem zatrudnienia, co w przypadku niektórych pracowników jest niezbędne. Wszyscy pracownicy są zatrudnieni przez park rozrywki przez co pojawia się tzw. wachlarz możliwości jeżeli chodzi o konkretne miejsce, w którym pracownik zajmuje dane stanowisko. Rozwiązaniem tej pułapki jest wprowadzenie powiązania *pracuje_w_hotelu* pomiędzy pracownikiem i hotelem (bądź punktem gastronomicznym, kasą czy atrakcją), co pozwoli nam na jasne zdefiniowanie miejsca, do którego jest przypisany pracownik na danym stanowisku.

4. Model logiczny

4.1 Charakterystyka modelu relacyjnego

Po zakończeniu projektowania, tworzenia i weryfikacji poprawności (funkcja ***Verify Model)*** modelu konceptualnego przy pomocy funkcji dostępnej w programie Toad Data Modeler 7.3 - ***Convert Model*** dokonana została konwersja do modelu relacyjnego. Podczas konwersji:

- każdy związek z krotnością “wiele do wielu” został zastąpiony dwoma związkami “jeden do wielu” (z obowiązkowością od strony “jeden”) wraz z utworzeniem tabeli łączącej
- do tabel łączących zostały zapisane klucze główne encji nadrzędnych, które oznaczone są na niebiesko
- identyfikujący atrybut każdej z encji stał się kluczem głównym tabeli oznaczonym na czerwono

- dla każdego związku “jeden do wielu” w tabeli od strony “wielu” pojawił się nowy atrybut - klucz obcy (odpowiadający kluczowi głównemu tabeli od strony “jeden”) oznaczony na zielono
- dla każdego związku “jeden do jeden” w tabeli potomka pojawił się nowy atrybut - klucz obcy (odpowiadający kluczowi głównemu tabeli rodzica) oznaczony na zielono
- tabele specjalizacji zostały uzupełnione o klucz główny encji nadrzędnej, który oznaczony jest na niebiesko
- tabele specjalizacji zostały powiązane z tabelą nadrzędną relacją “jeden do jeden” z domyślnie ustawioną nieobowiązkowością po stronie potomka i obowiązkowością po stronie rodzica
- typy danych uległy konwersji na takie, które będą zgodne z wybranym silnikiem bazy danych

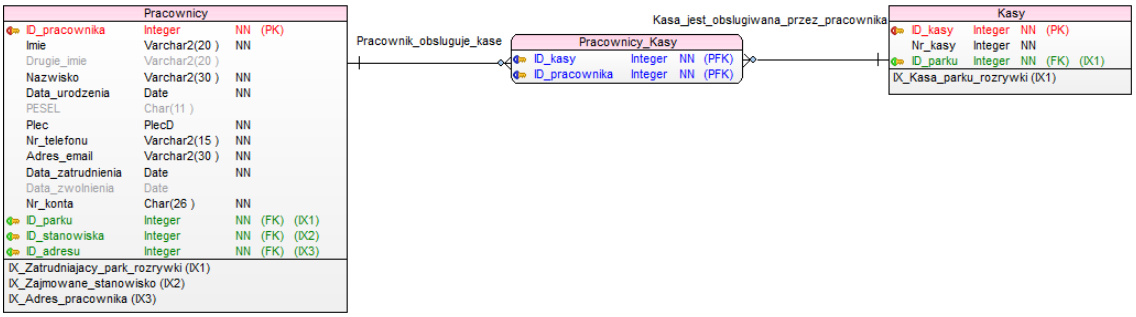
4.2 Usunięcie właściwości niekompatybilnych z modelem relacyjnym - przykłady

Usunięcie związków wiele do wielu

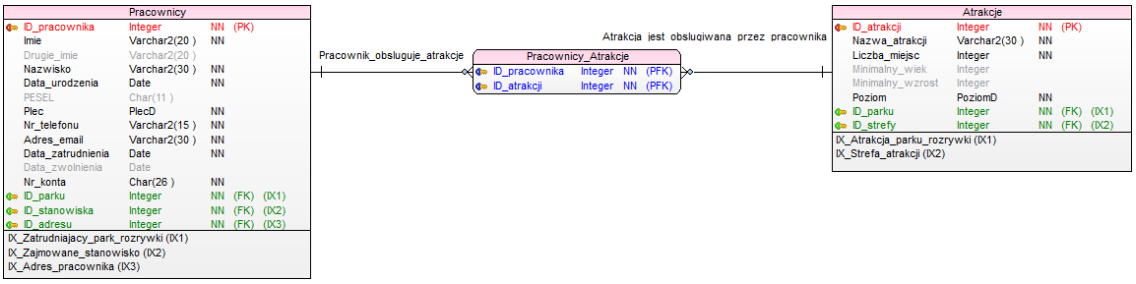
Między encją **Pracownik** a encją **Kasa** w modelu konceptualnym występował związek “wiele do wielu”, który został zastąpiony w modelu logicznym dwoma związkami “jeden do wielu” z obowiązkowością od strony jeden i tabelą łączącą.

Nadano jej nazwę **Pracownicy_Kasy**.

Celem tabeli **Pracownicy_Kasy** jest przechowywanie danych o tym, który pracownik obsługuje którą kasę.



Tabelę bridge’ującą, która utworzyła się między tabelami **Pracownicy** a **Atrakcje** nazwano **Pracownicy_Atrakcje**. Ma ona na celu przechowywanie informacji o tym, który pracownik obsługuje którą atrakcję.



Następną taką tabelą jest tabela **Zatrudnienia_gastronomia** utworzona między tabelami **Pracownicy** i **Punkty_gastronomiczne**. Przechowuje ona informacje o tym, w którym punkcie gastronomicznym pracuje który pracownik, a także informację o ważności badania sanepidowskiego wymaganego przy pracy w gastronomii.

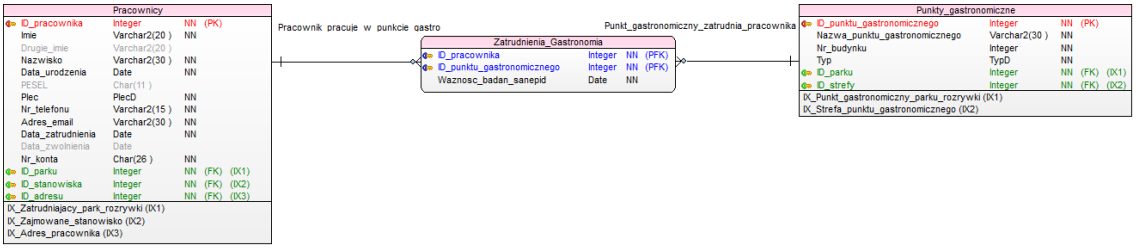
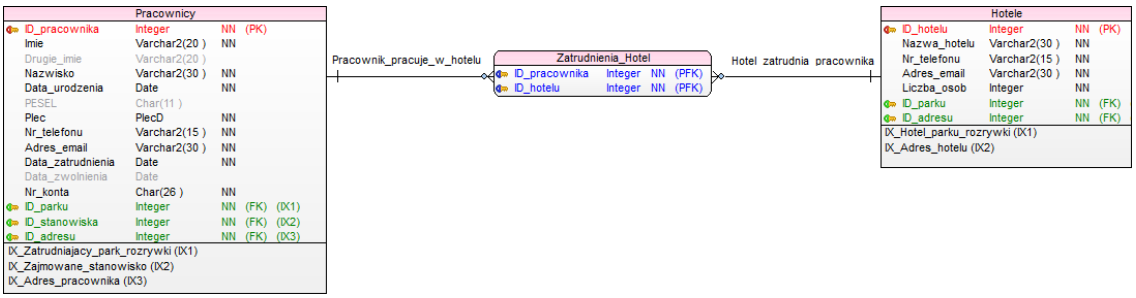


Tabela łącząca utworzyła się również między tabelami **Pracownicy** i **Hotele**. Została ona nazwana **Zatrudnienia_hotel** i przechowuje informacje o tym, który pracownik pracuje w którym hotelu.



Ponadto w celu odróżnienia relacji (tabele występujące w modelu logicznym) od encji (tabele występujące w modelu konceptualnym) nazwę każdej tabeli zmieniono na liczbę mnogą. Wszystkie inne tabele utworzone na etapie budowania modelu logicznego również posiadają nazwę w liczbie mnogiej (np. Wynagrodzenia, Adresy itd.).

4.3 Proces normalizacji – analiza i przykłady

Proces normalizacji ma na celu wyeliminowanie powtarzających się danych w relacyjnej bazie danych. Głównym założeniem jest trzymanie ich w jednym miejscu i linkowanie ich w razie potrzeby. Stosując się do jej zasad zwiększa się bezpieczeństwo danych oraz zmniejsza prawdopodobieństwo wystąpienia jakichkolwiek niespójności.

Pierwsza postać normalna zakłada, że każda wartość atrybutu w każdej krotce tej relacji jest wartością atomową oraz że nie występują powtarzające się grupy.

W naszym projekcie na poziomie konceptualnym istniały trzy atrybuty nie będące wartościami atomowymi:

- ➔ Pierwszy z nich określał adres i występował w encjach: **Park_rozrywki**, **Pracownik** oraz **Hotel** (był to odpowiednio **Adres_parku**, **Adres_zamieszkania**, **Adres_hotelu**). Atrybut ten był polem segmentowym, na które składały się informacje takie jak: ulica, numer budynku, numer lokalu, kod pocztowy, miejscowość oraz kraj. Aby osiągnąć 1PN utworzono nową relację **Adresy**, której atrybutami stały się wyżej wymienione szczegółowe dane dotyczące adresu. Relację tę połączono z tabelami **Parki_rozrywki**, **Pracownicy** oraz **Hotele**.
- ➔ Drugim był atrybut **Wlasciciel** występujący w encji **Park_rozrywki**. Było to pole segmentowe, na które składały się takie informacje jak: imię, drugie imię, nazwisko, data urodzenia, adres zamieszkania, pesel, numer telefonu, adres email. W celu osiągnięcia 1PN utworzono relację **Wlasciciele**, której atrybutami stały się wyżej wymienione informacje (oprócz adresu). Relację tę połączono z tabelami **Parki_rozrywki** oraz **Adresy** (właściciel posiada adres zamieszkania).
- ➔ Ostatnim był atrybut **Wynagrodzenie** znajdujący się w encji **Pracownik**. Jest to pole wielowartościowe, segmentowe, które przechowuje informacje o dacie wynagrodzenia, kwocie wynagrodzenia oraz kwocie premii. W celu osiągnięcia 1PN utworzono nową relację **Wynagrodzenia**, której atrybutami stały się wyżej wymienione informacje. Relację tę połączono z tabelą **Pracownicy**.

Druga postać normalna zakłada, że relacja musi być w 1PN oraz że każdy atrybut tej relacji nie wchodzący w skład żadnego klucza potencjalnego jest w pełni funkcyjnie zależny od wszystkich kluczy potencjalnych tej relacji. Musi być także spełniony warunek, że każdy atrybut nie wchodzący w skład klucza zależy od klucza, a nie tylko od jego części. Jeśli wszystkie klucze potencjalne w 1PN są kluczami prostymi to relacja jest w 2PN.

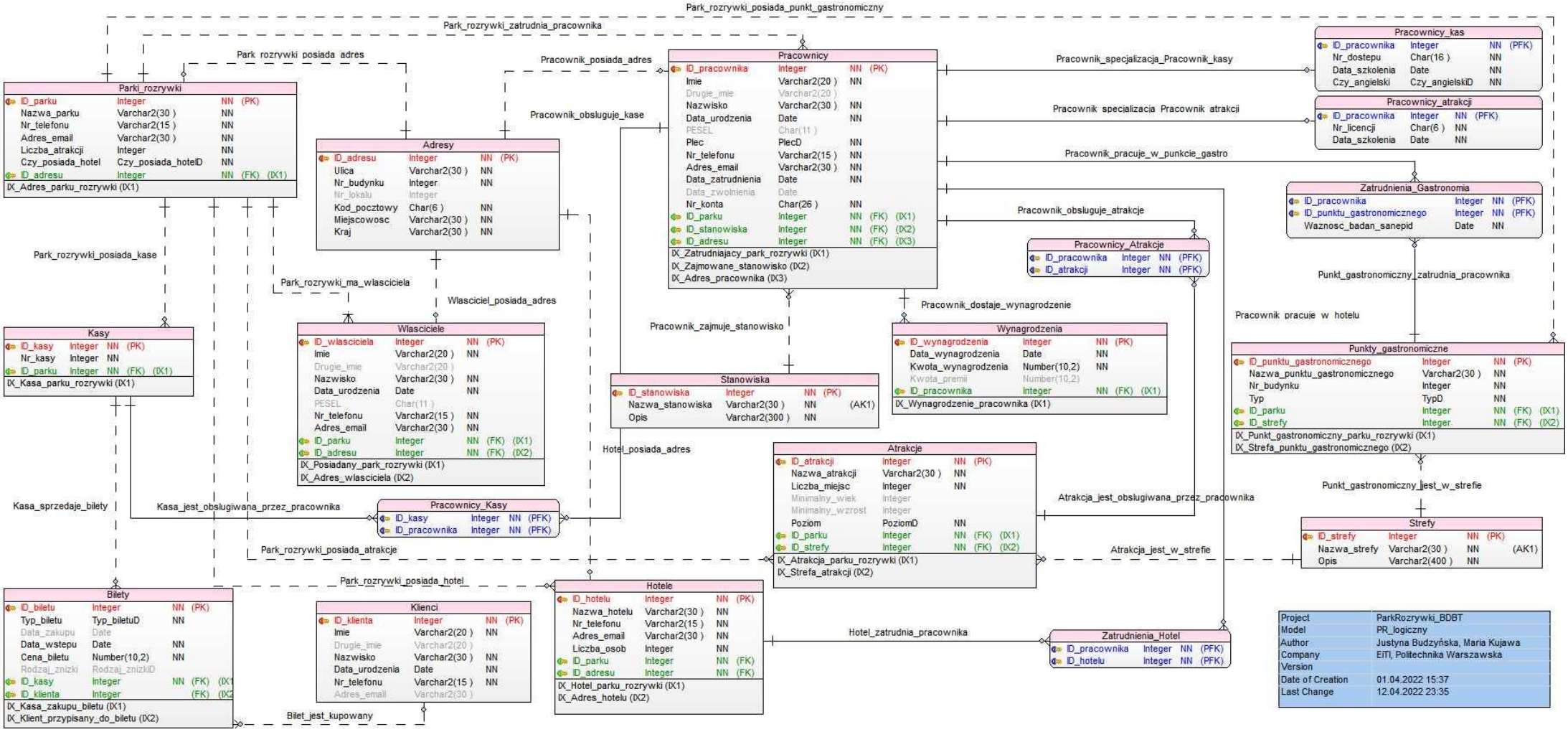
W naszym projekcie wszystkie klucze są kluczami prostymi. W związku z tym 2PN została osiągnięta automatycznie, bez wprowadzania jakichkolwiek zmian.

Trzecia postać normalna zakłada, że relacja musi być w 2PN oraz że każdy jej atrybut nie wchodzący w skład żadnego klucza potencjalnego nie jest przechodnio funkcyjnie zależny od żadnego klucza potencjalnego tej relacji. W naszym projekcie nie ma żadnych atrybutów, których nie dałoby się zidentyfikować wykorzystując klucz główny. Z tego wynika, że 3PN również została osiągnięta bez potrzeby wprowadzania jakichkolwiek dodatkowych zmian.

Podsumowanie

Cały model bazy danych został uproszczony w jak największym stopniu, nie zawiera żadnych pól wielowartościowych i segmentowych, a atrybuty dobrane zostały w taki sposób, aby precyzyjnie reprezentować każdą z relacji. Wszystkie klucze potencjalne relacji są kluczami prostymi i każdy atrybut nie wchodzący w skład żadnego klucza potencjalnego nie jest przechodnio funkcyjnie zależny od żadnego klucza potencjalnego tej relacji. Ostatecznie udało się osiągnąć 3PN w naszej bazie danych dotyczącej Parku Rozrywki.

4.4 Schemat ER na poziomie modelu logicznego



4.5 Więzy integralności

Warunki integralności danych dotyczą zapewnienia zgodności zawartości pól rekordu z typem rekordu, który został dla niego wcześniej określony. Ponadto, każde pole kluczowe ma być unikalne.

W naszym projekcie nie występuje już żadne pole segmentowe. Podczas próby osiągnięcia pierwszej postaci normalnej pola takie zostały zastąpione nowymi relacjami, które posiadają jedynie pola atomowe. Co za tym idzie, każde z pól może mieć jasno określony typ.

Zdecydowana większość pól jest określona jako obowiązkowe (NOT NULL) oraz posiada określony typ, jaki może przyjmować zawartość pola tego rekordu.

Przykładowo typ atrybutów dotyczących pieniędzy tj. Cena_biletu, Kwota_wynagrodzenia i Kwota_premii został określony jako Number(10,2) co oznacza, że będzie on przechowywał liczbę zmiennoprzecinkową z maksymalnie dwoma miejscami po przecinku (co jest zgodne z systemem pieniężnym). Powyższy typ danych oznacza również, że w polach mówiących o wartości pieniężnej nie mogą pojawić się wartości tekstowe.

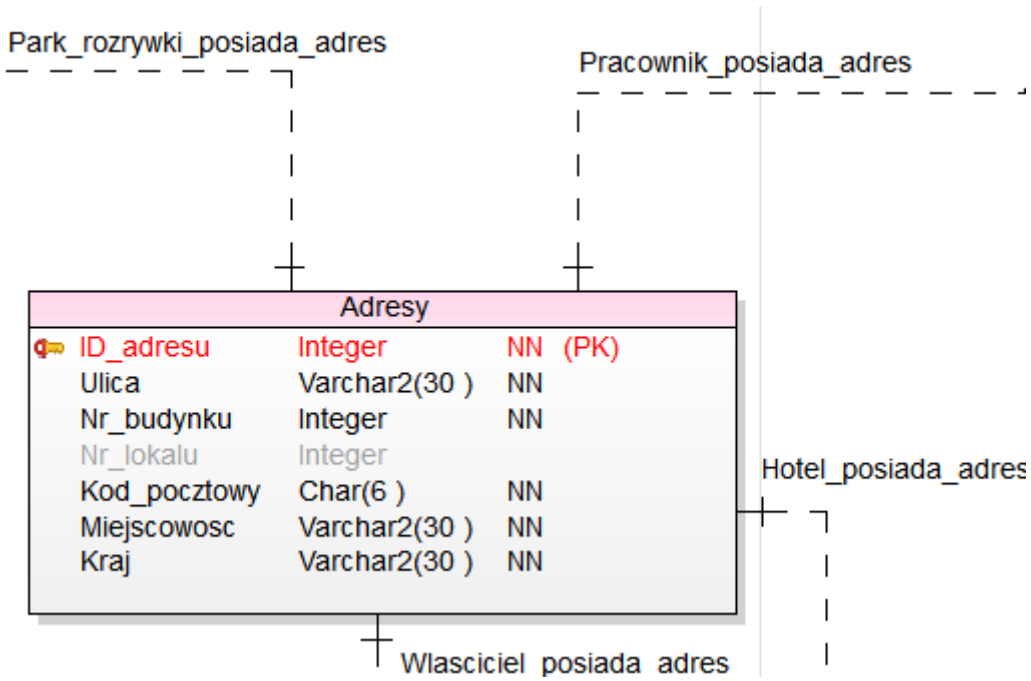
Warunkiem integralności danych jest również unikalność pola kluczowego, co także udało się spełnić w trakcie procesu normalizacji.

4.6 Proces denormalizacji – analiza i przykłady

Proces denormalizacji jest procesem odwrotnym do procesu normalizacji. Polega on na połączeniu niektórych znormalizowanych tabel w celu przyspieszenia dostępu do pewnych danych. Wadami takiego procesu jest mniejsza elastyczność systemu oraz zwiększenie skomplikowania projektu. Przeprowadzenie tego procesu jest uzasadnione tylko wtedy, gdy występuje duże prawdopodobieństwo, że system nie będzie spełniał wymagań z zakresu wydajności przetwarzania.

Nie zauważono potrzeby przeprowadzenia tego procesu w naszym projekcie.

Chcąc jednak przeanalizować jak mógłby wyglądać proces denormalizacji można się posłużyć przykładem relacji **Adresy**.



Jak przedstawia powyższa grafika relacja **Adresy** ma powiązanie z aż 4 innymi relacjami (**Parkami_rozrywki**, **Pracownikami**, **Wlascicielami** i **Hotelami**). W procesie denormalizacji należałoby usunąć relację Adresy i poprzenosić każdy z jej atrybutów osobno do każdej z 4 powiązanych z nią relacji. Taki zabieg wiązałby się z wysoką redundancją danych. Dlatego też, w tym przypadku nawet chęć przyspieszenia dostępu do informacji dotyczących adresu poszczególnej jednostki nie jest wystarczającym powodem do przeprowadzenia procesu denormalizacji ze względu na duże ryzyko powtarzania danych oraz wiążącego się z tym nadmiarowego zajmowania przestrzeni dyskowej.

5. Faza fizyczna

5.1 Projekt transakcji i weryfikacja ich wykonalności

Transakcja	Czy wykonalne?	Potrzebne dane
podgląd danych dotyczących parku rozrywki	TAK	Parki_rozrywki, Adresy, Wlasciciele
dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących parku rozrywki	TAK	Parki_rozrywki, Adresy, Wlasciciele

podgląd danych dotyczących właścicieli	TAK	Wlasciciele, Adresy
dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących właścicieli	TAK	Wlasciciele, Adresy
podgląd danych dotyczących adresów	TAK	Adresy
dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących adresów	TAK	Adresy
podgląd danych dotyczących pracowników	TAK	Pracownicy, Adresy, Wynagrodzenia, Stanowiska
dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących pracowników	TAK	Pracownicy, Adresy, Wynagrodzenia, Stanowiska
podgląd danych dotyczących specjalizacji pracownika - pracowników kasy	TAK	Pracownicy_kas, Pracownicy
dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących specjalizacji pracownika - pracowników kasy	TAK	Pracownicy_kas, Pracownicy
podgląd danych dotyczących specjalizacji pracownika - pracowników atrakcji	TAK	Pracownicy_atrakcji, Pracownicy
dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących specjalizacji pracownika - pracowników atrakcji	TAK	Pracownicy_atrakcji, Pracownicy
podgląd danych dotyczących stanowisk	TAK	Stanowiska
dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących stanowisk	TAK	Stanowiska
podgląd danych dotyczących wynagrodzeń	TAK	Wynagrodzenia
dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących wynagrodzeń	TAK	Wynagrodzenia
podgląd danych dotyczących punktów gastronomicznych znajdujących się na terenie parku rozrywki	TAK	Punkty_gastronomiczne, Strefy

dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących punktów gastronomicznych znajdujących się na terenie parku rozrywki	TAK	Punkty_gastronomiczne, Strefy
podgląd danych dotyczących zatrudnień w punktach gastronomicznych	TAK	Zatrudnienia_Gastronomia, Punkty_gastronomiczne, Pracownicy
dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących zatrudnień w punktach gastronomicznych	TAK	Zatrudnienia_Gastronomia, Punkty_gastronomiczne, Pracownicy
podgląd danych dotyczących hoteli	TAK	Hotele, Adresy
dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących hoteli	TAK	Hotele, Adresy
podgląd danych dotyczących zatrudnień w hotelu	TAK	Zatrudnienia_Hotel, Hotele, Pracownicy
dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących zatrudnień w hotelu	TAK	Zatrudnienia_Hotel, Hotele, Pracownicy
podgląd danych dotyczących kas	TAK	Kasy
dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących kas	TAK	Kasy
podgląd danych dotyczących kas, które są obsługiwane przez pracowników	TAK	Pracownicy_Kasy, Pracownicy, Kasy
dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących kas, które są obsługiwane przez pracowników	TAK	Pracownicy_Kasy, Pracownicy, Kasy
podgląd danych dotyczących biletów	TAK	Bilety
dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących biletów	TAK	Bilety
podgląd danych dotyczących klientów	TAK	Klienci
dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących klientów	TAK	Klienci
podgląd danych dotyczących atrakcji	TAK	Atrakcje, Strefy
dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących atrakcji	TAK	Atrakcje, Strefy
podgląd danych dotyczących atrakcji,	TAK	Pracownicy_Atrakcje, Pracownicy,

które są obsługiwane przez pracowników		Atrakcje
dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących atrakcji, które są obsługiwane przez pracowników	TAK	Pracownicy_Atrakcje, Pracownicy, Atrakcje
podgląd danych dotyczących stref	TAK	Strefy
dodawanie, usuwanie i modyfikacja już istniejących danych dotyczących stref	TAK	Strefy

5.2 Strojenie bazy danych – dobór indeksów

W niniejszym projekcie do strojenia bazy będą służyć następujące indeksy:

- ➔ **IX_Adres_parku_rozrywki -** wyszukiwanie adresu parku rozrywki

```
CREATE INDEX IX_Adres_parku_rozrywki ON Parki_rozrywki (ID_adresu)
```

- ➔ **IX_Kasa_parku_rozrywki -** wyszukiwanie parku rozrywki, do którego należy kasa

```
CREATE INDEX IX_Kasa_parku_rozrywki ON Kasy (ID_parku)
```

- ➔ **IX_Kasa_zakupu_biletu -** wyszukiwanie kasy, w której został zakupiony dany bilet

```
CREATE INDEX IX_Kasa_zakupu_biletu ON Bilety (ID_kasy)
```

- ➔ **IX_Klient_przypisany_do_biletu -** wyszukiwanie klienta, który zakupił dany bilet

```
CREATE INDEX IX_Klient_przypisany_do_biletu ON Bilety (ID_klienta)
```

- ➔ **IX_Posiadany_park_rozrywki -** wyszukiwanie parku rozrywki, który posiada właściciel

```
CREATE INDEX IX_Posiadany_park_rozrywki ON Wlasciciele (ID_parku)
```

- ➔ **IX_Adres_wlasciciela -** wyszukiwanie adresu zamieszkania właściciela

```
CREATE INDEX IX_Adres_wlasciciela ON Wlasciciele (ID_adresu)
```

➔ **IX_Zatrudniajacy_park_rozrywki -** wyszukiwanie parku rozrywki, który zatrudnia pracownika

```
CREATE INDEX IX_Zatrudniajacy_park_rozrywki ON Pracownicy (ID_parku)
```

➔ **IX_Zajmowane_stanowisko -** wyszukiwanie stanowiska, które zajmuje pracownik

```
CREATE INDEX IX_Zajmowane_stanowisko ON Pracownicy (ID_stanowiska)
```

➔ **IX_Adres_pracownika -** wyszukiwanie adresu zamieszkania pracownika

```
CREATE INDEX IX_Adres_pracownika ON Pracownicy (ID_adresu)
```

➔ **IX_Hotel_parku_rozrywki -** wyszukiwanie parku rozrywki, do którego należy hotel

```
CREATE INDEX IX_Hotel_parku_rozrywki ON Hotele (ID_parku)
```

➔ **IX_Adres_hotelu -** wyszukiwanie adresu hotelu

```
CREATE INDEX IX_Adres_hotelu ON Hotele (ID_adresu)
```

➔ **IX_Wynagrodzenie_pracownika -** wyszukiwanie pracownika, do którego przypisane jest dane wynagrodzenie

```
CREATE INDEX IX_Wynagrodzenie_pracownika ON Wynagrodzenia (ID_pracownika)
```

➔ **IX_Atrakcja_parku_rozrywki -** wyszukiwanie parku rozrywki, do którego należy atrakcja

```
CREATE INDEX IX_Atrakcja_parku_rozrywki ON Atrakcje (ID_parku)
```

➔ **IX_Strefa_atrakcji -** wyszukiwanie strefy parku rozrywki, w której znajduje się atrakcja

```
CREATE INDEX IX_Strefa_atrakcji ON Atrakcje (ID_strefy)
```

➔ **IX_Punkt_gastronomiczny_parku_rozrywki -** wyszukiwanie parku rozrywki, do którego należy dany punkt gastronomiczny

```
CREATE INDEX IX_Punkt_gastronomiczny_parku_rozrywki ON Punkty_gastronomiczne
(ID_parku)
```

➔ **IX_Strefa_punktu_gastrononomicznego** - wyszukiwanie strefy parku rozrywki, w której znajduje się dany punkt gastronomiczny

```
CREATE INDEX IX_Strefa_punktu_gastronomicznego ON Punkty_gastronomiczne
(ID_strefy)
```

5.3 Skrypt SQL zakładający bazę danych

```
/*
Created: 01.04.2022
Modified: 12.04.2022
Project: ParkRozrywki_BDBT
Model: PR_logiczny
Company: EiTI, Politechnika Warszawska
Author: Justyna Budzyńska, Maria Kujawa
Database: Oracle 19c
*/


-- Create sequences section -------------------------------------------------

CREATE SEQUENCE Parki_rozrywkiS
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE KasyS
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE BiletyS
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE AdresyS
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE WlascicieleS
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE KlienciS
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/
```

```
CREATE SEQUENCE PracownicyS
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE HoteleS
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE StanowiskaS
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE AtrakcjeS
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE WynagrodzeniaS
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE Punkty_gastronomiczneS
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE StrefyS
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

-- Create tables section -------------------------------------------------

-- Table Parki_rozrywki

CREATE TABLE Parki_rozrywki(
  ID_parku Integer NOT NULL,
  Nazwa_parku Varchar2(30 ) NOT NULL,
  Nr_telefonu Varchar2(15 ) NOT NULL,
  Adres_email Varchar2(30 ) NOT NULL,
  Liczba_atrakcji Integer NOT NULL,
  Czy_posiada_hotel Varchar2(3 ) NOT NULL
        CHECK (Czy_posiada_hotel IN ('Tak', 'Nie')),
  ID_adresu Integer NOT NULL
)
/
```

```
-- Create indexes for table Parki_rozrywki

CREATE INDEX IX_Adres_parku_rozrywki ON Parki_rozrywki (ID_adresu)
/

-- Add keys for table Parki_rozrywki

ALTER TABLE Parki_rozrywki ADD CONSTRAINT Park_rozrywkiPK PRIMARY KEY (ID_parku)
/

-- Table and Columns comments section

COMMENT ON COLUMN Parki_rozrywki.ID_parku IS 'Unikatowy identyfikator (ID) parku rozrwki.'
/
COMMENT ON COLUMN Parki_rozrywki.Nazwa_parku IS 'Nazwa parku rozrywki.'
/
COMMENT ON COLUMN Parki_rozrywki.Nr_telefonu IS 'Numer telefonu parku rozrywki.'
/
COMMENT ON COLUMN Parki_rozrywki.Adres_email IS 'Adres email parku rozrywki.'
/
COMMENT ON COLUMN Parki_rozrywki.Liczba_atrakcji IS 'Liczba atrakcji w parku rozrywki.'
/
COMMENT ON COLUMN Parki_rozrywki.Czy_posiada_hotel IS 'Flaga mówiąca o tym, czy park rozrywki posiada swój hotel.'
/

-- Table Pracownicy

CREATE TABLE Pracownicy(
  ID_pracownika Integer NOT NULL,
  Imie Varchar2(20 ) NOT NULL,
  Drugie_imie Varchar2(20 ),
  Nazwisko Varchar2(30 ) NOT NULL,
  Data_urodzenia Date NOT NULL,
  PESEL Char(11 ),
  Plec Char(1 ) NOT NULL
        CHECK (Plec IN ('K', 'M')),
  Nr_telefonu Varchar2(15 ) NOT NULL,
  Adres_email Varchar2(30 ) NOT NULL,
  Data_zatrudnienia Date NOT NULL,
  Data_zwolnienia Date,
  Nr_konta Char(26 ) NOT NULL,
  ID_parku Integer NOT NULL,
  ID_stanowiska Integer NOT NULL,
  ID_adresu Integer NOT NULL
)
/

-- Create indexes for table Pracownicy

CREATE INDEX IX_Zatrudniajacy_park_rozrywki ON Pracownicy (ID_parku)
/

CREATE INDEX IX_Zajmowane_stanowisko ON Pracownicy (ID_stanowiska)
/

CREATE INDEX IX_Adres_pracownika ON Pracownicy (ID_adresu)
/

-- Add keys for table Pracownicy

ALTER TABLE Pracownicy ADD CONSTRAINT PracownikPK PRIMARY KEY (ID_pracownika)
/

-- Table and Columns comments section

COMMENT ON COLUMN Pracownicy.ID_pracownika IS 'Unikatowy identyfikator (ID) pracownika.'
/
COMMENT ON COLUMN Pracownicy.Imie IS 'Imię pracownika.'
/
COMMENT ON COLUMN Pracownicy.Drugie_imie IS 'Drugie imię pracownika.'
/
COMMENT ON COLUMN Pracownicy.Nazwisko IS 'Nazwisko pracownika.'
/
```

```
COMMENT ON COLUMN Pracownicy.Data_urodzenia IS 'Data urodzenia pracownika.'
/
COMMENT ON COLUMN Pracownicy.PESEL IS 'PESEL pracownika.'
/
COMMENT ON COLUMN Pracownicy.Plec IS 'Flaga mówiąca o płci pracownika.'
/
COMMENT ON COLUMN Pracownicy.Nr_telefonu IS 'Numer telefonu pracownika.'
/
COMMENT ON COLUMN Pracownicy.Adres_email IS 'Adres email pracownika'
/
COMMENT ON COLUMN Pracownicy.Data_zatrudnienia IS 'Data zatrudnienia pracownika.'
/
COMMENT ON COLUMN Pracownicy.Data_zwolnienia IS 'Data zwolnienia pracownika.'
/
COMMENT ON COLUMN Pracownicy.Nr_konta IS 'Numer konta pracownika.'
/

-- Table Kasy

CREATE TABLE Kasy(
  ID_kasy Integer NOT NULL,
  Nr_kasy Integer NOT NULL,
  ID_parku Integer NOT NULL
)
/

-- Create indexes for table Kasy

CREATE INDEX IX_Kasa_parku_rozrywki ON Kasy (ID_parku)
/

-- Add keys for table Kasy

ALTER TABLE Kasy ADD CONSTRAINT KasaPK PRIMARY KEY (ID_kasy)
/

-- Table and Columns comments section

COMMENT ON COLUMN Kasy.ID_kasy IS 'Unikatowy identyfikator (ID) kasy biletowej.'
/
COMMENT ON COLUMN Kasy.Nr_kasy IS 'Numer kasy biletowej.'
/

-- Table Bilety

CREATE TABLE Bilety(
  ID_biletu Integer NOT NULL,
  Typ_biletu Varchar2(15 ) NOT NULL
        CHECK (Typ_biletu IN ('Jednodniowy', 'Trzydniowy', 'Tygodniowy')),
  Data_zakupu Date,
  Data_wstepu Date NOT NULL,
  Cena_biletu Number(10,2) NOT NULL,
  Rodzaj_znizki Varchar2(20 )
        CHECK (Rodzaj_znizki IN ('Ulgowy', 'Studencki', 'Dla seniora')),
  ID_kasy Integer NOT NULL,
  ID_klienta Integer
)
/

-- Create indexes for table Bilety

CREATE INDEX IX_Kasa_zakupu_biletu ON Bilety (ID_kasy)
/

CREATE INDEX IX_Klient_przypisany_do_biletu ON Bilety (ID_klienta)
/

-- Add keys for table Bilety

ALTER TABLE Bilety ADD CONSTRAINT BiletPK PRIMARY KEY (ID_biletu)
/
```

```
-- Table and Columns comments section

COMMENT ON COLUMN Bilety.ID_biletu IS 'Unikatowy identyfikator (ID) biletu.'
/
COMMENT ON COLUMN Bilety.Typ_biletu IS 'Typ biletu (jednodniowy, trzydniowy, tygodniowy).'
/
COMMENT ON COLUMN Bilety.Data_zakupu IS 'Data zakupu biletu.'
/
COMMENT ON COLUMN Bilety.Data_wstepu IS 'Data wstępu do parku rozrywki.'
/
COMMENT ON COLUMN Bilety.Cena_biletu IS 'Cena biletu.'
/
COMMENT ON COLUMN Bilety.Rodzaj_znizki IS 'Rodzaj zniżki (Ulgowy, Studencki, Dla seniora).'
/

-- Table Pracownicy_kas

CREATE TABLE Pracownicy_kas(
  ID_pracownika Integer NOT NULL,
  Nr_dostepu Char(16 ) NOT NULL,
  Data_szkolenia Date NOT NULL,
  Czy_angielski Varchar2(3 ) NOT NULL
        CHECK (Czy_angielski IN ('Tak', 'Nie'))
)
/

-- Add keys for table Pracownicy_kas

ALTER TABLE Pracownicy_kas ADD CONSTRAINT Unique_Identifier1 PRIMARY KEY (ID_pracownika)
/

-- Table and Columns comments section

COMMENT ON COLUMN Pracownicy_kas.Nr_dostepu IS 'Numer karty pozwalającej na dostęp do kasy.'
/
COMMENT ON COLUMN Pracownicy_kas.Data_szkolenia IS 'Data ostatniego szkolenia (pracownik kasy).'
/
COMMENT ON COLUMN Pracownicy_kas.Czy_angielski IS 'Flaga mówiąca o tym, czy dany pracownik kasy mówi też w języku angielskim.'
/

-- Table Pracownicy_atrakcji

CREATE TABLE Pracownicy_atrakcji(
  ID_pracownika Integer NOT NULL,
  Nr_licencji Char(6 ) NOT NULL,
  Data_szkolenia Date NOT NULL
)
/

-- Add keys for table Pracownicy_atrakcji

ALTER TABLE Pracownicy_atrakcji ADD CONSTRAINT Unique_Identifier2 PRIMARY KEY (ID_pracownika)
/

-- Table and Columns comments section

COMMENT ON COLUMN Pracownicy_atrakcji.Nr_licencji IS 'Numer licencji pozwalającej na obsługę atrakcji.'
/
COMMENT ON COLUMN Pracownicy_atrakcji.Data_szkolenia IS 'Data ostatniego szkolenia (pracownik kasy).'
/

-- Table Klienci

CREATE TABLE Klienci(
  ID_klienta Integer NOT NULL,
  Imie Varchar2(20 ) NOT NULL,
  Drugie_imie Varchar2(20 ),
  Nazwisko Varchar2(30 ) NOT NULL,
  Data_urodzenia Date NOT NULL,
  Nr_telefonu Varchar2(15 ) NOT NULL,
  Adres_email Varchar2(30 )
)
/
```

```
-- Add keys for table Klienci

ALTER TABLE Klienci ADD CONSTRAINT KlientPK PRIMARY KEY (ID_klienta)
/

-- Table and Columns comments section

COMMENT ON COLUMN Klienci.ID_klienta IS 'Unikatowy identyfikator (ID) klienta.'
/
COMMENT ON COLUMN Klienci.Imie IS 'Imie klienta.'
/
COMMENT ON COLUMN Klienci.Drugie_imie IS 'Drugie imię klienta.'
/
COMMENT ON COLUMN Klienci.Nazwisko IS 'Nazwisko klienta.'
/
COMMENT ON COLUMN Klienci.Data_urodzenia IS 'Data urodzenia klienta.'
/
COMMENT ON COLUMN Klienci.Nr_telefonu IS 'Numer telefonu klienta.'
/
COMMENT ON COLUMN Klienci.Adres_email IS 'Adres email klienta.'
/

-- Table Atrakcje

CREATE TABLE Atrakcje(
  ID_atrakcji Integer NOT NULL,
  Nazwa_atrakcji Varchar2(30 ) NOT NULL,
  Liczba_miejsc Integer NOT NULL,
  Minimalny_wiek Integer,
  Minimalny_wzrost Integer,
  Poziom Varchar2(20 ) NOT NULL
        CHECK (Poziom IN ('Dla każdego', 'Familijny', 'Normalny', 'Średniozaawansowany', 'Ekstremalny')),
  ID_parku Integer NOT NULL,
  ID_strefy Integer NOT NULL
)
/

-- Create indexes for table Atrakcje

CREATE INDEX IX_Atrakcja_parku_rozrywki ON Atrakcje (ID_parku)
/

CREATE INDEX IX_Strefa_atrakcji ON Atrakcje (ID_strefy)
/

-- Add keys for table Atrakcje

ALTER TABLE Atrakcje ADD CONSTRAINT AtrakcjaPK PRIMARY KEY (ID_atrakcji)
/

-- Table and Columns comments section

COMMENT ON COLUMN Atrakcje.ID_atrakcji IS 'Unikatowy identyfikator (ID) atrakcji.'
/
COMMENT ON COLUMN Atrakcje.Nazwa_atrakcji IS 'Nazwa atrakcji.'
/
COMMENT ON COLUMN Atrakcje.Liczba_miejsc IS 'Liczba miejsc na atrakcji.'
/
COMMENT ON COLUMN Atrakcje.Minimalny_wiek IS 'Minimalny wiek uczestnika atrakcji.'
/
COMMENT ON COLUMN Atrakcje.Minimalny_wzrost IS 'Minimalny wzrost uczestnika atrakcji.'
/
COMMENT ON COLUMN Atrakcje.Poziom IS 'Poziom trudności/zaawansowania atrakcji (Dla każdego, Familijny, Normalny,
Średniozaawansowany, Ekstremalny).'
/

-- Table Hotele

CREATE TABLE Hotele(
  ID_hotelu Integer NOT NULL,
  Nazwa_hotelu Varchar2(30 ) NOT NULL,
  Nr_telefonu Varchar2(15 ) NOT NULL,
  Adres_email Varchar2(30 ) NOT NULL,
```

```
  Liczba_osob Integer NOT NULL,
  ID_parku Integer NOT NULL,
  ID_adresu Integer NOT NULL
)
/

-- Create indexes for table Hotele

CREATE INDEX IX_Hotel_parku_rozrywki ON Hotele (ID_parku)
/

CREATE INDEX IX_Adres_hotelu ON Hotele (ID_adresu)
/

-- Add keys for table Hotele

ALTER TABLE Hotele ADD CONSTRAINT HotelPK PRIMARY KEY (ID_hotelu)
/

-- Table and Columns comments section

COMMENT ON COLUMN Hotele.ID_hotelu IS 'Unikatowy identyfikator (ID) hotelu.'
/
COMMENT ON COLUMN Hotele.Nazwa_hotelu IS 'Nazwa hotelu.'
/
COMMENT ON COLUMN Hotele.Nr_telefonu IS 'Numer telefonu hotelu.'
/
COMMENT ON COLUMN Hotele.Adres_email IS 'Adres email hotelu.'
/
COMMENT ON COLUMN Hotele.Liczba_osob IS 'Liczba miejsc w hotelu.'
/

-- Table Punkty_gastronomiczne

CREATE TABLE Punkty_gastronomiczne(
  ID_punktu_gastronomicznego Integer NOT NULL,
  Nazwa_punktu_gastronomicznego Varchar2(30 ) NOT NULL,
  Nr_budynku Integer NOT NULL,
  Typ Varchar2(30 ) NOT NULL
        CHECK (Typ IN ('Foodtruck', 'Restauracja', 'Kawiarnia', 'Automat')),
  ID_parku Integer NOT NULL,
  ID_strefy Integer NOT NULL
)
/

-- Create indexes for table Punkty_gastronomiczne

CREATE INDEX IX_Punkt_gastronomiczny_parku_rozrywki ON Punkty_gastronomiczne (ID_parku)
/

CREATE INDEX IX_Strefa_punktu_gastronomicznego ON Punkty_gastronomiczne (ID_strefy)
/

-- Add keys for table Punkty_gastronomiczne

ALTER TABLE Punkty_gastronomiczne ADD CONSTRAINT Punkt_gastronomicznyPK PRIMARY KEY (ID_punktu_gastronomicznego)
/

-- Table and Columns comments section

COMMENT ON COLUMN Punkty_gastronomiczne.ID_punktu_gastronomicznego IS 'Unikatowy identyfikator (ID) punktu gastronomicznego.'
/
COMMENT ON COLUMN Punkty_gastronomiczne.Nazwa_punktu_gastronomicznego IS 'Nazwa punktu gastronomicznego.'
/
COMMENT ON COLUMN Punkty_gastronomiczne.Nr_budynku IS 'Numer budynku, w którym znajduje się punkt.'
/
COMMENT ON COLUMN Punkty_gastronomiczne.Typ IS 'Typ punktu gastronomicznego (Foodtruck, Restauracja, Kawiarnia, Automat).'
/

-- Table Pracownicy_Atrakcje

CREATE TABLE Pracownicy_Atrakcje(
  ID_pracownika Integer NOT NULL,
```

```
  ID_atrakcji Integer NOT NULL
)
/

-- Table Zatrudnienia_Gastronomia

CREATE TABLE Zatrudnienia_Gastronomia(
  ID_pracownika Integer NOT NULL,
  ID_punktu_gastronomicznego Integer NOT NULL,
  Waznosc_badan_sanepid Date NOT NULL
)
/

-- Table and Columns comments section

COMMENT ON COLUMN Zatrudnienia_Gastronomia.Waznosc_badan_sanepid IS 'Ważność badań sanepidowskich.'
/

-- Table Zatrudnienia_Hotel

CREATE TABLE Zatrudnienia_Hotel(
  ID_pracownika Integer NOT NULL,
  ID_hotelu Integer NOT NULL
)
/

-- Table Wlasciciele

CREATE TABLE Wlasciciele(
  ID_wlasciciela Integer NOT NULL,
  Imie Varchar2(20 ) NOT NULL,
  Drugie_imie Varchar2(20 ),
  Nazwisko Varchar2(30 ) NOT NULL,
  Data_urodzenia Date NOT NULL,
  PESEL Char(11 ),
  Nr_telefonu Varchar2(15 ) NOT NULL,
  Adres_email Varchar2(30 ) NOT NULL,
  ID_parku Integer NOT NULL,
  ID_adresu Integer NOT NULL
)
/

-- Create indexes for table Wlasciciele

CREATE INDEX IX_Posiadany_park_rozrywki ON Wlasciciele (ID_parku)
/

CREATE INDEX IX_Adres_wlasciciela ON Wlasciciele (ID_adresu)
/

-- Add keys for table Wlasciciele

ALTER TABLE Wlasciciele ADD CONSTRAINT PK_Wlasciciele PRIMARY KEY (ID_wlasciciela)
/

-- Table and Columns comments section

COMMENT ON COLUMN Wlasciciele.ID_wlasciciela IS 'Unikatowy identyfikator (ID) właściciela.'
/
COMMENT ON COLUMN Wlasciciele.Imie IS 'Imię właściciela.'
/
COMMENT ON COLUMN Wlasciciele.Drugie_imie IS 'Drugie imię właściciela.'
/
COMMENT ON COLUMN Wlasciciele.Nazwisko IS 'Nazwisko właściciela.'
/
COMMENT ON COLUMN Wlasciciele.Data_urodzenia IS 'Data urodzenia właściciela.'
/
COMMENT ON COLUMN Wlasciciele.PESEL IS 'PESEL właściciela.'
/
COMMENT ON COLUMN Wlasciciele.Nr_telefonu IS 'Numer telefonu właściciela.'
/
COMMENT ON COLUMN Wlasciciele.Adres_email IS 'Adres email właściciela.'
/
```

```
-- Table Adresy

CREATE TABLE Adresy(
  ID_adresu Integer NOT NULL,
  Ulica Varchar2(30 ) NOT NULL,
  Nr_budynku Integer NOT NULL,
  Nr_lokalu Integer,
  Kod_pocztowy Char(6 ) NOT NULL,
  Miejscowosc Varchar2(30 ) NOT NULL,
  Kraj Varchar2(30 ) NOT NULL
)
/

-- Add keys for table Adresy

ALTER TABLE Adresy ADD CONSTRAINT PK_Adresy PRIMARY KEY (ID_adresu)
/

-- Table and Columns comments section

COMMENT ON COLUMN Adresy.ID_adresu IS 'Unikatowy identyfikator (ID) adresu.'
/
COMMENT ON COLUMN Adresy.Ulica IS 'Ulica.'
/
COMMENT ON COLUMN Adresy.Nr_budynku IS 'Numer budynku.'
/
COMMENT ON COLUMN Adresy.Nr_lokalu IS 'Numer lokalu.'
/
COMMENT ON COLUMN Adresy.Kod_pocztowy IS 'Kod pocztowy.'
/
COMMENT ON COLUMN Adresy.Miejscowosc IS 'Miejscowość.'
/
COMMENT ON COLUMN Adresy.Kraj IS 'Kraj.'
/

-- Table Stanowiska

CREATE TABLE Stanowiska(
  ID_stanowiska Integer NOT NULL,
  Nazwa_stanowiska Varchar2(30 ) NOT NULL,
  Opis Varchar2(300 ) NOT NULL
)
/

-- Add keys for table Stanowiska

ALTER TABLE Stanowiska ADD CONSTRAINT PK_Stanowiska PRIMARY KEY (ID_stanowiska)
/

ALTER TABLE Stanowiska ADD CONSTRAINT Nazwa UNIQUE (Nazwa_stanowiska)
/

-- Table and Columns comments section

COMMENT ON COLUMN Stanowiska.ID_stanowiska IS 'Unikatowy identyfikator (ID) stanowiska.'
/
COMMENT ON COLUMN Stanowiska.Nazwa_stanowiska IS 'Nazwa stanowiska.'
/
COMMENT ON COLUMN Stanowiska.Opis IS 'Opis stanowiska.'
/

-- Table Wynagrodzenia

CREATE TABLE Wynagrodzenia(
  ID_wynagrodzenia Integer NOT NULL,
  Data_wynagrodzenia Date NOT NULL,
  Kwota_wynagrodzenia Number(10,2) NOT NULL,
  Kwota_premii Number(10,2),
  ID_pracownika Integer NOT NULL
)
/
```

```
-- Create indexes for table Wynagrodzenia

CREATE INDEX IX_Wynagrodzenie_pracownika ON Wynagrodzenia (ID_pracownika)
/

-- Add keys for table Wynagrodzenia

ALTER TABLE Wynagrodzenia ADD CONSTRAINT PK_Wynagrodzenia PRIMARY KEY (ID_wynagrodzenia)
/

-- Table and Columns comments section

COMMENT ON COLUMN Wynagrodzenia.ID_wynagrodzenia IS 'Unikatowy identyfikator (ID) wynagrodzenia.'
/
COMMENT ON COLUMN Wynagrodzenia.Data_wynagrodzenia IS 'Data wynagrodzenia.'
/
COMMENT ON COLUMN Wynagrodzenia.Kwota_wynagrodzenia IS 'Kwota wynagrodzenia.'
/
COMMENT ON COLUMN Wynagrodzenia.Kwota_premii IS 'Kwota premii.'
/

-- Table Strefy

CREATE TABLE Strefy(
  ID_strefy Integer NOT NULL,
  Nazwa_strefy Varchar2(30 ) NOT NULL,
  Opis Varchar2(400 ) NOT NULL
)
/

-- Add keys for table Strefy

ALTER TABLE Strefy ADD CONSTRAINT PK_Strefy PRIMARY KEY (ID_strefy)
/

ALTER TABLE Strefy ADD CONSTRAINT Nazwa_strefy UNIQUE (Nazwa_strefy)
/

-- Table and Columns comments section

COMMENT ON COLUMN Strefy.ID_strefy IS 'Unikatowy identyfikator (ID) strefy.'
/
COMMENT ON COLUMN Strefy.Nazwa_strefy IS 'Nazwa strefy.'
/
COMMENT ON COLUMN Strefy.Opis IS 'Opis strefy.'
/

-- Table Pracownicy_Kasy

CREATE TABLE Pracownicy_Kasy(
  ID_kasy Integer NOT NULL,
  ID_pracownika Integer NOT NULL
)
/

-- Add keys for table Pracownicy_Kasy

ALTER TABLE Pracownicy_Kasy ADD CONSTRAINT PK_Pracownicy_Kasy PRIMARY KEY (ID_kasy,ID_pracownika)
/

-- Trigger for sequence Parki_rozrywkiS for column ID_parku in table Parki_rozrywki ---------
CREATE OR REPLACE TRIGGER ts_Parki_rozrywki_Parki_rozrywkiS BEFORE INSERT
ON Parki_rozrywki FOR EACH ROW
BEGIN
  :new.ID_parku := Parki_rozrywkiS.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Parki_rozrywki_Parki_rozrywkiS AFTER UPDATE OF ID_parku
ON Parki_rozrywki FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column ID_parku in table Parki_rozrywki as it uses sequence.');
END;
/
```

```
-- Trigger for sequence PracownicyS for column ID_pracownika in table Pracownicy ---------
CREATE OR REPLACE TRIGGER ts_Pracownicy_PracownicyS BEFORE INSERT
ON Pracownicy FOR EACH ROW
BEGIN
  :new.ID_pracownika := PracownicyS.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Pracownicy_PracownicyS AFTER UPDATE OF ID_pracownika
ON Pracownicy FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column ID_pracownika in table Pracownicy as it uses sequence.');
END;
/

-- Trigger for sequence KasyS for column ID_kasy in table Kasy ---------
CREATE OR REPLACE TRIGGER ts_Kasy_KasyS BEFORE INSERT
ON Kasy FOR EACH ROW
BEGIN
  :new.ID_kasy := KasyS.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Kasy_KasyS AFTER UPDATE OF ID_kasy
ON Kasy FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column ID_kasy in table Kasy as it uses sequence.');
END;
/

-- Trigger for sequence BiletyS for column ID_biletu in table Bilety ---------
CREATE OR REPLACE TRIGGER ts_Bilety_BiletyS BEFORE INSERT
ON Bilety FOR EACH ROW
BEGIN
  :new.ID_biletu := BiletyS.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Bilety_BiletyS AFTER UPDATE OF ID_biletu
ON Bilety FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column ID_biletu in table Bilety as it uses sequence.');
END;
/

-- Trigger for sequence KlienciS for column ID_klienta in table Klienci ---------
CREATE OR REPLACE TRIGGER ts_Klienci_KlienciS BEFORE INSERT
ON Klienci FOR EACH ROW
BEGIN
  :new.ID_klienta := KlienciS.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Klienci_KlienciS AFTER UPDATE OF ID_klienta
ON Klienci FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column ID_klienta in table Klienci as it uses sequence.');
END;
/

-- Trigger for sequence AtrakcjeS for column ID_atrakcji in table Atrakcje ---------
CREATE OR REPLACE TRIGGER ts_Atrakcje_AtrakcjeS BEFORE INSERT
ON Atrakcje FOR EACH ROW
BEGIN
  :new.ID_atrakcji := AtrakcjeS.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Atrakcje_AtrakcjeS AFTER UPDATE OF ID_atrakcji
ON Atrakcje FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column ID_atrakcji in table Atrakcje as it uses sequence.');
END;
/

-- Trigger for sequence HoteleS for column ID_hotelu in table Hotele ---------
CREATE OR REPLACE TRIGGER ts_Hotele_HoteleS BEFORE INSERT
ON Hotele FOR EACH ROW
```

```
BEGIN
  :new.ID_hotelu := HoteleS.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Hotele_HoteleS AFTER UPDATE OF ID_hotelu
ON Hotele FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column ID_hotelu in table Hotele as it uses sequence.');
END;
/

-- Trigger for sequence Punkty_gastronomiczneS for column ID_punktu_gastronomicznego in table Punkty_gastronomiczne ---------
CREATE OR REPLACE TRIGGER ts_Punkty_gastronomiczne_Punkty_gastronomiczneS BEFORE INSERT
ON Punkty_gastronomiczne FOR EACH ROW
BEGIN
  :new.ID_punktu_gastronomicznego := Punkty_gastronomiczneS.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Punkty_gastronomiczne_Punkty_gastronomiczneS AFTER UPDATE OF ID_punktu_gastronomicznego
ON Punkty_gastronomiczne FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column ID_punktu_gastronomicznego in table Punkty_gastronomiczne as it uses
sequence.');
END;
/

-- Trigger for sequence WlascicieleS for column ID_wlasciciela in table Wlasciciele ---------
CREATE OR REPLACE TRIGGER ts_Wlasciciele_WlascicieleS BEFORE INSERT
ON Wlasciciele FOR EACH ROW
BEGIN
  :new.ID_wlasciciela := WlascicieleS.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Wlasciciele_WlascicieleS AFTER UPDATE OF ID_wlasciciela
ON Wlasciciele FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column ID_wlasciciela in table Wlasciciele as it uses sequence.');
END;
/

-- Trigger for sequence AdresyS for column ID_adresu in table Adresy ---------
CREATE OR REPLACE TRIGGER ts_Adresy_AdresyS BEFORE INSERT
ON Adresy FOR EACH ROW
BEGIN
  :new.ID_adresu := AdresyS.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Adresy_AdresyS AFTER UPDATE OF ID_adresu
ON Adresy FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column ID_adresu in table Adresy as it uses sequence.');
END;
/

-- Trigger for sequence StanowiskaS for column ID_stanowiska in table Stanowiska ---------
CREATE OR REPLACE TRIGGER ts_Stanowiska_StanowiskaS BEFORE INSERT
ON Stanowiska FOR EACH ROW
BEGIN
  :new.ID_stanowiska := StanowiskaS.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Stanowiska_StanowiskaS AFTER UPDATE OF ID_stanowiska
ON Stanowiska FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column ID_stanowiska in table Stanowiska as it uses sequence.');
END;
/

-- Trigger for sequence WynagrodzeniaS for column ID_wynagrodzenia in table Wynagrodzenia ---------
CREATE OR REPLACE TRIGGER ts_Wynagrodzenia_WynagrodzeniaS BEFORE INSERT
ON Wynagrodzenia FOR EACH ROW
BEGIN
  :new.ID_wynagrodzenia := WynagrodzeniaS.nextval;
```

```
END;
/
CREATE OR REPLACE TRIGGER tsu_Wynagrodzenia_WynagrodzeniaS AFTER UPDATE OF ID_wynagrodzenia
ON Wynagrodzenia FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column ID_wynagrodzenia in table Wynagrodzenia as it uses sequence.');
END;
/

-- Trigger for sequence StrefyS for column ID_strefy in table Strefy ---------
CREATE OR REPLACE TRIGGER ts_Strefy_StrefyS BEFORE INSERT
ON Strefy FOR EACH ROW
BEGIN
  :new.ID_strefy := StrefyS.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Strefy_StrefyS AFTER UPDATE OF ID_strefy
ON Strefy FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column ID_strefy in table Strefy as it uses sequence.');
END;
/

-- Create foreign keys (relationships) section ------------------------------------------------

ALTER TABLE Pracownicy ADD CONSTRAINT Park_rozrywki_zatrudnia_pracownika FOREIGN KEY (ID_parku) REFERENCES Parki_rozrywki
(ID_parku)
/
ALTER TABLE Kasy ADD CONSTRAINT Park_rozrywki_posiada_kase FOREIGN KEY (ID_parku) REFERENCES Parki_rozrywki (ID_parku)
/
ALTER TABLE Bilety ADD CONSTRAINT Kasa_sprzedaje_bilety FOREIGN KEY (ID_kasy) REFERENCES Kasy (ID_kasy)
/
ALTER TABLE Bilety ADD CONSTRAINT Bilet_jest_kupowany FOREIGN KEY (ID_klienta) REFERENCES Klienci (ID_klienta)
/
ALTER TABLE Atrakcje ADD CONSTRAINT Park_rozrywki_posiada_atrakcje FOREIGN KEY (ID_parku) REFERENCES Parki_rozrywki (ID_parku)
/
ALTER TABLE Hotele ADD CONSTRAINT Park_rozrywki_posiada_hotel FOREIGN KEY (ID_parku) REFERENCES Parki_rozrywki (ID_parku)
/
ALTER TABLE Punkty_gastronomiczne ADD CONSTRAINT Park_rozrywki_posiada_punkt_gastronomiczny FOREIGN KEY (ID_parku) REFERENCES
Parki_rozrywki (ID_parku)
/
ALTER TABLE Wlasciciele ADD CONSTRAINT Park_rozrywki_ma_wlasciciela FOREIGN KEY (ID_parku) REFERENCES Parki_rozrywki (ID_parku)
/
ALTER TABLE Pracownicy ADD CONSTRAINT Pracownik_zajmuje_stanowisko FOREIGN KEY (ID_stanowiska) REFERENCES Stanowiska
(ID_stanowiska)
/
ALTER TABLE Punkty_gastronomiczne ADD CONSTRAINT Punkt_gastronomiczny_jest_w_strefie FOREIGN KEY (ID_strefy) REFERENCES Strefy
(ID_strefy)
/
ALTER TABLE Atrakcje ADD CONSTRAINT Atrakcja_jest_w_strefie FOREIGN KEY (ID_strefy) REFERENCES Strefy (ID_strefy)
/
ALTER TABLE Wynagrodzenia ADD CONSTRAINT Pracownik_dostaje_wynagrodzenie FOREIGN KEY (ID_pracownika) REFERENCES Pracownicy
(ID_pracownika)
/
ALTER TABLE Pracownicy_Kasy ADD CONSTRAINT Kasa_jest_obslugiwana_przez_pracownika FOREIGN KEY (ID_kasy) REFERENCES Kasy
(ID_kasy)
/
ALTER TABLE Pracownicy_Kasy ADD CONSTRAINT Pracownik_obsluguje_kase FOREIGN KEY (ID_pracownika) REFERENCES Pracownicy
(ID_pracownika)
/
ALTER TABLE Hotele ADD CONSTRAINT Hotel_posiada_adres FOREIGN KEY (ID_adresu) REFERENCES Adresy (ID_adresu)
/
ALTER TABLE Parki_rozrywki ADD CONSTRAINT Park_rozrywki_posiada_adres FOREIGN KEY (ID_adresu) REFERENCES Adresy (ID_adresu)
/
ALTER TABLE Pracownicy ADD CONSTRAINT Pracownik_posiada_adres FOREIGN KEY (ID_adresu) REFERENCES Adresy (ID_adresu)
/
ALTER TABLE Wlasciciele ADD CONSTRAINT Wlasciciel_posiada_adres FOREIGN KEY (ID_adresu) REFERENCES Adresy (ID_adresu)
/
```

5.4 Przykłady zapytań i poleceń SQL odnoszących się do bazy danych

➔ informacja o id pracownika i numerze PESEL wszystkich kobiet zatrudnionych w parku rozrywki

```
SELECT ID_PRACOWNIKA, PESEL FROM PRACOWNICY WHERE PLEC = 'K'
ORDER BY ID_PRACOWNIKA;
```

	ID_PRACOWNIKA	PESEL
1	2	79010550304
2	21	80012113547
3	22	88061764202
4	23	00320113363
5	24	96090431406
6	25	96051415768
7	41	96090431408
8	42	96051415788
9	61	87121250304

➔ informacja o ilości pracowników na danym stanowisku w parku rozrywki

```
SELECT COUNT(ID_PRACOWNIKA) ILOSC_PRACOWNIKOW, NAZWA_STANOWISKA
FROM PRACOWNICY JOIN STANOWISKA USING(ID_STANOWISKA)
GROUP BY NAZWA_STANOWISKA
ORDER BY NAZWA_STANOWISKA;
```

	ILOSC_PRACOWNIKOW	NAZWA_STANOWISKA
1	2	Animator
2	2	Kelner
3	1	Księgowy
4	1	Kucharz
5	3	Pracownik atrakcji
6	3	Pracownik hotelu
7	2	Pracownik kasy
8	1	Projektant atrakcji
9	1	Serwisant
10	1	Sprzedawca

➔ informacja o ilości biletów o danym typie, które zostały zakupione

```
SELECT COUNT(TYP_BILETU), TYP_BILETU FROM BILETY
WHERE DATA_ZAKUPU IS NOT NULL
GROUP BY TYP_BILETU;
```

	COUNT(TYP_BILETU)	TYP_BILETU
1	4	Trzydniowy
2	4	Jednodniowy
3	3	Tygodniowy

➔ informacja o id pracownika, imieniu i nazwisku wszystkich pracowników, których adres email kończy się na ‘@gmail.com’

```
SELECT ID_PRACOWNIKA, IMIE, NAZWISKO FROM PRACOWNICY
WHERE ADRES_EMAIL LIKE '%@gmail.com';
```

	ID_PRACOWNIKA	IMIE	NAZWISKO
1	2	Roksana	Kamińska
2	23	Marlena	Mazurek
3	24	Olga	Głowacka
4	28	Albert	Bak
5	29	Robert	Malinowski

➔ informacja o numerze PESEL, imieniu, nazwisku, kwocie wynagrodzenia i wysokości całkowitego wynagrodzenia pracowników (kwota wynagrodzenia + kwota premii)

```
SELECT PESEL, IMIE, NAZWISKO, W.KWOTA_WYNAGRODZENIA, SUM(W.KWOTA_WYNAGRODZENIA +
NVL(W.KWOTA_PREMII,0)) AS WYSOKOSC_CALEGO_WYNAGRODZENIA
FROM PRACOWNICY P JOIN WYNAGRODZENIA W ON (P.ID_PRACOWNIKA = W.ID_PRACOWNIKA)
GROUP BY PESEL, W.KWOTA_WYNAGRODZENIA, IMIE, NAZWISKO
ORDER BY WYSOKOSC_CALEGO_WYNAGRODZENIA DESC;
```

	PESEL	IMIE	NAZWISKO	KWOTA_WYNAGRODZENIA	WYSOKOSC_CALEGO_WYNAGRODZENIA
1	96110437217	Julian	Pawlak	5000	5500
2	96090483670	Albert	Bak	4000	4100
3	88061744039	Kordian	Kowalski	4000	4000
4	96090431408	Maria	Głowacka	3000	3100
5	00320108310	Robert	Malinowski	3000	3100
6	96090431406	Olga	Głowacka	3000	3100
7	96051415768	Justyna	Sadowska	3000	3100
8	96092400237	Ksawery	Sawicki	2500	2800
9	80012113547	Anna	Kowalska	2500	2800
10	96110437216	Aleksander	Pawlak	2500	2800
11	00320113363	Marlena	Mazurek	2500	2700
12	88061764202	Jadwiga	Drabina	2500	2700
13	80012166873	Jakub	Kamiński	2500	2500
14	96051415788	Karolina	Sadowska	2500	2500
15	96092400239	Karol	Sawicki	2000	2000
16	87121250304	Elżbieta	Bak	2000	2000
17	79010550304	Roksana	Kamińska	2000	2000

Bibliografia

1. *Slajdy z wykładów z przedmiotu Bazy Danych i Big Data (BDBT) - dr hab. inż. Marcin Kowalczyk*