Bartosz Dorobek 303930, Michał Cybulski 303916

prowadzący: dr hab. inż. Marcin Kowalczyk

BDBT

Projekt - Park Rozrywki Etap I

Politechnika Warszawska
Wydział Elektroniki i Technik Informacyjnych

24 kwietnia 2021

Spis treści

1. Zakres i cel projektu

Celem projektu było zaprojektowanie relacyjnej bazy danych na poziomie konceptualnym i logicznym oraz jej fizyczna implementacja. Stworzona przez nas baza danych jest oparta na rozwiązaniach firmy Oracle, a wykorzystanym językiem jest SQL.

Oprogramowanie użyte podczas realizacji projektu:

— Oracle Database 19c
— Toad Data Modeler
— Oracle SQL Developer

Temat naszego projektu była realizacja bazy danych dla parku rozrywki. Park rozrywki posiada punkty sprzedaży oraz segmenty, w których znajdują się przygotowane dla klientów atrakcje. Park rozrywki zatrudnia pracowników, którzy mogą pracować zarówno w punktach sprzedaży i obsługiwać przyjmowanych klientów, jak i przy atrkacjach w różnych sektorach. Dla klientów przygotowane są różne bilety.

2. Definicja systemu

Nasza baza danych była projektowana z zamiarem wykorzystania jej przez użytkowników wyspecjalizowanych, czyli pracowników parku rozrywki oraz użytkowników zewnętrznych, czyli klientów.

2.1. Perspektywy użytkowników

1. **Administrator** - Posiada dostęp i możliwość modyfikacji wszystkich danych. Ma uprawnienia administratora bazdy danych oracle.

2. **Kierownik parku** - Ma dostęp do wszystkich danych

3. **Księgowa** - Ma dostęp do danych wszystkich pracowników, szczególnie do danych związanych z zarobkami

4. **Pracownik** - Widzi swoje dane osobowe i kontaktowe, historie wynagrodzeń oraz obowiązki wynikające z zajmowango stanowiska.

5. **Pracownik - Sprzedawca** - Wykonuje zadaia z zakresu obsługi klienta, dlatego, ma dostęp do ich dancyh osobowych i kontaktowych. Może dodawać i anulować zamówienia, które się jeszcze nie aktywowały. Ma także ma wgląd w historię zamówień wszystkich klientów - zakupionych biletów. Ma podgląd informacji o biletach i rabatach. Widzi sekotory i wszyskite dane o dostępnych w parku atrakcjach z wyłączeniem informacji o przeglądach technicznych.

6. **Pracownik - Konserwator** - Ma dostęp do wszystkich danych o atrakcjach

7. **Klient** - Ma dostęp do swoich danych osobowych i kontaktowych. Widzi historie swoich zamówień. Może wyświetlać potrzebne informacje o sektorach i znajdujących się w nich atrakcjach.

3. Model konceptualny

3.1. Definicja zbiorów encji określonych w projekcie

Park rozrywki: główna encja bazy

Sektor: w nich znajdują się atrakcje, mogą być tematyczne

Atrakcja: atrakcje dostepne w parku rozrywki np. kolejka górska

Klient: osoba która zakupiła bilet jest kliet parku

Pracownik: pracownik parku rozrywki

Bilet: uprawnia do wejścia na dane sektory

Punkt sprzedazy: w nim sprzedawane są bilety klientom

Entity Name	Primary Unique Identifier	# Attributes
Atrakcja	id_atrakcji	14
Bilet	id_biletu	6
Klient	id_klienta	5
Park_Rozrywki	id_parku_rozrywki	5
Pracownik	id_pracownika	14
Punkt_sprzedazy	id_punktu_sprzedazy	4
Sektor	id_sektora	5

3.2. Ustalenie związków między encjami i ich typów

Relationship Name	Relationship Type	Relationship Between	Cardinality
Kupuje	Non-Identifying	Klient - Bilet	0..n - 1..m
Obsluguje_atrakcje	Non-Identifying	Pracownik - Atrakcja	0..n - 0..m
Posiada_atrakcje	Non-Identifying	Sektor - Atrakcja	1..1 - 0..m
Posiada_punkt_sprzedazy	Non-Identifying	Park_Rozrywki - Punkt_sprzedazy	1..1 - 0..m
Posiada_sektory	Non-Identifying	Park_Rozrywki - Sektor	1..1 - 0..m
Pozwala_na_wejscie	Non-Identifying	Bilet - Sektor	0..n - 1..m
Pracuje_w_punkcie_sprzedazy	Non-Identifying	Punkt_sprzedazy - Pracownik	0..n - 0..m
Pracuje_w_sektorze	Non-Identifying	Sektor - Pracownik	0..n - 0..m
Przyjmuje	Non-Identifying	Park_Rozrywki - Klient	1..1 - 0..m
Sprzedaje	Non-Identifying	Punkt_sprzedazy - Bilet	1..1 - 0..m
Zatrudnia	Non-Identifying	Park_Rozrywki - Pracownik	1..1 - 0..m

— **Park rozrywki - Sektor**
Park rozrywki może posiadać wiele sektorów, a każdy sektor jest w jednym parku rozrywki. Park na początku powstanie nie musi mieć żadnego sektora, natomiast sektor nie może istnieć poza parkiem rozrywki.

— **Park rozrywki - Pracownicy**
Park rozrywki zatrudnia wielu pracowników, a każdy z pracowników jest zatrudniony tylko w jednym parku. Na początku działalności park może nie zatrudniać żadnego pracownika, natomiast każdy pracownik podlega parkowi rozrywki.

— **Park rozrywki - Punkty sprzedazy**
Park rozrywki może posiadać wiele punktów sprzedaży, a każdy punkt sprzedaży jest w jednym parku rozrywki. Park na początku powstanie nie musi mieć żadnego punktu, natomiast punkt musi podlegać konkretnemu parkowi.

— **Park rozrywki - Klient**
Park rozrywki przyjmuje wielu klientów, ale może też nie przyjmować żadnego. Klient korzysta z danego parku rozrywki.

— **Punkty sprzedazy - Bilet**
Punkt sprzedaje wiele biletów, ale może też nie mieć żadnych sprzedanych biletów. Bilet jednak zawsze jest sprzedany z danego punktu.

— **Klient - Bilet**
Klient może kupić wiele biletów, ale by być klientem musi kupić conajmniej jeden. Bilet może być kupiony przez klienta lub wielu klientów, ale nie musi.

— **Bilet - Sektor**
Bilet pozwala najczęściej na wstęp do wielu sektorów, ale musi udostępniać conajmniej jednen sektor. Sektor jednak może istnieć nawt jeśli żaden bilet nie uprawnia na wejście do niego - np. na początku powstania lub kiedy jest wyłączony z użytku.

— **Sektor - Atrakcja**
Każda atrakcje znajduje się w danym sektorze. Setor zwykle posiada wiele atrakcji, ale np. na początku powstania może nie mieć żadnej.

— **Sektor - Pracownik**
Sektor może zatrudniać wielu pracowników. Pracownik może pracować na wilu sektorach. Na sektorze mogą nie pracować żadni pracownicy, gdy jest zamknięty.

— **Pracownik - Atrakcja**
Pracownik może pracować przy wielu atrakcjach. Przy atrakcji może pracować wielu pracowników. Gdy atrakcja jest nieczynna nie pracuje na niej żaden pracownik.

— **Punkt sprzedazy - Pracownik**
Pracownik może pracować w wielu punktach. Punkt sprzedaży może mieć wilu pracowników. W punkcie może też nie być żadnego pracownika.

3.3. Określenie atrybutów i ich dziedzin

Zastosowaną w projekcie dziedziną jest `StatusD` i jest zastosowana w atrybutach `Status` encji `Punkt sprzedazy, Sektor i Atrakcja`. Przyjmuje ona dwie wartości: 'Otwarte' lub 'Zamkniete'.

```
Status IN ('Otwarte','Zamkniete')
```

Park rozrywki (encja główna)			
Nazwa atrybutu	Typ i dziedzina	Wymagania	Opis
id_parku_rozrywki	Integer	Obowiązkowy	Unikalny identyfikator parku rozrywki
nazwa	VarChar(30)	Obowiązkowy	Nazwa parku rozrywki
adres	VarChar(100)	Obowiązkowy	Adres parku rozrywki, pole segmentowe {kod pocztowy, miasto, ulica, numer lokalu}
wlasciciel	VarChar(30)	Obowiązkowy	Właściciel parku rozrywki, pole segmentowe {imie, naziwsko}
data_zalozenia	Date	Obowiązkowy	Data założenia parku rozrywki

Sektor			
Nazwa atrybutu	Typ i dziedzina	Wymagania	Opis
id_sektora	Integer	Obowiązkowy	Unikalny identyfikator sektora
nazwa	VarChar(30)	Obowiązkowy	Nazwa sektora
data_zalozenia	Date	Obowiązkowy	Data założenia sektora
status	StatusD	Obowiązkowy	Status sektora (Otwarte/Zamkniete)
opis	VarChar(400)	Opcjonalny	Opis sektora

Punkt sprzedaży			
Nazwa atrybutu	Typ i dziedzina	Wymagania	Opis
id_punktu_sprzedazy	Integer	Obowiązkowy	Unikatowy identyfikator puntu sprzedaży
nazwa	VarChar(20)	Obowiązkowy	Nazwa kasy
liczba_kas	Integer	Obowiązkowy	Liczba stanowisk z kasą w punkcie sprzedaży
status	StatusD	Obowiązkowy	Status punktu (Otwarte/Zamkniete)

Atrakcje			
Nazwa atrybutu	Typ i dziedzina	Wymagania	Opis
id_atrakcji	Integer	Obowiązkowy	Unikalny identyfikator atrakcji
nazwa	VarChar(30)	Obowiązkowy	Nazwa atrkacji w parku rozrywki
producent	VarChar(30)	Obowiązkowy	Nazwa producenta atrakcji
wymagania_wiekowe	Integer	Obowiązkowy	Minimalny wiek uprawniający do skorzystania z atrakcji
status	StatusD	Obowiązkowy	Status atrakcji (Otwarte/Zamkniete)
data_powstania	Date	Obowiązkowy	Data powstania atrakcji
przeglad_techniczny	Date	Opcjonalny	Przegląd techniczny atrakcji
wysokosc	Integer	Opcjonalny	Wyskość atrakcji w metrach
predkosc	Integer	Opcjonalny	maksymalna prędkość atrakcji w km/h
przeciazenie	Integer	Opcjonalny	Wartość maksymalnego przeciążenia w standardowych jednostkach przyspieszenia ziemskiego G
przepustowosc	Integer	Opcjonalny	Liczba miejsc na pojedynczy przejazd
dlugosc_trasy	Integer	Opcjonalny	Długość trasy atrakcji
typ_atrakcji	VarChar(30)	Opcjonalny	Typ atrakcji
opis	VarChar(400)	Obowiązkowy	opis atrakcji

Pracownik			
Nazwa atrybutu	Typ i dziedzina	Wymagania	Opis
id_pracownika	Integer	Obowiązkowy	Unikalny identyfikator pracownika
imie	VarChar(30)	Obowiązkowy	Imię pracownika
nazwisko	VarChar(30)	Obowiązkowy	Nazwisko pracownika
data_urodzienia	Date	Obowiązkowy	Data urodzenia pracownika
PESEL	Character(11)	Opcjonalny	PESEL
nr_telefonu	VarChar(13)	Obowiązkowy	Numer telefonu pracownik
email	VarChar(30)	Obowiązkowy	Adres e-mail pracownika
stanowisko	VarChar(30)	Obowiązkowy	Stanowisko pracownika
nr_konta_bankowego	VarChar(26)	Obowiązkowy	Numer konta bankowego, na które pracownik otrzymuje wynagrodzenie
wynagrodzenie	Money	Opcjonalny	Miesięczne wynagrodzenie w PLN jakie otrzymuje pracownik
jezyk	Character(2)	Opcjonalny	Poziom znajomości języków obcych, pole wielowartościowe, pole segmentowe {język, poziom}
adres	VarChar(300)	Obowiązkowy	Adres zamieszkania pracownika, pole segmentowe
data_zatrudnienia	Date	Obowiązkowy	Data zatrudnienia pracownika
data_zwolnienia	Date	Opcjonalny	Data zwolnienia pracownika

Klient			
Nazwa atrybutu	Typ i dziedzina	Wymagania	Opis
id_klienta	Integer	Obowiązkowy	Unikatowy identyfikator klienta
imie	VarChar(30)	Obowiązkowy	imię klienta
nazwisko	VarChar(30)	Obowiązkowy	nazwisko klienta
email	VarChar(30)	Obowiązkowy	adres e-mail klienta
nr_telefonu	VarChar(30)	Obowiązkowy	numer telefonu klienta

Bilet			
Nazwa atrybutu	Typ i dziedzina	Wymagania	Opis
id_biletu	Integer	Obowiązkowy	Unikatowy identyfikator biletu
nazwa	VarChar(30)	Obowiązkowy	Nazwa biletu
ulga	VarChar(30)	Obowiązkowy	pole segmentowe {Czy przysługuje ulga, ile % zniżki}
cena	Money	Obowiązkowy	cena biletu
okres_waznosci	VarChar(30)	Opcjonalny	Okres ważności biletu, pole segmentowe {czas aktywacji, czas dezaktywacji}
data_zakupu	Timestamp	Opcjonalny	data i godzina zakupu bilet

3.4. Dodatkowe reguły integralnościowe (reguły biznesowe)

W zależności od rodzaju zakupionego biletu klinet ma dostęp do atrakcje w danych sektorach. Od rodzaju biletu może zależeć także okres ważności, czyli długość czasu jaki klient może spędzić w paru rozrywki. Na odwiedzających park rozrywki czekają zarówno ekstremalne atrakcje dla spragnionych adrenaliny, jak i dla wycieczek szkolnych i rodzin z dziećmi, dlatego w encji `Atrakcja` znajduje się grupa atrybutów z możliwością wartości pustej.

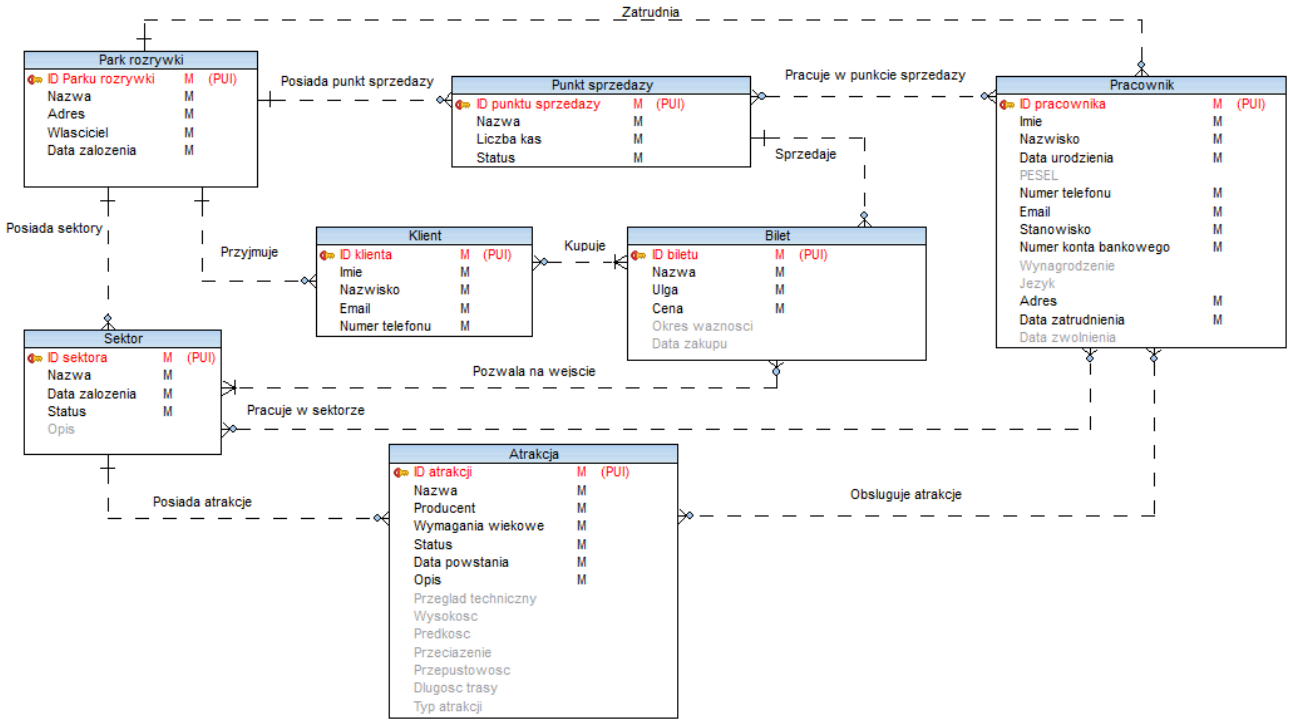
3.5. Klucze kandydujące i główne (decyzje projektowe)

Zdecydowaliśmy się od samego początku dla wszystkich encji ustawiać na klucze główne sztuczne identyfikatory numeryczne - id.

UI Type	Entity Name	Atribute Name	Unique Identifier Name
PUI	Atrakcja	id_atrakcji	Atrakcja_PK
PUI	Bilet	id_biletu	Bilet_PK
PUI	Klient	id_klienta	Klient_PK
PUI	Park_Rozrywki	id_parku_rozrywki	Park_rozrywki_PK
PUI	Pracownik	id_pracownika	Pracownik_PK
PUI	Punkt_sprzedazy	id_punktu_sprzedazy	Punkt_sprzedazy_PK
PUI	Sektor	id_sektora	Sektor_PK

Klucze kandydujące - minimalny zestaw atrybutów relacji, jednoznacznie identyfikujący każdą krotkę tej relacji. Dla Pracownika mógłby to być `Numer konta bankowego`

3.6. Schemat ER na poziomie konceptualnym

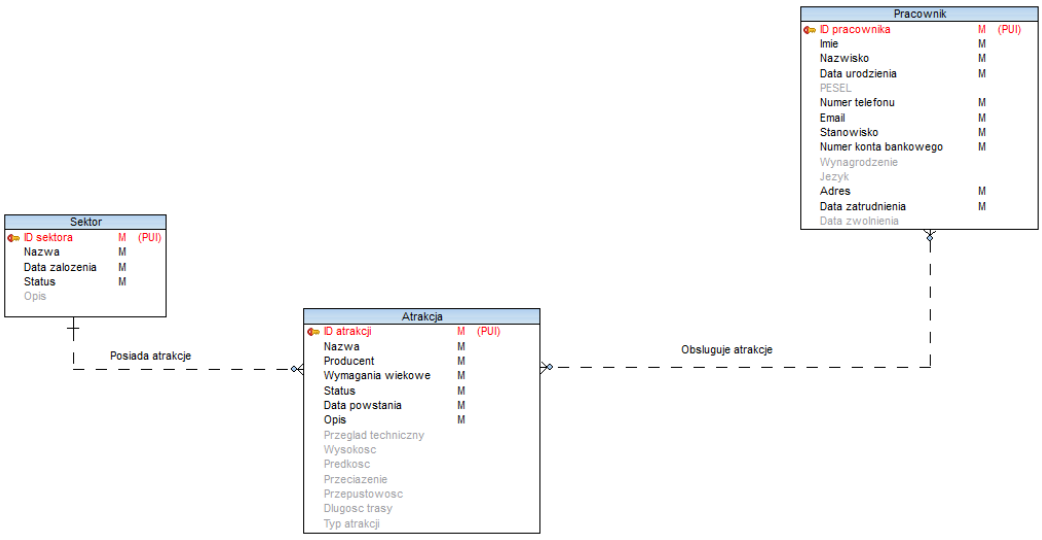


3.7. Problem pułapek szczelinowych i wachlarzowych

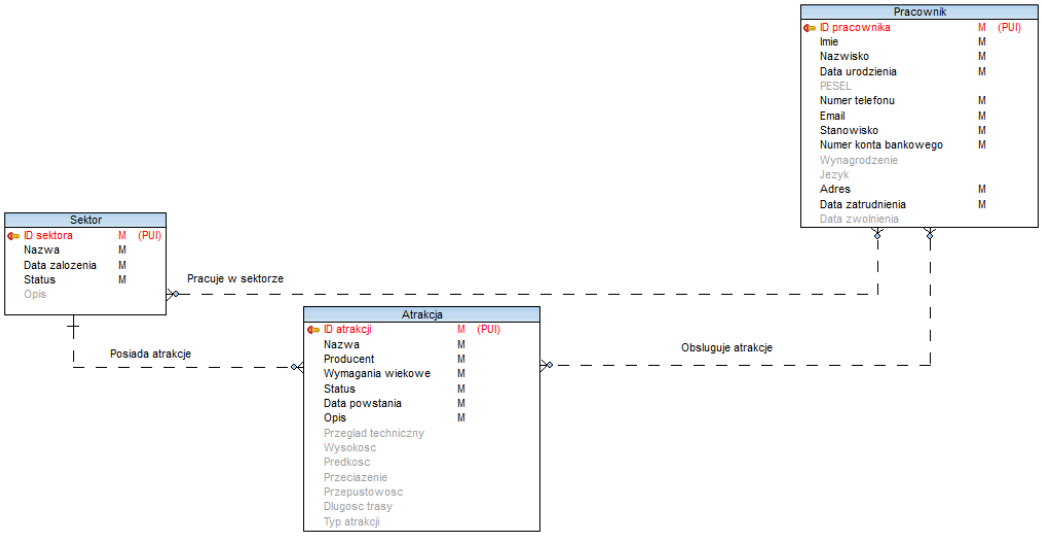
Pułapka szczelinowa – występuje, gdy model sugeruje istnienie związku pomiędzy zbiorami encji, ale nie istnieją ścieżki łączące pewne wystąpienia tych encji.

W naszym przypadku może nas interesować, kto pracuje w danym sektorze, a brak było bezpośredniego związku między encjami Pracownik i Sektor. Ponadto związki łączące encje Pracownika z Atrakcją mają minimalną krotność zero. Problem pułapki szczelinowej wystąpi, dla pracownika, który będzie pracował w sektorze, ale nie przy atrakcji.

Rozwiązaniem było utworzenie nowego związku - `Pracuje w sektorze`, bezpośrednio pomiędzy encjami `Sektor` i `Pracownik`.



Rys. 1. Przykład pułpaki szczelinowej

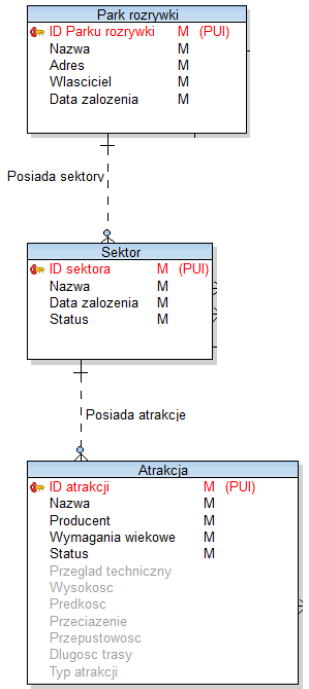


Rys. 2. Rozwiązanie problemu pułpaki szczelinowej

Pułapka wachlarzowa – występuje w sytuacji, gdy model przedstawia związek pomiędzy pewnymi zbiorami encji, ale wynikające z tego ścieżki pomiędzy wystąpieniami encji nie są jednoznaczne

Nie zaobserwowaliśmy w naszym modelu konceptualnym tego typu pułapki. Wystąpiłaby ona, gdybyśmy naszą encje główną `Park rozrywki` połączyli oddzielnie związkami typu jeden do wiel z encjami `Sektor` i `Atrakcja`. Uniknęliśmy problemu łącząc je hierarchcznie: Park z Sektorem, a Sektor z Atrakcją.

Zakładamy również, że w tym przypadku zastosowane rozwiązanie nie generuje problemu pułapki szczelinowej, ponieważ każda atrakcja musi przynależeć do sektora.



Rys. 3. Rozwiązanie problemu pułpaki wachlarzowej

4. Model Logiczny

4.1. Charaktetystyka modelu relacyjnego

Każdemu zbiorowi encji odpowiada tabela - relacja. Identyfikujący atrybut encji staje się kluczem głównym tabeli. Dla każdego związku „jeden do wiele" wstawiony został klucz główny tabeli ze strony jednej linii związku. Do tabeli reprezentującej stronę wiele linii związku wstawiamy klucz obcy.

4.2. Usunięcie właściwości niekompantybilnych z modelem relacyjnym

W pierwszej kolejności zmieniliśmy nazwy relacji z liczby pojedynczej na mnogą. Związki wiele do wielu zostały zastąpione dwoma związkami jeden do wielu z tablicą bridge'ującą. Przy automatycznej metodzie generowania modelu logicznego z wcześniej zbudowanego modelu konceptualnego relacje łączące tworzą się same, jednak istotnym było zweryfikowanie ustawienia mandatory powstałych związków. W modelu relacyjnym znikają również połączenia typu `Inheritance`, ale takich nie ma w naszym projekcie.

4.3. Proces normalizacji analiza i przykłady

Normalizacja bazy danych jest to proces mający na celu eliminację powtarzających się danych w relacyjnej bazie danych. Główna idea polega na trzymaniu danych w jednym miejscu, a w razie potrzeby linkowania do danych. Taki sposób tworzenia bazy danych zwiększa bezpieczeństwo danych i zmniejsza ryzyko powstania niespójności (w szczególności problemów anomalii takich jak anomalia wstawiania, usuwania, modyfikacji). Normalizacja przeprowadza bazę danych z jednego stanu spójnego (przed normalizacją) w inny stan spójny (po normalizacji).

Pierwsza postać normalna (1PN):

a) brak powtarzających się grup,
b) atrybuty atomowe - każda wartość atrybutu w każdej krotce relacji jest wartością elementraną.

Druga postać normalna (2PN):

a) dana relacja jest w drugiej postaci normalnej jeśli jest w 1PN,
b) każdy atrybut tej relacji nie wchodzący w skład żadnego klucza potencjalnego jest w pełni funkcyjnie zależny od wszystkich kluczy potencjalnych tej relacji,
c) każdy atrybut nie wchodzący w skład klucza zależy od klucza a nie od jego części,
d) wszystkie jej klucze potencjalne są kluczami prostymi

Trzecia postać normalna (3PN):

a) dana relacja jest w trzeciej postaci normalnej jeśli jest w 2PN,
b) każdy atrybut nie wchodzący w skład żadnego klucza potencjalnego nie jest przechodnio funkcyjnie zależny od żadnego klucza potencjalnego tej relacji.

Przykłady

By spełnić 1PN zamieniliśmy pola segmentowe i pola wielowartościowe na nowe relacje, relacje słownikowe, które łączą się związkami typu Non-Identifying z relacjami jako atrybuty, w których do tej pory funkcjonowały. W ten sposób z pól segmentowych głownej tablicy Park rozrywki powstały nowe relacje, takie jak Wlasciciele i Adresy. Dzięki temu zabiegowi wszystkie atrybuty wspomnianych relacji są atomowe. Ponadto zdobywamy większą uniwersalność modelu, ponieważ z relacji Adresy korzysta nie tylko Park rozrywki, ale też Właściciele i Pracownicy.

Caption	Adresy

Primary Key 'PK_<%OwnerName%>'

Attribute	Type	Parent Entity
id_adresu	PK	-----

Attributes

Key	Full Name	Domain	Data Type	N[1)]	U[2)]	C[3)]	D[4)]	Comments
PK	id_adresu		Integer	YES	NO	NO	NO	Unikatowy identyfikator adresu
	miasto		Varchar2(30)	YES	NO	NO	NO	Miasto
	Ulica		Varchar2(30)	YES	NO	NO	NO	Ulica
	nr_lokalu		Varchar2(30)	YES	NO	NO	NO	Numer lokalu
	kod_pocztyy		Char(6)	YES	NO	NO	NO	Kod pocztowy

Relationships

Full Name	Type	Parent Entity	Child Entity	Card.
Park_ma_adres	Non-identifying	Adresy	Park_Rozrywki	1:1
Wlasciciel_ma_adres	Non-identifying	Adresy	Wlasciciele	1:1
Pracownik_ma_adres	Non-identifying	Adresy	Pracownicy	1:1

Caption	Wlasciciele

Primary Key 'PK_<%OwnerName%>'

Attribute	Type	Parent Entity
id_wlasciciela	PK	-----

Attributes

Key	Full Name	Domain	Data Type	N[1)]	U[2)]	C[3)]	D[4)]	Comments
PK	id_wlascicielа		Integer	YES	NO	NO	NO	Unikatowy identyfikatow właściciela
	imie		Varchar2(30)	YES	NO	NO	NO	
	nazwisko		Varchar2(30)	YES	NO	NO	NO	
FK	id_adresu		Integer	YES	NO	NO	NO	
FK	id_parku_rozrywki		Integer	YES	NO	NO	NO	

Relationships

Full Name	Type	Parent Entity	Child Entity	Card.
Wlasciciel_ma_adres	Non-identifying	Adresy	Wlasciciele	1:1
Park_ma_wlasciciela	Non-identifying	Park_Rozrywki	Wlasciciele	1:N

Indexes

Full Name	Attributes	Unique
IX_Relationship4	id_adresu	NO
IX_Relationship5	id_parku_rozrywki	NO

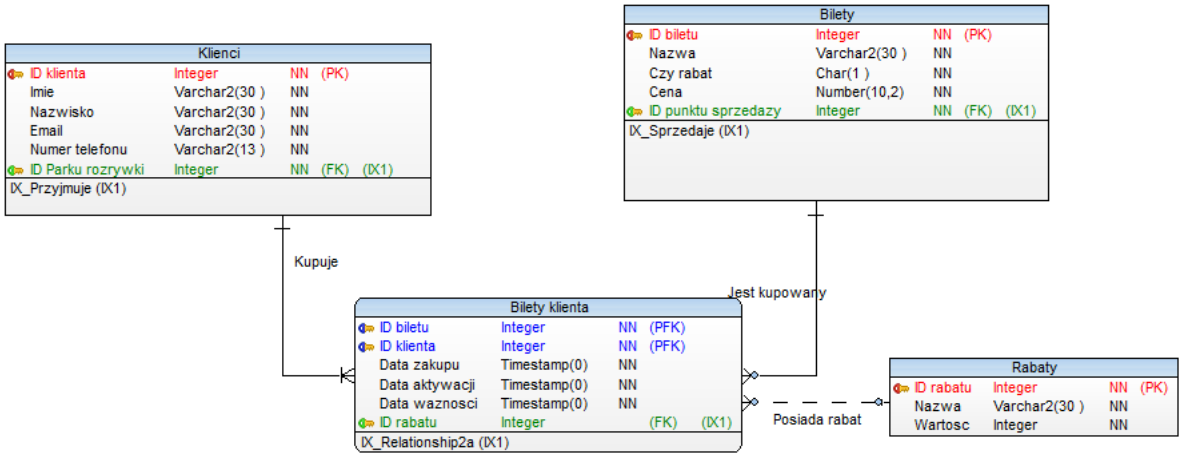
Na podobnej zasadzie tworzymy relacje:

— Stanowiska z pola wielowartościowego i segmentowego tablicy Pracownicy
— Wynagrodzenia z pola wielowartościowego i segmentowego tablicy Pracownicy
— Jezyki z pola wielowartościowego i segmentowego tablicy Pracownicy
— Poziomy jezykowa z pola wielowartościowego i segmentowego tablicy Pracownicy
— Rabaty z pola wielowartościowego i segmentowego tablicy Bilety

Zdecydowaliśmy się wprowadzić sztuczne, numeryczne, klucze proste (id) dla wszystkich relacji. Dzięki temu w prosty sposób osiągamy 2PN. Gwarantuje nam to, że wszystkie atrybuty spoza klucza zależą w sposób jednoznaczy od całości tego klucza prostego. Relacje słaba wyraża się przez nałożenie na klucz obcy dodatkowego warunku integralności - brak dopuszczania wartości pustej NN. Ponadto wydajność złączeń na wartościach kluczy numerycznych krótkich jest dużo wyższa niż na kluczach wielopolowych lub znakowych.

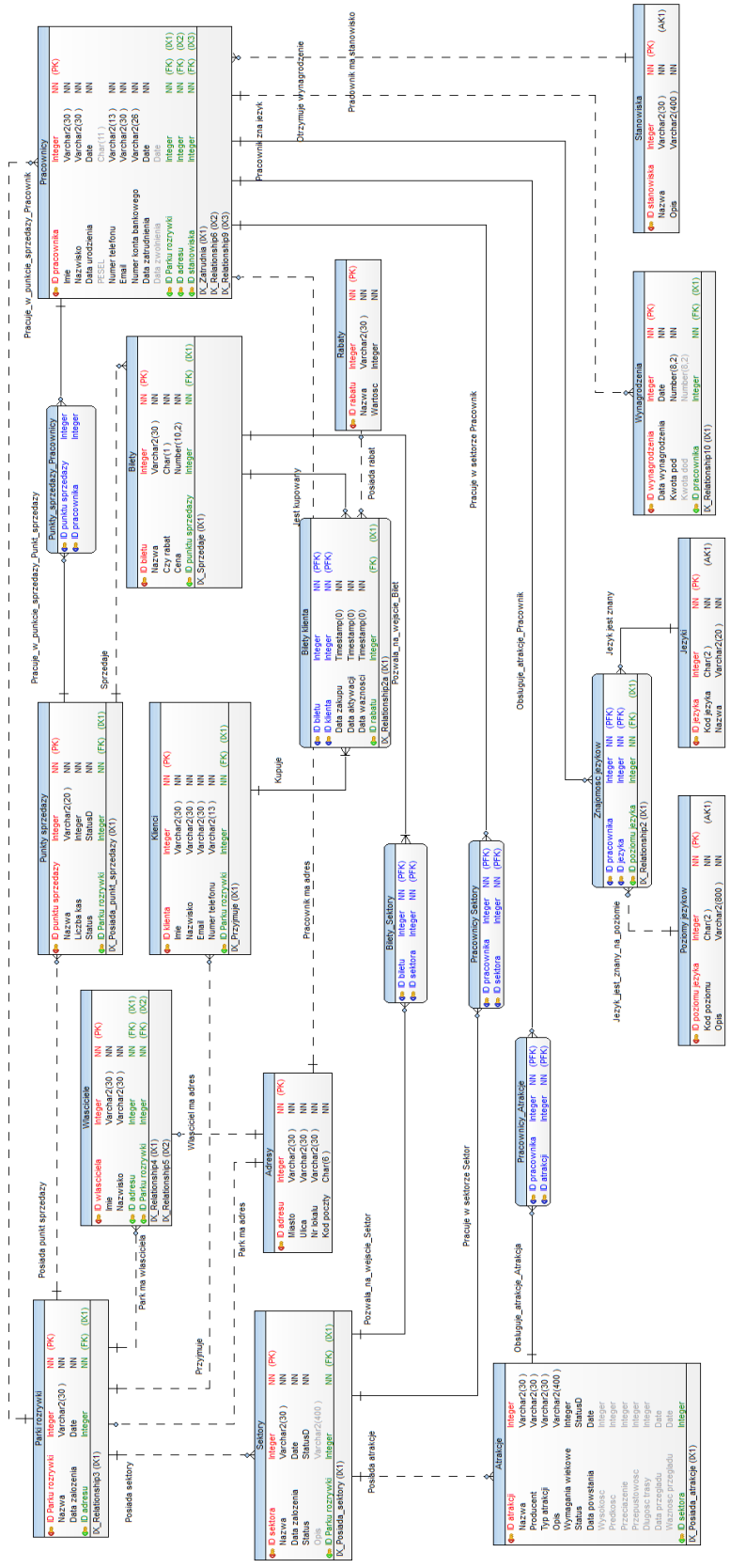
Unikając na etapie 1PN powtarzających się grup w prosty sposób otrzymujemy 3PN.Nie mamy atrybutów przechodnio wunkcyjnie zależnych od kluczy potencjalnych.

Ciekawym przykładem jest modyfikacja związków i relacji związanych z zakupem biletu.Ze związku N:M powstaje relacja bridge'ująca `Bilety klienta`. Z relacji Bilety wyodrębnia się relacja słownikowa `Rabaty`, które łączą się związkiem `0...1 to 0...n` z relacją `Bilety klienta`. Pole segmentowe `Okres waznosci` rozbijamy na dwa pola atomowe:`Data aktywacji Data waznosci` i przenosimy do relacji łączącej. Dzięki takiej strukturze jest odporna na anomalie aktualizacji, dołączania i usuwania. Klient może kupować wiele biletów. Bilety są reprezentacją rodzajów biletów i mogą istnieć niezależnie od klienta, a rabaty niezależnie od bieltu. Rabat łączy się z relacją bridge'ującą, bo to od konkretnego zakupu będzie zależeć, to jaki rabat zostanie przyznany kupującemu.



Z atrybutów Atrakcji `Typ atrakcji` i `Producent` też moglibyśmy utworzyć relacji, jednak zdecydowaliśmy, że informacje o tych danych interesują nas tylko w kontekście danej atrakcji i pozostawiliśmy je jako jej atrybuty.

4.4. Schemat ER na poziomie modelu logicznego



4.5. Więzy integralności

Dane przechowywane w bazie danych powinny spełniać wymogi integralności wynikające z założeń przyjętych podczas projektowania bazy. Reguły integralności bazy danych zapewniają poprawność i spójność danych w bazie. W bazie na tym etapie nie występują żadne pola segmentowe ani wielowartościowe.Wszystkie klucze główne posiadają niepowtarzalne wartości dzięki zaznaczeniu opcji PRIMARY KEY. Większość pól posiada zaznaczoną wartość NOT NULL. UNIQUE Zapewnia unikalność wartości w kolumnie, jednak w przeciwieństwie do PRIMARY KEY takich kluczy może być więcej niż jeden, oraz umożliwia występowanie wartości NULL. Foreign key jest to klucz obcy. Służy do definiowania relacji pomiędzy tabelami. Zapewnia że rekord w tabeli podrzędnej zawsze będzie miał swojego odpowiednika w tabeli nadrzędnej. Klucz obcy musi się odwoływać do kolumny (kolumn) w tabeli nadrzędnej, na których założony jest UNIQUE lub klucz główny.

Nie można wprowadzić wartości w polu klucza obcego tabeli połączonej, jeśli ta wartość nie istnieje w polu klucza głównego tabeli podstawowej. Np. nie możemy dodać rekordu opisującego Pracownia i w polu `id_stanowiska` wprowadzić identyfikator stanowiska nieistniejącego w tabeli `Stanowiska`.

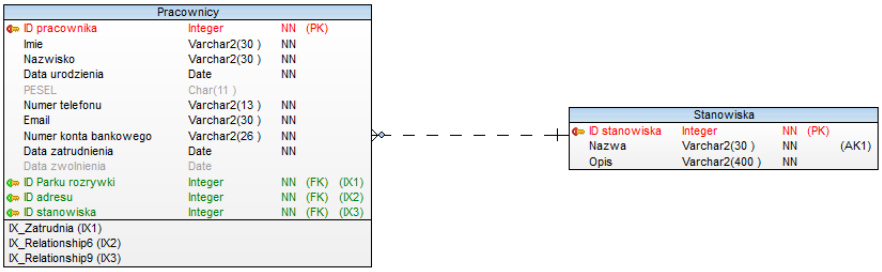
Nie można usunąć rekordu z tabeli podstawowej, jeśli w tabeli połączonej istnieją rekordy pasujące do niego. Np. nie możemy usunąć rekordu opisującego rabat, jeśli w tabeli `Bilety` istnieją bilety, które z tego rabatu korzystają. Można jednak zdecydować się na usunięcie rekordu podstawowego oraz wszystkich rekordów pokrewnych w ramach jednej operacji.

Nie można zmienić wartości klucza głównego w tabeli podstawowej, jeśli spowodowałoby to powstanie rekordów odłączonych. Nie można na przykład zmienić identyfikatora sektora w tabeli `Sektory`, jeśli w tabeli `Atrakcje` istnieją atrakcje przypisane do tego sektora. Można jednak zdecydować się na zaktualizowanie rekordu podstawowego oraz wszystkich rekordów pokrewnych w ramach jednej operacji, zaznaczając pole wyboru Kaskadowo aktualizuj pola pokrewne.

4.6. Proces denormalizacji - analiza i przykłady

Denormalizacja bazy danych jest to wprowadzenie kontrolowanej redundancji danych do bazy danych w celu przyspieszenia wykonywania na niej operacji (np. obsługiwania zapytań). Dzięki denormalizacji bazy unika się operacji połączeń tabel.

Moglibyśmy zdenormalizować np. relacje `Pracownicy` włączając w nią atrybuty relacji `Stanowiska`. Dzięki temu usprawnilibyśmy proces wyszukiwania informacji o tym jakie stanowisko zajmuje pracownik. Z drugiej strony, gdybyśmy chcieli zaktualizować wymagania dotyczące danego stanowiska, czyli atrybut opis stanowiska, musielibyśmy przeprowadzać taką aktualizację dla każdego pracownika oddzielnie. Uznaliśmy, że w naszym projekcie nie będziemy przeprowadzać żadnej denormalizacji, ponieważ dla naszej bazy wiązało by się to z większymy stratami, niż zyskami.



5. Faza Fizyczna

5.1. Projekt transakcji i weryfikacji ich wykonalności

Nazwa Transakcji	Wykorzystane zasoby	Realizowalność
Liczba sprzedanych biletów	Encja 'Bilety'	Tak
Liczba klientów	Encja 'Klienci'	Tak
Znajomość języka pracownika	Relacja 'pracownik_zna_jezyk'	Tak
Wynagrodzeniela administracji	Relacja 'Otrzymuje wynagrodzenie'	Tak
Modyfikacja rabatu	Encja 'Rabaty'	Tak
Zmiana statusu atrakcji	Encja 'Atrakcje'	Tak
Dodawanie/modyfikacja wynagrodzeń	Encja 'Wynagrodzenia'	Tak
Dodawanie/Modyfikacja atrakcji	Encja 'Atrakcje'	Tak
Modyfikacja danych osobowych	Encja 'Pracownicy'	Tak
Dodawanie/modyfikacja sektorów	Encja 'Sektory'	Tak
Ustalenia dnia specjalnej zniżki	Nie istnieją	Nie

5.2. Przykłady zapytań i poleceń SQL odnoszących się do bazy danych

```sql
--- liczba sprzedanych biletów ---
SELECT SUM(CENA) AS Suma_cen_Biletów, COUNT(ID_BILETU)
 AS Liczba_biletów FROM BILETY;
--- Wynagrodzenie dla Administracji---
SELECT IMIE, NAZWISKO, NR_KONTA_BANKOWEGO, Kwota_pod, Kwota_dod, Data_wynagrodzenia
 FROM PRACOWNICY NATURAL JOIN Wynagrodzenia;
---Liczba klientów---
SELECT COUNT(id_klienta) AS LICZBA_KLIENTOW FROM KLIENCI;

--- Modyfikacja rabatu---
UPDATE Rabaty SET wartosc=30 WHERE id_rabatu=1;

--- Modyfikacja statusu atrakcji ---
update atrakcje
  set status = case
                 when status = 'Otwarte' then 'Zamkniete'
                 else 'Otwarte'
               end
where id_atrakcji=1;
```

5.3. Strojenie bazy danych - dobór indeksów

Indeksowanie jest podstawowym mechanizmem wykorzystywanym w celu optymalizacji baz danych.

```sql
-- Create indexes for table Atrakcje
CREATE INDEX IX_Posiada_atrakcje ON Atrakcje (id_sektora)
-- Create indexes for table Park_Rozrywki
CREATE INDEX IX_Relationship3 ON Park_Rozrywki (id_adresu)
 -- Create indexes for table Punkty_sprzedazy
CREATE INDEX IX_Posiada_punkt_sprzedazy ON Punkty_sprzedazy (id_parku_rozrywki)
-- Create indexes for table Klienci
CREATE INDEX IX_Przyjmuje ON Klienci (id_parku_rozrywki)
-- Create indexes for table Pracownicy
CREATE INDEX IX_Zatrudnia ON Pracownicy (id_parku_rozrywki)
CREATE INDEX IX_Relationship6 ON Pracownicy (id_adresu)
CREATE INDEX IX_Relationship9 ON Pracownicy (id_stanowiska)
-- Create indexes for table Bilety
CREATE INDEX IX_Sprzedaje ON Bilety (id_punktu_sprzedazy)
-- Create indexes for table Sektory
CREATE INDEX IX_Posiada_sektory ON Sektory (id_parku_rozrywki)
```

```
-- Create indexes for table Wlasciciele
CREATE INDEX IX_Relationship4 ON Wlasciciele (id_adresu)
CREATE INDEX IX_Relationship5 ON Wlasciciele (id_parku_rozrywki)
-- Create indexes for table Wynagrodzenia
CREATE INDEX IX_Relationship10 ON Wynagrodzenia (id_pracownika)
-- Create indexes for table znajomosc_jezykow
CREATE INDEX IX_Relationship2 ON znajomosc_jezykow (id_poziomu_jezyka)
-- Create indexes for table Bilety_klienta
CREATE INDEX IX_Relationship2a ON Bilety_klienta (id_rabatu)
```

5.4. Skrypt SQL zakładający bazę danych

```
/*
Created: 21.11.2020
Modified: 28.11.2020
Model: Park_Rozrywki_MR
Version: ee0e73e
Database: Oracle 19c
*/


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

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

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

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

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

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

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

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

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

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

CREATE SEQUENCE RabatSeq
 INCREMENT BY 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

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

CREATE SEQUENCE StanowiskoSeq
```

```
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

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

-- Table Atrakcje

CREATE TABLE Atrakcje(
  id_atrakcji Integer NOT NULL,
  nazwa Varchar2(30 ) NOT NULL,
  producent Varchar2(30 ) NOT NULL,
  typ_atrakcji Varchar2(30 ) NOT NULL,
  opis Varchar2(400 ) NOT NULL,
  wymagania_wiekowe Integer NOT NULL,
  status Varchar2(9 ) NOT NULL
        CHECK (Status IN ('Otwarte','Zamkniete')),
  data_powstania Date NOT NULL,
  wysokosc Integer,
  predkosc Integer,
  przeciazenie Integer,
  przepustowosc Integer,
  dlugosc_trasy Integer,
  data_przegladu Date,
  waznosc_przegladu Date,
  id_sektora Integer NOT NULL
)
/

-- Create indexes for table Atrakcje

CREATE INDEX IX_Posiada_atrakcje ON Atrakcje (id_sektora)
/

-- Add keys for table Atrakcje

ALTER TABLE Atrakcje ADD CONSTRAINT Atrakcja_PK PRIMARY KEY (id_atrakcji)
/

-- Table and Columns comments section

COMMENT ON COLUMN Atrakcje.typ_atrakcji IS 'Typ atrakcji'
/
COMMENT ON COLUMN Atrakcje.opis IS 'Opis atrakcji'
/
COMMENT ON COLUMN Atrakcje.data_przegladu IS 'Data wykonania przeglądu technicznego
↪  atrakcji'
/
COMMENT ON COLUMN Atrakcje.waznosc_przegladu IS 'Data ważności przeglądu technicznego
↪  atrakcji'
/

-- Table Park_Rozrywki
```

```
CREATE TABLE Park_Rozrywki(
  id_parku_rozrywki Integer NOT NULL,
  nazwa Varchar2(30 ) NOT NULL,
  data_zalozenia Date NOT NULL,
  id_adresu Integer NOT NULL
)
/

-- Create indexes for table Park_Rozrywki

CREATE INDEX IX_Relationship3 ON Park_Rozrywki (id_adresu)
/

-- Add keys for table Park_Rozrywki

ALTER TABLE Park_Rozrywki ADD CONSTRAINT Park_rozrywki_PK PRIMARY KEY (id_parku_rozrywki)
/

-- Table Punkty_sprzedazy

CREATE TABLE Punkty_sprzedazy(
  id_punktu_sprzedazy Integer NOT NULL,
  nazwa Varchar2(20 ) NOT NULL,
  liczba_kas Integer NOT NULL,
  status Varchar2(9 ) NOT NULL
        CONSTRAINT CheckConstraintA3 CHECK (Status IN ('Otwarte','Zamkniete'))
        CHECK (Status IN ('Otwarte','Zamkniete')),
  id_parku_rozrywki Integer NOT NULL
)
/

-- Create indexes for table Punkty_sprzedazy

CREATE INDEX IX_Posiada_punkt_sprzedazy ON Punkty_sprzedazy (id_parku_rozrywki)
/

-- Add keys for table Punkty_sprzedazy

ALTER TABLE Punkty_sprzedazy ADD CONSTRAINT Punkt_sprzedazy_PK PRIMARY KEY
↪  (id_punktu_sprzedazy)
/

-- Table Klienci

CREATE TABLE Klienci(
  id_klienta Integer NOT NULL,
  imie Varchar2(30 ) NOT NULL,
  nazwisko Varchar2(30 ) NOT NULL,
  email Varchar2(30 ) NOT NULL,
  nr_telefonu Varchar2(13 ) NOT NULL,
  id_parku_rozrywki Integer NOT NULL
)
/

-- Create indexes for table Klienci
```

```
CREATE INDEX IX_Przyjmuje ON Klienci (id_parku_rozrywki)
/

-- Add keys for table Klienci

ALTER TABLE Klienci ADD CONSTRAINT Klient_PK PRIMARY KEY (id_klienta)
/

-- Table Pracownicy

CREATE TABLE Pracownicy(
  id_pracownika Integer NOT NULL,
  imie Varchar2(30 ) NOT NULL,
  nazwisko Varchar2(30 ) NOT NULL,
  data_urodzenia Date NOT NULL,
  PESEL Char(11 ),
  nr_telefonu Varchar2(13 ) NOT NULL,
  email Varchar2(30 ) NOT NULL,
  nr_konta_bankowego Varchar2(26 ) NOT NULL,
  data_zatrudnienia Date NOT NULL,
  data_zwolnienia Date,
  id_parku_rozrywki Integer NOT NULL,
  id_adresu Integer NOT NULL,
  id_stanowiska Integer NOT NULL
)
/

-- Create indexes for table Pracownicy

CREATE INDEX IX_Zatrudnia ON Pracownicy (id_parku_rozrywki)
/

CREATE INDEX IX_Relationship6 ON Pracownicy (id_adresu)
/

CREATE INDEX IX_Relationship9 ON Pracownicy (id_stanowiska)
/

-- Add keys for table Pracownicy

ALTER TABLE Pracownicy ADD CONSTRAINT Pracownik_PK PRIMARY KEY (id_pracownika)
/

-- Table Bilety

CREATE TABLE Bilety(
  id_biletu Integer NOT NULL,
  nazwa Varchar2(30 ) NOT NULL,
  czy_rabat Char(1 ) NOT NULL,
  cena Number(10,2) NOT NULL,
  id_punktu_sprzedazy Integer NOT NULL
)
/

-- Create indexes for table Bilety
```

```
CREATE INDEX IX_Sprzedaje ON Bilety (id_punktu_sprzedazy)
/

-- Add keys for table Bilety

ALTER TABLE Bilety ADD CONSTRAINT Bilet_PK PRIMARY KEY (id_biletu)
/

-- Table and Columns comments section

COMMENT ON COLUMN Bilety.czy_rabat IS 'Czy przysługuje rabat (Boolean)'
/

-- Table Sektory

CREATE TABLE Sektory(
  id_sektora Integer NOT NULL,
  nazwa Varchar2(30 ) NOT NULL,
  data_zalozenia Date NOT NULL,
  status Varchar2(9 ) NOT NULL
        CONSTRAINT CheckConstraintA1 CHECK (Status IN ('Otwarte','Zamkniete'))
        CHECK (Status IN ('Otwarte','Zamkniete')),
  opis Varchar2(400 ),
  id_parku_rozrywki Integer NOT NULL
)
/

-- Create indexes for table Sektory

CREATE INDEX IX_Posiada_sektory ON Sektory (id_parku_rozrywki)
/

-- Add keys for table Sektory

ALTER TABLE Sektory ADD CONSTRAINT Sektor_PK PRIMARY KEY (id_sektora)
/

-- Table and Columns comments section

COMMENT ON COLUMN Sektory.opis IS 'Opis sektora'
/

-- Table Punkty_sprzedazy_Pracownicy

CREATE TABLE Punkty_sprzedazy_Pracownicy(
  id_punktu_sprzedazy Integer NOT NULL,
  id_pracownika Integer NOT NULL
)
/

-- Table Pracownicy_Atrakcje

CREATE TABLE Pracownicy_Atrakcje(
  id_pracownika Integer NOT NULL,
  id_atrakcji Integer NOT NULL
```

```
)
/

-- Table Bilety_Sektory

CREATE TABLE Bilety_Sektory(
  id_biletu Integer NOT NULL,
  id_sektora Integer NOT NULL
)
/

-- Table Rabaty

CREATE TABLE Rabaty(
  id_rabatu Integer NOT NULL,
  nazwa Varchar2(30 ) NOT NULL,
  wartosc Integer NOT NULL
)
/

-- Add keys for table Rabaty

ALTER TABLE Rabaty ADD CONSTRAINT PK_Rabaty PRIMARY KEY (id_rabatu)
/

-- Table and Columns comments section

COMMENT ON COLUMN Rabaty.id_rabatu IS 'Unikatowy identyfikator rabatu'
/
COMMENT ON COLUMN Rabaty.nazwa IS 'Nazwa rabatu'
/
COMMENT ON COLUMN Rabaty.wartosc IS 'ile % rabatu'
/

-- Table Adresy

CREATE TABLE Adresy(
  id_adresu Integer NOT NULL,
  miasto Varchar2(30 ) NOT NULL,
  Ulica Varchar2(30 ) NOT NULL,
  nr_lokalu Varchar2(30 ) NOT NULL,
  kod_poczty Char(6 ) 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 adresu'
/
COMMENT ON COLUMN Adresy.miasto IS 'Miasto'
/
```

```
COMMENT ON COLUMN Adresy.Ulica IS 'Ulica'
/
COMMENT ON COLUMN Adresy.nr_lokalu IS 'Numer lokalu'
/
COMMENT ON COLUMN Adresy.kod_poczty IS 'Kod pocztowy'
/

-- Table Wlasciciele

CREATE TABLE Wlasciciele(
  id_wlasciciela Integer NOT NULL,
  imie Varchar2(30 ) NOT NULL,
  nazwisko Varchar2(30 ) NOT NULL,
  id_adresu Integer NOT NULL,
  id_parku_rozrywki Integer NOT NULL
)
/

-- Create indexes for table Wlasciciele

CREATE INDEX IX_Relationship4 ON Wlasciciele (id_adresu)
/

CREATE INDEX IX_Relationship5 ON Wlasciciele (id_parku_rozrywki)
/

-- 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 identyfikatow właściciela'
/

-- Table Stanowiska

CREATE TABLE Stanowiska(
  id_stanowiska Integer NOT NULL,
  nazwa Varchar2(30 ) NOT NULL,
  opis Varchar2(400 ) 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)
/

-- Table and Columns comments section

COMMENT ON COLUMN Stanowiska.id_stanowiska IS 'Unikatowy identyfikator stanowiska'
```

```
/
COMMENT ON COLUMN Stanowiska.nazwa 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_pod Number(8,2) NOT NULL,
  kwota_dod Number(8,2),
  id_pracownika Integer NOT NULL
)
/

-- Create indexes for table Wynagrodzenia

CREATE INDEX IX_Relationship10 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
↪  wynagrodzenia'
/
COMMENT ON COLUMN Wynagrodzenia.data_wynagrodzenia IS 'Data wypłaty wynagrodzenia'
/
COMMENT ON COLUMN Wynagrodzenia.kwota_pod IS 'Kwota podstawowa'
/
COMMENT ON COLUMN Wynagrodzenia.kwota_dod IS 'Kwota dodatkowa'
/

-- Table Jezyki

CREATE TABLE Jezyki(
  id_jezyka Integer NOT NULL,
  kod_jezyka Char(2 ) NOT NULL,
  nazwa Varchar2(20 ) NOT NULL
)
/

-- Add keys for table Jezyki

ALTER TABLE Jezyki ADD CONSTRAINT PK_Jezyki PRIMARY KEY (id_jezyka)
/

ALTER TABLE Jezyki ADD CONSTRAINT kod_jezyka UNIQUE (kod_jezyka)
/
```

```
-- Table and Columns comments section

COMMENT ON COLUMN Jezyki.id_jezyka IS 'Unikatowy identyfikator języka'
/
COMMENT ON COLUMN Jezyki.kod_jezyka IS 'Kod języka zgodnie ze standardem ISO'
/
COMMENT ON COLUMN Jezyki.nazwa IS 'Nazwa języka'
/

-- Table znajomosc_jezykow

CREATE TABLE znajomosc_jezykow(
  id_pracownika Integer NOT NULL,
  id_jezyka Integer NOT NULL,
  id_poziomu_jezyka Integer NOT NULL
)
/

-- Create indexes for table znajomosc_jezykow

CREATE INDEX IX_Relationship2 ON znajomosc_jezykow (id_poziomu_jezyka)
/

-- Add keys for table znajomosc_jezykow

ALTER TABLE znajomosc_jezykow ADD CONSTRAINT PK_znajomosc_jezykow PRIMARY KEY
↪  (id_pracownika,id_jezyka)
/

-- Table Poziomy_jezykow

CREATE TABLE Poziomy_jezykow(
  id_poziomu_jezyka Integer NOT NULL,
  kod_poziomu Char(2 ) NOT NULL,
  Opis Varchar2(800 ) NOT NULL
)
/

-- Add keys for table Poziomy_jezykow

ALTER TABLE Poziomy_jezykow ADD CONSTRAINT PK_Poziomy_jezykow PRIMARY KEY
↪  (id_poziomu_jezyka)
/

ALTER TABLE Poziomy_jezykow ADD CONSTRAINT kod_poziomu UNIQUE (kod_poziomu)
/

-- Table and Columns comments section

COMMENT ON COLUMN Poziomy_jezykow.id_poziomu_jezyka IS 'Unikatowy identyfikatow języka'
/
COMMENT ON COLUMN Poziomy_jezykow.kod_poziomu IS 'Kod poziomu'
/
COMMENT ON COLUMN Poziomy_jezykow.Opis IS 'Opis wymagań'
/
```

```
-- Table Bilety_klienta

CREATE TABLE Bilety_klienta(
  id_biletu Integer NOT NULL,
  id_klienta Integer NOT NULL,
  data_zakupu Timestamp(0) NOT NULL,
  data_aktywacji Timestamp(0) NOT NULL,
  data_waznosci Timestamp(0) NOT NULL,
  id_rabatu Integer
)
/

-- Create indexes for table Bilety_klienta

CREATE INDEX IX_Relationship2a ON Bilety_klienta (id_rabatu)
/

-- Add keys for table Bilety_klienta

ALTER TABLE Bilety_klienta ADD CONSTRAINT PK_Bilety_klienta PRIMARY KEY
↪  (id_biletu,id_klienta)
/

-- Table and Columns comments section

COMMENT ON COLUMN Bilety_klienta.data_zakupu IS 'Data i godzina zakupu biletu'
/
COMMENT ON COLUMN Bilety_klienta.data_aktywacji IS 'Data i godzina, o której bilet się
↪  aktywuje'
/
COMMENT ON COLUMN Bilety_klienta.data_waznosci IS 'Data i godzina, o której bilet traci
↪  ważność'
/

-- Table Pracownicy_Sektory

CREATE TABLE Pracownicy_Sektory(
  id_pracownika Integer NOT NULL,
  id_sektora Integer NOT NULL
)
/

-- Add keys for table Pracownicy_Sektory

ALTER TABLE Pracownicy_Sektory ADD CONSTRAINT PK_Pracownicy_Sektory PRIMARY KEY
↪  (id_pracownika,id_sektora)
/

-- Trigger for sequence AtrakcjaSeq for column id_atrakcji in table Atrakcje ---------
CREATE OR REPLACE TRIGGER ts_Atrakcje_AtrakcjaSeq BEFORE INSERT
ON Atrakcje FOR EACH ROW
BEGIN
  :new.id_atrakcji := AtrakcjaSeq.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Atrakcje_AtrakcjaSeq 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 Park_rozrywkiSeq for column id_parku_rozrywki in table
↪  Park_Rozrywki ---------
CREATE OR REPLACE TRIGGER ts_Park_Rozrywki_Park_rozrywkiSeq BEFORE INSERT
ON Park_Rozrywki FOR EACH ROW
BEGIN
  :new.id_parku_rozrywki := Park_rozrywkiSeq.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Park_Rozrywki_Park_rozrywkiSeq AFTER UPDATE OF
↪  id_parku_rozrywki
ON Park_Rozrywki FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column id_parku_rozrywki in table
  ↪  Park_Rozrywki as it uses sequence.');
END;
/

-- Trigger for sequence Punkt_SprzedazySeq for column id_punktu_sprzedazy in table
↪  Punkty_sprzedazy ---------
CREATE OR REPLACE TRIGGER ts_Punkty_sprzedazy_Punkt_SprzedazySeq BEFORE INSERT
ON Punkty_sprzedazy FOR EACH ROW
BEGIN
  :new.id_punktu_sprzedazy := Punkt_SprzedazySeq.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Punkty_sprzedazy_Punkt_SprzedazySeq AFTER UPDATE OF
↪  id_punktu_sprzedazy
ON Punkty_sprzedazy FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column id_punktu_sprzedazy in table
  ↪  Punkty_sprzedazy as it uses sequence.');
END;
/

-- Trigger for sequence KlientSeq for column id_klienta in table Klienci ---------
CREATE OR REPLACE TRIGGER ts_Klienci_KlientSeq BEFORE INSERT
ON Klienci FOR EACH ROW
BEGIN
  :new.id_klienta := KlientSeq.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Klienci_KlientSeq 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 PracownikSeq for column id_pracownika in table Pracownicy
↪  ---------
CREATE OR REPLACE TRIGGER ts_Pracownicy_PracownikSeq BEFORE INSERT
ON Pracownicy FOR EACH ROW
BEGIN
  :new.id_pracownika := PracownikSeq.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Pracownicy_PracownikSeq 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 BiletSeq for column id_biletu in table Bilety ---------
CREATE OR REPLACE TRIGGER ts_Bilety_BiletSeq BEFORE INSERT
ON Bilety FOR EACH ROW
BEGIN
  :new.id_biletu := BiletSeq.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Bilety_BiletSeq 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 SektorSeq for column id_sektora in table Sektory ---------
CREATE OR REPLACE TRIGGER ts_Sektory_SektorSeq BEFORE INSERT
ON Sektory FOR EACH ROW
BEGIN
  :new.id_sektora := SektorSeq.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Sektory_SektorSeq AFTER UPDATE OF id_sektora
ON Sektory FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column id_sektora in table Sektory as it
  ↪  uses sequence.');
END;
/

-- Trigger for sequence RabatSeq for column id_rabatu in table Rabaty ---------
CREATE OR REPLACE TRIGGER ts_Rabaty_RabatSeq BEFORE INSERT
ON Rabaty FOR EACH ROW
BEGIN
  :new.id_rabatu := RabatSeq.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Rabaty_RabatSeq AFTER UPDATE OF id_rabatu
ON Rabaty FOR EACH ROW
BEGIN
```

```
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column id_rabatu in table Rabaty as it
  ↪  uses sequence.');
END;
/

-- Trigger for sequence AdresSeq for column id_adresu in table Adresy ---------
CREATE OR REPLACE TRIGGER ts_Adresy_AdresSeq BEFORE INSERT
ON Adresy FOR EACH ROW
BEGIN
  :new.id_adresu := AdresSeq.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Adresy_AdresSeq 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 WlascicielSeq for column id_wlasciciela in table Wlasciciele
↪  ---------
CREATE OR REPLACE TRIGGER ts_Wlasciciele_WlascicielSeq BEFORE INSERT
ON Wlasciciele FOR EACH ROW
BEGIN
  :new.id_wlasciciela := WlascicielSeq.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Wlasciciele_WlascicielSeq 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 StanowiskoSeq for column id_stanowiska in table Stanowiska
↪  ---------
CREATE OR REPLACE TRIGGER ts_Stanowiska_StanowiskoSeq BEFORE INSERT
ON Stanowiska FOR EACH ROW
BEGIN
  :new.id_stanowiska := StanowiskoSeq.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Stanowiska_StanowiskoSeq 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 WynagrodzenieSeq for column id_wynagrodzenia in table
↪  Wynagrodzenia ---------
CREATE OR REPLACE TRIGGER ts_Wynagrodzenia_WynagrodzenieSeq BEFORE INSERT
ON Wynagrodzenia FOR EACH ROW
```

```
BEGIN
  :new.id_wynagrodzenia := WynagrodzenieSeq.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Wynagrodzenia_WynagrodzenieSeq 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;
/


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

ALTER TABLE Pracownicy ADD CONSTRAINT Zatrudnia FOREIGN KEY (id_parku_rozrywki) REFERENCES
↪  Park_Rozrywki (id_parku_rozrywki)
/



ALTER TABLE Bilety ADD CONSTRAINT Sprzedaje FOREIGN KEY (id_punktu_sprzedazy) REFERENCES
↪  Punkty_sprzedazy (id_punktu_sprzedazy)
/



ALTER TABLE Sektory ADD CONSTRAINT Posiada_sektory FOREIGN KEY (id_parku_rozrywki)
↪  REFERENCES Park_Rozrywki (id_parku_rozrywki)
/



ALTER TABLE Atrakcje ADD CONSTRAINT Posiada_atrakcje FOREIGN KEY (id_sektora) REFERENCES
↪  Sektory (id_sektora)
/



ALTER TABLE Klienci ADD CONSTRAINT Przyjmuje FOREIGN KEY (id_parku_rozrywki) REFERENCES
↪  Park_Rozrywki (id_parku_rozrywki)
/



ALTER TABLE Punkty_sprzedazy ADD CONSTRAINT Posiada_punkt_sprzedazy FOREIGN KEY
↪  (id_parku_rozrywki) REFERENCES Park_Rozrywki (id_parku_rozrywki)
/



ALTER TABLE Park_Rozrywki ADD CONSTRAINT Park_ma_adres FOREIGN KEY (id_adresu) REFERENCES
↪  Adresy (id_adresu)
/
```

```
ALTER TABLE Wlasciciele ADD CONSTRAINT Wlasciciel_ma_adres FOREIGN KEY (id_adresu)
↪  REFERENCES Adresy (id_adresu)
/



ALTER TABLE Wlasciciele ADD CONSTRAINT Park_ma_wlasciciela FOREIGN KEY (id_parku_rozrywki)
↪  REFERENCES Park_Rozrywki (id_parku_rozrywki)
/



ALTER TABLE Pracownicy ADD CONSTRAINT Pracownik_ma_adres FOREIGN KEY (id_adresu)
↪  REFERENCES Adresy (id_adresu)
/



ALTER TABLE Pracownicy ADD CONSTRAINT Pracownik_ma_stanowisko FOREIGN KEY (id_stanowiska)
↪  REFERENCES Stanowiska (id_stanowiska)
/



ALTER TABLE Wynagrodzenia ADD CONSTRAINT Otrzymuje_wynagrodzenie FOREIGN KEY
↪  (id_pracownika) REFERENCES Pracownicy (id_pracownika)
/



ALTER TABLE znajomosc_jezykow ADD CONSTRAINT Pracownik_zna_jezyk FOREIGN KEY
↪  (id_pracownika) REFERENCES Pracownicy (id_pracownika)
/



ALTER TABLE znajomosc_jezykow ADD CONSTRAINT Jezyk_jest_znany FOREIGN KEY (id_jezyka)
↪  REFERENCES Jezyki (id_jezyka)
/



ALTER TABLE Bilety_klienta ADD CONSTRAINT Jest_kupowany FOREIGN KEY (id_biletu) REFERENCES
↪  Bilety (id_biletu)
/



ALTER TABLE Bilety_klienta ADD CONSTRAINT Kupuje FOREIGN KEY (id_klienta) REFERENCES
↪  Klienci (id_klienta)
/
```

```
ALTER TABLE Pracownicy_Sektory ADD CONSTRAINT Pracuje_w_sektorze_Pracownik FOREIGN KEY
↪  (id_pracownika) REFERENCES Pracownicy (id_pracownika)
/



ALTER TABLE Pracownicy_Sektory ADD CONSTRAINT Pracuje_w_sektorze_Sektor FOREIGN KEY
↪  (id_sektora) REFERENCES Sektory (id_sektora)
/



ALTER TABLE znajomosc_jezykow ADD CONSTRAINT Jezyk_jest_znany_na_poziomie FOREIGN KEY
↪  (id_poziomu_jezyka) REFERENCES Poziomy_jezykow (id_poziomu_jezyka)
/



ALTER TABLE Bilety_klienta ADD CONSTRAINT Posiada_rabat FOREIGN KEY (id_rabatu) REFERENCES
↪  Rabaty (id_rabatu)
/
```

5.5. Skrypt SQL wypełniający bazę danych

```
---Adresy---

INSERT INTO Adresy (id_adresu, miasto, ulica, nr_lokalu, kod_poczty)
VALUES (1, 'Warszawa', 'Księcia Janusza', 39, '01-452');

INSERT INTO Adresy (id_adresu, miasto, ulica, nr_lokalu, kod_poczty)
VALUES (2, 'Gdańsk', 'Główna', 3, '80-442');

INSERT INTO Adresy (id_adresu, miasto, ulica, nr_lokalu, kod_poczty)
VALUES (3, 'Warszawa', 'Złota', 5, '00-120');

INSERT INTO Adresy (id_adresu, miasto, ulica, nr_lokalu, kod_poczty)
VALUES (4, 'Gdańsk', 'Główna', 6, '80-442');

INSERT INTO Adresy (id_adresu, miasto, ulica, nr_lokalu, kod_poczty)
VALUES (5, 'Gdańsk', 'Główna', 7, '80-442');


--- Parki rozrywki ---
INSERT INTO Park_rozrywki (id_parku_rozrywki, nazwa, data_zalozenia, id_adresu)
VALUES (1, 'Park Ustronie', TO_DATE('1909-02-07', 'YYYY-MM-DD'), 5);

INSERT INTO Park_rozrywki (id_parku_rozrywki, nazwa, data_zalozenia, id_adresu)
VALUES (2, 'Park Gdańsk', TO_DATE('1909-02-07', 'YYYY-MM-DD'), 2);

--- Sektory ---
INSERT INTO Sektory (id_sektora, nazwa, data_zalozenia, status, opis, id_parku_rozrywki)
VALUES (1, 'Sektor śmiechu', TO_DATE('2010-02-05', 'YYYY-MM-DD'),
 'Otwarte', 'Przyjemne miejsce dla wszystkich!', 2);
```

```
INSERT INTO Sektory (id_sektora, nazwa, data_zalozenia, status, opis, id_parku_rozrywki)
VALUES (2, 'Sektor zabawy', TO_DATE('2011-04-02', 'YYYY-MM-DD'),
 'Otwarte', 'Wspaniała zabawa', 1);

INSERT INTO Sektory (id_sektora, nazwa, data_zalozenia, status, opis, id_parku_rozrywki)
VALUES (3, 'Sektor ekstremalny', TO_DATE('12-03-02', 'YYYY-MM-DD'),
 'Otwarte', 'Przyjemne miejsce dla wszystkich!', 2);

INSERT INTO Sektory (id_sektora, nazwa, data_zalozenia, status, opis, id_parku_rozrywki)
VALUES (4, 'sektor atrakcji wodnych', TO_DATE('2013-12-12', 'YYYY-MM-DD'),
 'Otwarte', 'Wspaniała zabawa', 2);

--- Wlasciciele ---
INSERT INTO Wlasciciele (id_wlasciciela, imie, nazwisko, id_adresu, id_parku_rozrywki)
VALUES (1, 'Jan', 'Wielki', 2, 2);

INSERT INTO Wlasciciele (id_wlasciciela, imie, nazwisko, id_adresu, id_parku_rozrywki)
VALUES (2, 'Jan', 'Nowak', 3, 1);

--- Punkty Sprzedaży ---
INSERT INTO Punkty_sprzedazy (id_punktu_sprzedazy, nazwa, liczba_kas, status,
↪  id_parku_rozrywki)
VALUES (1, 'Wielki', 2, 'Otwarte', 2);

INSERT INTO Punkty_sprzedazy (id_punktu_sprzedazy, nazwa, liczba_kas, status,
↪  id_parku_rozrywki)
VALUES (2, 'Niewielki', 2, 'Otwarte', 1);

--- Klienci ---
INSERT INTO klienci (id_klienta, imie, nazwisko, email, nr_telefonu, id_parku_rozrywki)
VALUES (1, 'Jan', 'Malutki','malyalewariat@gmail.com', '+48223123176', 1);

INSERT INTO klienci (id_klienta, imie, nazwisko, email, nr_telefonu, id_parku_rozrywki)
VALUES (2, 'Jan', 'Chudy','JanChudy@gmail.com', '+48223147176', 2);

--- Stanowisko ---
INSERT INTO Stanowiska (id_stanowiska, nazwa, opis)
VALUES (1, 'Administrator', 'Zajmuje się administracją parku');

INSERT INTO Stanowiska (id_stanowiska, nazwa, opis)
VALUES (2, 'Pracownik punktu sprzedaży', 'Zajmuje się punktem sprzedaży');

INSERT INTO Stanowiska (id_stanowiska, nazwa, opis)
VALUES (3, 'Konserwator', 'Zajmuje się konserwacją atrakcji');

--- Pracownicy ---

INSERT INTO Pracownicy (id_pracownika, imie, nazwisko, data_urodzenia, nr_telefonu, email,
                        nr_konta_bankowego, data_zatrudnienia, id_adresu,
                        ↪  id_parku_rozrywki, id_stanowiska)
VALUES (1, 'Jan', 'Chudy',TO_DATE('2000-02-07', 'YYYY-MM-DD'), '+48993123564',
 'Chudziak@gmail.com', '161030133515458614581674 85', TO_DATE('2019-02-07', 'YYYY-MM-DD'),
 ↪  5 , 2, 1);

INSERT INTO Pracownicy (id_pracownika, imie, nazwisko, data_urodzenia, nr_telefonu, email,
```

```
                        nr_konta_bankowego, data_zatrudnienia, id_adresu,
                        ↪  id_parku_rozrywki, id_stanowiska)
VALUES (2, 'Jan', 'Wartki',TO_DATE('1973-12-07', 'YYYY-MM-DD'), '+48673123564',
 'WartkiJanek@gmail.com', '411240485214013509889887O2', TO_DATE('2009-02-07',
 ↪  'YYYY-MM-DD'), 3, 1, 3);

--- Punkt_sprzedazy_pracownicy ---
INSERT INTO Punkty_sprzedazy_pracownicy (id_punktu_sprzedazy, id_pracownika)
VALUES (1, 2);

--- Pracownicy_sektory ---
INSERT INTO Pracownicy_sektory (id_pracownika, id_sektora)
VALUES (2, 1);

--- Rabat ---
INSERT INTO Rabaty (id_rabatu, nazwa, wartosc)
VALUES (1, 'Dla studentów', 50);

INSERT INTO Rabaty (id_rabatu, nazwa, wartosc)
VALUES (2, 'Dla seniorów', 60);

INSERT INTO Rabaty (id_rabatu, nazwa, wartosc)
VALUES (3, 'Dla dzieci do lat 3', 80);


--- Bilety ---

INSERT INTO Bilety (id_biletu, nazwa, czy_rabat, cena,  id_punktu_sprzedazy)
VALUES (1, 'Normalny', 1, 20.99, 1);

INSERT INTO Bilety (id_biletu, nazwa, czy_rabat, cena, id_punktu_sprzedazy)
VALUES (2, 'Premium', 1, 40.99, 2);


--- Bilety_sektory ---
INSERT INTO Bilety_sektory (id_biletu, id_sektora)
VALUES (1, 1);

INSERT INTO Bilety_sektory (id_biletu, id_sektora)
VALUES (2, 2);

--- Wynagrodzenia ---
INSERT INTO Wynagrodzenia (id_wynagrodzenia, data_wynagrodzenia,
 kwota_pod, kwota_dod, id_pracownika)
VALUES (1, TO_DATE('2020-02-01', 'YYYY-MM-DD'),3000, 100, 1);

INSERT INTO Wynagrodzenia (id_wynagrodzenia, data_wynagrodzenia,
 kwota_pod, kwota_dod, id_pracownika)
VALUES (2, TO_DATE('2020-02-01', 'YYYY-MM-DD'),2000, 200, 2);

INSERT INTO Wynagrodzenia (id_wynagrodzenia, data_wynagrodzenia,
 kwota_pod, kwota_dod, id_pracownika)
VALUES (3, TO_DATE('2020-02-01', 'YYYY-MM-DD'),3000, 200, 1);

INSERT INTO Wynagrodzenia (id_wynagrodzenia, data_wynagrodzenia,
 kwota_pod, kwota_dod, id_pracownika)
```

```
VALUES (4, TO_DATE('2020-02-01', 'YYYY-MM-DD'),2000, 200, 2);

INSERT INTO Wynagrodzenia (id_wynagrodzenia, data_wynagrodzenia,
 kwota_pod, kwota_dod, id_pracownika)
VALUES (5, TO_DATE('2020-02-01', 'YYYY-MM-DD'),3000, 200, 1);

INSERT INTO Wynagrodzenia (id_wynagrodzenia, data_wynagrodzenia,
 kwota_pod, kwota_dod, id_pracownika)
VALUES (6, TO_DATE('2020-02-01', 'YYYY-MM-DD'),2000, 500, 2);

--- Jezyki ---
INSERT INTO Jezyki (id_jezyka, kod_jezyka, nazwa)
VALUES (1, 'en', 'angielski');

INSERT INTO Jezyki (id_jezyka, kod_jezyka, nazwa)
VALUES (2, 'de', 'niemiecki');

INSERT INTO Jezyki (id_jezyka, kod_jezyka, nazwa)
VALUES (3, 'it', 'włoski');

INSERT INTO Jezyki (id_jezyka, kod_jezyka, nazwa)
VALUES (4, 'pl', 'polski');

INSERT INTO Jezyki (id_jezyka, kod_jezyka, nazwa)
VALUES (5, 'cz', 'Czeski');

--- Poziomy_Jezykow ---
INSERT INTO Poziomy_jezykow (id_poziomu_jezyka, kod_poziomu, opis)
VALUES (1, 'A1', 'Poziom A1');
INSERT INTO Poziomy_Jezykow (id_poziomu_jezyka, kod_poziomu, opis)
VALUES (2, 'A2', 'Poziom A2');
INSERT INTO Poziomy_Jezykow (id_poziomu_jezyka, kod_poziomu, opis)
VALUES (3, 'B1', 'Poziom B1');
INSERT INTO Poziomy_Jezykow (id_poziomu_jezyka, kod_poziomu, opis)
VALUES (4, 'B2', 'Poziom B2');
INSERT INTO Poziomy_Jezykow (id_poziomu_jezyka, kod_poziomu, opis)
VALUES (5, 'C1', 'Poziom C1');
INSERT INTO Poziomy_Jezykow (id_poziomu_jezyka, kod_poziomu, opis)
VALUES (6, 'C2', 'Poziom C2');

--- Znajomosc_jezykow ---
INSERT INTO Znajomosc_jezykow (id_pracownika, id_jezyka, id_poziomu_jezyka)
VALUES (1, 3, 3);
INSERT INTO Znajomosc_jezykow (id_pracownika, id_jezyka, id_poziomu_jezyka)
VALUES (2, 1, 2);
INSERT INTO Znajomosc_jezykow (id_pracownika, id_jezyka, id_poziomu_jezyka)
VALUES (2, 2, 3);
INSERT INTO Znajomosc_jezykow (id_pracownika, id_jezyka, id_poziomu_jezyka)
VALUES (2, 3, 6);

--- Atrakcje ---
INSERT INTO Atrakcje (id_atrakcji, nazwa, producent, typ_atrakcji, opis,
 wymagania_wiekowe, status, data_powstania,
                              wysokosc, predkosc, przeciazenie, przepustowosc,
                              ↪  dlugosc_trasy,
                               data_przegladu, waznosc_przegladu, id_sektora)
```

```
VALUES (1, 'Diabelski Młyn', 'Tool', 'karuzela', 'Karuzela obracająca się na poziomej
↪  osi', 12,
 'Otwarte', TO_DATE('1990-02-25', 'YYYY-MM-DD'),
        32, 1, 0, 20, 0, TO_DATE('2018-02-25 ', 'YYYY-MM-DD'),TO_DATE('2021-02-21',
        ↪  'YYYY-MM-DD'), 1);
INSERT INTO Atrakcje (id_atrakcji, nazwa, producent, typ_atrakcji,
 opis, wymagania_wiekowe, status, data_powstania,
                               wysokosc, predkosc, przeciazenie, przepustowosc,
                                dlugosc_trasy, data_przegladu, waznosc_przegladu,
                                ↪  id_sektora)
VALUES (2, 'Diabelski Młyn', 'Tool', 'karuzela', 'Karuzela obracająca się na poziomej
↪  osi',
 12, 'Otwarte', TO_DATE('1990-02-25', 'YYYY-MM-DD'),
        32, 1, 0, 20, 0, TO_DATE('2020-02-21 ', 'YYYY-MM-DD'),TO_DATE('2023-02-21',
        ↪  'YYYY-MM-DD'), 2);
INSERT INTO Atrakcje (id_atrakcji, nazwa, producent, typ_atrakcji, opis,
                                wymagania_wiekowe, status, data_powstania,
                               wysokosc, predkosc, przeciazenie, przepustowosc,
                               ↪  dlugosc_trasy,
                                data_przegladu, waznosc_przegladu, id_sektora)
VALUES (3, 'Kolejka górska - góral', 'Nany', 'kolejka',
 'Kolejka górska z wieloma gwałtownymi zakrętami', 9, 'Otwarte',TO_DATE('1990-02-25',
 ↪  'YYYY-MM-DD'),
        32, 1, 0, 20, 0, TO_DATE('2020-02-21 ', 'YYYY-MM-DD'),TO_DATE('2021-02-21',
        ↪  'YYYY-MM-DD'), 1);
INSERT INTO Atrakcje (id_atrakcji, nazwa, producent, typ_atrakcji, opis,
 wymagania_wiekowe, status, data_powstania,
                               wysokosc, predkosc, przeciazenie,
                                przepustowosc, dlugosc_trasy, data_przegladu,
                                 waznosc_przegladu, id_sektora)
VALUES (4, 'Kolejka górska - góral', 'Nany', 'kolejka',
 'Kolejka górska z wieloma gwałtownymi zakrętami', 9, 'Otwarte', TO_DATE('1990-02-25',
 ↪  'YYYY-MM-DD'),
        32, 1, 0, 20, 0, TO_DATE('2017-02-25 ', 'YYYY-MM-DD'),TO_DATE('2020-02-21',
        ↪  'YYYY-MM-DD'), 1);

--- Pracownicy_atrakcje ---
INSERT INTO Pracownicy_atrakcje (id_pracownika, id_atrakcji)
VALUES (2, 1);
INSERT INTO Pracownicy_atrakcje (id_pracownika, id_atrakcji)
VALUES (2, 3);

--- Bilety Klienta ---
INSERT INTO Bilety_Klienta (id_biletu, id_klienta, data_zakupu, data_aktywacji,
↪  data_waznosci, id_rabatu)
VALUES (1, 1, TO_TIMESTAMP('2018-02-24 20:00:00 ','YYYY-MM-DD HH24:MI:SS'),
        TO_TIMESTAMP('2020-02-25 08:00:00 ','YYYY-MM-DD HH24:MI:SS'),
         TO_TIMESTAMP('2021-02-25 16:00:00','YYYY-MM-DD HH24:MI:SS'),1);
```

Literatura

[1] www.jsystems.pl
[2] www.informatyka.orawskie.pl/?pl_sbd,22