Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska

Bazy danych i Big Data
(projekt cz. I)

Sprawozdanie z projektu bazy danych

Lena Zubik, Kacper Nowakowski

Warszawa, 2021

Spis treści

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

Celem projektu było zaprojektowanie uproszczonej bazy danych odwzorowującej działanie klubu lekkoatletycznego. Zakres projektu obejmuje stworzenie modelu konceptualnego, stworzenie na jego podstawie modelu logicznego i przekształcenie go do modelu fizycznego. Utworzona baza danych będzie oparta o rozwiązania firmy Oracle.

Oprogramownie użyte podczas realizacji projektu:
- Toad Data Modeler
- SQL Developer
- Oracle Database

1.1. Założenia funkcjonalne

Realizowany projekt dotyczy **klubu lekkoatletycznego**. Klub ten zajmuje się głównie przeprowadzaniem **treningów** w swoich placówkach rozsianych po całym kraju. Treningi te są przeprowadzane przez **trenerów** pracujących w danej placówce, od których wymagane jest posiadanie aktualnej **licencji**. Każdy trening wymaga określenia **miejsca treningu** oraz ewentualnie potrzebnego **sprzętu** wraz z jego **ilością**. Przy tworzeniu nowego treningu niezbędny będzie także **przedział czasowy** czy **cena za osobę**. Opcjonalną zaś opcją będzie ustawienie **przedziału wiekowego** uczestników, dla których trening ten jest przewidziany. Treningi te są oferowane **klubowiczom**, którzy muszą wpierw wykupić **członkostwo**. Klub ten zapisuje także informacje o **osiągnięciach** klubowiczów oraz trenerów uwzględniając **zawody** oraz **dyscyplinę**, w której uzyskano dano osiągnięcie.

2. Definicja systemu

2.1. Wyróżnione funkcjonalności systemu

1. Dodawanie, modyfikowanie i usuwanie informacji o klubie
2. Podgląd informacji o klubie
3. Dodawanie, modyfikowanie i usuwanie osób z listy klubowiczów i ich danych
4. Podgląd danych personalnych i kontaktowych klubowiczów
5. Dodawanie, modyfikowanie i usuwanie osób z listy pracowników i ich danych
6. Podgląd danych personalnych i kontaktowych pracowników
7. Dodawanie, modyfikowanie i usuwanie treningów i ich wymagań
8. Podgląd zaplanowanych treningów
9. Dodawanie, modyfikowanie i usuwanie informacji o osiągnięciach trenerów i klubowiczów
10. Podgląd informacji o osiągnięciach trenerów i klubowiczów
11. Dodawanie, modyfikowanie i usuwanie informacji o sprzęcie
12. Podgląd informacji o sprzęcie
13. Dodawanie, modyfikowanie i usuwanie informacji o miejscach treningów
14. Podgląd informacji dotyczących miejsc treningów

2.2. Perspektywy użytkowników

W systemie zostali uwzględnieni tacy użytkownicy jak: administrator, właściciel klubu, pracownicy w tym trenerzy oraz klubowicze w tym zawodnicy z licencją.

2.2.1. Administrator

Posiada dostęp do wszystkich funkcjonalności systemu i modyfikowania bazy danych. Ma uprawnienia administratora bazy danych Oracle.

2.2.2. Właściciel klubu

Posiada dostęp od wszystkich danych przechowywanych w bazie. Może edytować dane pracownika takie, jak na przykład wynagrodzenie.

2.2.3. Pracownik

Ma dostęp do swoich danych. Może modyfikować, dodawać i usuwać informacje o treningach oraz wydarzeniach. Jest odpowiedzialny za obsługę klientów. Może edytować i dodawać nowych klubowiczów. Może dodawać i edytować informacje o osiągnięciach.

2.2.4. Trener

Trener ma takie same uprawnienia co pracownik. Jest zobowiązany do prowadzenia treningów.

2.2.5. Klubowicz

Ma dostęp do swoich danych, może je modyfikować. Ma podgląd na dostępne treningi, może się na nie zapisywać.

3. Model konceptualny

3.1. Definicja zbiorów encji określonych w projekcie (decyzje projektowe)

W projekcie zostały wyróżnione encje:

1. Klub
2. Pracownik
3. Trener (uszczególnienie encji pracownika)
4. Klubowicz
5. Sprzęt
6. Miejsce treningu
7. Trening

3.2. Ustalenie związków między encjami i ich typów

Trening - Klubowicz 0..n - 0..m (związek binarny) Klubowicz zapisuje się na trening. Klubowicz może zapisać się na wiele treningów ale może nie być zapisany na żaden w danym momencie. Na nowo stworzony trening może nie być zapisany jeszcze żaden klubowicz, ale ogólnie może się zapisywać na niego wiele klubowiczów.

Miejsce treningu - Trening 1..n - 0..m (związek binarny) Trening musi odbywać się przynajmniej w jednym miejscu, ale może w wielu. Miejsce treningu nie musi mieć przypisanego żadnego treningu w danym momencie, ale może mieć ich wiele.

Klub - Pracownik 1..1 - 0..m (związek binarny) Pracownik pracuje w jednej i tylko jednej placówce klubu. Klub początkowo może nie mieć przypisanego żadnego pracownika, ale ostatecznie będzie miał przypisanych ich wielu.

Klub - Sprzęt 1..1 - 0..m (związek binarny) Sprzęt musi znajdować się tylko w jednej placówce klubu. Klub początkowo może nie mieć przypisanego żadnego sprzętu, ale ostatecznie będzie miał przypisanego go wiele.

Klub - Miejsce treningu 1..1 - 0..m (związek binarny) Miejsce treningu musi znajdować się tylko w jednej placówce klubu. Klub początkowo może nie mieć przypisanego żadnego miejsca, ale ostatecznie będzie miał przypisanych ich wiele.

Klub - Trening 1..1 - 0..m (związek binarny) Trening musi odbywać się tylko jednej w placówce klubu. Klub może początkowo nie oferować żadnego treningu, ale ostatecznie może oferować ich wiele.

Sprzęt - Trening 0..n - 0..m (związek binarny)Trening nie musi korzystać z żadnego sprzętu, ale może korzystać z wielu sprzętów jednocześnie. Dany sprzęt nie musi być przypisany żadnemu treningowi, ale może być przypisany wielu treningom naraz.

Trener - Trening 1..n - 0..m (związek binarny) Trening musi być prowadzony przynajmniej przez jednego trenera, ale może być prowadzony przez wielu trenerów. Trener w danym momencie nie musi być przypisany na żaden z treningów, ale może być przypisany na wiele treningów naraz.

3.3. Określenie atrybutów i ich dziedzin

Nazwa atrybutu	Dziedzina	Typ danej	Wymagania dostępności	Opis
ID_klubu		SmallInt	Pole wymagane	Unikatowy identyfikator klubu.
Nazwa		Varchar2(35)	Pole wymagane	Nazwa klubu.
Adres		Varchar2(400)	Pole wymagane	Adres klubu, pole segmentowe (miasto, ulica, numer lokalu, kod pocztowy, poczta).
Data założenia		Date	Pole wymagane	Data założenia klubu.
Właściciel		Varchar(300)	Pole wymagane	Właściciel klubu, pole segmentowe (imię, nazwisko), pole wielowartościowe.

Tab. 3.1. Klub

Nazwa atrybutu	Dziedzina	Typ danej	Wymagania dostępności	Opis
ID_pracownika		Integer	Pole wymagane	Unikatowy identyfikator pracownika.
Imię		Varchar2(20)	Pole wymagane	Imię pracownika.
Nazwisko		Varchar2(30)	Pole wymagane	Nazwisko pracownika.
Data urodzenia		Date	Pole wymagane	Data urodzenia pracownika.
PESEL		Character(11)	Pole wymagane	Numer PESEL pracownika.
Płeć	PłecD	Character(1)	Pole wymagane	Płeć pracownika.
Adres		Varchar(400)	Pole wymagane	Adres klubu, pole segmentowe (miasto, ulica, numer lokalu, kod pocztowy).
Stanowisko	StanowiskoD	Varchar(25)	Pole wymagane	Stanowisko pracownika, wyróżnione stanowiska to: administrator, księgowy, fizjoterapeuta, medyk.
Data zatrudnienia		Date	Pole wymagane	Data zatrudnienia pracownika.
Termin umowy		Date	Pole niewymagane	Termin podpisanej umowy.
Numer konta		Character(26)	Pole wymagane	Numer konta pracownika.
Email		Varchar(50)	Pole niewymagane	Email pracownika.
Telefon		Varchar(12)	Pole niewymagane	Numer telefonu pracownika.

Tab. 3.2. Pracownik

Nazwa atrybutu	Dziedzina	Typ danej	Wymagania dostępności	Opis
ID_klubowicza		Integer	Pole wymagane	Unikatowy identyfikator klubowicza.
Imię		Varchar2(20)	Pole wymagane	Imię klubowicza.
Nazwisko		Varchar2(30)	Pole wymagane	Nazwisko klubowicza.
Data urodzenia		Date	Pole wymagane	Data urodzenia klubowicza.
PESEL		Character(11)	Pole wymagane	Numer PESEL klubowicza.
Płeć	PłecD	Character(1)	Pole wymagane	Płeć klubowicza.
Adres		Varchar(400)	Pole wymagane	Adres klubu, pole segmentowe (miasto, ulica, numer lokalu, kod pocztowy).
Numer konta		Character(26)	Pole wymagane	Numer konta klubowicza.
Email		Varchar(50)	Pole niewymagane	Email klubowicza.
Telefon		Varchar(12)	Pole niewymagane	Numer telefonu klubowicza.
Licencja		Vrachar(400)	Pole niewymagane	Licencja klubowicza, pole segmentowe (numer licencji, data wydania, data ważności, organ wystawiający, miasto wydania).
Data dołączenia		Data	Pole wymagane	Data dołączenia klubowicza do klubu.
Data ważności członkostwa		Data	Pole wymagane	Ważność członkostwa danego klubowicza.
Osiągnięcie		Varchar(400)	Pole niewymagane	Osiągnięcie klubowicza, pole segmentowe (rok, tytuł, miejsce zdobycia), pole wielowartościowe.

Tab. 3.3. Klubowicz

Nazwa atrybutu	Dziedzina	Typ danej	Wymagania dostępności	Opis
Licencja		Varchar(400)	Pole wymagane	Unikatowy identyfikator trenera.
Specjalizacja	SpecjalizacjaD	Varchar(400)	Pole wymagane	Specjalizacja trenera, zostały wyróżnione specjalizacje: techniczny, wytrzymałościowy, coach.
Osiągnięcie		Varchar2(400)	Pole niewymagane	Osiągnięcie pracownika, pole segmentowe (rok, tytuł, miejsce zdobycia tytułu), pole wielowartościowe.

Tab. 3.4. Trener

Nazwa atrybutu	Dziedzina	Typ danej	Wymagania dostępności	Opis
ID_treningu		Integer	Pole wymagane	Unikatowy identyfikator treningu.
Nazwa		Varchar(30)	Pole wymagane	Nazwa treningu.
Data rozpoczęcia		Date	Pole wymagane	Czas rozpoczęcia treningu.
Data zakończenia treningu		Date	Pole wymagane	Data zakończenia treningu.
Cena za osobę		Money	Pole wymagane	Cena za osobę.
Max liczba trenujących		SmallInt	Pole wymagane	Maksymalna możliwa liczba trenujących.
Aktualna liczba zapisanych		Integer	Pole wymagane	Aktualna liczba zapisanych klubowiczów.
Wiek minimalny		SmallInt	Pole niewymagane	Minimalny wiek uczestnictwa.
Wiek maksymalny		SmallInt	Pole niewymagane	Maksymalny wiek uczestnictwa.
Opis		Varchar(400)	Pole niewymagane	Opis treningu.

Tab. 3.5. Trening

Nazwa atrybutu	Dziedzina	Typ danej	Wymagania dostępności	Opis
ID_sprzętu		Integer	Pole wymagane	Unikatowy identyfikator sprzętu.
Nazwa		Varchar(20)	Pole wymagane	Nazwa sprzętu.
Ilość		Integer	Pole wymagane	Ilość sprzętu.
Opis		Varchar(400)	Pole niewymagane	Opis sprzętu.

Tab. 3.6. Sprzęt

Nazwa atrybutu	Dziedzina	Typ danej	Wymagania dostępności	Opis
ID_miejsca_treningu		Integer	Pole wymagane	Unikatowy identyfikator miejsca treningu.
Nazwa		Varchar(30)	Pole wymagane	Nazwa treningu.
Maksymalna liczba trenujących		Integer	Pole wymagane	Maksymalna liczba trenujących w danych miejscu treningu.
Opis		Varchar(400)	Pole niewymagane	Opis danego miejsca treningu.

Tab. 3.7. Miejsce Treningu

3.4. Dodatkowe reguły integralnościowe (reguły biznesowe)

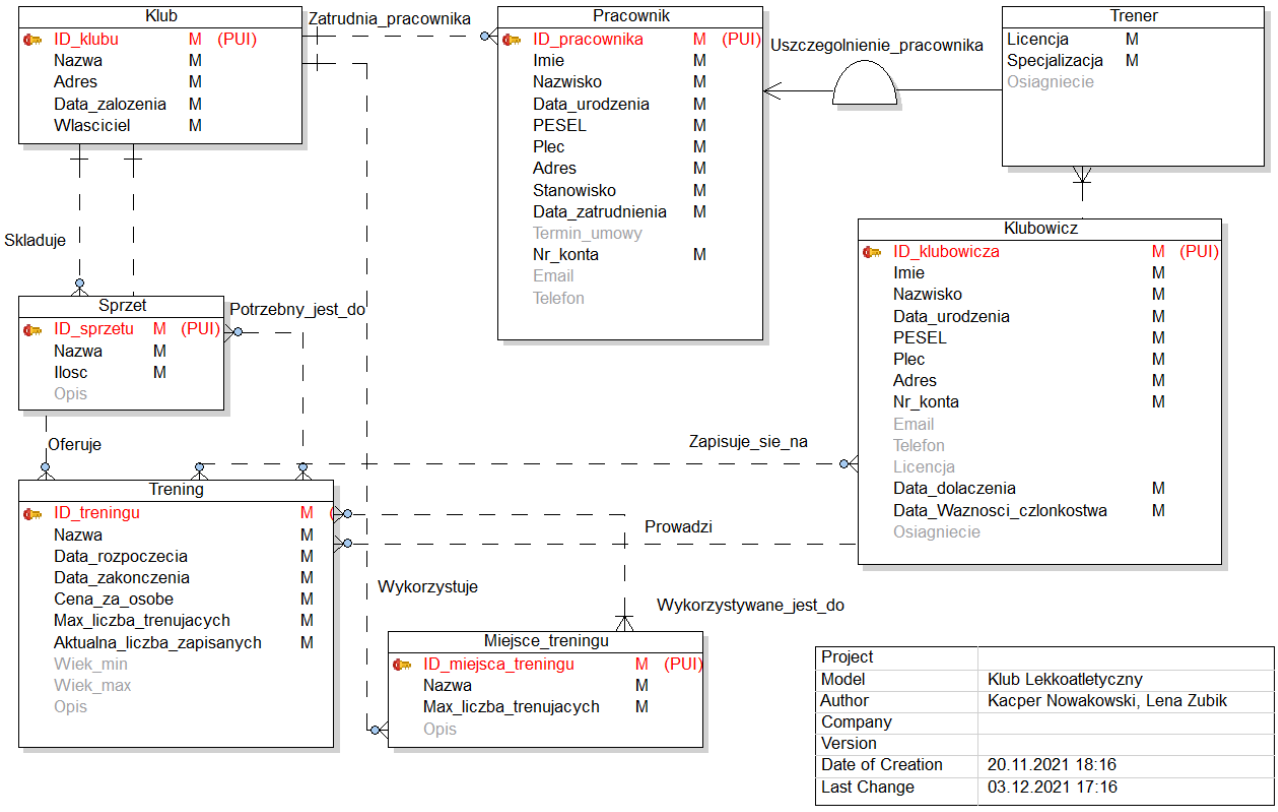
W projekcie wyodrębniono 2 reguły integralnościowe - `Stanowisko` oraz `Specjalizacja`. Reguła `Stanowisko` dotyczy encji **Pracownik** i ogranicza wartości atrybutu *Stanowisko*. Możliwymi wartościami są dostępne dla pracowników stanowiska, które powinny być odgórnie ustalone, np. administrator, księgowy, medyk. Zaś reguła `Specjalizacja` tyczy się encji **Trener** i ogranicza atrybut *Specjalizacja*. Dostępnymi wartościami są możliwe do uzyskania przez trenera specjalizacje takie jak trener techniczny, wytrzymałościowy lub coach (trener od przygotowanie mentalnego zawodników).

3.5. Klucze kandydujące i główne (decyzje projektowe)

Zdecydowano się na użycie kluczy sztucznych - wygenerowanie nr ID.

Encja	Klucze kandydujące	Wyłoniony klucz główny
Klub	ID_klubu	ID_klubu
Pracownik	ID_pracownika, PESEL	ID_pracownika
Trener	ID_trenera, Nr licencji	ID_trenera
Klubowicz	ID_klubowicza, PESEL	ID_klubowicza
Sprzęt	ID_sprzetu	ID_sprzetu
Miejsce treningu	ID_miejsca_treningu	ID_miejsca_treningu
Trening	ID_treningu	ID_treningu

3.6. Schemat ER na poziomie konceptualnym

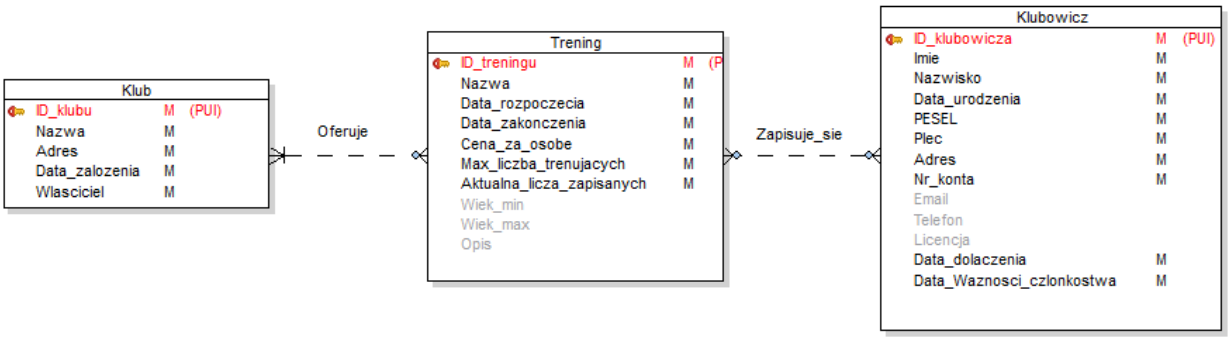


Rys. 3.1. Model konceptualny

3.7. Problem pułapek szczelinowych i wachlarzowych – analiza i przykłady

3.7.1. Pułapka wachlarzowa

Początkowo relacje pomiędzy encjami **Klub-Trening-Klubowicz** wyglądały następująco:

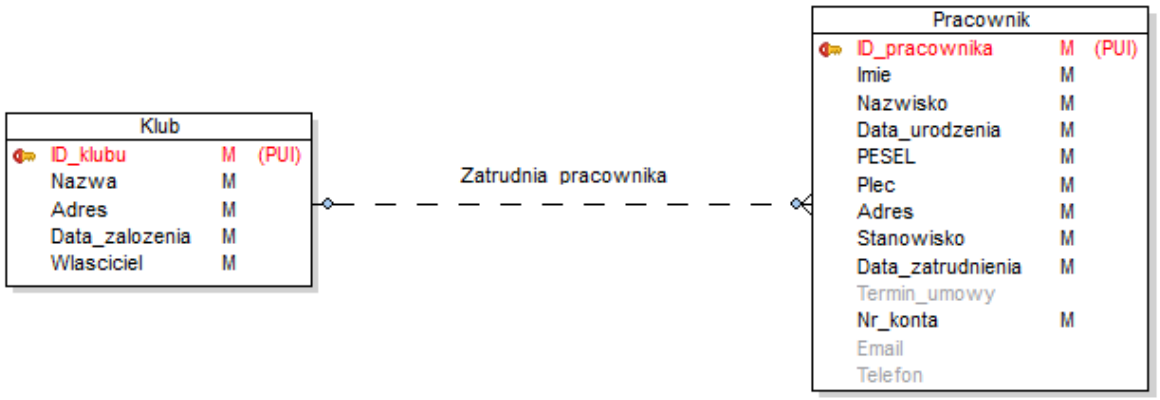


Rys. 3.2. Relacje pomiędzy encjami z problemem wachlarzowym

W takim wypadku **Trening** mógł być prowadzony przez kilka **Klubów**. Jeżeli nastąpi sytuacja, w której dwóch **Klubowiczów** zapisze się na jeden **Trening** prowadzony przez kilka **Klubów** to nie będzie można stwierdzić do jakiego **Klubu** zapisał się dany **Klubowicz**. Rozwiązaniem tego problemu było ograniczenie, że dany **Trening** może być prowadzony przez tylko jeden **Klub**. Dlatego zmodyfikowano relację **Klub-Trening** z 1..n - 0..m do 1..1 - 0..m.

3.7.2. Pułapka szczelinowa

W trakcie projektowania zauważono potencjalne rozwiązanie, które stworzyłoby pułapkę szczelinową. Otóż typ uczestnictwa relacji pomiędzy encjami **Klub** i **Pracownik** mógłby być obustronnie nieobowiązkowy.



Rys. 3.3. Relacje pomiędzy encjami z problemem szczelinowym

Takie ustawienia oznaczają, że **Pracownik** nie musi być przypisany do jakiegokolwiek **Klubu**. Konsekwencją tego jest niemożność określenia do jakiego **Klubu** należy **Pracownik** a więc też w szczególnym przypadku **Trener**, który został przypisany do prowadzenia jakiegoś **Treningu** oferowanego przez jeden **Klub**. Wynika z tego, że tworząc kolejny **Trening** nie można stwierdzić czy podany **Trener** należy do "naszego" **Klubu** co jest sytuacją niepożądaną. Dlatego też ta relacja wymaga obecności przynajmniej jednego **Klubu**.

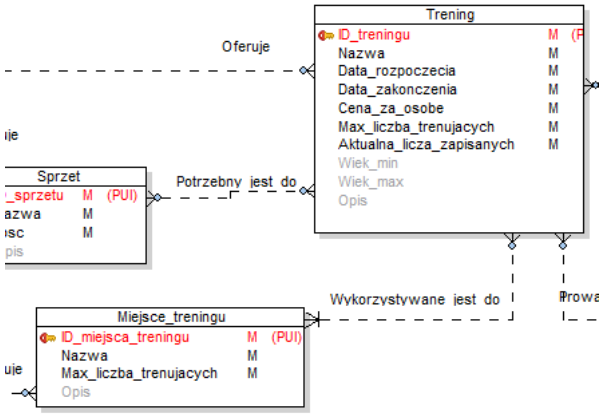
4. Model logiczny

4.1. Charakterystyka modelu relacyjnego

Relacyjny model danych jest obecnie najbardziej popularnym modelem używanym w systemach baz danych. Jest on oparty o oparty jest o algebrę relacji.Podstawowe elementy modelu to relacje i więzi. Podstawową strukturą danych jest relacja będąca podzbiorem iloczynu kartezjańskiego dwóch wybranych zbiorów reprezentujących dopuszczalne wartości. Dzięki wykorzystaniu relacyjnego modelu danych zyskujemy uporządkowaną strukturę, w której łatwiej jest rozwiązywać problemy semantyki, spójności i redundancji danych, co z punktu widzenia projektanta i dewelopera korzystającego z bazy danych jest szczególnie istotne.

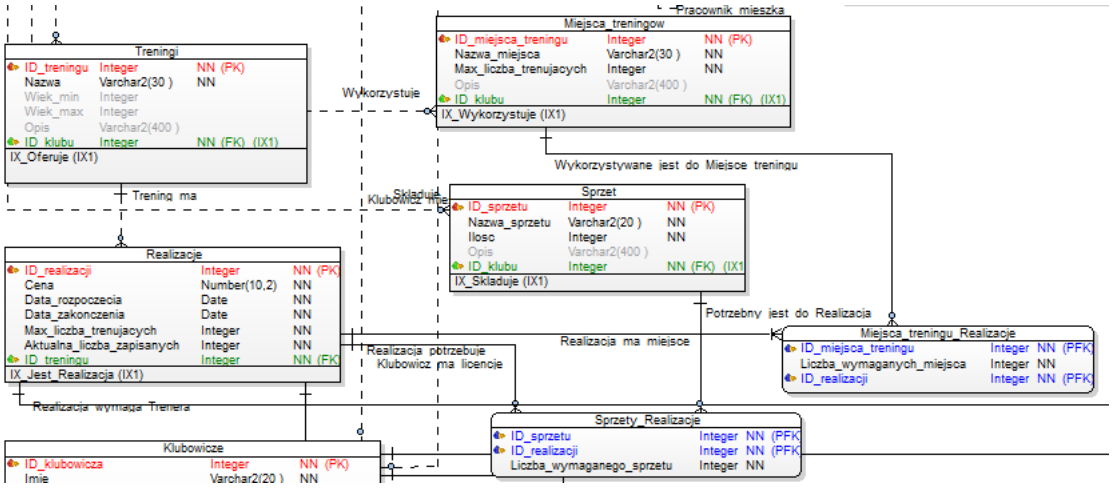
4.2. Usunięcie właściwości niekompatybilnych z modelem relacyjnym - przykłady

Właściwości niekompatybilne z modelem relacyjnym objawiają się w postaci związków wiele do wielu. Takie związku w modelu konceptualnym pojawiają się pomiędzy encjami **Trening Klubowicz**, **Trening Trener**, **Trening Miejsce Treningu** oraz **Trening Sprzęt**. Takie związki w modelu logicznym można zrealizować poprzez wykorzystanie tabeli bridge'ującej, która musi mieć atrybuty takie jak klucze główne dołączonych encji oraz opcjonalne atrybuty uszczegóławiające związek. Rozwiązanie przykładowej niekompatybilności zostało zaprezentowane poniżej.



Rys. 4.1. Związek Trening Sprzęt w modelu konceptualnym

Związek ten w modelu logicznym został przekształcony do postaci z rysunku 4.2.



Rys. 4.2. Związek Trening Sprzęt poprzez relację Realizacji w modelu logicznym

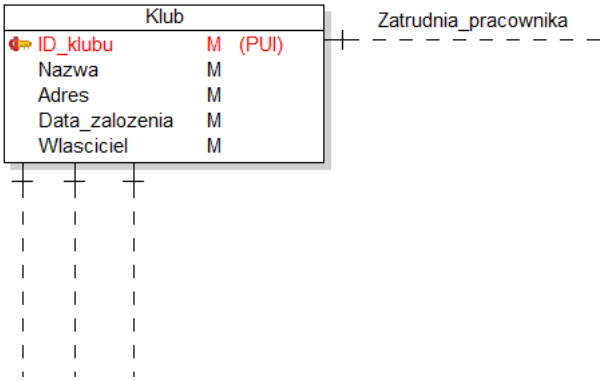
W modelu logicznym rozszerzono ten związek o informację o ilości wymaganego miejsca przez konkretny trening, co pozwoli na dokładniejszą kontrolę organizacji treningów klubie

4.3. Proces normalizacji - analiza i przykłady

Normalizacja baz danych odbywa się poprzez dostosowanie struktury bazy danych do postaci normalnej. Postać normalna jest to postać bazy danych w której nie występuje redundancja danych czyli ich powtarzalność oraz zależność. Występują trzy podstawowe postaci normalne:
- pierwsza postać normalna
- druga postać normalna
- trzecia postać normalna

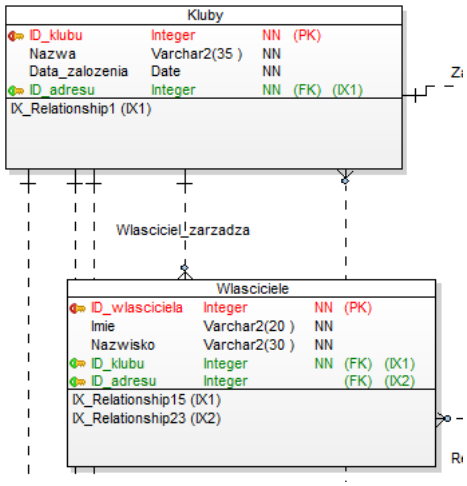
4.3.1. Pierwsza postać normalna

Relacja jest w pierwszej postaci normalnej, jeśli każda wartość atrybutu w każdej krotce tej relacji jest wartością atomową oraz jeśli w relacji nie ma powtarzających się grup.



Rys. 4.3. Relacja Klub przed normalizacją

Relacja Kub z rysunku 4.3 ma atrybut Właściciel, który nie jest polem atomowym. Właściciel charakteryzuje imię, nazwisko i adres.



Rys. 4.4. Relacja Klub po normalizacji

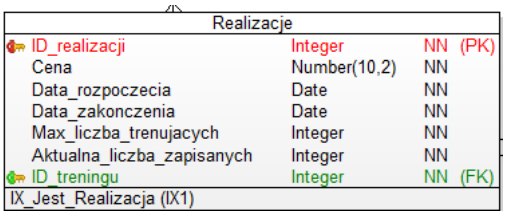
Dokonano dekompozycji widocznej na rysunku 4.4. Wydzielono relację Właściciel i jej atrybut adres także został wydzielony do osobnej relacji (byłaby to grupa powtarzająca się), dlatego Właściciel zamiast atrybutu adres posiada ID adresu.

4.3.2. Druga postać normalna

Aby uzyskać drugą postać normalną relacji, wymagane jest osiągnięcie pierwszej postaci normalnej oraz zapewnienie, że każdy atrybut tej relacji niewchodzący w skład żadnego klucza kandydującego jest w pełni funkcyjnie zależny od wszystkich kluczy kandydujących tej relacji. Oznacza to, że relacja ma drugą postać normalną jeżeli wszystkie klucze potencjalne są kluczami prostymi. W projekcie każda relacja posiada swój własny klucz sztuczny, który jest prosty co implikuje, że jest w 2PN.

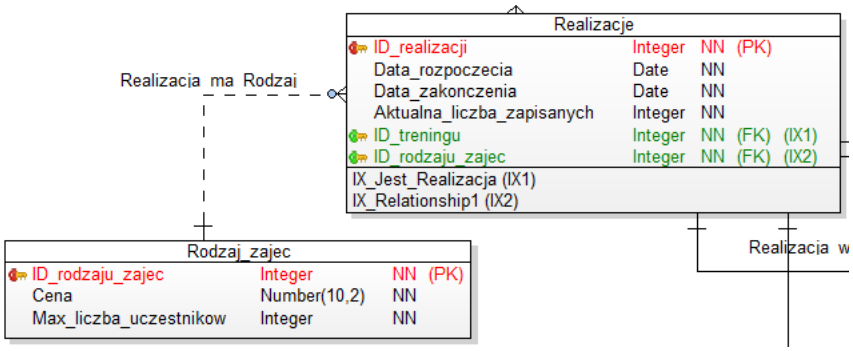
4.3.3. Trzecia postać normalna

Osiągnięcie trzeciej postaci normalnej jest możliwe, gdy relacja jest w drugiej postaci normalnej i każdy jej atrybut niewchodzący w skład żadnego klucza kandydującego nie jest przechodnio (tranzytywnie) funkcyjnie zależny od żadnego klucza kandydującego tej relacji. W zbudowanym modelu konceptualnym pojawiła się zależność tranzytywnie przechodnia w tabeli **Realizacje** między ceną danej realizacji i maksymalną liczą uczestników, co widoczne jest na rysunku 4.5.



Rys. 4.5. Relacja Realizacja przed normalizacją

Gdy liczba uczestników jest mniejsza, zajęcia kosztują więcej. Szczególnym przypadkiem zajęć z małą ilością uczestników są zajęcia indywidualne, za które jest określona stawka. Zależność ta została przeniesiona do realizacji **Rodzaj**, co zostało przedstawione na rysunku 4.6.



Rys. 4.6. Relacja Realizacja po normalizacji

4.4. Więzy integralności

Więzy integralności służą ochronie przed utraceniem spójności danych w trakcie ich modyfikacji. W projekcie do każdej stworzonej relacji, oprócz relacji **Trener** (jest uszczegółowieniem relacji **Pracownik**), wykorzystano klucze sztuczne. Podjęto decyzję projektową o wykorzystaniu dodanych kluczy sztucznych jako **kluczy głównych**. Gwarantują one unikalność rekordów. W projekcie pojawiają się **klucze obce** ze względu na istniejące związki pomiędzy relacjami, dzięki nim mamy pewność, że rekord w tabeli "parent" ma swojego odpowiednika w tabeli "child". Kolejnym więzem chroniącym spójność danych jest określenie niektórych atrybutów relacji jako "mandatory". Dzięki temu mamy pewność, że użytkownik nie może wprowadzić wybrakowanego rekordu, co mogłoby spowodować końcowy brak spójności danych przechowywanych w bazie.

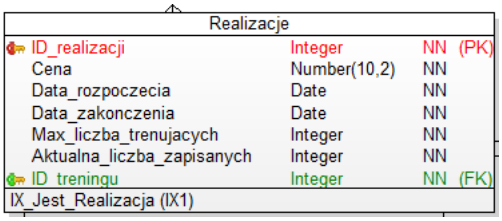
4.5. Proces denormalizacji - analiza i przykłady

4.5.1. Łączenie specjalizacji

Łączenie specjalizacji jest dobrym pomysłem, gdy wyszczególniono mało (a zwłaszcza jedną) specjalizację encji. W takim wypadku istnieje możliwość wciągnięcia atrybutów specjalizacji do realizacji, z której ta specjalizacja wynika. W takim wypadku wciągnięte do relacji atrybuty stają się nieobowiązkowe. Takiego zabiegu nie zastosowano w przypadku wyszczególnienia relacji **Trenera** z relacji **Pracownika**.

4.5.2. Wprowadzenie powtarzających się grup

Wprowadzenie powtarzających się grup do danej relacji jest zabiegiem pozwalającym na łatwiejsze odpytanie bazy danych o konkretne informacje (długość i złożoność takiego zapytania jest znacznie mniejsza). Tak postąpiono w przypadku zależności ceny realizacji i maksymalnej ilości uczestników, mogących brać w niej udział, w relacji **Realizacja**.

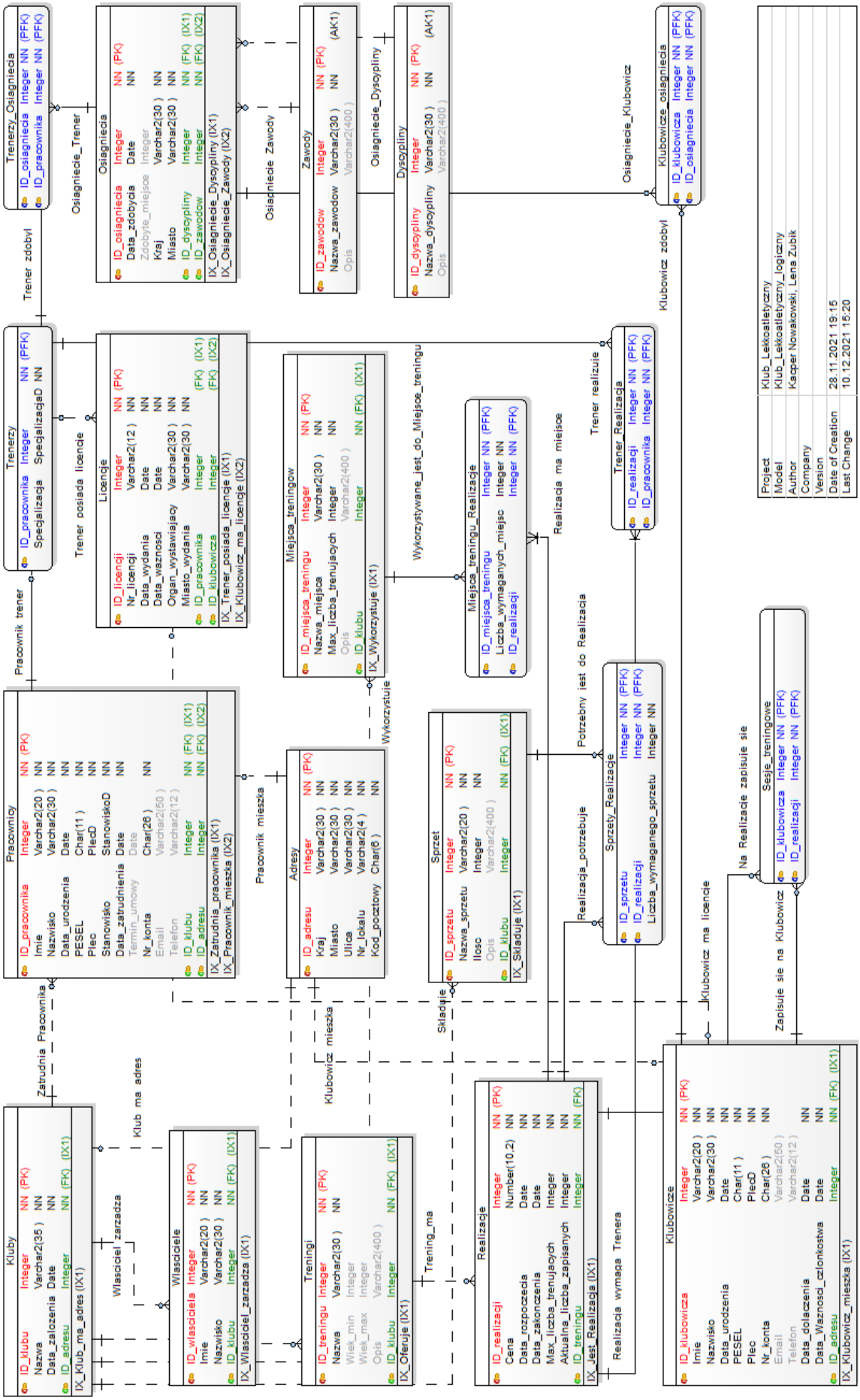


Rys. 4.7. Relacja Realizacja po denormalizacji

4.5.3. Tworzenie tabel skrótowych (agregatów)

Uznano, że żadna przewidywana operacja nie wprowadza konieczności większej agregacji danych, dlatego ze względu na integralność danych, nie podjęto takiego kroku.

4.6. Schemat ER na poziomie modelu logicznego



Rys. 4.8. Model logiczny

5. Faza fizyczna

5.1. Projekt transakcji i weryfikacja ich wykonalności

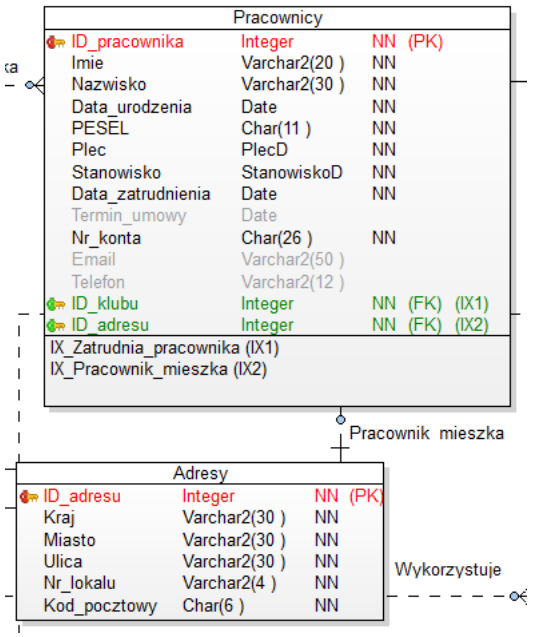
Poniżej zamieszczono przewidziane transakcje, które zostały pogrupowane wg obszarów wyróżnionych w punkcie 2.1 z uwzględnieniem warunków ich realizacji.

Obszar	Transakcja	Zasób	Zawsze możliwe?	Uwagi
Obszar administratora	Zdefiniowanie klubu	Kluby	Nie	Musi istnieć adres klubu.
	Dodawanie informacji o klubie	Kluby	Tak	Brak
	Modyfikowanie informacji o klubie	Kluby	Tak	Brak
	Usuwanie informacji o klubie	Kluby	Tak	Brak
	Podgląd informacji o klubie	Kluby	Tak	Brak
	Dodawanie pracownika	Pracownicy	Nie	Musi istnieć adres pracownika i klub, do którego przynależy.
Obszar pracownika	Modyfikacja informacji o pracowniku	Klubowicze	Tak	Brak
	Usuwanie informacji o pracowniku	Klubowicze	Tak	Brak
	Podgląd informacji o pracowniku	Klubowicze	Tak	Brak
	Dodawanie klubowicza	Klubowicze	Nie	Musi istnieć adres klubowicza.
	Modyfikacja danych klubowicza	Klubowicze	Tak	Brak
	Usuwanie klubowicza	Klubowicze	Tak	Brak
	Dodawanie informacji o licencjach trenerów	Licencje	Tak	Brak
	Podgląd informacji o licencjach	Licencje	Tak	Brak
	Zarządzanie treningami	Treningi		
Obszar pracownika – trenera (dodatkowe funkcje)	Dodawanie informacji o licencjach zawodników	Licencje	Tak	Brak
	Modyfikowanie informacji o licencjach zawodników	Licencje	Tak	Brak
	Usuwanie informacji o licencjach zawodników	Licencje	Tak	Brak
Obszar klubowicza	Dodawanie swoich danych	Klubowicze	Tak	Założono, że zostanie założone konto przez pracownika a dodanie danych to uzupełnienie informacji.
	Modyfikacja swoich danych	Klubowicze	Tak	Brak
	Podgląd swoich danych	Klubowicze	Tak	Brak
	Dopisywanie się do konkretnych realizacji treningu	Treningi	Nie	Musi istnieć realizacja treningu z przypisanym trenerem.

5.2. Strojenie bazy danych – dobór indeksów

Indeksy to dodatkowa informacja dla tabeli, która przechowuje rekordy innej tabeli, połączonej związkiem, posortowane rosnąco według pewnej, ustalonej kolumny. Posortowaniu najczęściej podlegają tabele z dużą liczbą rekordów dzięki czemu nie jest dokonywany pełny skan tabeli podczas odpytywania o rekordy z pewną wartością jakiegoś atrybutu.

Powołując się na źródła w fazie projektowania stwierdzono, że indeksowane zostaną klucze obce. Nie indeksowano żadnych innych kolumn gdyż nie zostało to uznane za niezbędne a każde nadmiarowe indeksowanie może spowolnić działanie operacji DML, ponieważ indeksy muszą być aktualizowane za każdym razem, gdy tabela ulega nawet najmniejszej modyfikacji. W projekcie zostały zdefiniowane indeksy zaproponowane przez narzędzie Toad Modeler.



Rys. 5.1. Przykład indeksów Zatrudnia_pracownika i Pracownik_mieszka

Indeksy bywają pomocne, gdy w tabeli **Pracownicy** jest wiele rekordów i potrzebne jest otrzymanie ich adresów np. chcąc otrzymać pracowników, którzy mieszkają w konkretnej okolicy. Odwołując się do odpowiedniej grupy indeksów można odnaleźć konkretne krotki z tabeli Pracownicy, które spełniają zadane zapytanie. Użycie indeksów jest szczególnie efektywne, gdy danych jest dużo. Klub sportowy nie jest małą placówką, więc zdefiniowanie takiego indeksu przyśpieszy operacje wyszukiwania adresów. Dobrze jest pamiętać, że dodawanie indeksów w przesadnej ilości ma swoje negatywne konsekwencje (mogą ostatecznie zajmować dużo pamięci) i należy zastanowić się, czy dodanie nowego jest uzasadnione.

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

5.3.1. Dodanie kilku adresów

```
INSERT ALL
    INTO adresy (kraj, miasto, ulica, nr_lokalu, kod_pocztowy) VALUES
    ('Polska', 'Warszawa', 'Wojaka', '35', '03-499')
    INTO adresy (kraj, miasto, ulica, nr_lokalu, kod_pocztowy) VALUES
    ('Polska', 'Gdańsk', 'Toruńska', '15', '04-499')
    INTO adresy (kraj, miasto, ulica, nr_lokalu, kod_pocztowy) VALUES
    ('Polska', 'Kraków', 'Wawelska', '99', '05-499')
```

	ID_ADRESU	KRAJ	MIASTO	ULICA	NR_LOKALU	KOD_POCZTOWY
1	1	Polska	Warszawa	Postępu	1206	02-499
2	2	Polska	Warszawa	Wojaka	35	03-499
3	3	Polska	Gdańsk	Toruńska	15	04-499
4	4	Polska	Kraków	Wawelska	99	05-499

Rys. 5.2. Efekt po dodaniu kilku adresów komendą **INSERT**

5.3.2. Treningi oferowane przez konkretny klub posortowane alfabetycznie po ich nazwach

	ID_TRENINGU	NAZWA	WIEK_MIN	WIEK_MAX	OPIS	ID_KLUBU
1	1	Dla poczatkujacych	(null)	(null)	(null)	1
2	2	Dla zaawansowanych	(null)	(null)	(null)	1
3	3	Dla amatorów	(null)	(null)	(null)	1
4	4	Personalny	(null)	(null)	(null)	1
5	5	Coaching	(null)	(null)	(null)	2

Rys. 5.3. Przykładowe uzupełnienie relacji **Treningi**

```
SELECT t.ID_TRENINGU, t.NAZWA, k.NAZWA FROM kluby k INNER JOIN treningi t
    ON k.ID_KLUBU=t.ID_KLUBU
    WHERE k.NAZWA='DZIK' ORDER BY t.NAZWA;
```

	ID_TRENINGU	NAZWA	NAZWA_1
1	3	Dla amatorów	DZIK
2	1	Dla poczatkujacych	DZIK
3	2	Dla zaawansowanych	DZIK
4	4	Personalny	DZIK

Rys. 5.4. Odpowiedź na zapytanie

5.4. Skrypt SQL zakładający bazę danych

```
/*
Created: 28.11.2021
Modified: 08.12.2021
Project: Klub_Lekkoatletyczny
Model: Klub_Lekkoatletyczny_logiczny
Author: Kacper Nowakowski, Lena Zubik
Database: Oracle 12c Release 2
*/


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

CREATE SEQUENCE KlubSeq1
```

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

CREATE SEQUENCE WlascicielSeq1
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE TreningSeq1
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE RealizacjaSeq1
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE KlubowiczSeq1
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE PracownikSeq1
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE AdresSeq1
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
```

```
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE SprzetSeq1
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE LicencjaSeq1
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE Miejsce_treningowSeq1
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE DyscyplinaSeq1
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE OsiagniecieSeq1
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/

CREATE SEQUENCE ZawodySeq1
 INCREMENT BY 1
 START WITH 1
 NOMAXVALUE
 NOMINVALUE
 CACHE 20
/
```

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

-- Table Kluby

CREATE TABLE Kluby(
  ID_klubu Integer NOT NULL,
  Nazwa Varchar2(35 ) NOT NULL,
  Data_zalozenia Date NOT NULL,
  ID_adresu Integer NOT NULL
)
/

-- Create indexes for table Kluby

CREATE INDEX IX_Klub_ma_adres ON Kluby (ID_adresu)
/

-- Add keys for table Kluby

ALTER TABLE Kluby ADD CONSTRAINT KlubPK PRIMARY KEY (ID_klubu)
/

-- Table Pracownicy

CREATE TABLE Pracownicy(
  ID_pracownika Integer NOT NULL,
  Imie Varchar2(20 ) NOT NULL,
  Nazwisko Varchar2(30 ) NOT NULL,
  Data_urodzenia Date NOT NULL,
  PESEL Char(11 ) NOT NULL,
  Plec Char(1 ) NOT NULL
        CHECK (Plec IN ('K', 'M')),
  Stanowisko Varchar2(25 ) NOT NULL
        CHECK (Stanowisko IN ('administrator', 'księgowy', 'fizjoterapeuta', 'medyk')),
  Data_zatrudnienia Date NOT NULL,
  Termin_umowy Date,
  Nr_konta Char(26 ) NOT NULL,
  Email Varchar2(50 ),
  Telefon Varchar2(12 ),
  ID_klubu Integer NOT NULL,
  ID_adresu Integer NOT NULL
)
/

-- Create indexes for table Pracownicy

CREATE INDEX IX_Zatrudnia_pracownika ON Pracownicy (ID_klubu)
/

CREATE INDEX IX_Pracownik_mieszka ON Pracownicy (ID_adresu)
```

```
/

-- Add keys for table Pracownicy

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

-- Table Klubowicze

CREATE TABLE Klubowicze(
  ID_klubowicza Integer NOT NULL,
  Imie Varchar2(20 ) NOT NULL,
  Nazwisko Varchar2(30 ) NOT NULL,
  Data_urodzenia Date NOT NULL,
  PESEL Char(11 ) NOT NULL,
  Plec Char(1 ) NOT NULL
        CHECK (Plec IN ('K', 'M')),
  Nr_konta Char(26 ) NOT NULL,
  Email Varchar2(50 ),
  Telefon Varchar2(12 ),
  Data_dolaczenia Date NOT NULL,
  Data_Waznosci_czlonkostwa Date NOT NULL,
  ID_adresu Integer NOT NULL
)
/

-- Create indexes for table Klubowicze

CREATE INDEX IX_Klubowicz_mieszka ON Klubowicze (ID_adresu)
/

-- Add keys for table Klubowicze

ALTER TABLE Klubowicze ADD CONSTRAINT KlubowiczPK PRIMARY KEY (ID_klubowicza)
/

-- Table Sprzet

CREATE TABLE Sprzet(
  ID_sprzetu Integer NOT NULL,
  Nazwa_sprzetu Varchar2(20 ) NOT NULL,
  Ilosc Integer NOT NULL,
  Opis Varchar2(400 ),
  ID_klubu Integer NOT NULL
)
/

-- Create indexes for table Sprzet

CREATE INDEX IX_Skladuje ON Sprzet (ID_klubu)
/
```

```
-- Add keys for table Sprzet

ALTER TABLE Sprzet ADD CONSTRAINT SprzetPK PRIMARY KEY (ID_sprzetu)
/

-- Table Miejsca_treningow

CREATE TABLE Miejsca_treningow(
  ID_miejsca_treningu Integer NOT NULL,
  Nazwa_miejsca Varchar2(30 ) NOT NULL,
  Max_liczba_trenujacych Integer NOT NULL,
  Opis Varchar2(400 ),
  ID_klubu Integer NOT NULL
)
/

-- Create indexes for table Miejsca_treningow

CREATE INDEX IX_Wykorzystuje ON Miejsca_treningow (ID_klubu)
/

-- Add keys for table Miejsca_treningow

ALTER TABLE Miejsca_treningow ADD CONSTRAINT Miejsce_treninguPK PRIMARY KEY (ID_miejsca_
treningu)
/

-- Table Treningi

CREATE TABLE Treningi(
  ID_treningu Integer NOT NULL,
  Nazwa Varchar2(30 ) NOT NULL,
  Wiek_min Integer,
  Wiek_max Integer,
  Opis Varchar2(400 ),
  ID_klubu Integer NOT NULL
)
/

-- Create indexes for table Treningi

CREATE INDEX IX_Oferuje ON Treningi (ID_klubu)
/

-- Add keys for table Treningi

ALTER TABLE Treningi ADD CONSTRAINT TreningPK PRIMARY KEY (ID_treningu)
/

-- Table and Columns comments section
```

```
COMMENT ON COLUMN Treningi.Nazwa IS 'Nazwa treningu.'
/

-- Table Sesje_treningowe

CREATE TABLE Sesje_treningowe(
  ID_klubowicza Integer NOT NULL,
  ID_realizacji Integer NOT NULL
)
/

-- Table Trener_Realizacja

CREATE TABLE Trener_Realizacja(
  ID_realizacji Integer NOT NULL,
  ID_pracownika Integer NOT NULL
)
/

-- Table Miejsca_treningu_Realizacje

CREATE TABLE Miejsca_treningu_Realizacje(
  ID_miejsca_treningu Integer NOT NULL,
  Liczba_wymaganych_miejsc Integer NOT NULL,
  ID_realizacji Integer NOT NULL
)
/

-- Table and Columns comments section

COMMENT ON COLUMN Miejsca_treningu_Realizacje.Liczba_wymaganych_miejsc IS 'Liczba
wymaganych miejsc dla danego terningu.'
/

-- Table Sprzety_Realizacje

CREATE TABLE Sprzety_Realizacje(
  ID_sprzetu Integer NOT NULL,
  ID_realizacji Integer NOT NULL,
  Liczba_wymaganego_sprzetu Integer NOT NULL
)
/

-- Table and Columns comments section

COMMENT ON COLUMN Sprzety_Realizacje.Liczba_wymaganego_sprzetu IS 'Liczba
wymaganego sprzętu.'
/

-- Table Wlasciciele
```

```
CREATE TABLE Wlasciciele(
  ID_wlasciciela Integer NOT NULL,
  Imie Varchar2(20 ) NOT NULL,
  Nazwisko Varchar2(30 ) NOT NULL,
  ID_klubu Integer NOT NULL
)
/

-- Create indexes for table Wlasciciele

CREATE INDEX IX_Wlasciciel_zarzadza ON Wlasciciele (ID_klubu)
/

-- Add keys for table Wlasciciele

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

-- Table and Columns comments section

COMMENT ON TABLE Wlasciciele IS 'Encja właścicela.'
/
COMMENT ON COLUMN Wlasciciele.ID_wlasciciela IS 'Unikatowy identyfikator właściciela.'
/
COMMENT ON COLUMN Wlasciciele.Imie IS 'Imię właściciela.'
/
COMMENT ON COLUMN Wlasciciele.Nazwisko IS 'Nazwisko właściciela.'
/

-- Table Adresy

CREATE TABLE Adresy(
  ID_adresu Integer NOT NULL,
  Kraj Varchar2(30 ) NOT NULL,
  Miasto Varchar2(30 ) NOT NULL,
  Ulica Varchar2(30 ) NOT NULL,
  Nr_lokalu Varchar2(4 ) NOT NULL,
  Kod_pocztowy 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 TABLE Adresy IS 'Encja adres.'
/
```

```
COMMENT ON COLUMN Adresy.ID_adresu IS 'Unikatowy idnetyfikator adresu.'
/
COMMENT ON COLUMN Adresy.Kraj IS 'Kraj zamieszkania.'
/
COMMENT ON COLUMN Adresy.Miasto IS 'Miato w adresie.'
/
COMMENT ON COLUMN Adresy.Ulica IS 'Ulica adresu.'
/
COMMENT ON COLUMN Adresy.Nr_lokalu IS 'Numer lokalu z adresu.'
/
COMMENT ON COLUMN Adresy.Kod_pocztowy IS 'Kod pocztowy adresu.'
/

-- Table Realizacje

CREATE TABLE Realizacje(
  ID_realizacji Integer NOT NULL,
  Cena Number(10,2) NOT NULL,
  Data_rozpoczecia Date NOT NULL,
  Data_zakonczenia Date NOT NULL,
  Max_liczba_trenujacych Integer NOT NULL,
  Aktualna_liczba_zapisanych Integer NOT NULL,
  ID_treningu Integer NOT NULL
)
/

-- Create indexes for table Realizacje

CREATE INDEX IX_Jest_Realizacja ON Realizacje (ID_treningu)
/

-- Add keys for table Realizacje

ALTER TABLE Realizacje ADD CONSTRAINT PK_Realizacje PRIMARY KEY (ID_realizacji)
/

-- Table and Columns comments section

COMMENT ON TABLE Realizacje IS 'Encja realizacji treningu.'
/
COMMENT ON COLUMN Realizacje.ID_realizacji IS 'Unikatowy identyfikator realizacji treningu.'
/
COMMENT ON COLUMN Realizacje.Cena IS 'Cena za osobę danej realizacji.'
/
COMMENT ON COLUMN Realizacje.Data_rozpoczecia IS 'Data rozpoczęcia realizacji danego
treningu.'
/
COMMENT ON COLUMN Realizacje.Data_zakonczenia IS 'Data zakończenia realizacji danego
treningu.'
/
COMMENT ON COLUMN Realizacje.Max_liczba_trenujacych IS 'Maksymalna liczba trenujących.'
```

```
/
COMMENT ON COLUMN Realizacje.Aktualna_liczba_zapisanych IS 'Aktualna liczba zapisanych
uczestników danej realizacji.'
/

-- Table Licencje

CREATE TABLE Licencje(
  ID_licencji Integer NOT NULL,
  Nr_licencji Varchar2(12 ) NOT NULL,
  Data_wydania Date NOT NULL,
  Data_waznosci Date NOT NULL,
  Organ_wystawiajacy Varchar2(30 ) NOT NULL,
  Miasto_wydania Varchar2(30 ) NOT NULL,
  ID_pracownika Integer,
  ID_klubowicza Integer
)
/

-- Create indexes for table Licencje

CREATE INDEX IX_Trener_posiada_licencje ON Licencje (ID_pracownika)
/

CREATE INDEX IX_Klubowicz_ma_licencje ON Licencje (ID_klubowicza)
/

-- Add keys for table Licencje

ALTER TABLE Licencje ADD CONSTRAINT PK_Licencje PRIMARY KEY (ID_licencji)
/

-- Table and Columns comments section

COMMENT ON TABLE Licencje IS 'Encja licencji.'
/
COMMENT ON COLUMN Licencje.ID_licencji IS 'Unikatowy identyfikator licencji.'
/
COMMENT ON COLUMN Licencje.Nr_licencji IS 'Numer licencji.'
/
COMMENT ON COLUMN Licencje.Data_wydania IS 'Data wydania.'
/
COMMENT ON COLUMN Licencje.Data_waznosci IS 'Data ważności.'
/
COMMENT ON COLUMN Licencje.Organ_wystawiajacy IS 'Organ wystawiający licencji.'
/
COMMENT ON COLUMN Licencje.Miasto_wydania IS 'Miasto wydania licencji.'
/

-- Table Dyscypliny
```

```
CREATE TABLE Dyscypliny(
  ID_dyscypliny Integer NOT NULL,
  Nazwa_dyscypliny Varchar2(30 ) NOT NULL,
  Opis Varchar2(400 )
)
/

-- Add keys for table Dyscypliny

ALTER TABLE Dyscypliny ADD CONSTRAINT PK_Dyscypliny PRIMARY KEY (ID_dyscypliny)
/

ALTER TABLE Dyscypliny ADD CONSTRAINT Nazwa_dyscypliny UNIQUE (Nazwa_dyscypliny)
/

-- Table and Columns comments section

COMMENT ON TABLE Dyscypliny IS 'Encja dyscypliny.'
/
COMMENT ON COLUMN Dyscypliny.ID_dyscypliny IS 'Unikatowy identyfikator dyscypliny.'
/
COMMENT ON COLUMN Dyscypliny.Nazwa_dyscypliny IS 'Nazwa dyscypliny.'
/

-- Table Osiagniecia

CREATE TABLE Osiagniecia(
  ID_osiagniecia Integer NOT NULL,
  Data_zdobycia Date NOT NULL,
  Zdobyte_miejsce Integer,
  Kraj Varchar2(30 ) NOT NULL,
  Miasto Varchar2(30 ) NOT NULL,
  ID_dyscypliny Integer NOT NULL,
  ID_zawodow Integer NOT NULL
)
/

-- Create indexes for table Osiagniecia

CREATE INDEX IX_Osiagniecie_Dyscypliny ON Osiagniecia (ID_dyscypliny)
/

CREATE INDEX IX_Osiagniecie_Zawody ON Osiagniecia (ID_zawodow)
/

-- Add keys for table Osiagniecia

ALTER TABLE Osiagniecia ADD CONSTRAINT PK_Osiagniecia PRIMARY KEY (ID_osiagniecia)
/

-- Table and Columns comments section
```

```
COMMENT ON TABLE Osiagniecia IS 'Encja osiągnięcia.'
/
COMMENT ON COLUMN Osiagniecia.ID_osiagniecia IS 'Uniktowy identyfikator osiągnięcia.'
/
COMMENT ON COLUMN Osiagniecia.Data_zdobycia IS 'Data zdobycia osiągnięcia.'
/
COMMENT ON COLUMN Osiagniecia.Zdobyte_miejsce IS 'Zdobyte miejsce danego osiągnięcia.'
/
COMMENT ON COLUMN Osiagniecia.Kraj IS 'Krak, w którym zdobyto osiągnięcie.'
/
COMMENT ON COLUMN Osiagniecia.Miasto IS 'Miasto, w którym zdobyto osiągnięcie.'
/

-- Table Trenerzy_Osiagniecia

CREATE TABLE Trenerzy_Osiagniecia(
  ID_osiagniecia Integer NOT NULL,
  ID_pracownika Integer NOT NULL
)
/

-- Add keys for table Trenerzy_Osiagniecia

ALTER TABLE Trenerzy_Osiagniecia ADD CONSTRAINT PK_Trenerzy_Osiagniecia PRIMARY KEY
(ID_osiagniecia,ID_pracownika)
/

-- Table Klubowicze_osiagniecia

CREATE TABLE Klubowicze_osiagniecia(
  ID_klubowicza Integer NOT NULL,
  ID_osiagniecia Integer NOT NULL
)
/

-- Add keys for table Klubowicze_osiagniecia

ALTER TABLE Klubowicze_osiagniecia ADD CONSTRAINT PK_Klubowicze_osiagniecia PRIMARY KEY
(ID_klubowicza,ID_osiagniecia)
/

-- Table Zawody

CREATE TABLE Zawody(
  ID_zawodow Integer NOT NULL,
  Nazwa_zawodow Varchar2(30 ) NOT NULL,
  Opis Varchar2(400 )
)
/
```

```
-- Add keys for table Zawody

ALTER TABLE Zawody ADD CONSTRAINT PK_Zawody PRIMARY KEY (ID_zawodow)
/

ALTER TABLE Zawody ADD CONSTRAINT Nazwa_zawodow UNIQUE (Nazwa_zawodow)
/

-- Table and Columns comments section

COMMENT ON TABLE Zawody IS 'Encja zwodów.'
/
COMMENT ON COLUMN Zawody.ID_zawodow IS 'Unikatowy identyfikator zawodów.'
/

-- Table Trenerzy

CREATE TABLE Trenerzy(
  ID_pracownika Integer NOT NULL,
  Specjalizacja Varchar2(25 ) NOT NULL
        CHECK (Specjalizacja IN ('techniczny', 'wytrzymałościowy', 'coach'))
)
/

-- Add keys for table Trenerzy

ALTER TABLE Trenerzy ADD CONSTRAINT PK_Trenerzy PRIMARY KEY (ID_pracownika)
/

-- Table and Columns comments section

COMMENT ON TABLE Trenerzy IS 'Encja trener.'
/
COMMENT ON COLUMN Trenerzy.Specjalizacja IS 'Specjalizacja trenera.'
/

-- Trigger for sequence KlubSeq1 for column ID_klubu in table Kluby ---------
CREATE OR REPLACE TRIGGER ts_Kluby_KlubSeq1 BEFORE INSERT
ON Kluby FOR EACH ROW
BEGIN
  :new.ID_klubu := KlubSeq1.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Kluby_KlubSeq1 AFTER UPDATE OF ID_klubu
ON Kluby FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column ID_klubu in table Kluby as it
  uses sequence.');
END;
/
```

```
-- Trigger for sequence PracownikSeq1 for column ID_pracownika in table Pracownicy ---------
CREATE OR REPLACE TRIGGER ts_Pracownicy_PracownikSeq1 BEFORE INSERT
ON Pracownicy FOR EACH ROW
BEGIN
  :new.ID_pracownika := PracownikSeq1.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Pracownicy_PracownikSeq1 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 KlubowiczSeq1 for column ID_klubowicza in table Klubowicze ---------
CREATE OR REPLACE TRIGGER ts_Klubowicze_KlubowiczSeq1 BEFORE INSERT
ON Klubowicze FOR EACH ROW
BEGIN
  :new.ID_klubowicza := KlubowiczSeq1.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Klubowicze_KlubowiczSeq1 AFTER UPDATE OF ID_klubowicza
ON Klubowicze FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column ID_klubowicza in table Klubowicze
  as it uses sequence.');
END;
/

-- Trigger for sequence SprzetSeq1 for column ID_sprzetu in table Sprzet ---------
CREATE OR REPLACE TRIGGER ts_Sprzet_SprzetSeq1 BEFORE INSERT
ON Sprzet FOR EACH ROW
BEGIN
  :new.ID_sprzetu := SprzetSeq1.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Sprzet_SprzetSeq1 AFTER UPDATE OF ID_sprzetu
ON Sprzet FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column ID_sprzetu in table Sprzet as
  it uses sequence.');
END;
/

-- Trigger for sequence Miejsce_treningowSeq1 for column ID_miejsca_treningu in table
Miejsca_treningow ---------
CREATE OR REPLACE TRIGGER ts_Miejsca_treningow_Miejsce_treningowSeq1 BEFORE INSERT
ON Miejsca_treningow FOR EACH ROW
BEGIN
  :new.ID_miejsca_treningu := Miejsce_treningowSeq1.nextval;
```

```
END;
/
CREATE OR REPLACE TRIGGER tsu_Miejsca_treningow_Miejsce_treningowSeq1 AFTER UPDATE OF
ID_miejsca_treningu
ON Miejsca_treningow FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column ID_miejsca_treningu in table
  Miejsca_treningow as it uses sequence.');
END;
/

-- Trigger for sequence TreningSeq1 for column ID_treningu in table Treningi ---------
CREATE OR REPLACE TRIGGER ts_Treningi_TreningSeq1 BEFORE INSERT
ON Treningi FOR EACH ROW
BEGIN
  :new.ID_treningu := TreningSeq1.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Treningi_TreningSeq1 AFTER UPDATE OF ID_treningu
ON Treningi FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column ID_treningu in table Treningi
  as it uses sequence.');
END;
/

-- Trigger for sequence WlascicielSeq1 for column ID_wlasciciela in table Wlasciciele ------
CREATE OR REPLACE TRIGGER ts_Wlasciciele_WlascicielSeq1 BEFORE INSERT
ON Wlasciciele FOR EACH ROW
BEGIN
  :new.ID_wlasciciela := WlascicielSeq1.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Wlasciciele_WlascicielSeq1 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 AdresSeq1 for column ID_adresu in table Adresy ---------
CREATE OR REPLACE TRIGGER ts_Adresy_AdresSeq1 BEFORE INSERT
ON Adresy FOR EACH ROW
BEGIN
  :new.ID_adresu := AdresSeq1.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Adresy_AdresSeq1 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 RealizacjaSeq1 for column ID_realizacji in table Realizacje --------
CREATE OR REPLACE TRIGGER ts_Realizacje_RealizacjaSeq1 BEFORE INSERT
ON Realizacje FOR EACH ROW
BEGIN
  :new.ID_realizacji := RealizacjaSeq1.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Realizacje_RealizacjaSeq1 AFTER UPDATE OF ID_realizacji
ON Realizacje FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column ID_realizacji in table Realizacje
  as it uses sequence.');
END;
/

-- Trigger for sequence LicencjaSeq1 for column ID_licencji in table Licencje ---------
CREATE OR REPLACE TRIGGER ts_Licencje_LicencjaSeq1 BEFORE INSERT
ON Licencje FOR EACH ROW
BEGIN
  :new.ID_licencji := LicencjaSeq1.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Licencje_LicencjaSeq1 AFTER UPDATE OF ID_licencji
ON Licencje FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column ID_licencji in table Licencje
  as it uses sequence.');
END;
/

-- Trigger for sequence DyscyplinaSeq1 for column ID_dyscypliny in table Dyscypliny --------
CREATE OR REPLACE TRIGGER ts_Dyscypliny_DyscyplinaSeq1 BEFORE INSERT
ON Dyscypliny FOR EACH ROW
BEGIN
  :new.ID_dyscypliny := DyscyplinaSeq1.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Dyscypliny_DyscyplinaSeq1 AFTER UPDATE OF ID_dyscypliny
ON Dyscypliny FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column ID_dyscypliny in table Dyscypliny
  as it uses sequence.');
END;
/

-- Trigger for sequence OsiagniecieSeq1 for column ID_osiagniecia in table Osiagniecia -----
```

```
CREATE OR REPLACE TRIGGER ts_Osiagniecia_OsiagniecieSeq1 BEFORE INSERT
ON Osiagniecia FOR EACH ROW
BEGIN
  :new.ID_osiagniecia := OsiagniecieSeq1.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Osiagniecia_OsiagniecieSeq1 AFTER UPDATE OF ID_osiagniecia
ON Osiagniecia FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column ID_osiagniecia in table Osiagniecia
  as it uses sequence.');
END;
/

-- Trigger for sequence ZawodySeq1 for column ID_zawodow in table Zawody ---------
CREATE OR REPLACE TRIGGER ts_Zawody_ZawodySeq1 BEFORE INSERT
ON Zawody FOR EACH ROW
BEGIN
  :new.ID_zawodow := ZawodySeq1.nextval;
END;
/
CREATE OR REPLACE TRIGGER tsu_Zawody_ZawodySeq1 AFTER UPDATE OF ID_zawodow
ON Zawody FOR EACH ROW
BEGIN
  RAISE_APPLICATION_ERROR(-20010,'Cannot update column ID_zawodow in table Zawody
  as it uses sequence.');
END;
/


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

ALTER TABLE Pracownicy ADD CONSTRAINT Zatrudnia_Pracownika FOREIGN KEY (ID_klubu)
REFERENCES Kluby (ID_klubu)
/



ALTER TABLE Sprzet ADD CONSTRAINT Skladuje FOREIGN KEY (ID_klubu)
REFERENCES Kluby (ID_klubu)
/



ALTER TABLE Miejsca_treningow ADD CONSTRAINT Wykorzystuje FOREIGN KEY (ID_klubu)
REFERENCES Kluby (ID_klubu)
/



ALTER TABLE Treningi ADD CONSTRAINT Oferuje FOREIGN KEY (ID_klubu)
```

```
REFERENCES Kluby (ID_klubu)
/



ALTER TABLE Kluby ADD CONSTRAINT Klub_ma_adres FOREIGN KEY (ID_adresu)
REFERENCES Adresy (ID_adresu)
/



ALTER TABLE Realizacje ADD CONSTRAINT Trening_ma FOREIGN KEY (ID_treningu)
REFERENCES Treningi (ID_treningu)
/



ALTER TABLE Sesje_treningowe ADD CONSTRAINT Na_Realizacje_zapisuje_sie FOREIGN KEY
(ID_realizacji) REFERENCES Realizacje (ID_realizacji)
/



ALTER TABLE Trener_Realizacja ADD CONSTRAINT Realizacja_wymaga_Trenera FOREIGN KEY
(ID_realizacji) REFERENCES Realizacje (ID_realizacji)
/



ALTER TABLE Sprzety_Realizacje ADD CONSTRAINT Realizacja_potrzebuje FOREIGN KEY
(ID_realizacji) REFERENCES Realizacje (ID_realizacji)
/



ALTER TABLE Klubowicze_osiagniecia ADD CONSTRAINT Klubowicz_zdobyl FOREIGN KEY
(ID_klubowicza) REFERENCES Klubowicze (ID_klubowicza)
/



ALTER TABLE Trenerzy_Osiagniecia ADD CONSTRAINT Osiagniecie_Trener FOREIGN KEY
(ID_osiagniecia) REFERENCES Osiagniecia (ID_osiagniecia)
/



ALTER TABLE Klubowicze_osiagniecia ADD CONSTRAINT Osiagniecie_Klubowicz FOREIGN KEY
(ID_osiagniecia) REFERENCES Osiagniecia (ID_osiagniecia)
/
```

```
ALTER TABLE Osiagniecia ADD CONSTRAINT Osiagniecie_Dyscypliny FOREIGN KEY
(ID_dyscypliny) REFERENCES Dyscypliny (ID_dyscypliny)
/


ALTER TABLE Osiagniecia ADD CONSTRAINT Osiagniecie_Zawody FOREIGN KEY
(ID_zawodow) REFERENCES Zawody (ID_zawodow)
/


ALTER TABLE Wlasciciele ADD CONSTRAINT Wlasciciel_zarzadza FOREIGN KEY
(ID_klubu) REFERENCES Kluby (ID_klubu)
/


ALTER TABLE Pracownicy ADD CONSTRAINT Pracownik_mieszka FOREIGN KEY
(ID_adresu) REFERENCES Adresy (ID_adresu)
/


ALTER TABLE Klubowicze ADD CONSTRAINT Klubowicz_mieszka FOREIGN KEY
(ID_adresu) REFERENCES Adresy (ID_adresu)
/


ALTER TABLE Trenerzy ADD CONSTRAINT Pracownik_trener FOREIGN KEY
(ID_pracownika) REFERENCES Pracownicy (ID_pracownika)
/


ALTER TABLE Trenerzy_Osiagniecia ADD CONSTRAINT Trener_zdobyl FOREIGN KEY
(ID_pracownika) REFERENCES Trenerzy (ID_pracownika)
/


ALTER TABLE Trener_Realizacja ADD CONSTRAINT Trener_realizuje FOREIGN KEY
(ID_pracownika) REFERENCES Trenerzy (ID_pracownika)
/


ALTER TABLE Licencje ADD CONSTRAINT Trener_posiada_licencje FOREIGN KEY
```

```
(ID_pracownika) REFERENCES Trenerzy (ID_pracownika)
/



ALTER TABLE Licencje ADD CONSTRAINT Klubowicz_ma_licencje FOREIGN KEY
(ID_klubowicza) REFERENCES Klubowicze (ID_klubowicza)
/



ALTER TABLE Miejsca_treningu_Realizacje ADD CONSTRAINT Realizacja_ma_miejsce FOREIGN KEY
(ID_realizacji) REFERENCES Realizacje (ID_realizacji)
/
```

6. Bibliografia

— materiały udostępnione przez dr hab. inż. Marcina Kowalczyka
— https://slidetodoc.com/powtrzenie-techniki-zbierania-informacji-analiza-dokumentacji-wywiady-obserwacja/
— https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model
— https://mansfeld.pl/bazy-danych/indeksowanie-baz-danych-mysql/