Bazy Danych 2 Projekt

**Zakład bukmacherski**

Termin zajęć: Piątek, 13:15–15:00

Kacper Pabian 226123 Dr inż. Paweł Głuchowski

Artur Łuszcz 226013

**Rozdział 1**

**1.Model konceptualny bazy danych.**

* 1. **Opis świata rzeczywistego**.

Zakłady bukmacherskie w formie aplikacji jest to rodzaj zakładów losowych, w których należy przewidzieć wynik spotkania sportowego. Zakłady bukmacherskie, w tym przypadku, przyjmowane są za pośrednictwem aplikacji co pozwala na nie zatrudnianie bukmachera i tym samym na redukcje kosztów. Pieniądze potrzebne do obstawienia pobierane są z indywidualnego konta, które użytkownik musi uprzednio doładować. Tradycyjnie kupony oraz potwierdzenia zakładów były przekazywane w formie papierowej, ale z powodu rozwoju technologii, zdecydowano na zmianę tradycyjnej formy na rzecz komunikacji z użytkownikiem drogą SMS.

* 1. **Użytkownik.**

Ze względu na nowoczesne podejście do użytkownika, jedną z funkcji, które zapewnia aplikacja jest dostęp do panelu użytkownika, który umożliwia:

1. Zarejestrowanie się.
2. Zalogowanie się do aplikacji.
3. Zasilenie konta.
4. Obstawienie wyników.
5. Sprawdzenie historii.
6. Wypłacenie środków.

Aby użytkownik mógł korzystać z punktów 3-6 musi spełnić pewne wymagania. Mianowicie musi być zarejestrowany oraz zalogowany do aplikacji. Obstawienie wyników jest możliwe pod warunkiem, że administrator dodał wydarzenie sportowe do aplikacji. Kapitał użytkownika będzie trzymany w portfelu powiązanym z indywidualnym kontem. Użytkownik może sprawdzić historię swoich obstawień, w której skład wchodzą informacje z potwierdzeń.

* 1. **Administrator.**

Rozszerzeniem grupy użytkowników są administratorzy. Są odpowiedzialni za zarządzaniem bazą danych od strony technicznej. Do ich możliwości należy:

1. Dodawanie nowych zakładów.
2. Modyfikacja istniejących zakładów.
3. Usuwanie istniejących zakładów.
   1. **Przepisy.**

Administrator ponosi odpowiedzialność za poprawność danych. Zakład bukmacherski ma prawo do cofnięcia błędnych zakładów do 1 godziny po zakończeniu.

* 1. **Dane techniczne.**

Użytkownik może przeglądać zakłady i obstawiać jedynie za pośrednictwem aplikacji. Aplikacja zostanie wykonana w technologii Java.

**2. Wymaganie funkcjonalne i niefunkcjonalne.**

**2.1 Wymagania funkcjonalne**

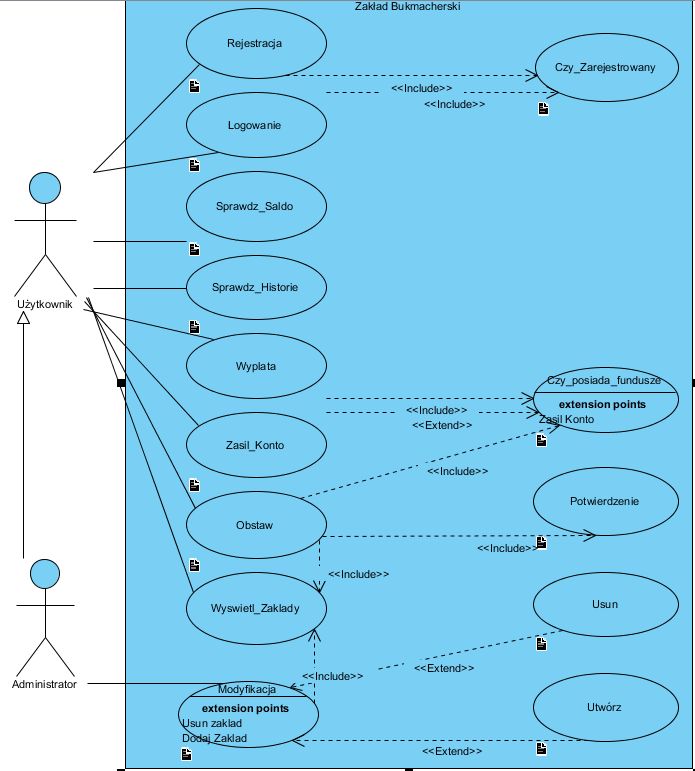
1. Użytkownik ma możliwość zarejestrowania i logowania się do systemu.
2. Administrator może dodawać nowe zakłady i zmieniać już istniejące.
3. Użytkownik może w łatwy sposób wybrać istniejący zakład i dokonać obstawienia.
4. Dostępne do obstawiania wydarzenia sportowe są dodawane przez moderatorów aplikacji.
5. Aplikacja umożliwia użytkownikowi zasilenie konta realną walutą, którą można wykorzystać jedynie do obstawiania zakładów.
6. Użytkownik może wykonać jednorazowo 1 kupon zakładami( 5 zakładów).
7. Użytkownik otrzymuje potwierdzenie obstawianego zakładu   
   z informacjami o zakładzie, kursie oraz postawionej kwocie.
8. Historia zakładów użytkownika jest mu dostępna przez miesiąc od jego zakończenia.
9. Aplikacja pozwala na sprawdzenie stanu konta.

**2.2 Wymagania niefunkcjonalne**

1. System powinien pracować w technologii Java i posiadać nowoczesny interfejs użytkownika.
2. System nie powinien udostępniać osobom trzecim informacji o użytkownikach.

**3.** **Specyfikacja wymagań funkcjonalnych za pomocą diagramu przypadków użycia.**

**3.1 Diagram przypadków użycia.**



Rysunek Diagram przypadków użycia

**3.2 Scenariusze przypadków użycia.**

**3.2.1 Nazwa PU: Rejestracja**

Cel: Dodanie danych użytkownika do bazy.

Warunki początkowe: Uruchomienie programu jako aplikacji.

Warunki końcowe: Zarejestrowanie danych użytkownika, pojawienie się nowej danej.

Przebieg 1:

1. Aplikacja prosi o podanie danych użytkownika tj. e-mail, login i hasło.
2. Użytkownik wprowadza dane.
3. Wywołanie PU Czy\_Zrejestrowany.
4. PU Czy\_Zrejestrowany zwrócił wynik negatywny.
5. Utworzenie nowej danej.
6. Wyświetlenie komunikatu o pomyślnym zarejestrowaniu.

Przebieg 2 (alternatywny kiedy PU Czy\_Zrejestrowany zwrócił wynik pozytywny):

1. Aplikacja prosi o podanie danych użytkownika tj. e-mail, login i hasło.
2. Użytkownik wprowadza dane.
3. Wywołanie PU Czy\_Zrejestrowany.
4. PU Czy\_Zrejestrowany zwrócił wynik pozytywny.
5. Wyświetlenie komunikatu o błędzie (podane dane są już zajęte).

**3.2.2 Nazwa PU: Logowanie**

Cel: Zalogowanie się przez użytkownika do aplikacji

Warunki początkowe: Uruchomienie programu jako aplikacji.

Warunki końcowe: Zalogowanie się użytkownika do systemu. Pojawienie się komunikatu o błędzie logowania jeżeli użytkownik wpiszę złe dane dostępowe.

Przebieg 1:

1. Aplikacja prosi o podanie danych dostępowych.
2. Użytkownik wprowadza swoje dane do systemu.
3. Wywołanie PU Czy\_Zarejestrowany.
4. PU Czy\_Zrejestrowany zwrócił wynik pozytywny.
5. Zalogowanie się do systemu i wyświetlenie komunikatu o pomyślnym zalogowaniu.

Przebieg 2 (alternatywny kiedy PU Czy\_Zrejestrowany zwrócił wynik negatywny):

1. Aplikacja prosi o podanie danych dostępowych.
2. Użytkownik wprowadza swoje dane do systemu.
3. Wywołanie PU Czy\_Zarejestrowany.
4. PU Czy\_Zrejestrowany zwrócił wynik negatywny.
5. Wyświetlenie komunikatu o błędnym logowaniu.

**3.2.3 Nazwa PU: Sprawdz\_Historie**

Cel: Sprawdzenie historii przeprowadzonych obstawień.

Warunki początkowe: Musi być uruchomiony wcześniej PU Logowanie.

Warunki końcowe: Zwraca informację o ilości oraz przebiegu obstawień przez użytkownika.

Scenariusz:

1. Wyświetla informację o historii obstawień użytkownika z przeciągu 30 dni wstecz.

**3.2.4 Nazwa PU: Czy\_Zarejestrowany**

Cel: Sprawdzenie czy w bazie znajduje się użytkownik o podanych danych.

Warunki początkowe: Jest uruchamiany z następujących punktów użycia: PU Logowanie i PU Rezerwacja.

Warunki końcowe: Zwraca wynik, określający, czy podane dane znajdują się w bazie

Przebieg 1:

1. Porównuje podane dane z danymi znajdującymi się w bazie.
2. Dane zostały odnalezione w bazie.
3. PU zwraca wynik pozytywny.

Przebieg 2(dane nie zostały odnalezione w bazie):

1. Porównuje podane dane z danymi znajdującymi się w bazie.
2. Dane nie zostały odnalezione w bazie.
3. PU zwraca wynik negatywny.

**3.2.5 Nazwa PU: Zasil\_Konto**

Cel: Zasilenie konta w systemie.

Warunki początkowe: Musi być uruchomiony wcześniej PU Logowanie. Jest uruchamiany z następujących punktów użycia: PU Czy\_Posiada\_Fundusze.

Warunki końcowe: Zwiększenie się środków jakie posiadał użytkownik w systemie.

Scenariusz:

1. Wyświetlenie interfejsu, w którym użytkownik poda kwotę jaką chce przelać na konto oraz wybierze metodę płatności.
2. Użytkownik wprowadza kwotę i wybiera metodę płatności.
3. Jeśli użytkownik nie dokona płatności przejdź do punktu 5.
4. Użytkownik dokonuje płatności.
5. Wyświetlenie informacji o obecnym stanie konta.

**3.2.6 Nazwa PU: Obstaw**

Cel: Wykonanie obstawienia

Warunki początkowe: Musi być uruchomiony wcześniej PU Logowanie.

Warunki końcowe: Utworzenie nowej danej odzwierciedlającej obstawienie. Posiadającej takie informacje jak: rodzaj zakładu, na jaki mecz, jaka kwota została zastawiona oraz jaka jest możliwa wygrana.

Przebieg 1:

1. Wywołanie PU Wyswietl\_Zaklady.
2. Użytkownik zaznacza odpowiednie informacje.
3. Należy wywołać PU Czy\_Posiada\_Fundusze aby sprawdzić czy użytkownik posiada wystarczającą liczbę funduszy aby dokonać obstawienia.
4. PU Czy\_Posiada\_Fundusze zwrócił wynik pozytywny.
5. Utworzenie nowej danej odzwierciedlającej obstawienie.
6. Wywołanie PU Potwierdzenie.

Przebieg 2(PU Czy\_Posiada\_Fundusze zwrócił wynik negatywny):

1. Wywołanie PU Wyswietl\_Zaklady.
2. Użytkownik zaznacza odpowiednie informacje.
3. Należy wywołać PU Czy\_Posiada\_Fundusze aby sprawdzić czy użytkownik posiada wystarczającą liczbę funduszy aby dokonać obstawienia.
4. PU Czy\_Posiada\_Fundusze zwrócił wynik negatywny.
5. Zakończenie PU i wyświetlenie informacji o braku środków.

Przebieg 3 (PU Wyswietl\_Zaklady nie wyświetlił zakładów):

1. Wywołanie PU Wyswietl\_Zaklady.
2. Na interfejsie nie wyświetliły się żadne zakłady.
3. Zakończenie PU i wyświetlenie informacji, że nie ma dostępnych zakładów

**3.2.7 Nazwa PU: Czy\_Posiada\_Fundusze**

Cel: Porównanie wymaganych środków z tymi znajdującymi się na koncie użytkownika.

Warunki początkowe: Jest uruchamiany z następujących punktów użycia: PU Obstaw i PU Wyplata.

Warunki końcowe: Zwraca wynik określający czy użytkownik posiada wystarczająco funduszy na koncie aby wykonać działanie.

Przebieg 1:

1. Porównanie środków na koncie użytkownika z kwotą potrzebną do wykonania działania.
2. Na koncie jest wystarczająco środków- zwróć wynik pozytywny.

Przebieg 2(środków na koncie jest za mało):

1. Porównanie środków na koncie użytkownika z kwotą potrzebną do wykonania działania.
2. Środków na koncie jest za mało.
3. Wywołaj PU Zasil\_Konto lub przejdź do kroku 5.
4. Wróć do kroku 1.
5. Zwróć wynik negatywny.

**3.2.8 Nazwa PU: Potwierdzenie**

Cel: Przedstawienie użytkownikowi informacji o wykonanym obstawieniu.

Warunki początkowe: Jest uruchamiany z następujących punktów użycia: PU Obstaw.

Warunki końcowe: Użytkownik otrzymuję informację, w której znajduję się ilość postawionych pieniędzy, data, możliwa wygrana i co zostało obstawione.

Scenariusz:

1. Wyświetlenie potwierdzenia użytkownikowi.

**3.2.9 Nazwa PU: Wyswietl\_Zaklady**

Cel: Wyświetlenie listy dostępnych zakładów.

Warunki początkowe: Jest uruchamiany z następujących punktów użycia: PU Obstaw i PU Modyfikacja.

Warunki końcowe: Porównanie obecnej daty oraz daty zakończenia znajdującej się przy zakładach. Wyświetlenie aktualnych zakładów

Przebieg 1:

1. Porównaj aktualną datę oraz datę informującą o zakończeniu się zakładu.
2. Wyświetl aktualne zakłady.

**3.210 Nazwa PU: Sprawdz\_Saldo**

Cel: Przedstawienie użytkownikowi informacji o stanie konta.

Warunki początkowe: Musi być uruchomiony wcześniej PU Logowanie.

Warunki końcowe: Użytkownik otrzymuję informację o stanie swojego konta.

Scenariusz:

1. Wyświetlenie informacji o stanie konta.

**3.2.11 Nazwa PU: Modyfikacja**

Cel: Modyfikacja istniejących zakładów.

Warunki początkowe: Musi być uruchomiony wcześniej PU Logowanie. Użytkownik musi się zalogować jako administrator.

Warunki końcowe: Informacja o zakładzie zostanie zmodyfikowana.

Scenariusz:

1. Wywołanie PU Wyswietl\_Zaklady.i umożliwienie wyboru jednego z nich administratorowi.
2. Administrator wybiera zakład.
3. Wyświetlenie się informacji o wybranym zakładzie oraz umożliwienie ich modyfikacji.
4. Modyfikacja informacji przez administratora.
5. Zaktualizowanie zakładu.

**3.2.12 Nazwa PU: Utworz**

Cel: Stworzenie nowego zakładu

Warunki początkowe: Musi być uruchomiony wcześniej PU Logowanie. Użytkownik musi się zalogować jako administrator. Wywołanie przez PU Modyfikacja

Warunki końcowe: Dodanie nowej danej informującej o zakładzie.

Scenariusz:

1. Wyświetlenie interfejsu umożliwiającego wprowadzenie danych.
2. Administrator wprowadza informacje o zakładzie tj. data utworzenia, data zakończenia, kurs, nazwa, rodzaj zakładu.
3. Dodanie nowej danej do systemu.

**3.2.13 Nazwa PU: Usun**

Cel: Usunięcie możliwości wyboru zakładu podczas obstawiania.

Warunki początkowe: Musi być uruchomiony wcześniej PU Logowanie. Użytkownik musi się zalogować jako administrator. Wywołanie przez PU Modyfikacja

Warunki końcowe: Usunięcie danej odzwierciedlającej istniejący zakład.

Scenariusz:

1. Wywołanie PU Wyswietl\_Zaklady
2. Administrator wybiera zakład.
3. Usunięcie danej z systemu.

**3.2.14 Nazwa PU: Wyplata**

Cel: Przelanie środków z konta w systemie na konto bankowe użytkownika

Warunki początkowe: Musi być uruchomiony wcześniej PU Logowanie.

Warunki końcowe: Wypłacenie środków do konta użytkownika.

Przebieg 1:

1.Wyświetlenie się interfejsu, w którym można wpisać kwotę do wypłaty.

2. Użytkownik podaje kwotę.

3. Wywołanie PU Czy\_Posiada\_Fundusze.

4. PU Czy\_Posiada\_Fundusze zwrócił wynik pozytywny

5. Wykonanie operacji i wyświetlenie komunikatu z potwierdzeniem.

Przebieg 2 (PU Czy\_Posiada\_Fundusze zwrócił wynik negatywny):

1.Wyświetlenie się interfejsu, w którym można wpisać kwotę do wypłaty.

2. Użytkownik podaje kwotę.

3. Wywołanie PU Czy\_Posiada\_Fundusze.

4. PU Czy\_Posiada\_Fundusze zwrócił wynik negatywny.

**4. Identyfikacja diagramu związków encji na podstawie analizy scenariuszy poszczególnych przypadków użycia.**

**4.1 Określenie tabel.**

Użytkownik-{Uzytkownik\_ID, Imie, Nazwisko, e-mail, Nr\_telefonu}.

Konto-{Login, Uzytkownik\_ID, Uprawnienia\_ID, Haslo, Srodki}.

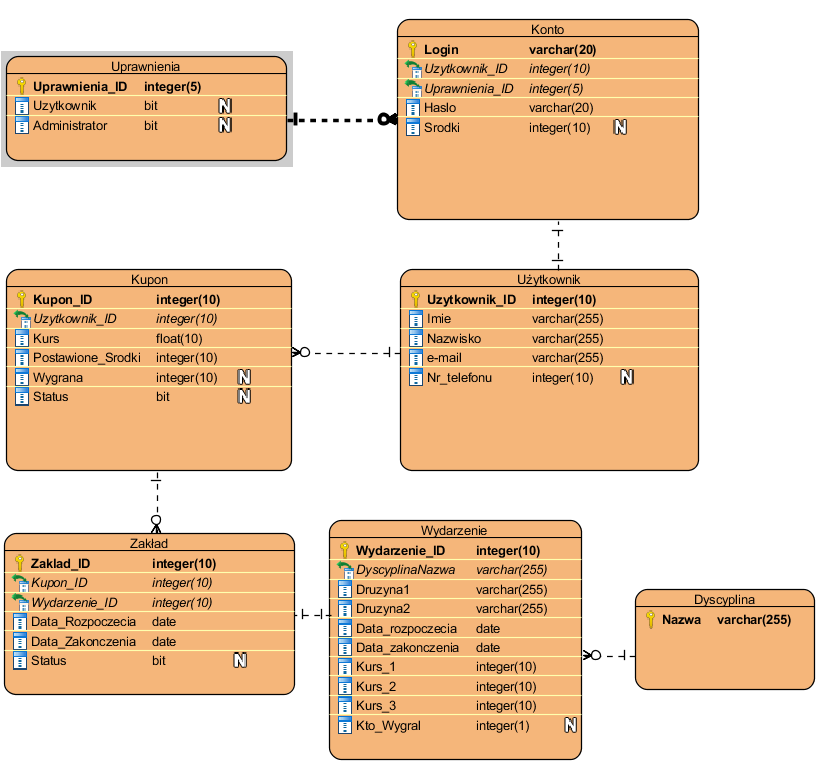
Uprawnienia-{Uprawnienia\_ID, Uzytkownik, Administrator}.

Kupon-{Kupon\_ID, Uzytkownik\_ID, Kurs, Postawione\_Srodki, Wygrana, Status}.

Zaklad-{Zaklad\_ID, Kupon\_ID, Wydarzenie\_ID, Data\_Rozpoczecia, Data\_Zakonczenia, Status}.

Wydarzenie-{Wydarzenie\_ID, DyscyplinaNazwa, Druzyna1, Druzyna2, Data\_Rozpoczecia, Data\_Zakonczenia, Kurs\_1, Kurs\_2, Kurs\_3, Kto\_Wygral}.

Dyscyplina-{Nazwa}.



Rysunek Diagram encji

**Rozdział 2**

1. **Analiza liczby instancji każdej encji**

**Obliczenia:**

* Liczba kont jest równa liczbie użytkowników.
* Zgodznie z założeniami w wymaganiach funkcjonalnych na jeden kupon przypada maksymalnie 5 zakładów, więc liczba wierszy Zakładu będzie 5 razy większa od liczby Kuponu

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tabela** | **Maksymalna liczba wierszy** | **Średnia liczba wierszy** | **Rozmiar kolumn** | **Maksymalny rozmiar tabeli** | **Średni rozmiar tabeli** |
| Użytkownik | 2 000 | 1 000 | 5 | 10 000 | 5 000 |
| Konto | 2 000 | 1 000 | 4 | 8 000 | 4 000 |
| Uprawnienia | 10 | 2 | 3 | 30 | 6 |
| Kupon | 3 000 | 2 000 | 6 | 18 000 | 12 000 |
| Zakład | 15 000 | 10 000 | 6 | 80 000 | 60 000 |
| Wydarzenie | 10 000 | 5 000 | 10 | 100 000 | 50 000 |
| Dyscyplina | 50 | 25 | 1 | 50 | 25 |

1. **Analiza użycia identyfikująca podstawowe rodzaje transakcji**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tabela** | **Wstawianie** | **Modyfikacja** | **Usuwanie** | **Wyszukiwanie** |
| Użytkownik | Często | Rzadko | Brak | Bardzo Często |
| Konto | Często | Często | Brak | Bardzo Często |
| Uprawnienia | Bardzo Rzadko | Brak | Bardzo Rzadko | Bardzo Często |
| Kupon | Bardzo Często | Często | Rzadko | Często |
| Zakład | Bardzo Często | Często | Rzadko | Często |
| Wydarzenie | Bardzo Często | Często | Rzadko | Często |
| Dyscyplina | Bardzo Rzadko | Brak | Bardzo Rzadko | Bardzo Często |

1. **Sformułowanie wymagań dotyczących dostępu – określenie częstości wykonania operacji na danych**

|  |  |  |  |
| --- | --- | --- | --- |
| **Lp.** | **Operacja** | **Częstotliwość wykonywania** | **Maksymalny czas wykonania zapytania** |
| 1. | Raport historii zakładów – na żądanie | Do 500 razy dziennie | 3 s |
| 2. | Rejestracja użytkownika | Do 200 razy dziennie | 3 s |
| 3. | Logowanie | Do 500 razy dziennie | 3 s |
| 4. | Sprawdzenie statusu kuponu | Do 500 razy dziennie | 3 s |
| 5. | Zmiana statusu zakładu | Do 1000 razy dziennie | 3 s |
| 6. | Dodanie zakładu | Do 1000 razy dziennie | 3 s |
| 7. | Dodanie wydarzenia | Do 100 razy dziennie | 3 s |
| 8. | Sprawdzenie salda | Do 1000 razy dziennie | 3 s |
| 9. | Generowanie potwierdzeń | Do 1000 razy dziennie | 3 s |
| 10. | Wypłata | Do 1000 razy dziennie | 3 s |
| 11. | Zasilenie konta | Do 1000 razy dziennie | 3 s |

Opis przebiegów operacji wymienionych w tabeli powyżej:

1. Raport historii zakładów to operacja wywoływana na żądanie użytkownika, polega ona na uzyskaniu zapamiętanych informacji o kuponach wykonanych przez użytkownika w ciągu ostatnich 30 dni – potrzebne są tabele: *Użytkownik, Zakład, Kupon*.
2. Rejestracja użytkownika to operacja dodająca rekord do tabel *Użytkownik, Konto* oraz zdefiniowaniem uprawnień nowego użytkownika znajdujących się w tabeli *Uprawnienia.*
3. Logowanie to operacja, która pozwala na znalezienie użytkownika w bazie dzięki unikalnych danych i nadaje dostęp do konta użytkownika. Wymagane tabele: *Użytkownik, Konto, Uprawnienia*.
4. Operacja, która pozwala na sprawdzenie statusu kuponu, który informuje o wygranej albo przegranej. Wymagane tabele: *Kupon*.
5. Operacja ma na celu zmiany statusu zakładu w zależności od przebiegu wydarzenia. Automatycznie wiąże się to z ewentualną zmianą statusu w kuponie. Wymagane tabele: *Kupon, Zakład, Wydarzenie*.
6. Operacja pozwala na dodanie nowego zakładu. Jeśli użytkownik nie posiada utworzonego kuponu to dodanie pierwszego zakładu tworzy nowy kupon. Podczas wykonywania tej operacji sprawdzane są środki na koncie użytkownika. Wymagane tabele: *Kupon, Zakład, Wydarzenie, Konto, Użytkownik*.
7. Operacja pozwala administratorowi na utworzenie nowego wydarzenia sportowego powiązanego z odpowiednią dyscypliną. Wymagane tabele: *Wydarzenie, Dyscyplina.*
8. Wyświetlenie informacji o środkach znajdujących się na koncie użytkownika. Wymagane tabele: *Użytkownik, Konto.*
9. Operacja pozwala na wyświetlenie informacji zawartych w nowo utworzonym zakładzie. Wymagane tabele: *Zakład, Wydarzenie.*
10. Wypłata środków z konta. Wymagane tabele: *Konto, Użytkownik*.
11. Wpłata środków na konto. Wymagane tabele: *Konto, Użytkownik*.
12. **Analiza integralności.**

Integralność łączy w sobie formalną poprawność bazy danych i procesów przetwarzania, poprawność fizycznej organizacji danych, zgodność ze schematem bazy danych, zgodność z ograniczeniami integralności oraz z regułami dostępu.

Jednym ze sposobów zachowania integralności danych są więzy integralności, czyli warunki, które powinny być spełnione przez określony podzbiór danych z bazy. Spełnianie tych warunków świadczy, że baza danych jest w stanie spójnym.

* **Integralność encji.**

Odnosi się do pojedynczej tabeli, w której powinien istnieć klucz pierwotny. Jeżeli danej kolumnie nałożyliśmy warunek PRIMARY KEY, to automatycznie zostaną nałożone jej warunki NOT NULL oraz UNIQUE.

W naszych tabelach umieściliśmy takie atrybuty, które jednoznacznie charakteryzują klucz pierwotny danej tabeli. W większości przypadków są to identyfikatory ID lecz w niektórych przypadkach są to inne kolumny jak np. Nazwa w tabeli Dyscyplina.

* **Integralność krotki.**

Zakłada się, że każda krotka opisuje jeden obiekt świata rzeczywistego a wartość krotki powinna odpowiadać elementowi świata rzeczywistego. Na wartości przyjmowane przez krotki można nałożyć niezależne więzy, które muszą być spełnione przez wszystkie krotki niezależnie. Więzy te to:

- zawężenie dziedziny atrybutu poprzez podanie przedziału wartości, listy możliwych zawartości,

- podanie formatu wartości, np. VARCHAR(20),

- zadeklarowanie konieczności występowania jakiejś wartości - NOT NULL,

- zdefiniowanie niepowtarzalnych wartości atrybutu – UNIQUE

* **Integralność odwołań.**

Dotyczy modelowania powiązań pomiędzy obiektami w rzeczywistości. W bazach relacyjnych realizowane jest to z wykorzystaniem klucza pierwotnego i odpowiadającego mu wartością klucza obcego. Tworząc klucz obcy, definiujemy związek między tabelą klucza pierwotnego i tabelą klucza obcego. Związek taki powstaje podczas połączenia kolumn tych samych typów danych z każdej tabeli.

W relacyjnych bazach danych integralność odwołań dotyczy sytuacji, kiedy tabela A zawiera klucz obcy będący równocześnie kluczem pierwotnym tabeli B. Warunek integralności odwołań ustala, że dla każdego wiersza tabeli A musi istnieć taki wiersz w tabeli B, że wartość klucza obcego i pierwotnego są jednakowe. Integralność ta może zostać naruszona poprzez usunięcie wiersza w tabeli, której klucz główny jest kluczem obcym w innej tabeli. W celu uniknięcia sytuacji odwoływania się do nieistniejących elementów zdecydowaliśmy się na różne rozwiązania. Jednym z nich będzie wpisywanie wartości NULL zamiast odwołania do konkretnego wiersza w innej tabeli. Mimo wszystko w naszym systemie operacja usuwania nie zdarza się często, dlatego liczba takich sytuacji jest mocno ograniczona.

* **Asercje**

Asercje to ograniczenia występujące w schemacie jak niezależne od tabel obiekty, służą do kontroli wartości wprowadzanych do tabel. Stosuje się je przede wszystkim do pól, których wartości musi być ograniczona oraz wartość ta powinna być sprawdzona pod kryterium spełniania określonych wymagań. Dla systemu bazy zakładu bukmacherskiego asercje można efektywnie wykorzystać przy kontroli wpisywanych numerów telefonów czy też adresów e-mail:

- W tabeli Użytkownicy numer telefonu musi posiadać 9 cyfr.

- W tabeli Użytkownicy email musi posiadać znak ‘@’.

- W tabeli Konto hasło musi posiadać cyfry.

- W tabeli Kupon zarówno kurs, postawione środki i wygrana nie mogą być liczbą ujemną.

- W tabeli Zakład i w tabeli Wydarzenie data zakończenia nie może być mniejsza od daty rozpoczęcia.

W ramach potrzeby istnieje możliwość dodawania kolejnych asercji np. w razie wystąpienia błędów czy w celu usprawnienia działania bazy.

1. **Dostrajanie bazy danych pod względem wydajności.**

Aby zapewnić wydajność naszej bazy danych przepowadziliśmy kilka optymalizacji, które mają na celu poprawienie szybkości i ogólnego działania bazy danych.

* **Użycie węzłów integralności.**

Zabieg ten nie pozwala tak zmodyfikować danych w tabelach aby straciły one spójność.

Każda tabela posiada swój unikalny klucz główny, który jednoznacznie identyfikuje każdą krotkę, dzięki czemu, oprócz spełnienia warunków normalizacji bazy danych, może występować jako łącznik w obrębie innych tabel.

|  |  |
| --- | --- |
| Nazwa tabeli | Klucz główny |
| Uprawnienia | Uprawnienia\_ID |
| Konto | Login |
| Użytkownik | Użytkownik\_ID |
| Kupon | Kupon\_ID |
| Zakład | Zakład\_ID |
| Wydarzenie | Wydazenie\_ID |
| Dyscyplina | Nazwa |

Wybrane przez nas atrybuty będą posiadać wartość NOT NULL. Zabrania to wpiasania wartości NULL do kolumny.

Przykład:

* nałożenie wartości NOT NULL na atrybut hasła w tabeli Konto, wymusza podanie hasła podczas tworzenia nowego konta użytkownika.

Niektóre tabele będą posiadać klucz obcy służący do zdefiniowania relacji, który zapewnia, że rekord w tabeli podrzędnej zawsze będzie miał swojego odpowiednika w tabeli nadrzędnej.

|  |  |
| --- | --- |
| Nazwa Tabeli | Klucz obcy |
| Konto | Użytkownik\_ID, Uprawnienia\_ID |
| Kupon | Użytkownik\_ID |
| Zakład | Kupon\_ID, Wydarzenie\_ID |
| Wydarzenie | DyscyplinaNazwa |

* **Denormalizacja**

Denormalizacja bazy polega na wprowadzenie kontrolowanej nadmierności do bazy danych w celu przyśpieszenia wykonywania na niej operacji.

Nasza baza danych została dobrze przemyślana podczas tworzenia, dzięki czemu w obecnym etapie projektu nie stwierdziliśmy konieczności używania denormalizacji.

* **Indeksowanie**

Poza kluczami głównymi, które równocześnie są indeksami uznaliśmy, że dalsze indeksowanie nie wpłynie na poprawę w działaniu bazy danych.

**Rozdział 3**

1. **Fizyczny model bazy danych.**

Baza wraz ze wszystkimi tabelami została zaimplementowana w wybranym przez nas środowisku Oracle. Baza działa na lokalnym serwerze Oracle. Przy tworzeniu użyliśmy także dodatkowego frameworku Oracle Application Express. Diagram związków pomiędzy gotowymi tabelami dołączony jest do opisu niniejszego etapu projektu. Załączony został również skrypt SQL generujący całą stworzoną bazę danych. Dołączony zostały także plik z zapytaniami generującymi losowe dane do testów.

1. **Zbiór zapytań zoptymalizowanych.**

W celu przeprowadzenia testów napisaliśmy serie zapytań do bazy danych. Zapytania te są odwzorowaniem operacji jakie będą wykonywane podczas korzystania z bazy danych.

Przedstawione poniżej zapytania zostały napisane na podstawie czynności, które w naszej aplikacji wykonują się najczęściej. Przy, niektórych zapytaniach potrzebne było wyświetlenie danych nawet z 3 tabel.

* Wyszukanie użytkownika na podstawie loginu i hasła.

SELECT uzytkownik\_id, imie, nazwisko FROM uzytkownik WHERE uzytkownik\_id=(SELECT uzytkownik\_id FROM konto WHERE login ='DnhbTvCwAEYbVLHRHGth' AND haslo='lgPpYfLYHQIqAxsKrduV');

* Wyszukanie dostępnych środków na podstawie ID użytkownika.

SELECT srodki FROM konto WHERE uzytkownik\_id=2;

* Wyszukanie wszystkich zakładów wchodzących w skład kuponów należących do danego użytkownika.

SELECT zaklad\_id, status FROM zaklad WHERE kupon\_id =ANY(select kupon\_id FROM kupon WHERE uzytkownik\_id=2);

* Wyszukanie wydarzeń wchodzących w skład zakładów należących do danego kuponu.

SELECT wydarzenie\_id FROM wydarzenie WHERE wydarzenie\_id =ANY(SELECT wydarzenie\_id FROM zaklad WHERE kupon\_id=3);

* Generowanie potwierdzenia.

SELECT uzytkownik.imie, uzytkownik.nazwisko, kupon\_Id,kurs,postawione\_srodki,wygrana AS "mozliwa wygrana", konto.srodki AS "pozostale srodki na koncie" FROM kupon JOIN konto ON kupon.uzytkownik\_id = konto.uzytkownik\_id JOIN uzytkownik ON kupon.uzytkownik\_id = uzytkownik.uzytkownik\_id WHERE kupon.kupon\_id= 4 ;

Wydajność bazy danych sprawdzaliśmy poprzez sprawdzanie informacji zawartych w kuponie przy maksymalnej\* liczbie kuponów.

\*Liczba ta została określona w *Rozdziale 2*. w podpunkcie *1. Analiza instancji każdej encji.*

Przykładowe zapytanie:

SELECT \* FROM kupon WHERE kupon\_id =3;

Wynik:



Jak widać wyżej zapytanie wykonuje się bardzo szybko nawet dla maksymalnej liczby krotek.

1. **Polityka bezpieczeństwa.**

Tworząc aplikację opartą na bazie danych, aspekt bezpieczeństwa jest niezwykle istotny. Aby zapobiec atakom ze strony hakerów oraz wyciekom informacji w naszej aplikacji zastosujemy poniżej wymienione metody:

* weryfikacja uprawnień użytkownika przy próbie dostępu do danych,
* kilka metod ochrony procesu logowania i rejestracji:
* czas oczekiwania między nieudanymi próbami logowania,
* odzyskiwanie hasła poprzez email, bądź pytanie pomocnicze,
* haszowanie hasła w aplikacji przy procesie jego wpisywania,
* hasło będzie musiało się składać z co najmniej 8 liter, w tym jednej wielkiej, 1 cyfry oraz znaku specjalnego,
* okresowe informacje o zmianie hasła na inne niż 5 poprzednich,
* moderatorzy są odpowiedzialni za utrzymanie w tajemnicy danych poufnych, do których dostęp został im powierzony, a także danych dostępowych.