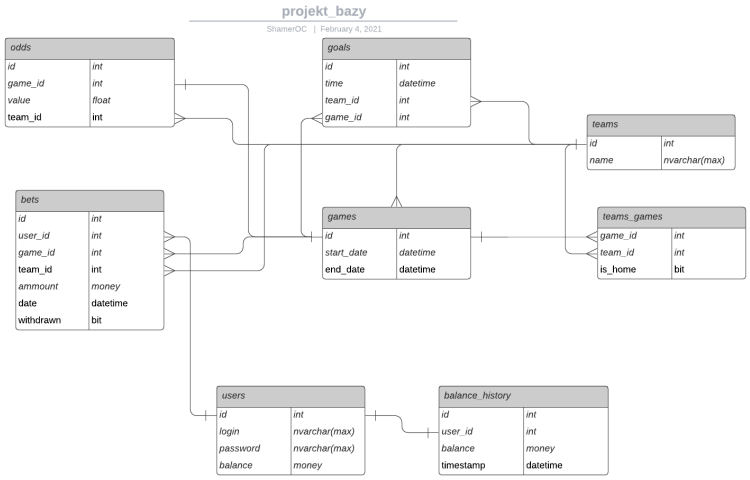
Baza danych serwisu bukmacherskiego

Krzysztof Wydrzyński

4 lutego 2021

1 Opis projektu

Stworzony przeze mnie projekt to bazy danych dla prostego serwisu bukmacherskiego.
Podczas procesu tworzenia za cel przyjąłem sobie implementacje najważniejszych tabel, widoków i funkcji, na których opiera się cała logika serwisu. Kod SQL dostępny jest na githubie pod tym linkiem: github.com/shameroc/databases-project (znajduje się tutaj także skrypt tworzenia tabel)
Założyłem, że w serwisie będzie dostępna jedynie opcja zakładów na wygraną spotkania. Dodanie innych możliwości zakładów nie stanowi problemu, baza jest zbudowana na tyle modularnie, że aby dodać taką funkcjonalność, to wystarczy dodać tabelę z rodzajami zakładów, oraz zmodyfikować tabelę *bets*. Zakłady składać można jedynie do czasu startu meczu, wyzwalacz blokuje złożenie zakładu po rozpoczęciu, co uniemożliwia popularny obecnie *live betting*, dodanie takiej opcji mocno skomplikowałoby bazę danych. Zakłady są wypłacane za pośrednictwem procedury $refresh\_balances()$, która to przeszukuje wszystkie rekordy tabeli *bets*, które nie zostały wypłacone, a następnie realizuje transakcje. Dostępna jest również historia portfela.



Rysunek 1: Diagram ER

2 Widoki, funkcje, procedury, wyzwalacze

2.1 Widoki

2.1.1 avalible_bets

Pierwszy zaimplementowany przeze mnie widok jest to avalible_bets. Pokazuje on wszystkie mecze, na które można postawić zakład.

```
WITH CTE
AS (
 SELECT DISTINCT tg.game_id, (
 SELECT t.name
 FROM teams_games AS tg1
 JOIN teams AS t ON t.id = tg1.team_id
 WHERE game_id = tg.game_id AND is_home = 1
 ) AS team_1,
 (
 SELECT t.name
 FROM teams_games AS tg1
  JOIN teams AS t ON t.id = tg1.team_id
 WHERE game_id = tg.game_id AND is_home = 0
 ) AS team_2
 FROM teams_games AS tg
)

SELECT cte.team_1 AS home, cte.team_2 AS away, g.start_date, g.end_date
FROM games AS g
JOIN CTE as cte ON cte.game_id = g.id
WHERE GETDATE() < g.start_date
```

CTE łączy każde 2 rekordy z tabeli *teams_games*, ze wspólnym identyfikatorem gry i wstawia je do jednego rekordu. Następnie wyrażenie *SELECT* wypisuje wszystkie rekordy z *games* takie, że obecna data jest mniejsza od daty rozpoczęcia meczu, oraz łączy wspomniane wcześniej CTE.

2.1.2 playing_users

Kolejny widok to playing_users. Pokazuje on wszystkich graczy, którzy mają obecnie postawiony zakład i nie został jeszcze wypłacony.

```
CREATE OR ALTER VIEW playing_users AS
SELECT b.game_id, u.login, b.ammount, t.name AS bet_on
FROM bets AS b
JOIN users AS u ON b.id=u.id
JOIN teams AS t On b.team_id=t.id
WHERE b.withdrawn=0
```

2.2 Funkcje

2.2.1 getActiveBets

Pierwsza funkcja to getActiveBets. Zwraca ona wszystkie zakłady danego użytkownika, wraz z ilością postawionych pieniędzy i potencjalną wygraną. Jako argument przyjmuje id użytkownika, a zwraca tabelę.

```
CREATE OR ALTER FUNCTION getActiveBets(@user_id INT)
RETURNS TABLE
AS
RETURN (
  SELECT b.id, b.ammount, t.name, b.ammount AS bet, (b.ammount * o.value) AS win
  FROM bets AS b
  JOIN teams AS t ON t.id=b.team_id
  JOIN odds AS o ON o.game_id=b.game_id AND o.team_id=b.team_id
  WHERE b.user_id=@user_id AND b.withdrawn=0
)
```

2.2.2 getBets

Drugą funkcją jest getBets. Jako argument przyjmuję id gry, a zwraca wszystkie złożone na nią zakłady, wraz z potrzebnymi danymi

```
CREATE OR ALTER FUNCTION getBets(@game_id INT)
RETURNS TABLE
AS
RETURN(
 SELECT u.login, o.value AS course, b.ammount, CAST((o.value * b.ammount) AS MONEY) AS win, t.name, b
 FROM bets AS b
 JOIN users AS u ON b.user_id=u.id
 JOIN odds AS o ON o.game_id=b.game_id AND o.team_id=b.team_id
 JOIN teams AS t ON t.id=b.team_id
 WHERE b.game_id=@game_id
);
```

2.2.3 getWinner

Trzecią funkcją jest getWinner. Jako argument przyjmuje ona id gry, a zwraca id wygranego w tej grze. Robi to przez zliczenie goli, które wystąpiły w tej grze. Kiedy gra się jeszcze nie zakończyła zwraca -1, a kiedy wystąpił remis, to zwraca 0.

```
CREATE OR ALTER FUNCTION getWinner(@game_id INT)
RETURNS INT
AS
BEGIN
 DECLARE @endDate DATETIME;
 DECLARE @result INT
 SET @result=-1
 SET @endDate = (SELECT end_date FROM games WHERE id=@game_id)
 IF (@endDate < GETDATE())
 BEGIN
  DECLARE @host INT
  DECLARE @guest INT
  SET @host=(SELECT team_id FROM teams_games WHERE game_id=@game_id AND is_home=1)
  SET @guest=(SELECT team_id FROM teams_games WHERE game_id=@game_id AND is_home=0)
  DECLARE @host_goals INT;
  DECLARE @guest_goals INT;
  SET @host_goals = (SELECT COUNT(id)
                                FROM goals
```

```
                                WHERE game_id=@game_id AND team_id=@host)
  SET @guest_goals = (SELECT COUNT(id)
                                FROM goals
                                WHERE game_id=@game_id AND team_id=@guest)
   IF (@host_goals > @guest_goals) SET @result=@host
   ELSE IF (@host_goals < @guest_goals) SET @result=@guest
   ELSE SET @result=0
 END
 RETURN @result
END
```

2.3 Procedury

2.3.1 placeBet

Procedura placeBet umożliwia postawienie zakładu. Jako argument przyjmuje id użytkownika, id gry, id zespołu, oraz wartość zakładu. W procedurze mamy transakcje, ponieważ mamy ograniczenie, które mówi, że kwota na koncie nie może być ujemna. Gdy próbujemy postawić więcej niż mamy w portfelu, to transakcja jest cofana. Gdy chcemy postawić na grę, która się już rozpoczęła, to wyzwalacz to blokuje, również sprawdzane jest czy stawiamy na zespół, który uczestniczy w danym meczu.

```
CREATE OR ALTER PROC placeBet(@user_id INT, @game_id INT, @team_id INT, @value MONEY)
AS
DECLARE @TRC INT = @@TRANCOUNT
IF @TRC=0
        BEGIN TRAN Tr1
ELSE
        SAVE TRAN Tr1
BEGIN TRY
        UPDATE users SET balance -= @value
        WHERE id=@user_id
        DECLARE @startDate DATETIME
        SET @startDate = (SELECT start_date FROM games WHERE id=@game_id)
        IF @startDate < GETDATE()
                DECLARE @x INT =1/0

        IF (SELECT COUNT(team_id) FROM teams_games WHERE team_id=@team_id AND game_id=@game_id) < 1
                DECLARE @a INT =1/0

        INSERT INTO bets
                (user_id, game_id, team_id, ammount, date, withdrawn)
        VALUES
                (@user_id, @game_id, @team_id, @value, GETDATE(), 0)

        IF @TRC=0
                COMMIT TRAN
END TRY
BEGIN CATCH
        IF @TRC=0
                ROLLBACK TRAN
        ELSE
                IF XACT_STATE() <> -1
```

```
                        ROLLBACK TRAN Tr1
        RAISERROR('Insufficent balance, or incorect game_id',1,1)
END CATCH
```

2.3.2 refreshBalances

Procedura refreshBalances pozwala na wypłacanie wygranych. Z założenia powinna ona być uruchamiana, kiedy gra się skończyła. Wykonują się w niej 2 zapytania $UPDATE$, które dodają fundusze od konta gracza, wtedy i tylko wtedy, gdy gra w zakładzie się już skończyła, gra jest wygrana (użyłem tutaj wcześniej napisanej funkcji $getWinner$), oraz czy nie wygrana nie została już wcześniej wypłacona. Po tej operacji flaga $withdrawn$ zmieniana jest na $true$.

```
CREATE OR ALTER PROC refreshBalances
AS
DECLARE @TRC INT = @@TRANCOUNT
IF @TRC=0
        BEGIN TRAN Tr1
ELSE
        SAVE TRAN Tr1
BEGIN TRY
        DECLARE @table TABLE (id INT)

        INSERT INTO @table
        SELECT b.id
        FROM bets AS b
        JOIN games AS g ON g.id=b.game_id AND g.start_date < GETDATE()
        WHERE withdrawn=0

        IF (NOT EXISTS (SELECT 1 FROM @table))
                DECLARE @a INT=1/0

         UPDATE users
         SET balance += o.value*b.ammount
         FROM bets b
         JOIN odds AS o ON o.game_id=b.game_id AND o.team_id=b.team_id
         WHERE b.id IN (SELECT id FROM @table)

         UPDATE bets
         SET withdrawn=1
         FROM games AS g
         WHERE b.id IN (SELECT id FROM @table)

         IF @TRC=0
                COMMIT TRAN
END TRY
BEGIN CATCH
        IF @TRC=0
                ROLLBACK TRAN
        ELSE
                IF XACT_STATE() <> -1
                        ROLLBACK TRAN Tr1
```

```
        RAISERROR('Nothing to refresh',1,1)
END CATCH
```

2.3.3 addGame

Procedura addGame dodaje gry do tabeli *games*. Najpierw sprawdzana jest poprawność daty, potem poprawność zespołów (czy nie są takie same, czy oba istnieją), a następnie dodaje do tabeli.

```
CREATE OR ALTER PROC addGame(@team1 INT, @team2 INT, @start_date DATETIME, @end_date DATETIME)
AS
DECLARE @TRC INT = @@TRANCOUNT
IF @TRC=0
        BEGIN TRAN Tr1
ELSE
        SAVE TRAN Tr1
BEGIN TRY
        IF @end_date < @start_date
                DECLARE @a INT=1/0

        IF (SELECT COUNT(id) FROM teams WHERE id=@team1 OR id=@team2) <> 2
                DECLARE @b INT=1/0

        DECLARE @id INT
        DECLARE @table table (id INT)
        INSERT INTO games
                (start_date, end_date)
        OUTPUT inserted.id INTO @table
        VALUES
                (@start_date, @end_date)

        SELECT @id = id FROM @table

        INSERT INTO teams_games
                (game_id, team_id, is_home)
        VALUES
                (@id, @team1, 1),
                (@id, @team2, 0)

        IF @TRC=0
                COMMIT TRAN
END TRY
BEGIN CATCH
        IF @TRC=0
                ROLLBACK TRAN
        ELSE
                IF XACT_STATE() <> -1
                        ROLLBACK TRAN Tr1
        RAISERROR('error happend',1,1)
END CATCH
```

2.4 Wyzwalacze

2.4.1 goalAfterGameEnd

Wyzwalacz goalAfterGameEnd blokuje dodanie do tabeli *goals* goli w meczach, które się już skończyły.

```
CREATE OR ALTER TRIGGER goalAfterGameEnd
ON goals
AFTER INSERT
AS
        DECLARE @game INT = (SELECT game_id FROM inserted)
        IF (SELECT end_date FROM games WHERE id=@game) < GETDATE()
        BEGIN
                ROLLBACK
                RAISERROR('Game has already ended',1,1)
        END
```

2.4.2 betAfterGameStart

Wyzwalacz betAfterGameStart blokuje dodanie do tabeli *bets* zakładów, dla meczy, które się już rozpoczęły

```
CREATE OR ALTER TRIGGER betAfterGameStart
ON goals
AFTER INSERT
AS
        DECLARE @game INT = (SELECT game_id FROM inserted)
        IF (SELECT start_date FROM games WHERE id=@game) < GETDATE()
        BEGIN
                ROLLBACK
                RAISERROR('Game has started',1,1)
        END
```

2.4.3 balanceHistory

Wyzwalacz balanceHistory tworzy historię portfela użytkownika.

```
CREATE OR ALTER TRIGGER balanceHistory
ON users
AFTER UPDATE
AS
        INSERT INTO balance_history
        (user_id, balance, timestamp)
        (SELECT id, balance, GETDATE()
        FROM deleted)
```