Wyższa Szkoła Bankowa w Gdańsku
Wydział Finansów i Zarządzania

Bazy Danych

Projekt zaliczeniowy

Autor : Marcin Kubicki

Gdańsk, 28.03.2022

Informacje wstępne

1.1 Cel i zakres projektu

Celem mojego projektu było zrealizowanie funkcjonującej bazy danych, która obsługuje system wypożyczeń, przechowuje informacje o klientach i pracownikach oraz śledzi obecny stan magazynu i dostaw.

1.2 Odbiorca rozwiązania

Odbiorcą projektu docelowo jest mały biznes, o stałej pojedynczej lokacji, funkcjonujący głównie stacjonarnie.

1.3 Założenia

Podczas pracy nad projektem chciałem, aby spełniał on możliwie najprościej następujące założenia.

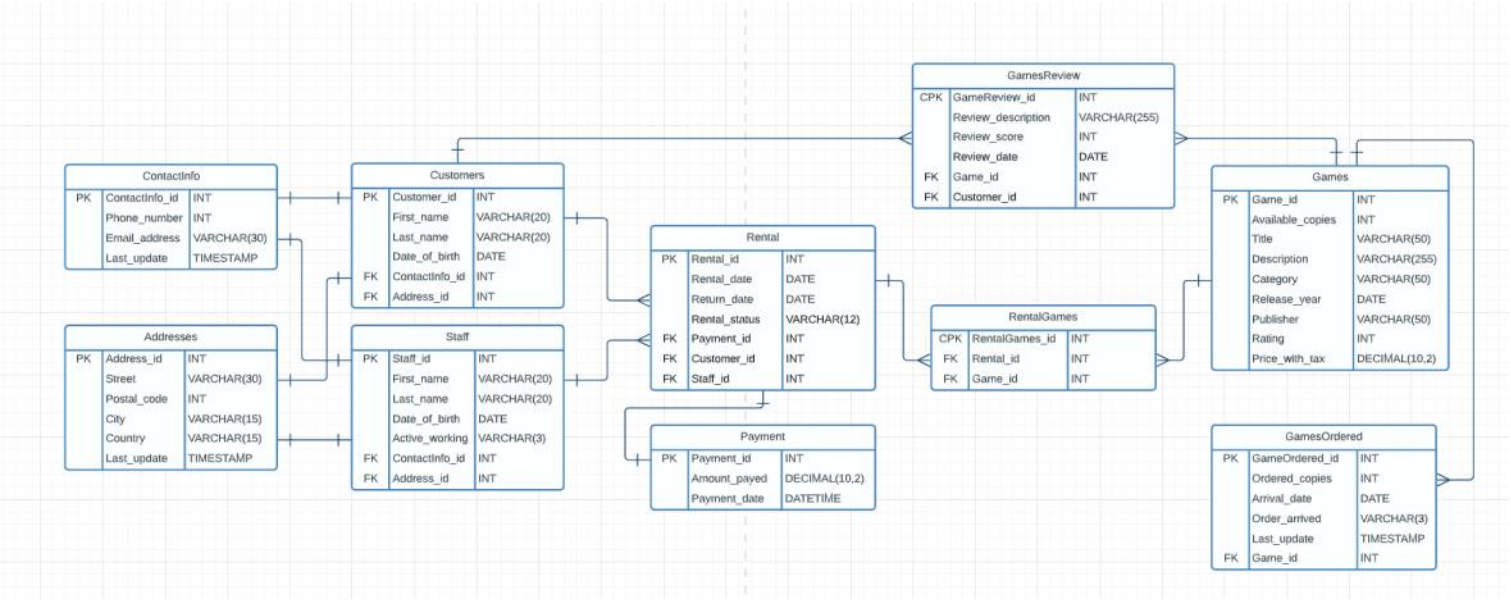
- Ewidencja i zarządzanie danymi o klientach
- Ewidencja i zarządzanie danymi o pracownikach
- Prowadzenie spisu gier dostępnych na sklepie
- Ewidencjonowanie danych z każdej płatności wraz z informacjami o ilości wypożyczonych gier i terminie zwrotu wypożyczeń.
- Prowadzenie własnej bazy recenzji wypożyczonych gier od klientów sklepu
- Śledzenie informacji o dostawach gier od dostawców biznesu.

1.4 Wykorzystane technologie

- Baza SQL Server oraz system RDBMS Microsoft SQL Server Management Studio do zarządzania bazą.
- Aplikacja webowa lucidchart umożliwiająca tworzenie diagramów ERD.

Obiekty bazy danych

2.1 Diagram bazy danych



2.2 Tabele

Tabela 1: Customers

Customers		
PK	Customer_id	INT
	First_name	VARCHAR(20)
	Last_name	VARCHAR(20)
	Date_of_birth	DATE
FK	ContactInfo_id	INT
FK	Address_id	INT

```
CREATE TABLE [Customers] (
  [Customer_id] INT PRIMARY KEY,
  [First_name] VARCHAR(20),
  [Last_name] VARCHAR(20),
  [Date_of_birth] DATE,
  [ContactInfo_id] INT FOREIGN KEY
  REFERENCES ContactInfo(ContactInfo_id),
  [Address_id] INT FOREIGN KEY
  REFERENCES Addresses(Address_id),
);
```

Przechowuje podstawowe informacje o klientach wypożyczalni. Dane dotyczące ich zamieszkania lub kontaktowe umieszczane są w osobnych tabelach *Addresses* oraz *ContacInfo*, z którymi połączona jest tabela *Customers*.

Obiekty bazy danych

Tabela 2: Staff

Staff		
PK	Staff_id	INT
	First_name	VARCHAR(20)
	Last_name	VARCHAR(20)
	Date_of_birth	DATE
	Active_working	VARCHAR(3)
FK	ContactInfo_id	INT
FK	Address_id	INT

```
CREATE TABLE [Staff] (
  [Staff_id] INT PRIMARY KEY,
  [First_name] VARCHAR(20),
  [Last_name] VARCHAR(20),
  [Date_of_birth] DATE,
  [Active_working] VARCHAR(3),
  [ContactInfo_id] INT FOREIGN KEY
  REFERENCES ContactInfo(ContactInfo_id),
  [Address_id] INT FOREIGN KEY
  REFERENCES Addresses(Address_id),
);
```

Przechowuje podstawowe informacje o pracownikach wypożyczalni. Dane dotyczące ich zamieszkania lub kontaktowe umieszczane są w osobnych tabelach *Addresses* oraz *ContacInfo*, z którymi połączona jest tabela *Staff*. Kolumna *Active_working* mówi o tym, czy pracownik w dalszym ciągu pracuje w firmie.

Tabela 3: Addresses

Addresses		
PK	Address_id	INT
	Street	VARCHAR(30)
	Postal_code	INT
	City	VARCHAR(15)
	Country	VARCHAR(15)
	Last_update	TIMESTAMP

```
CREATE TABLE [Addresses] (
  [Address_id] INT PRIMARY KEY,
  [Address] VARCHAR(30),
  [Postal_code] INT,
  [City] VARCHAR(15),
  [Country] VARCHAR(15),
  [Last_update] TIMESTAMP,
);
```

Przechowuje informace o miejscu zamieszkania pracowników i klientów z tabel *Customers* oraz *Staff*. Zawiera podstawowe dane adresowe takie jak ulica, kod pocztowy, miasto oraz kraj obecnego zamieszkania.

Obiekty bazy danych

Tabela 4: ContactInfo

ContactInfo		
PK	ContactInfo_id	INT
	Phone_number	INT
	Email_address	VARCHAR(30)
	Last_update	TIMESTAMP

```
CREATE TABLE [ContactInfo] (
  [ContactInfo_id] INT PRIMARY KEY,
  [Phone_number] INT,
  [Email_address] VARCHAR(30),
  [Last_update] TIMESTAMP,
);
```

Przechowuje informacje o danych kontaktowych dla pracowników i klientów z tabel *Customers* oraz *Staff*. Zawiera dwie kolumny przechowujące odpowiednio numer telefonu i adres mailowy oraz kolumnę *Last_update* informującą o czasie dokonania ostatnich zmian wewnątrz tabeli.

Tabela 5: Rental

Rental		
PK	Rental_id	INT
	Rental_date	DATE
	Return_date	DATE
	Rental_status	VARCHAR(12)
FK	Payment_id	INT
FK	Customer_id	INT
FK	Staff_id	INT

```
CREATE TABLE [Rental] (
  [Rental_id] INT PRIMARY KEY,
  [Rental_date] DATE,
  [Return_date] DATE,
  [Rental_status] VARCHAR(10),
  [Payment_id] INT FOREIGN KEY
  REFERENCES Payment(Payment_id),
  [Customer_id] INT FOREIGN KEY
  REFERENCES Customers(Customer_id),
  [Staff_id] INT FOREIGN KEY
  REFERENCES Staff(Staff_id),
);
```

Przechowuje informacje o wypożyczeniach. Zawiera połączenie relacją many-to-many do tabeli *Games*, za pomocą tabeli łącznikowej *RentalGames*, dzięki czemu pojedyncze wypożyczenie może zawierać wiele pozycji z tabeli gier. Dodatkowo odwołania do tabel *Payment*, *Customer* oraz *Staff* pozwalają na dodanie do każdej transakcji uiszczoną kwotę, imię i nazwisko klienta oraz imię i nazwisko pracownika, który obsługiwał dane wypożyczenie.

Tabela 6: Payment

Payment		
PK	Payment_id	INT
	Amount_payed	DECIMAL(10,2)
	Payment_date	DATETIME

```
CREATE TABLE [Payment] (
  [Payment_id] INT PRIMARY KEY,
  [Amount_payed] DECIMAL(10,2),
  [Payment_date] DATETIME,
);
```

Przechowuje informacje o płatnościach. Połączona z tabelą *Rental*. Zawiera dwie kolumny informujące odpowiednio o zapłaconej kwocie oraz o dokładnym czasie dokonania transakcji.

Tabela 7: RentalGames

RentalGames		
CPK	RentalGames_id	INT
FK	Rental_id	INT
FK	Game_id	INT

```
CREATE TABLE [RentalGames] (
  CONSTRAINT RentalGames_id PRIMARY KEY
(Rental_id, Game_id),

  [Rental_id] INT FOREIGN KEY
  REFERENCES Rental(Rental_id),
  [Game_id] INT FOREIGN KEY
  REFERENCES Games(Game_id),
);
```

Tabela łącznikowa umożliwiająca relację many-to-many pomiędzy tabelami *Rental* oraz *Games*. Jej primary key jest kluczem złożonym z primary key'ów tabel *Rental* oraz *Games*.

Obiekty bazy danych

Tabela 8: Games

Games		
PK	Game_id	INT
	Available_copies	INT
	Title	VARCHAR(50)
	Description	VARCHAR(255)
	Category	VARCHAR(50)
	Release_year	DATE
	Publisher	VARCHAR(50)
	Rating	INT
	Price_with_tax	DECIMAL(10,2)

```
CREATE TABLE [Games] (
  [Game_id] INT PRIMARY KEY,
  [Available_copies] INT,
  [Title] VARCHAR(50),
  [Description] VARCHAR(255),
  [Category] VARCHAR(50),
  [Release_year] DATE,
  [Publisher] VARCHAR(50),
  [Rating] INT,
  [Price_without_tax] DECIMAL(10,2),
  [Price_with_tax] DECIMAL(10,2),
);
```

Przechowuje informacje na temat dostępnych do wypożyczenia gier. Każdy unikalny numer id, nadany przez kolumnę *Game_id*, oznacza pojedynczą grę opisaną poprzez pozostałe kolumny, ukazując jej tytuł, opis, kategorię, rok wydania, wydawcę oraz ocenę pobraną z serwisów agregujących recenzje. Powiązana relacją many-to-many, za pośrednictwem tabeli *RentalGames*, z tabelą *Rental*, dzięki czemu możliwe jest włączenie wielu tytułów do jednego wypożyczenia. Odwołanie do niej mają również tabele *GamesReview* oraz *GamesOrdered* opisane w dalszej części konspektu.

Tabela 9: GamesOrdered

GamesOrdered		
PK	GameOrdered_id	INT
	Ordered_copies	INT
	Arrival_date	DATE
	Order_arrived	VARCHAR(3)
	Last_update	TIMESTAMP
FK	Game_id	INT

```
CREATE TABLE [GamesOrdered] (
  [GameOrdered_id] INT PRIMARY KEY,
  [Ordered_copies] INT,
  [Arrival_date] DATE,
  [Order_arrived] VARCHAR(3),
  [Last_update] TIMESTAMP,
  [Game_id] INT FOREIGN KEY
  REFERENCES Games(Game_id),
);
```

Przechowuje informacje o zamówionych grach dla wypożyczalni. Kolumna *Order_arrived* mówi o tym, czy dane zamówienie zostało już zrealizowane, czy jest w trakcie realizacji. Zawiera odwołanie do tabeli *Games*, dzięki czemu istnieje możliwość implementacji odpowiedniej procedury/triggera, który automatycznie zwiększy ilość gier w kolumnie *Available_copies* po dostarczeniu zamówienia.

Tabela 10: GamesReview

GamesReview		
CPK	GameReview_id	INT
	Review_description	VARCHAR(255)
	Review_score	INT
	Review_date	DATE
FK	Game_id	INT
FK	Customer_id	INT

```
CREATE TABLE [GamesReview] (
  CONSTRAINT GameReview_id PRIMARY KEY
   (Game_id, Customer_id),
  [Review_description] VARCHAR(255),
  [Review_score] INT,
  [Review_date] DATE,
  [Game_id] INT FOREIGN KEY
  REFERENCES Games(Game_id),
  [Customer_id] INT FOREIGN KEY
  REFERENCES Customers(Customer_id),
);
```

Tabela umożliwia zapisywanie recenzji gier bezpośrednio od klientów wypożyczalni. Zawiera klucz złożony z primary key'ów z tabel *Games* oraz *Customers*, dzięki czemu możliwe jest powiązanie recenzji z konkretnym klientem oraz konkretną grą, której dotyczy recenzja.

Funkcjonalności bazy danych

3.1 Widoki

Widok 1: CustomerFullDetail

```
CREATE VIEW CustomerFullDetail AS
SELECT C.First_name [Imię klienta], C.Last_name [Nazwisko klienta],
       C.Date_of_birth [Rok urodzenia klienta],
       CI.Phone_number [Telefon], CI.Email_address [E-mail],
         A.Street [Ulica], A.Postal_code [Kod pocztowy], A.City [Miasto], A.Country [Kraj]
FROM Customers AS [C]
      LEFT OUTER JOIN Addresses AS [A] ON C.Address_id = A.Address_id
      LEFT OUTER JOIN ContactInfo AS [CI] ON C.ContactInfo_id = CI.ContactInfo_id
```

	Imię klienta	Nazwisko klienta	Rok urodzenia klienta	Telefon	E-mail	Ulica	Kod pocztowy	Miasto	Kraj
1	Marek	Krowicki	1989-12-20	800880	NULL	ul Długa 1	80801	Gdańsk	Polska
2	Alicja	Brzeszkot	NULL	917538294	NULL	ul. Sienkiewicza 21	80720	Gdańsk	Polska
3	Jan	Kowalski	1965-03-07	746382647	Jan.kowalski@gmail.com	ul. Niepamiętna 50C	80846	Gdynia	Polska
4	Emil	Nowak	1989-06-08	393113793	Emil.Nowak@gmail.com	ul. Kwiatowa 20	80180	Gdańsk	Polska
5	Danuta	Stenka	1961-10-10	NULL	NULL	ul. Łąkowa 66	80801	Gdańsk	Polska

Widok zbiera informacje z tabel *Customers*, *Addresses* oraz *ContactInfo*. Został stworzony, aby móc w szybki sposób podejrzeć pełne dane dotyczące klientów wypożyczalni. Użyłem klauzuli *LEFT OUTER JOIN*, aby zauważyć ewentualnych klientów, którzy nie mają powiązania z pozostałymi tabelami.

Widok 2: FullGameReview

```
CREATE VIEW FullGameReview AS
SELECT G.Title [Tytuł],
         GR.Review_description [Recenzja gry], Gr.Review_score [Ocena gry],
         C.First_name [Imię klienta], C.Last_name [Nazwisko klienta]
FROM GamesReview AS [GR]
      INNER JOIN Games AS [G] ON GR.Game_id = G.Game_id
      INNER JOIN Customers AS [C] ON GR.Customer_id = C.Customer_id
```

	Tytuł	Recenzja gry	Ocena gry	Imię klienta	Nazwisko klienta
1	Wiedźmin 3	Niesamowity tytuł. Jestem dumny z naszego kraju,...	10	Marek	Krowicki
2	Worms 3D	Gra mojego dzieciństwa. Pamiętam, kiedyś to były...	8	Emil	Nowak

Widok oparty jest na tabelach *Games*, *Customers* i *GamesReview*. Przedstawia dane recenzji gier wypożyczalni dokonanych przez klientów. Tabela *GamesReview* posiada compound key złożony z primary key'ów pozostałych tabel, dzięki czemu baza jest w stanie precyzyjnie określić jakiego tytułu dotyczy recenzja oraz który klient ją napisał.

Widok 3: FullRentalInfo

```
CREATE VIEW FullRentalInfo AS
SELECT R.Rental_id AS [id_zamowienia], R.Rental_date, R.Rental_status,
      G.Title, P.Amount_payed,
      C.Last_name AS [nazwisko klienta], S.Last_name AS [nazwisko pracownika]
FROM Rental AS [R]
      LEFT OUTER JOIN RentalGames AS [RG] ON R.Rental_id = RG.Rental_id
      LEFT OUTER JOIN Games AS [G] ON G.Game_id = RG.Game_id
      LEFT OUTER JOIN Payment AS [P] ON R.Payment_id = P.Payment_id
      LEFT OUTER JOIN Customers as [C] ON R.Customer_id = C.Customer_id
      LEFT OUTER JOIN Staff as [S] On R.Staff_id = S.Staff_id
```

	id_zamowienia	Rental_date	Rental_status	Title	Amount_payed	nazwisko klienta	nazwisko pracownika
1	1	2022-01-23	Zakonczony	Call of Duty Vanguard	240.79	Kowalski	Kubicki
2	2	2022-03-15	Aktywny	Dark Souls 2	238.37	Kowalski	Szymański
3	2	2022-03-15	Aktywny	Far Cry 3	238.37	Kowalski	Szymański
4	2	2022-03-15	Aktywny	Assasin's Creed 2	238.37	Kowalski	Szymański
5	3	2022-03-10	Po terminie	Call of Duty Vanguard	312.18	Brzeszkot	Kubicki
6	3	2022-03-10	Po terminie	Assasin's Creed 2	312.18	Brzeszkot	Kubicki
7	4	2021-12-15	Zakonczony	Far Cry 3	83.49	Krowicki	Dąbrowski
8	5	2021-12-27	Po terminie	GTA V	199.79	Nowak	Andrzejewska

Widok oparty na tabelach *Rental*, *RentalGames*, *Games*, *Payment*, *Customers* oraz *Staff*. Tabela *RentalGames* jest tabelą łącznikową umożliwiającą relację many-to-many pomiędzy tabelami *Rental* oraz *Games*. Widok służy szybkiemu podglądowi wszystkich wypożyczeń wraz ze wszystkimi informacjami dotyczącymi danego wypożyczenia takimi jak: gry, które dane wypożyczenie obejmuje, płatnośći, nazwiska klientów i pracowników. Jako typ powiązania tabel wybrałem *LEFT OUTER JOIN*, aby zareagować na sytuacje, gdy tabeli *Rental* brakuje danych umieszczanych w powiązanych tabelach.

3.2 Procedury składowe

Procedura 1: GetRentalInfo

```
CREATE PROCEDURE GetRentalInfo @id INT
AS
  SELECT R.Rental_status,
  S.Last_name AS [Pracownik obslugujacy],
  C.First_name AS [Imie klienta], C.Last_name AS [Nazwisko klienta],
  P.Amount_payed AS [Platnosc], P.Payment_date AS [Data platnosci]
  FROM Rental AS [R]
  INNER JOIN Staff AS [S] ON S.Staff_id = R.Staff_id
  INNER JOIN Customers AS [C] ON C.Customer_id = R.Customer_id
  INNER JOIN Payment AS [P] ON P.Payment_id = R.Payment_id
  WHERE R.Rental_id = @id


  EXEC GetRentalInfo 1
```

Procedura wyświetla kolumnę *Rental_status* tabeli *Rental* i kilka innych kolumn dla powiązanych tabel po przyjęciu odpowiedniego parametru wejściowego, którym jest id wypożyczenia (kolumna *Rental_id* z tabeli *Rental*). Dzięki niej w łatwy sposób można odnaleźć interesujące nas wypożyczenie, aby sprawdzić związane z nim informacje.

Procedura 2: UpdateAvailableCopies

```
CREATE PROCEDURE UpdateAvailableCopies (@game_id INT, @copies INT)
AS
UPDATE Games
      SET Available_copies = @copies
      FROM Games
      WHERE Game_id = @game_id


EXEC UpdateAvailableCopies 7, 25
```

Procedura zmienia ilość dostępnych kopii danej gry poprzez zmianę wartości kolumny *Available_copies* w tabeli *Games* przyjmując jako parametry wejściowe odpowiednio id danej gry jako pierwszy parametr oraz ilość kopii po zmianie jako parametr drugi.

3.3 Funkcje

Funkcja 1: HowManyDaysUntil

```
CREATE FUNCTION dbo.HowManyDaysUntil (@destinedDate DATE)
RETURNS INT
AS
BEGIN
      RETURN DATEDIFF(day,GETDATE(),@destinedDate)
END

SELECT dbo.HowManyDaysUntil('2022-12-25')
```

Funkcja, po przyjęciu parametru wejściowego typu DATE, zwraca ilość dni pomiędzy aktualną datą, a podanym parametrem. W zamyśle służy szybkiemu sprawdzeniu ile dni dzieli wypożyczalnie od danego terminu, który może być dla niej ważny z różnych powodów, np. w poleceniu SELECT, poniżej stworzonej funkcji, zapytanie wylicza ile dni pozostało do sezonu świątecznego.

Funkcja 2: AgeOfEmployee

```
CREATE FUNCTION dbo.AgeOfEmployee (@ColumnName DATE)
RETURNS INT
AS
BEGIN
      RETURN datediff(year,@ColumnName,getdate())
END

SELECT dbo.AgeOfEmployee(date_of_birth), First_name FROM Staff
```

Funkcja, po przyjęciu parametru wejściowego typu DATE, zwraca ilość lat pomiędzy podaną kolumną, a obecną datą. Służy wyliczaniu wieku pracowników. Zapytanie *SELECT* wysyła jako parametr kolumnę *date_of_birth* z tabeli *Staff*, dorzucając dodatkowo kolumnę *First_name* dla lepszej orientacji w uzyskanych danych.

3.4 Wyzwalacze/Triggery

Trigger 1: OverbookingRentalTime

```
CREATE TRIGGER OverbookingRentalTime
ON Rental
AFTER UPDATE, INSERT
AS
	IF EXISTS
		(
		SELECT *
		FROM inserted
		WHERE DATEDIFF(day,Rental_date,getdate()) > 14
		)
	BEGIN
		UPDATE Rental
		SET Rental_status = 'Po terminie' WHERE Rental_status LIKE '%Aktywn%'
	END
```

Trigger oparty został na tabeli *Rental* i jest wywoływany poprzez polecenia *UPDATE* oraz *INSERT*. Pomysłem na powyższy wyzwalacz była potrzeba automatyzacji zmian statusu wypożyczeń w kolumnie *Rental_status*. W zamyśle tabeli w tej kolumnie przewiduje się jedynie trzy typy statusu: Aktywny, Zakonczony oraz Po terminie. Przyjmując, że każde wypożyczenie w firmie trwa 14 dni, założyłem, że jeśli różnica dni między *Rental_date*, a obecną datą, jest większa niż 14 dni oraz status jest równy 'Aktywny', wówczas następuje update kolumny *Rental_status* na 'Po terminie'. Dzięki drugiej części warunku wyzwalacz nie wykona polecenia *UPDATE*, jeśli różnica wynosi powyżej 14 dni ale dla wypożyczenia, które posiada status 'Zakonczony', a więc zostało już sfinalizowane. Dodałem również wildcard % z obu stron, aby wyzwalacz zadziałał przy ewentualnym omyłkowym wstawieniu spacji, bądź znaku przed i po statusie 'Aktywny'.