POLITECHNIKA RZESZOWSKA
im. IGNACEGO ŁUKASIEWICZA

Katedra
Informatyki i Automatyki
Politechnika Rzeszowska

Bazy danych

Dokumentacja Projektu

pt.: „Implementacja bazy danych systemu rezerwacyjnego Workshoper”

POLITECHNIKA RZESZOWSKA
im. IGNACEGO ŁUKASIEWICZA

Katedra
Informatyki i Automatyki
Politechnika Rzeszowska

Data wykonania: 15.1.2019

Grupa: EFZI III - L03
Artur Czernia

Spis Treści

Wstęp

Bazy danych są jednym z najważniejszych działów IT. To dzięki bazom danych istnieją serwisy i aplikacje – zarówno takie małe, jak chociażby programy sprzedażowe dla prywatnych sklepów, jak i takie duże jak Facebook, Google czy systemy bankowe. Jest to również jeden z najstarszych działów – bazy danych podobne do takich jakich znamy dziś powstawały już na początku lat 80. XX wieku. Wcześniej również istniały bazy danych, jednakże były one głównie kartotekowe. Z pewnością cały czas jest to rozwojowa dziedzina – pojawiają się nowe typy silników bazodanowych – obiektowe, nierelacyjne, dokumentowe, jak i istniejące typy – takie jak relacyjne – cały czas są rozwijane o nowe funkcje – jak np. wsparcie dla rozwiązań chmurowych, czy przyśpieszone silniki zapytań.

Określenie tematyki i zakresu projektu, przedstawienie zagadnień związanych z tematem

Projekt zakłada zaprojektowanie i stworzenie bazy danych dla systemu Workshoper, to jest systemu umożliwiającemu znalezienie najbliższego warztatu samochodowego i rezerwację wizyty.

Określenie funkcji bazy danych i ich priorytetu

Baza danych zostanie wykorzystana jako rdzeń systemu. Będzie służyła do przechowywania danych – warsztatów, klientów, zarezerwowanych wizytach itd. Sama baza będzie zespolona z systemem backendowym, który to będzie obsługiwał wszelką logikę i komunikację z klientami systemu. Od projektu i stworzenia bazy danych będzie zależała prędkość systemu oraz jakość oferowanych usług – zły projekt przekłada się na duże spadki wydajności w bardzo dużym stopniu. Jest to najczęstrzy element systemów powodujący spadki wydajności.

Wybór technologii i typu bazy danych do zrealizowania projektu

Do stworzenia systemu zostanie wykorzystany Microsoft SQL Server 2017 w wersji Express Edition. Jest to relacyjna baza danych tworzona przez firmę Microsoft. Wybór ten jest podyktowany przede wszystkim pewnością autora co do solidności i wydajności oferowanego narzędzia. Również dodatkowe funkcje oferowane przez ten silnik jak funkcje okna, czy replikacja pozioma pokazuje, że jest to idealne narzędzie do tego typu systemu.

Wybór narzędzi do zrealizowania projektu

Do stworzenia bazy danych zostanie użyte Microsoft SQL Server Management Studio 2017. Jest to autorskie narzędzie firmy Microsoft do komunikacji z ich bazami danych. Oprócz typowych czynności związanych z tworzeniem i administracją bazy danych, zawiera on kilka przydatnych narzędzi, które niejednokrotnie pozwalają przyśpieszyć i polepszyć pracę z bazami danych firmy Microsoft. Do takich narzędzi niewątpliwie można zaliczyć profiler zapytań, rysowane diagramy wykonania zapytań, czy możliwość wygenerowania diagramów bazy danych.

Prezentacja przygotowanego repozytorium z opisem

https://github.com/aczernia/aczerniabdproject

Diagram Logiczny

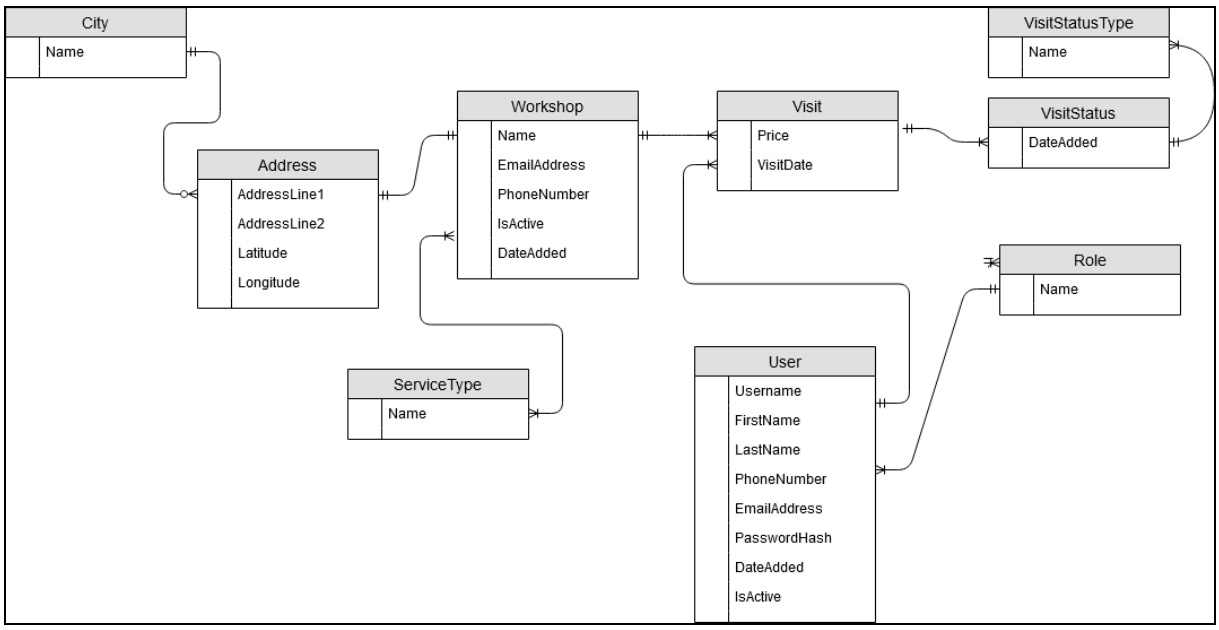
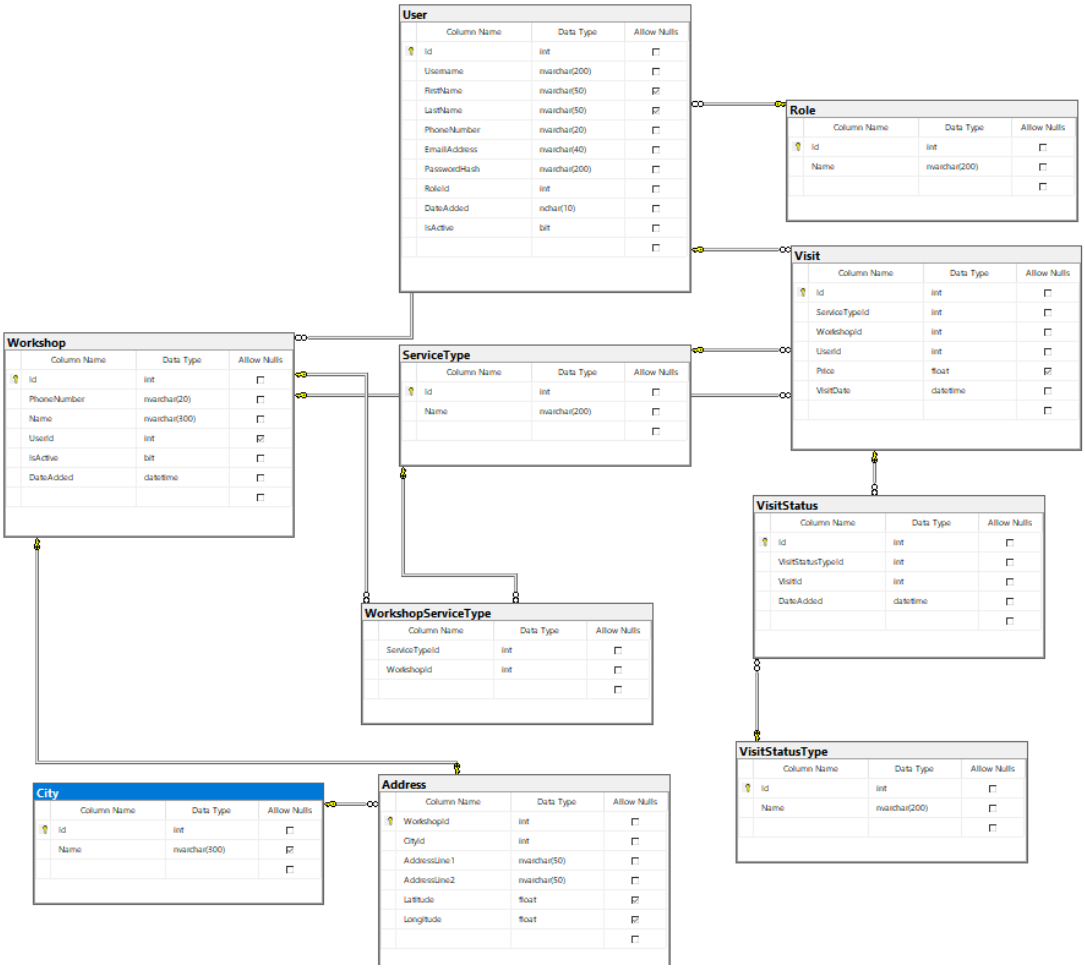


Diagram ERD


User
Column Name Data Type Allow Nulls
Id int
Username nvarchar(200)
FirstName nvarchar(50)
LastName nvarchar(50)
PhoneNumber nvarchar(20)
EmailAddress nvarchar(40)
PasswordHash nvarchar(200)
RoleId int
DateAdded nchar(10)
IsActive bit
Role
Column Name Data Type Allow Nulls
Id int
Name nvarchar(200)
Visit
Column Name Data Type Allow Nulls
Id int
ServiceTypeId int
WorkshopId int
UserId int
Price float
VisitDate datetime
Workshop
Column Name Data Type Allow Nulls
Id int
PhoneNumber nvarchar(20)
Name nvarchar(300)
UserId int
IsActive bit
DateAdded datetime
ServiceType
Column Name Data Type Allow Nulls
Id int
Name nvarchar(200)
VisitStatus
Column Name Data Type Allow Nulls
Id int
VisitStatusTypeId int
VisitId int
DateAdded datetime
WorkshopServiceType
Column Name Data Type Allow Nulls
ServiceTypeId int
WorkshopId int
VisitStatusType
Column Name Data Type Allow Nulls
Id int
Name nvarchar(200)
City
Column Name Data Type Allow Nulls
Id int
Name nvarchar(300)
Address
Column Name Data Type Allow Nulls
WorkshopId int
CityId int
AddressLine1 nvarchar(50)
AddressLine2 nvarchar(50)
Latitude float
Longitude float

Opis tabel

Workshop – jedna z głównych tabeli systemu, zawiera spis wszystkich warsztatów samochodowych systemu

Address – tabela w relacji 1 – 1 z tabelą Workshop – zawiera dane adresowe poszczególnych warsztatów, oddzielenie nastąpiło w celu zoptymalizowania sposobu wyszukiwania najlepszego warsztatu

City – tabela z miastami połączona z tabelą Address

ServiceType – tabela zawierająca opisy usług oferowanych przez system

WorkshopServiceType – tabela łącząca zawierająca oferowane przez warsztat usługi

User – tabela zawierająca użytkowników, zarówno klientów, jak i „właścicieli" warsztatów samochodowych – zawiera informacje potrzebne do logowania

Role – tabela zawierająca role w systemie

Visit – tabela zawierająca listę wizyt zarezerowanych przez użytkowników w systemie

VisitStatus – tabela zawierająca statusy poszczególnych wizyt

VisitStatusType – tabela słownikowa zawierająca możliwe statusy dla wizyt

Problemy powstałe w trakcie tworzenia diagramów

Największym problemem była kwestia rozmieszczenia tabeli Address jako osobnej tabeli w relacji – jednak w wyniku testów, zdecydowano się umieścić dane adresowe w osobnej tabeli

Skrypt tworzący bazę danych

```
CREATE TABLE dbo.Address(
	WorkshopId int NOT NULL PRIMARY KEY,
	CityId int NOT NULL,
	AddressLine1 nvarchar(50) NOT NULL,
	AddressLine2 nvarchar(50) NOT NULL,
	Latitude float NULL,
	Longitude float NULL,
)

CREATE TABLE dbo.City(
	Id int NOT NULL PRIMARY KEY IDENTITY(1,1),
	Name nvarchar(300) NULL,
)

CREATE TABLE dbo.[Role](
	Id int NOT NULL PRIMARY KEY IDENTITY(1,1),
	Name nvarchar(200) NOT NULL,
)

CREATE TABLE dbo.ServiceType(
	Id int NOT NULL PRIMARY KEY IDENTITY(1,1),
	Name nvarchar(200) NOT NULL,
)

CREATE TABLE dbo.[User](
	Id int NOT NULL PRIMARY KEY IDENTITY(1,1),
	Username nvarchar(200) NOT NULL,
	FirstName nvarchar(50) NULL,
	LastName nvarchar(50) NULL,
	PhoneNumber nvarchar(20) NOT NULL,
	EmailAddress nvarchar(40) NOT NULL,
	PasswordHash nvarchar(200) NOT NULL,
	RoleId int NOT NULL,
	DateAdded nchar(10) NOT NULL,
	IsActive bit NOT NULL,
)


CREATE TABLE dbo.Visit(
	Id int NOT NULL PRIMARY KEY IDENTITY(1,1),
	ServiceTypeId int NOT NULL,
	WorkshopId int NOT NULL,
	UserId int NOT NULL,
	Price float NULL,
	VisitDate datetime NOT NULL,
)
```

```
CREATE TABLE dbo.VisitStatus(
	Id int NOT NULL PRIMARY KEY IDENTITY(1,1),
	VisitStatusTypeId int NOT NULL,
	VisitId int NOT NULL,
	DateAdded datetime NOT NULL,
)



CREATE TABLE dbo.VisitStatusType(
	Id int NOT NULL PRIMARY KEY IDENTITY(1,1),
	Name nvarchar(200) NOT NULL,
)


CREATE TABLE dbo.Workshop(
	Id int NOT NULL PRIMARY KEY IDENTITY(1,1),
	PhoneNumber nvarchar(20) NOT NULL,
	Name nvarchar(300) NOT NULL,
	UserId int NULL,
	IsActive bit NOT NULL,
	DateAdded datetime NOT NULL,
)


CREATE TABLE dbo.WorkshopServiceType(
	ServiceTypeId int NOT NULL,
	WorkshopId int NOT NULL
)

ALTER TABLE dbo.Address  WITH CHECK ADD  CONSTRAINT FK_Address_City FOREIGN KEY(CityId)
REFERENCES dbo.City (Id)

ALTER TABLE dbo.Address CHECK CONSTRAINT FK_Address_City

ALTER TABLE dbo.Address  WITH CHECK ADD  CONSTRAINT FK_Address_Workshop FOREIGN
KEY(WorkshopId) REFERENCES dbo.Workshop (Id)

ALTER TABLE dbo.Address CHECK CONSTRAINT FK_Address_Workshop

ALTER TABLE dbo."User"  WITH CHECK ADD  CONSTRAINT FK_User_Role FOREIGN KEY(RoleId)
REFERENCES dbo."Role" (Id)

ALTER TABLE dbo."User" CHECK CONSTRAINT FK_User_Role

ALTER TABLE dbo.Visit  WITH CHECK ADD  CONSTRAINT FK_Visit_ServiceType FOREIGN
KEY(ServiceTypeId) REFERENCES dbo.ServiceType (Id)

ALTER TABLE dbo.Visit CHECK CONSTRAINT FK_Visit_ServiceType

ALTER TABLE dbo.Visit  WITH CHECK ADD  CONSTRAINT FK_Visit_User FOREIGN KEY(UserId)
REFERENCES dbo."User" (Id)
```

Skrypt wypełnienia danymi bazy danych

```
INSERT INTO dbo.Role VALUES ('Admin')
INSERT INTO dbo.Role VALUES ('Workshop')
INSERT INTO dbo.Role VALUES ('Client')

INSERT INTO dbo.City VALUES ('Rzeszów');
INSERT INTO dbo.City VALUES ('Lublin');
INSERT INTO dbo.City VALUES ('Białystok');

INSERT INTO dbo.VisitStatusType VALUES ('Zaplanowa')
INSERT INTO dbo.VisitStatusType VALUES ('Rozpoczęta')
INSERT INTO dbo.VisitStatusType VALUES ('Zakończona')
INSERT INTO dbo.VisitStatusType VALUES ('Anulowana')

INSERT INTO dbo.ServiceType(Name) VALUES ('Wymiana wału')
INSERT INTO dbo.ServiceType(Name) VALUES ('Wymiana klocków hamulcowych')
INSERT INTO dbo.ServiceType(Name) VALUES ('Wymiana chłodnicy')
```

Funkcje bazodanowe

1. Dodanie serwisu do warsztatu samochodowego

```
CREATE PROCEDURE [dbo].[AddServiceToWorkshop]
	@WorkshopId int,
	@ServiceTypeId int
AS
BEGIN
	BEGIN TRANSACTION

	INSERT INTO WorkshopServiceType(ServiceTypeId, WorkshopId) VALUES (@ServiceTypeId,
@WorkshopId);

	COMMIT TRANSACTION
END
```

2. Rozpoczęcie wizyty

```
CREATE PROCEDURE [dbo].[BeginVisit]
	@VisitId int
AS
BEGIN
	BEGIN TRANSACTION

	INSERT INTO VisitStatus(VisitStatusTypeId, VisitId, DateAdded) VALUES (2, @VisitId,
CURRENT_TIMESTAMP)

	COMMIT TRANSACTION
END
```

3. Zakończenie wizyty

```
CREATE PROCEDURE [dbo].[FinishVisit]
	@VisitId int,
	@TotalPrice float
AS
BEGIN
	BEGIN TRANSACTION

	UPDATE Visit SET Price = @TotalPrice WHERE Id = @VisitId

	INSERT INTO VisitStatus(VisitStatusTypeId, VisitId, DateAdded) VALUES (3, @VisitId,
CURRENT_TIMESTAMP)

	COMMIT TRANSACTION
END
```

4. Usługi obsługiwane przez warsztat

```
CREATE PROCEDURE [dbo].[GetServicesForWorkshop]
	@WorkshopId int
AS
BEGIN
	SELECT st.Name
	FROM Workshop w
	INNER JOIN WorkshopServiceType wst on wst.WorkshopId = w.Id
	INNER JOIN ServiceType st on wst.ServiceTypeId = st.Id
	WHERE w.Id = @WorkshopId
END
```

5. Pobranie wizyt dla klienta

```
CREATE PROCEDURE [dbo].[GetVisitsByUserId]
	@UserId int,
	@WorkshopId int
AS
BEGIN
	SELECT (u.FirstName + ' ' + u.LastName) as Name, st.Name as ServiceName, v.VisitDate
	FROM Visit v
	INNER JOIN "User" u on u.Id = v.UserId
	INNER JOIN "ServiceType" st on st.Id = v.ServiceTypeId
	WHERE v.WorkshopId = @WorkshopId AND u.Id = @UserId
END
```

6. Rejestracja wizyty

```
CREATE PROCEDURE [dbo].[RegisterVisit]
	@UserId int,
	@WorkshopId int,
	@ServiceTypeId int,
	@VisitDate datetime
AS
BEGIN
	BEGIN TRANSACTION
	INSERT INTO "Visit"(ServiceTypeId, WorkshopId, UserId, Price, VisitDate)
		VALUES (@ServiceTypeId, @WorkshopId, @UserId, null, @VisitDate);

	declare @VisitId int;
	SET @VisitId = @@IDENTITY

	INSERT INTO "VisitStatus"(VisitStatusTypeId, VisitId, DateAdded)
		VALUES (1, @VisitId, CURRENT_TIMESTAMP)

	COMMIT TRANSACTION
END
```

7. Rejestracja klienta

```
CREATE PROCEDURE [dbo].[RegisterClient]
	@Username nvarchar(200),
	@FirstName nvarchar(200),
	@LastName nvarchar(200),
	@PhoneNumber nvarchar(200),
	@EmailAddress nvarchar(20),
	@PasswordHash nvarchar(30)
AS
BEGIN
	BEGIN TRANSACTION
	INSERT INTO "User"(Username, FirstName, LastName, PhoneNumber, EmailAddress,
PasswordHash, RoleId, DateAdded, IsActive)
		VALUES (@Username, @FirstName, @LastName, @PhoneNumber, @EmailAddress,
@PasswordHash, 3, CURRENT_TIMESTAMP, 1);

	COMMIT TRANSACTION
END
GO
```

8. Wyliczenie przychodu warsztatu

```
CREATE PROCEDURE [dbo].[CalculateIncomeByMonth]
	@WorkshopId int
AS
BEGIN
	SELECT CAST(YEAR(v.VisitDate) as varchar(4)) + ' - ' + CAST(MONTH(v.VisitDate) as
varchar(4)),  SUM(v.Price)
	FROM visit v
	WHERE v.WorkshopId = @WorkshopId
	GROUP BY CAST(YEAR(v.VisitDate) as varchar(4)) + ' - ' + CAST(MONTH(v.VisitDate) as
varchar(4))
END
```

9. Rejstracja warsztatu

```
CREATE PROCEDURE [dbo].[RegisterWorkshop]
	@Name nvarchar(200),
	@PhoneNumber nvarchar(200),
	@EmailAddress nvarchar(200),
	@CityId int,
	@AddressLine1 nvarchar(200),
	@AddressLine2 nvarchar(200),
	@Latitude float,
	@Longitude float,
	@Username nvarchar(200),
	@PasswordHash nvarchar(200)
AS
BEGIN
	BEGIN TRANSACTION
	INSERT INTO "User"(Username, FirstName, LastName, PhoneNumber, EmailAddress,
PasswordHash, RoleId, DateAdded, IsActive)
		VALUES (@Username, '', '', @PhoneNumber, @EmailAddress, @PasswordHash, 2,
CURRENT_TIMESTAMP, 1);

	declare @UserId int;
	SET @UserId = @@IDENTITY

	INSERT INTO "Workshop"(PhoneNumber, Name, UserId, IsActive, DateAdded)
		VALUES (@PhoneNumber, @Name, @UserId, 1, CURRENT_TIMESTAMP);

	declare @WorkshopId int;
	SET @WorkshopId = @@IDENTITY

	INSERT INTO "Address"(WorkshopId, CityId, AddressLine1, AddressLine2, Latitude,
Longitude)
		VALUES (@WorkshopId, @CityId, @AddressLine1, @AddressLine2, @Latitude,
@Longitude)

	COMMIT TRANSACTION
END
```

10. Lista najczęstrzych usług warsztatu

```
CREATE PROCEDURE [dbo].[GetMostSellingServicesByWorkshopId]
	@WorkshopId int
AS
BEGIN
	SELECT count(v.Id) as VisitNumber, st.name as VisitName
	FROM Visit v
	INNER JOIN ServiceType st on st.Id = v.ServiceTypeId
	WHERE v.WorkshopId = @WorkshopId
	GROUP BY st.Id, st.Name
	ORDER BY count(v.Id) DESC
END
```

11. Lista zaplanowanych wizyt na dziś

```
CREATE PROCEDURE [dbo].[GetPlannedVisitsByWorkshopId]
	@WorkshopId int
AS
BEGIN
	SELECT c.ClientName, c.VisitId, c.VisitDate, c.ServiceType, c.StatusTypeName,
c.StatusTypeId
	FROM (
		SELECT (u.FirstName + ' ' + u.LastName) as ClientName, v.Id as VisitId,
st.Name as ServiceType, v.VisitDate as VisitDate, ROW_NUMBER() OVER (PARTITION BY v.ID
ORDER BY vs.DateAdded DESC) as rn, vst.Name as StatusTypeName, vst.Id as StatusTypeId,
vs.DateAdded as PlannedDate, v.WorkshopId
		FROM Visit v
		INNER JOIN ServiceType st on st.Id = v.ServiceTypeId
		INNER JOIN VisitStatus vs on vs.VisitId = v.Id
		INNER JOIN VisitStatusType vst on vst.Id = vs.VisitStatusTypeId
		INNER JOIN "User" u on u.ID = v.UserId) as c
	WHERE c.rn = 1 AND StatusTypeId = 1 AND CAST(CURRENT_TIMESTAMP as Date) =
CAST(c.VisitDate as Date) AND c.WorkshopId = @WorkshopId
END
```