

Bazy Danych
Autor: Alicja Marciniak

BAZA DANYCH ILUSTRUJĄCA MIASTA I JEGO MIESZKAŃCÓW WRAZ Z APLIKACJĄ

1. Środowisko programistyczne

Do napisania bazy oraz prostej aplikacji użyłam:



























- mySQL oraz SQL – użyte do pracy z bazą danych. Wymagane w projekcie elementy zostały napisane bezpośrednio w konsoli, w celu przyswojenia struktury kwerend. Do części powtarzalnej pracy – takiej jak wprowadzanie rekordów – posłużyłam się narzędziem phpMyAdmin, w celu przyspieszenia pracy i uniknięcia błędów.
- PHP – jako język komunikacji bazy z aplikacją. Korzystałam z PHP7 oraz komunikowałam się z serwrem dzięki pakietowi LAMP.
- HTML oraz CSS – użyte do stworzenia bardzo prostego interfejsu gracznego aplikacji.

2. WYMAGANIA PROJEKTU:

	MINIMALNA LICZBA
1. table count	15
2. joins left	4
inner	
right	
3. union	1
4. transactions	5
5. functions	2
6. full-text search	1
7. triggers	2
8. views	2
9. constraints unique	5
foreign	
10. subqueries	2
11. like	2
12. between	2

- 13. aggregation functions 3
- 14. order 4

3. Struktura plików

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 add_inhabitant.php	2017-02-22 17:06	1.3K	
 city.jpg	2017-02-23 01:46	136K	
 city.png	2017-02-22 16:33	1.0M	
 city_index.html	2017-02-23 04:37	2.3K	
 get_doctors.php	2017-02-23 01:28	1.6K	
 get_family_members.php	2017-02-23 03:23	2.0K	
 get_family_relationships.php	2017-02-22 18:03	1.0K	
 get_hospitals.php	2017-02-23 01:38	1.3K	
 get_illnesses.php	2017-02-23 01:43	1.3K	
 get_inhabitant.php	2017-02-23 00:35	1.6K	
 get_pets.php	2017-02-22 18:12	1.1K	
 get_places.php	2017-02-23 01:16	1.6K	
 get_places_categories.php	2017-02-23 02:29	1.0K	
 get_private_properties.php	2017-02-23 02:31	1.0K	
 get_professions.php	2017-02-23 02:50	1.2K	
 get_public_places.php	2017-02-23 02:32	1.0K	
 get_society.php	2017-02-23 02:55	1.6K	
 get_species.php	2017-02-22 18:20	1.0K	
 inabitant_removed.php	2017-02-23 04:37	1.7K	
 inabitant_updated.php	2017-02-23 04:20	3.6K	
 inhabitant_added.php	2017-02-23 03:58	3.6K	
 mysqli_connect.php	2017-02-22 13:26	290	
 remove_inabitant.php	2017-02-23 04:11	741	
 style.css	2017-02-23 04:37	1.7K	
 update_inhabitant.php	2017-02-23 04:16	1.3K	

4. Link do filmu ilustrującego działanie aplikacji:

<https://drive.google.com/file/d/0B46ybbgvxPL8cXJHNHJLZVhRcWM/view>

W czasie trwania prezentuję wszystkie główne tablice znajdujące się w bazie (do tablic niebędących głównymi zaliczam tablice powstałe z widoków oraz pomocnicze junction tables do relacji many-to-many).

Ponadto prezentowane są również funkcjonalności dodawania i odejmowania użytkownika.

Podczas dodawania wymagane są wszystkie pola poza datą śmierci, zaś mechanizm usuwania oparty jest o index zwracany przez bazę.

5. Wszystkie pliki oraz dump (city_inhabitants.sql) bazy znajdują się w skompresowanym folderze PROJECT.zip

6. Struktura bazy:

(poniższe screeny zostały wykonane w opraciu o wizualizację programu bezpośrednio podpiętego do bazy danych – MySQL WorkBench)

a) schemat bazy – link do screena:

<https://drive.google.com/file/d/0B46ybbgvxPL8LWZnLUZhMlp0U2s/view?usp=sharing>

b) tabele

Name	Engine	Version	Row Format	Rows	Avg Row Length	Data Length	Max Data Length	Index Length	Data Free	Auto Increment
Doctors	InnoDB	10	Dynamic	2	8192	16.0 KiB	0.0 bytes	32.0 KiB	0.0 bytes	3
FamilyMembers	InnoDB	10	Dynamic	15	1092	16.0 KiB	0.0 bytes	64.0 KiB	0.0 bytes	0
FamilyRelations	InnoDB	10	Dynamic	14	1170	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes	15
Hospitals	InnoDB	10	Dynamic	2	8192	16.0 KiB	0.0 bytes	32.0 KiB	0.0 bytes	3
Illnesses	InnoDB	10	Dynamic	12	1365	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes	0
Inhabitants	InnoDB	10	Dynamic	9	1820	16.0 KiB	0.0 bytes	48.0 KiB	0.0 bytes	21
Pets	InnoDB	10	Dynamic	10	1638	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes	12
Places	InnoDB	10	Dynamic	10	1638	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes	11
PlacesCategories	InnoDB	10	Dynamic	10	1638	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes	11
PrivateProperties	InnoDB	10	Dynamic	5	3276	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes	6
Professions	InnoDB	10	Dynamic	13	1260	16.0 KiB	0.0 bytes	32.0 KiB	0.0 bytes	0
PublicPlaces	InnoDB	10	Dynamic	6	2730	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes	7
Society	InnoDB	10	Dynamic	0	0	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes	0
Species	InnoDB	10	Dynamic	9	1820	16.0 KiB	0.0 bytes	32.0 KiB	0.0 bytes	16
_InhabitantID_DoctorID	InnoDB	10	Dynamic	10	1638	16.0 KiB	0.0 bytes	32.0 KiB	0.0 bytes	11
_InhabitantID_IllnessID	InnoDB	10	Dynamic	12	1365	16.0 KiB	0.0 bytes	32.0 KiB	0.0 bytes	13
_InhabitantID_PetID	InnoDB	10	Dynamic	14	1170	16.0 KiB	0.0 bytes	32.0 KiB	0.0 bytes	15



c) widoki

Info	Tables	Columns	Indexes	Triggers	Views	Stored Procedures	Functions	Grants	Events
Name									
LifeSentenceFullSearchForLove									
Pets\$Owners									
ShowIrishmen									
ShowPeople&Illnesses									
ShowWomen									

1) LifeSentenceFullSearchLove – widok, który pokazuje wszystkie wyniki full-text search dla frazy „love” w kolumnie z życiowym cytatem każdego mieszkańca

- 2) Pets\$Owners – widok, pokazujący wszystkie zwierzęta wraz z ich wszystkimi właścicielami (join)
- 3) ShowPeople&Illnesses – pokazuje wszystkich mieszkańców i ich choroby (jeśli jakieś mają) (join)
- 4) ShowWoman – pokazuje wszystkie kobiety, zgodnie z polskim kryterium imienia – kończące się na „a” (like)
- 5) ShowIrishmen – pokazuje Irlandczyków zgodnie z typowo irlandzkim początkiem nazwiska w formie „O’- ” - jak O’Hara (like)

d) triggery

Info	Tables	Columns	Indexes	Triggers	Views	Stored Procedures	Functions	Grants	Events
Name	Event	Table	Timing	Created	SQL Mode				
 PopulationIncrease	INSERT	Inhabitants	AFTER	2017-02-23 14:45:35.66	NO_AUTO_VALUE_ON_ZERO				
 PopulationDecrease	DELETE	Inhabitants	AFTER	2017-02-23 14:45:35.83	NO_AUTO_VALUE_ON_ZERO				

Triggery reagują na dodanie lub usunięcie mieszkańca z bazy – analogicznie zwiększając lub zmniejszając populację miasta

d) transakcje w procedurach

Info	Tables	Columns	Indexes	Triggers	Views	Stored Procedures	Functions	Grants	Events
Name	Type	Definer	Modified	Created	Security Type				
addNewPet	PROCEDURE	root@localhost	2017-02-23 18:11:05	2017-02-23 18:11:05	DEFINER				
addNewSpecies	PROCEDURE	root@localhost	2017-02-23 15:59:36	2017-02-23 15:59:36	DEFINER				
removePet	PROCEDURE	root@localhost	2017-02-23 18:18:37	2017-02-23 18:18:37	DEFINER				
removeSpecies	PROCEDURE	root@localhost	2017-02-23 18:20:01	2017-02-23 18:20:01	DEFINER				
updateAmountOfSpecies	PROCEDURE	root@localhost	2017-02-23 16:05:23	2017-02-23 16:05:23	DEFINER				

Transakcje związane są z dodawaniem zwierzęcia do bazy. W tabeli ze zwierzętami istnieje indeks wskazujący na jego gatunek, zaś w tabeli z gatunkami znajduje się kolumna zliczająca wszystkie zwierzęta danego gatunku.

Procedury związane z dodaniem (addSpecies) i odjęciem (removeSpecies) gatunku opierają się tylko na prostej transakcji dodania/odjęcia, a ilość zwierząt danego gatunku jest ustawiana domyślnie na 0.

Procedury dodania/odjęcia zwierzęcia (addPet oraz removePet) składają się również z takich transakcji, ale po ich wykonaniu wywoływana jest wewnątrz procedury inna procedura pomocnicza – procedura aktualizująca liczbę zwierząt (updateAmountOfSpecies)

e) union

Użyłam unii przy tworzeniu tablicy PlacesCategories, która jest złączeniem zawartości tablic PublicPlaces oraz PrivateProperties.

f) between oraz aggregation functions

Zostały użyte do podliczenia statystyk miasta w tablicy Society.

PRZYKŁADOWE ZAPISANE PRZEZE MNIE KWERENDY:

(niestety część z nich nie jest zapisana w dumpie bazy, dlatego je tu umieszczam)

```
SELECT * FROM `Inhabitants` WHERE `Inhabitants`.`FirstName` LIKE '%a' // zwraca kobiety
```

```
SELECT * FROM `Inhabitants` WHERE `Inhabitants`.`LastName` LIKE 'O"% // zwraca  
Irlandczykow
```

```
// WYPISUJE MIESZKAŃCA I DRUGĄ OSOBĘ WRAZ Z RELACJĄ W JAKIEJ JEST DLA  
TEGO MIESZKAŃCA
```

```
SELECT  
I1.FirstName,  
I1.LastName,  
FamilyRelations.Relation,  
I2.FirstName,  
I2.LastName,  
FamilyMembers.FamilyName  
FROM FamilyMembers  
INNER JOIN Inhabitants AS I1  
    ON FamilyMembers.`Inhabitant1ID`=I1.`InhabitantID`  
INNER JOIN FamilyRelations  
    ON FamilyMembers.`FamilyRelationID`=FamilyRelations.`FamilyRelationID`  
INNER JOIN Inhabitants AS I2  
    ON FamilyMembers.`Inhabitant2ID`=I2.`InhabitantID`  
ORDER BY `I1`.`FirstName` ASC
```

```
// WYPISUJE CHOROBY DLA KAŻDEGO MIESZKAŃCA
```

```
SELECT IH.FirstName, IH.LastName, IL.Name  
FROM Inhabitants AS IH  
LEFT JOIN __InhabitantID_IlnessID AS __IN_IL  
    ON IH.InhabitantID = __IN_IL.InhabitantID  
LEFT JOIN Illnesses AS IL  
    ON __IN_IL.IlnessID = IL.IlnessID
```

```
// WYPISUJE WŁAŚCICIELI DLA KAŻDEGO ZWIERZAKA
```

```
SELECT P.Name, IH.FirstName, IH.LastName  
FROM Inhabitants AS IH  
RIGHT JOIN __InhabitantID_PetID AS __IN_P  
    ON IH.InhabitantID = __IN_P.InhabitantID  
RIGHT JOIN Pets AS P  
    ON __IN_P.PetId = P.PetID
```

```
// LICZBA ZWIERZĄT DANEGO GATUNKU
```

```
UPDATE `Species` SET `Amount` = (SELECT COUNT(1) FROM `Pets`  
    WHERE `Pets`.`SpeciesID`=`Species`.`SpeciesID`)
```

```
//LOWER CASE NA WSZYSTKICH NAZWACH
```

```
SELECT Professions.ProffesionID, LCASE(Name), Professions.WorkPlaceID FROM `Professions`
```

```
// UPER CASE NA WSZYSTKICH SZPITALACH
```

```
SELECT Hospitals.HospitalID, UCASE(HospitalName) FROM `Hospitals`
```

```
// Wszyscy z wiekiem większym niż średnia
```

```
SELECT * FROM Inhabitants WHERE Age > (SELECT AVG(Age) FROM Inhabitants)
```

```
// wyświetla doktorów wraz z danymi opartymi o zewnętrzne indeksy w tablicy Doctors
```

```
SELECT D.DoctorID, I.FirstName, I.LastName, D.Specialization. H.Name
```

```
FROM Inhabitants AS I
```

```
INNER JOIN Doctors AS D
```

```
ON D.InhabitantID = I.InhabitantID
```

```
INNER JOIN Hospitals AS H
```

```
ON D.HospitalID = H.HospitalID;
```

```
// wyświetla miejsca wraz z danymi opartymi o zewnętrzne indeksy w tablicy Places
```

```
SELECT P.PlaceID, P.Name, P.Street, P.Number, P.Flat, C.Name Category
```

```
FROM Places AS P
```

```
INNER JOIN PlacesCategories AS C
```

```
ON P.PlaceCategoryID = C.PlaceCategoryID;
```

```
//procedura pomocnicza zaktualizacji ilości zwierząt
```

```
CREATE PROCEDURE updateAmountOfSpecies()
```

```
BEGIN
```

```
DECLARE exit handler for sqlexception
```

```
BEGIN
```

```
ROLLBACK;
```

```
END;
```

```
DECLARE exit handler for sqlwarning
```

```
BEGIN
```

```
ROLLBACK;
```

```
END;
```

```
START TRANSACTION;
```

```
UPDATE `Species` SET `Amount` = (SELECT COUNT(1) FROM `Pets`
```

```
WHERE `Pets`.`SpeciesID`=`Species`.`SpeciesID`);
```

```
COMMIT;
```

```
END //
```

```
//procedura dodania zwierzęcia
```

```
CREATE PROCEDURE addNewPet(IN Name varchar(128), SpeciesID int)
```

```
BEGIN
```

```
DECLARE exit handler for sqlexception
```

```
BEGIN
```

```
ROLLBACK;
```

```
END;
```

```
DECLARE exit handler for sqlwarning
```

```
BEGIN
```

```
ROLLBACK;
```

```
END;
```

```
START TRANSACTION;
```

```
INSERT INTO Pets (Name, SpeciesID) VALUES (Name, SpeciesID);
COMMIT;
CALL updateAmountOfSpecies();
END //
```

```
// usunięcie zwierzęcia
CREATE PROCEDURE removePet(IN PetID int)
BEGIN
DECLARE exit handler for sqlexception
BEGIN
ROLLBACK;
END;
DECLARE exit handler for sqlwarning
BEGIN
ROLLBACK;
END;
START TRANSACTION;
DELETE FROM Pets WHERE Pets.PetID = PetID;
COMMIT;
CALL updateAmountOfSpecies();
END //
```

```
//usunięcie gatunku
CREATE PROCEDURE removeSpecies(IN SpeciesID int)
BEGIN
DECLARE exit handler for sqlexception
BEGIN
ROLLBACK;
END;
DECLARE exit handler for sqlwarning
BEGIN
ROLLBACK;
END;
START TRANSACTION;
DELETE FROM Species WHERE Species.SpeciesID = SpeciesID;
COMMIT;
END //
```