POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

PROJEKT Z BAZ DANYCH

System obsługi hotelu

Termin zajęć: Środa, 13:15–15:00

AUTOR/AUTORZY:
Bartłomiej Lis 226227

PROWADZĄCY ZAJĘCIA:
dr inż. Roman Ptak, W4/K9

Wrocław, 2019 r.

Spis treści:

1. Wstęp

1.1. Cel projektu

Stworzenie bazy danych i aplikacji webowej do obsługi rezerwacji hotelu.

1.2. Zakres projektu

Projekt będzie polegał na stworzeniu bazy danych która będzie przechowywać dane dotyczące hotelu, klientów, pracowników i rezerwacji. Stworzenie aplikacji webowej która pozwoli w szybki i prosty sposób na utworzenie rezerwacji i sprawdzenie konkretnych danych.

2. Analiza wymagań

Wybór i opracowanie wstępnych założeń dotyczących wybranych tematów projektów.

2.1. Opis działania i schemat logiczny systemu

Hotel usytuowany we Wrocławiu o wysokości 3 pięter. Na pierwszych dwóch znajdują się pokoje dwuosobowe na trzecim natomiast apartamentowce które mieszczą po cztery osoby. Liczba pokoi dwuosobowych 20 (10 na piętro) i 4 apartamentowce. Hotel oferuje możliwość rezerwacji mailowej i telefonicznej. Każda potwierdzona rezerwacja jest zapisywana i widoczna w bazie danych (dane wynajmującego, termin, opłata). Aplikacja webowa pozwala łatwo śledzić i sprawdzać informacje o gościach, wolnych pokojach i opłatach.

2.2. Wymagania funkcjonalne

1.Pracownik :

- wprowadzenie rezerwacji
- modyfikacja rezerwacji
- usuwanie rezerwacji
- potwierdzenie zameldowania
- modyfikacja meldunku
- sprawdzenie czy są wolne pokoje
- dodawanie danych gości
- sprawdzenie danych gości
- modyfikacja danych gości
- przeglądanie historii rezerwacji
- wyszukiwanie rezerwacji w danym przedziale czasu
- obliczanie kosztu pobytu

- generowanie rachunku

2.Administrator:

- posiada wszystkie funkcjonalności pracownika
- usuwanie oraz modyfikacja rachunku
- zarządzanie danymi pracowników (dodawanie, usuwanie itp.)

2.3. Wymagania niefunkcjonalne

2.3.1. Wykorzystywane technologie i narzędzia

Baza danych powstanie w mySQL natomiast aplikacja webowa powstanie w ASP.net.

2.3.2. Wymagania dotyczące rozmiaru bazy danych

Baza danych będzie niewielka ze względu na małą liczbę pokoi w hotelu. Wraz z wzrostem popularności hotelu baza będzie posiadać dane kilku tysięcy klientów oraz podobną ilość rezerwacji. Dane klientów będą przetrzymywane przez 5 lat. Przewidujemy w dobrych warunkach około czterech tysięcy gości rocznie i około dwóch tysięcy pobytów. Liczba pracowników będzie oscylować w okolicy pięciu osób. Dane dotyczące samego hotelu, pokoi nie będzie się zmieniać.

2.3.3. Wymagania dotyczące bezpieczeństwa systemu

Ze względu na możliwy dostęp do bazy danych przez internet, zastosowano szyfrowanie pomiędzy użytkownikiem oraz serwerem, potwierdzone certyfikatem SSL. Zastosowano system archiwizacji bazy danych jak i całego systemu. Napisano skrypt który codziennie w godzinach nocnych wykonuje backup wszystkich danych oraz wysyła je na dysk twardy. W przypadku niepowodzenia/powodzenia backupu administrator zostaje poinformowany. W przypadku błędu systemu lub użytkownika będzie możliwe przywrócenie bazy danych oraz całego systemu.

2.4.Przyjęte założenia projektowe

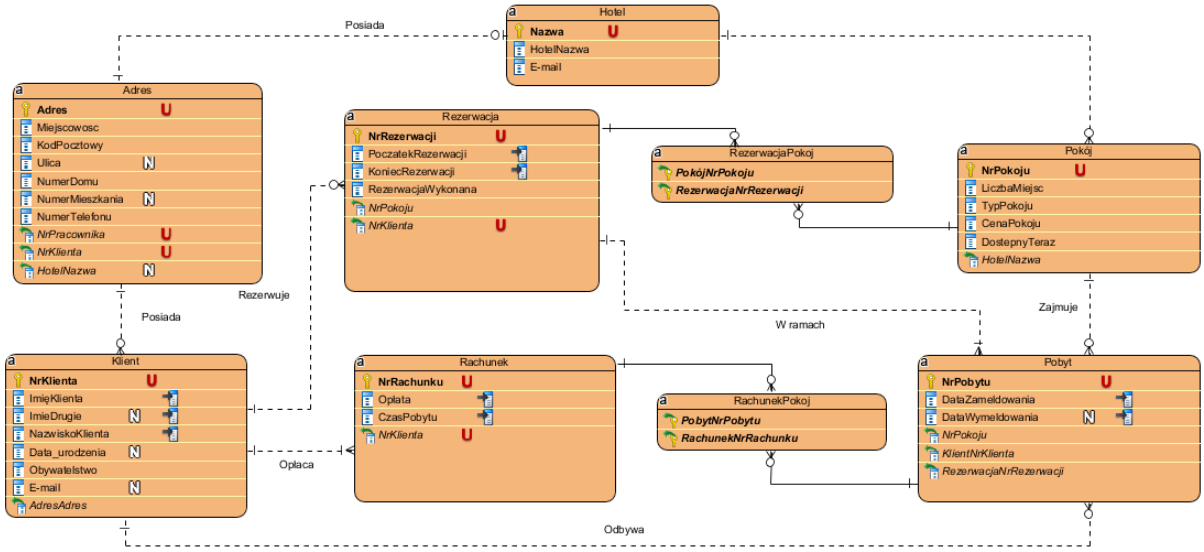
Ze względu na przyjęte dane, baza nie będzie zbyt duża. Pozwoli w prosty sposób sprawdzić informacje i rezerwację miejsc hotelowych z poziomu aplikacji webowej. Administrator hotelu będzie miał dodatkowe możliwości dodawania nowych kont pracowników, zmieniania ich danych osobowych oraz dostęp do usunięcia bądź edycji rachunku przy losowych problemach. Prosty system zabezpieczający logowania pracowników i archiwizacji daje bezpieczeństwo przed utratą danych.

3. Projekt systemu

Projekt i struktury bazy danych, mechanizmów zapewniania poprawności przechowywanych informacji, oraz kontroli dostępu do danych.

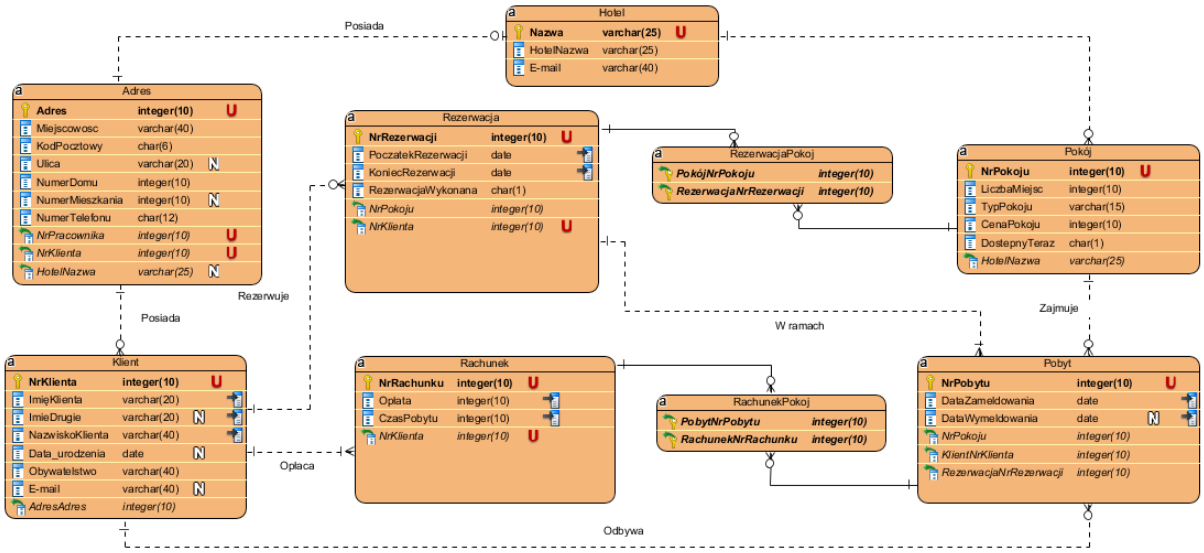
3.1. Projekt bazy danych

3.1.1. Analiza rzeczywistości i uproszczony model konceptualny



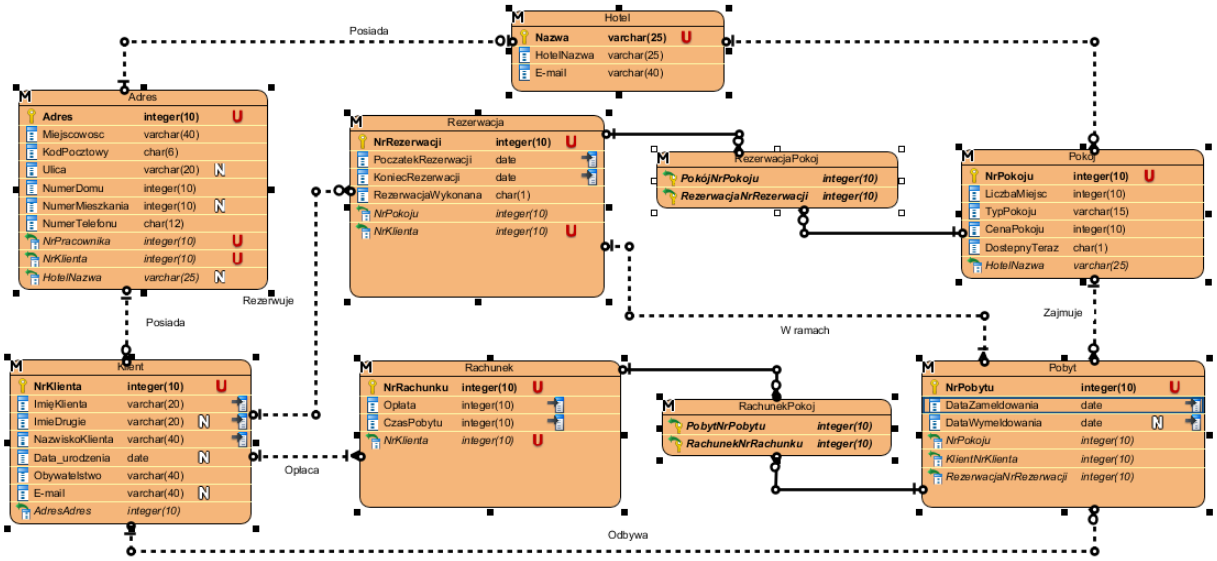
3.1. Model konceptualny stworzony w programie Visual Paradigm dla systemu bazodanowego MySql.

3.1.2. Model logiczny i normalizacja



3.2. Model logiczny stworzony w programie Visual Paradigm dla systemu bazodanowego MySql.

3.1.3. Model fizyczny i ograniczenia integralności danych



3.3. Model fizyczny stworzony w programie Visual Paradigm dla systemu bazodanowego MySql.

3.1.4. Inne elementy schematu – mechanizmy przetwarzania danych

Sekwencje w przy tworzeniu kluczy będą tworzone przy pomocy funkcji „autoincrement” systemu MySql.

Widoki:

wolne_ten_miesiac – jest to widok, który wyświetla informacje o nr pokoi oraz termiach dostępności w danym miesiącu. Wykorzystuje informacje z tabel rezerwacja oraz pokój.

platnosc_pokoj – jest to widok, który wyświetla dane o nr rezerwacji, nr pobytu na określony pokój przez danego klienta(nr klienta) oraz cenie pobytu. Wykorzystuje do tego złączenie tabel: pobyt, klient, rachunek.

historia_rezerwacji – jest to widok, który wyświetla informacje na temat wszystkich gości przebywających kiedykolwiek w hotelu oraz okresie rezerwacji na podstawie tabeli rezerwacja.

goscie_teraz – jest to widok, który wyświetla informacje na temat wszystkich gości obecnie znajdujących się hotelu. Korzysta z tabeli pobyt.

dane_klient- jest to widok, który wyświetla informacje na temat określonego klienta. Korzysta z tabeli klient.

Wyzwalacze:

mozliwosc_rezerwacji – jest to wyzwalacz przypięty do tabeli rezerwacja, uruchamia się on przed dodaniem nowej krotki do tabeli rezerwacja i sprawdza czy można zarezerwować określony pokój w określonym przedziale czasowym, w tym celu sprawdza zawartość tabeli rezerwacja oraz pobyt.

duze_litery_klient – wyzwalacz przypięty do tabeli klient, który w momencie wprowadzania nowego klienta będzie sprawdzał czy pola Imię, DrugieImię, Nazwisko zaczynają się od dużej litery w przeciwnym wypadku podmienia je.

duze_litery_adres- wyzwalacz przypięty do tabeli adres, który w momencie podawania nowego adresu będzie sprawdzał czy pola Miejscowosc oraz Ulica zaczynają się od dużej litery, w przeciwnym wypadku podmienia je.

poprawny_nr_tel – wyzwalacz przypięty do tabeli adres, który w momencie wpisywania wartości do kolumny NumerTelefonu sprawdza czy spełniają one założenia wskazujące na poprawność wprowadzanego numeru.

Procedury składowane:

PokazRachunekKlienta – procedura przyjmująca jako argument id klienta i zwracająca jego sumę zobowiązań oraz czas pobytu.

SprawdzDostepnePokoje – procedura przyjmująca jako argument pożądany przedział czasowy i zwracająca numery pokoi dostępnych w danym przedziale.

WykonajRezerwacje – procedura przyjmująca jako argument początek i koniec rezerwacji, nrKlienta, nr pokoju i dokonuje wpisu rezerwacji do bazy danych.

ZweryfikujKlienta – procedura przyjmująca jako argument nazwisko Klienta oraz zwracająca informacje czy osoba o danym nazwisku posiada aktywny pobyt.

3.1.5. Projekt mechanizmów bezpieczeństwa na poziomie bazy danych

	Pracownik	Administrator
Hotel	Wyświetlanie	wyświetlanie modyfikacja usuwanie dodawanie
Klient	wyświetlanie modyfikacja usuwanie dodawanie	wyświetlanie modyfikacja usuwanie dodawanie
Adres	wyświetlanie modyfikacja usuwanie dodawanie	wyświetlanie modyfikacja usuwanie dodawanie
Rezerwacja	wyświetlanie modyfikacja usuwanie dodawanie	wyświetlanie modyfikacja usuwanie dodawanie
Pobyt	wyświetlanie modyfikacja usuwanie dodawanie	wyświetlanie modyfikacja usuwanie dodawanie
Rachunek	wyświetlanie dodawanie	wyświetlanie modyfikacja usuwanie dodawanie
Pokoj	Wyświetlanie	wyświetlanie modyfikacja usuwanie dodawanie

3.2. Projekt aplikacji użytkownika

3.2.1. Architektura aplikacji i diagramy projektowe

3.2.2. Interfejs graficzny i struktura menu

3.2.3. Projekt wybranych funkcji systemu

Rejestracja Pracownika. Funkcja rejestrująca pracownika w systemie. Administrator po podaniu przez pracownika danych (imię, nazwisko, email) zakłada konto dla pracownika.

Logowanie pracownika/administratora. Funkcja logowania administratora w systemie. Jej celem jest sprawdzenie poprawności login'u i hasła. Po poprawnym zidentyfikowaniu następuje przekierowanie użytkownika do panelu pracownika/administratora.

Przypomnienie hasła. Funkcja umożliwiająca zmianę hasła konta administratora/pracownika systemu. W wyniku podania poprawnego email wysyła za pomocą poczty elektronicznej wygenerowany link umożliwiający zmianę hasła.

Przegląd historii pobytu. Funkcja umożliwia przeglądanie pobytów, które się już zakończyły. Umożliwiać będzie przeglądanie pobytów w określonych ramach czasowych, dla danego gościa, dla danego pokoju.

Przegląd rezerwacji. Funkcja umożliwia przeglądanie złożonych rezerwacji.

Przegląd wolnych pokoi. Funkcja umożliwia przeglądanie wolnych pokoi w wybranym terminie.

Przegląd rezerwacji dla danego pokoju. Funkcja umożliwia przegląd rezerwacji konkretnego pokoju.

3.2.4. Metoda podłączania do bazy danych – integracja z bazą danych

Wymiana danych pomiędzy interfejsem użytkownika (przeglądarka WWW), a bazą danych będzie odbywać się przy pomocy modułu PHP. Moduł ten jest odpowiedzialny za walidację danych, ich przetwarzanie oraz dwukierunkową wymianę z bazą danych. Komunikacja z bazą danych odbywa się za pomocą biblioteki ADOdb w wersji 4.80 lub wyższej. System bazodanowy aplikacji będzie oparty o bazę danych MySQL w wersji 5.2 lub wyższej uruchomiony pod systemem operacyjnym Linux z zainstalowanym serwerem HTTP Apache 1.3.x lub wyższą z obsługą PHP w wersji 4.3.x lub wyższej. Komunikacja użytkownika z aplikacją będzie odbywać się przy pomocy przeglądarki stron.

3.2.5. Projekt zabezpieczeń na poziomie aplikacji

Głównym zabezpieczeniem systemu będzie login pracownika i hasło. Hasła będą przechowywane w oddzielnej bazie. Przechowywane hasło będzie zhaszowane.

4. Implementacja systemu baz danych

Do zaimplementowania bazy danych użyto zestawu aplikacji Xampp + phpMyAdmin.
Wersja silnika bazy danych: 10.1.38-MariaDB

4.1. Tworzenie tabel i definiowanie ograniczeń

Elementy bazy danych takie jak Tablice i definiowanie ograniczeń zostały stworzone przy pomocy narzędzi dostępnych w pakiecie phpMyAdmin .
W poniższym przykładzie ukazano sposób w jaki zostały stworzone tabele „klient" oraz „adres" oraz jak zaimplementowano ograniczenie dla tabeli „klient".

```
CREATE TABLE `klient` (
  `NrKlienta` int(10) NOT NULL,
  `ImieKlienta` varchar(20) NOT NULL,
  `ImieDrugie` varchar(20) DEFAULT NULL,
  `NazwiskoKlienta` varchar(40) NOT NULL,
  `DataUrodzenia` date DEFAULT NULL,
  `Obywatelstwo` varchar(40) NOT NULL,
  `Email` varchar(40) DEFAULT NULL,
  `FkAdres` int(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
CREATE TABLE `adres` (
  `IdAdres` int(10) NOT NULL,
  `Miejscowosc` varchar(40) NOT NULL,
  `KodPocztowy` char(6) NOT NULL,
  `Ulica` varchar(20) DEFAULT NULL,
  `NrDomu` varchar(10) NOT NULL,
  `NrMieszkania` varchar(10) DEFAULT NULL,
  `NrTelefonu` char(12) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
ALTER TABLE `klient`
  ADD CONSTRAINT `klient_ibfk_1` FOREIGN KEY (`FkAdres`) REFERENCES `adres` (`IdAdres`);
```

4.2. Implementacja mechanizmów przetwarzania danych

Niektóre mechanizmy przetwarzania danych takie jak indeksy oraz sekwencje zostały zaimplementowane przy pomocy gotowych narzędzi pakietu phpMyAdmin już podczas tworzenia tabel. Poniżej przedstawiono indeksy oraz sekwencje dla tabeli „rezerwacja”.

```
ALTER TABLE `rezerwacja`
  ADD PRIMARY KEY (`NrRezerwacji`),
  ADD KEY `PocRezIdx` (`PoczatekRezerwacji`),
  ADD KEY `KonRezIdx` (`KoniecRezerwacji`),
  ADD KEY `FkKlient` (`FkKlient`);

ALTER TABLE `rezerwacja`
  MODIFY `NrRezerwacji` int(10) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=10;
```

Mechanizmy przetwarzania danych takie jak widoki, wyzwalacze oraz procedury składowane utworzone zostały przy pomocy czystego języka SQL. Poniżej przedstawiono po jednym przykładzie każdego z nich.

```
CREATE VIEW `wolne_ten_miasiac`  AS
 select `t1`.`NrPokoju` AS `NrPokoju`,
        `t1`.`LiczbaMiejsc` AS `LiczbaMiejsc`,
        `t1`.`Pietro` AS `Pietro`,
        `t1`.`TypPokoju` AS `TypPokoju`,
        `t1`.`CenaPokoju` AS `CenaPokoju`
 from ((`pokoj` `t1` join `rezerwacja` `t2`)
            join `rezerwacja_pokoj` `t3`)
 where ((`t1`.`NrPokoju` = `t3`.`NrPokoju`)
        and (`t2`.`NrRezerwacji` = `t3`.`NrRezerwacji`)
        and (month(`t2`.`PoczatekRezerwacji`) <> month(curdate()))
        and (`t1`.`DostepnyTeraz` = '1')) ;
```

Dodatkowo: Platnosc_pokoj, hotel_dane, historia_rezerwacji, goscie_teraz.

```
CREATE TRIGGER `wstaw_date` BEFORE INSERT ON `rezerwacja` FOR EACH ROW BEGIN
    SET NEW.DataOperacji=NOW();
END
```

Dodatkowo: ustaw_dostepny(tabela pokoj), ustaw_niedostepny(tabela pokoj)

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `generuj_rachunek` (IN `id_klient` INT)  NO SQL
BEGIN
DECLARE x, y, z INT DEFAULT 0;
SELECT (pobyt.KoniecZameldowania - pobyt.PoczatekZameldowania),
  (pobyt.KoniecZameldowania - pobyt.PoczatekZameldowania) *
  pokoj.CenaPokoju, pobyt.NrPobytu
  INTO x,y,z
FROM pobyt, pokoj
WHERE pobyt.FkKlient=id_klient
    AND pobyt.FkPokoj=pokoj.NrPokoju;

 INSERT INTO rachunek VALUES (null, y, x,id_klient);
 INSERT INTO rachunek_pobyt VALUES(z,LAST_INSERT_ID());

END$$
```

Dodatkowo: dostępne_pokoje, wykonaj_rezerwacje, znajdz_klienta

4.3. Implementacja uprawnień i innych zabezpieczeń

Dostęp do bazy danych będzie miał jedynie administrator oraz oraz pracownik.

Uprawnienia administratora:

```
1  GRANT ALL PRIVILEGES ON *.* TO 'Administrator'@'localhost'
2  IDENTIFIED BY PASSWORD '*05A8A1A73083F816772592F13D11C8AA5CCD9681' WITH GRANT OPTION;
3
4  GRANT ALL PRIVILEGES ON `obsluga\_hotelu`.* TO 'Administrator'@'localhost' WITH GRANT OPTION;
5
6  GRANT SELECT, INSERT (FkNazwa, LiczbaMiejsc, DostepnyTeraz, Pietro, TypPokoju, CenaPokoju),
7  UPDATE (FkNazwa, LiczbaMiejsc, DostepnyTeraz, Pietro, TypPokoju, CenaPokoju),
8  DELETE, CREATE, DROP, INDEX, ALTER, CREATE VIEW, SHOW VIEW, TRIGGER
9  ON `obsluga_hotelu`.`pokoj` TO 'Administrator'@'localhost' WITH GRANT OPTION;
10
11 GRANT SELECT, INSERT (FkAdres, HotelNazwa, Email),
12 UPDATE (FkAdres, HotelNazwa, Email),
13 DELETE, CREATE, DROP, INDEX, ALTER, CREATE VIEW, SHOW VIEW, TRIGGER
14 ON `obsluga_hotelu`.`hotel` TO 'Administrator'@'localhost' WITH GRANT OPTION;
15
16 GRANT SELECT, INSERT (CzasPobytu, Oplata, FkKlient),
17 UPDATE (CzasPobytu, Oplata, FkKlient),
18 DELETE, CREATE, DROP, INDEX, ALTER, CREATE VIEW, SHOW VIEW, TRIGGER
19 ON `obsluga_hotelu`.`rachunek` TO 'Administrator'@'localhost' WITH GRANT OPTION;
20
21 GRANT SELECT, INSERT (KodPocztowy, Miejscowosc, NrTelefonu, NrDomu, Ulica, NrMieszkania),
22 UPDATE (KodPocztowy, Miejscowosc, NrTelefonu, NrDomu, Ulica, NrMieszkania),
23 DELETE, CREATE, DROP, INDEX, ALTER, CREATE VIEW, SHOW VIEW, TRIGGER
24 ON `obsluga_hotelu`.`adres` TO 'Administrator'@'localhost' WITH GRANT OPTION;
```

```
GRANT SELECT, INSERT (KoniecRezerwacji, PoczatekRezerwacji, FkKlient, DataOperacji),
UPDATE (KoniecRezerwacji, PoczatekRezerwacji, FkKlient, DataOperacji),
DELETE, CREATE, DROP, INDEX, ALTER, CREATE VIEW, SHOW VIEW, TRIGGER
ON `obsluga_hotelu`.`rezerwacja` TO 'Administrator'@'localhost' WITH GRANT OPTION;

GRANT SELECT, INSERT (NazwiskoKlienta, ImieKlienta, Obywatelstwo, ImieDrugie, FkAdres, DataUrodzenia, Email),
UPDATE (NazwiskoKlienta, ImieKlienta, Obywatelstwo, ImieDrugie, FkAdres, DataUrodzenia, Email),
DELETE, CREATE, DROP, INDEX, ALTER, CREATE VIEW, SHOW VIEW, TRIGGER
ON `obsluga_hotelu`.`klient` TO 'Administrator'@'localhost' WITH GRANT OPTION;

GRANT SELECT, INSERT (KoniecZameldowania, PoczatekZameldowania, FkRezerwacja, FkPokoj, FkKlient),
UPDATE (KoniecZameldowania, PoczatekZameldowania, FkRezerwacja, FkPokoj, FkKlient),
DELETE, CREATE, DROP, INDEX, ALTER, CREATE VIEW, SHOW VIEW, TRIGGER ON `obsluga_hotelu`.`pobyt`
TO 'Administrator'@'localhost' WITH GRANT OPTION;
```

Uprawnienia pracownika:

```
GRANT SELECT, INSERT, UPDATE, DELETE, FILE ON *.* TO 'pracownik'@'localhost'
IDENTIFIED BY PASSWORD '*05A8A1A73083F816772592F13D11C8AA5CCD9681';

GRANT ALL PRIVILEGES ON `obsluga_hotelu`.* TO 'pracownik'@'localhost';

GRANT SELECT ON `obsluga_hotelu`.`hotel` TO 'pracownik'@'localhost';

GRANT SELECT,
INSERT (KoniecZameldowania, PoczatekZameldowania, FkRezerwacja, FkPokoj, FkKlient),
UPDATE (KoniecZameldowania, PoczatekZameldowania, FkRezerwacja, FkPokoj, FkKlient),
DELETE ON `obsluga_hotelu`.`pobyt` TO 'pracownik'@'localhost';

GRANT SELECT,
INSERT (NazwiskoKlienta, ImieKlienta, Obywatelstwo, ImieDrugie, FkAdres, DataUrodzenia, Email),
UPDATE (NazwiskoKlienta, ImieKlienta, Obywatelstwo, ImieDrugie, FkAdres, DataUrodzenia, Email),
DELETE ON `obsluga_hotelu`.`klient` TO 'pracownik'@'localhost';

GRANT SELECT,
INSERT (KodPocztowy, Miejscowosc, NrTelefonu, NrDomu, Ulica, NrMieszkania),
UPDATE (KodPocztowy, Miejscowosc, NrTelefonu, NrDomu, Ulica, NrMieszkania),
DELETE ON `obsluga_hotelu`.`adres` TO 'pracownik'@'localhost';

GRANT SELECT,
INSERT (KoniecRezerwacji, PoczatekRezerwacji, FkKlient, DataOperacji),
UPDATE (KoniecRezerwacji, PoczatekRezerwacji, FkKlient, DataOperacji),
DELETE ON `obsluga_hotelu`.`rezerwacja` TO 'pracownik'@'localhost';

GRANT SELECT,
INSERT (FkNazwa, LiczbaMiejsc, DostepnyTeraz, Pietro, TypPokoju, CenaPokoju),
UPDATE (FkNazwa, LiczbaMiejsc, DostepnyTeraz, Pietro, TypPokoju, CenaPokoju)
ON `obsluga_hotelu`.`pokoj` TO 'pracownik'@'localhost';

GRANT SELECT,
INSERT (CzasPobytu, Oplata, FkKlient),
UPDATE (CzasPobytu, Oplata, FkKlient),
DELETE
ON `obsluga_hotelu`.`rachunek` TO 'pracownik'@'localhost';
```

4.4. Testowanie bazy danych na przykładowych danych

Wstawianie danych:

Poprawne:

✔ Wstawionych rekordów: 1.
Id wstawionego wiersza: 4

```
INSERT INTO `klient` (`NrKlienta`, `ImieKlienta`, `ImieDrugie`, `NazwiskoKlienta`,
`DataUrodzenia`, `Obywatelstwo`, `Email`, `FkAdres`) VALUES (NULL, 'Przemyslaw', 'Stanislaw',
'Kwiatkowski', '1992-05-21', 'Polska', 'kwiatkowski@wp.pl', '1');
```

✔ Wstawionych rekordów: 1.
Id wstawionego wiersza: 2

```
INSERT INTO `adres` (`IdAdres`, `Miejscowosc`, `KodPocztowy`, `Ulica`, `NrDomu`, `NrMieszkania`,
`NrTelefonu`) VALUES (NULL, 'Poznan', '12-130', 'Kwiatkowa', '5', '2', NULL);
```

✔ Wstawionych rekordów: 1.

```
INSERT INTO `pokoj` (`NrPokoju`, `LiczbaMiejsc`, `Pietro`, `TypPokoju`, `CenaPokoju`,
`DostepnyTeraz`, `FkNazwa`) VALUES ('6', '2', '2', 'dwuosobowy', '100', '1', '1');
```

Niepoprawne:

```
1 INSERT INTO `klient` (`NrKlienta`, `ImieKlienta`, `ImieDrugie`, `NazwiskoKlienta`,
  `DataUrodzenia`, `Obywatelstwo`, `Email`, `FkAdres`) VALUES (NULL, 'Piotr', NULL,
  'Bałtroczyk', NULL, 'Polska', NULL, NULL);
```

```
#1048 - Kolumna 'FkAdres' nie może być null
```

```
1 INSERT INTO `rezerwacja` (`NrRezerwacji`, `PoczatekRezerwacji`, `KoniecRezerwacji`,
  `DataOperacji`, `FkKlient`) VALUES (NULL, '2019-05-16', '2019-05-25', NULL, '33');
```

```
#1452 - Cannot add or update a child row: a foreign key
   constraint fails (`obsluga_hotelu`.`rezerwacja`,
CONSTRAINT `rezerwacja_ibfk_1` FOREIGN KEY (`FkKlient`)
         REFERENCES `klient` (`NrKlienta`))
```

-brak klienta o nr 33 w tabeli klient.

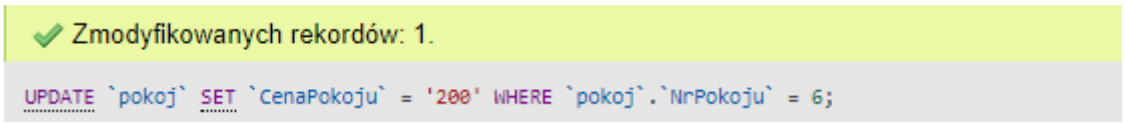
```
1 INSERT INTO `rachunek` (`NrRachunku`, `Oplata`, `CzasPobytu`, `FkKlient`) VALUES (NULL,
  '200', '3', '50');
```

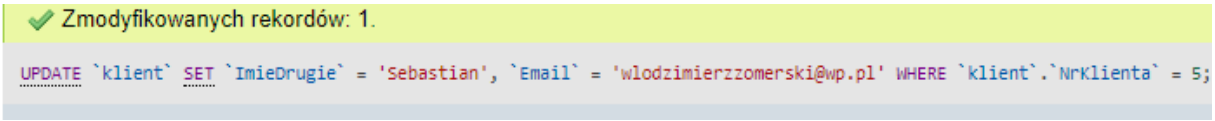
```
#1452 - Cannot add or update a child row: a foreign key
    constraint fails (`obsluga_hotelu`.`rachunek`,
 CONSTRAINT `rachunek_ibfk_1` FOREIGN KEY (`FkKlient`)
         REFERENCES `klient` (`NrKlienta`))
```

- brak klienta o nr 50 w tabeli klient

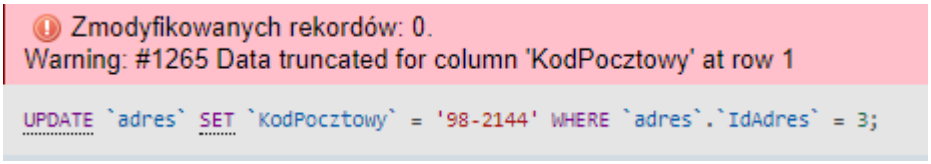
Modyfikacja danych:

Poprawna:



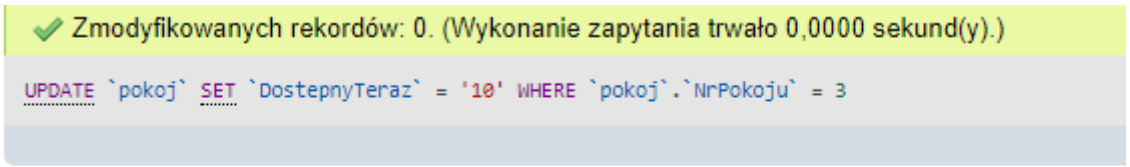


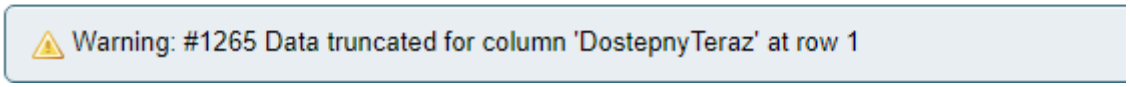
Niepoprawna:



- zbyt wiele znaków w kodzie pocztowym

-Pole KoniecRezerwacji jest wymagane





- nie wprowadzono zmian. Pole DostepnyTeraz jest typu bool.

Usuwanie danych:

Poprawne:

Zmodyfikowanych rekordów: 1. (Wykonanie zapytania trwało 0,0000 sekund(y).)

```
DELETE FROM adres WHERE adres.IdAdres = 22
```

Zmodyfikowanych rekordów: 1. (Wykonanie zapytania trwało 0,0000 sekund(y).)

```
DELETE FROM klient WHERE klient.NrKlienta=17
```

Zmodyfikowanych rekordów: 0. (Wykonanie zapytania trwało 0,0000 sekund(y).)

```
DELETE FROM rezerwacja WHERE NrRezerwacji=5
```

Niepoprawne:

```
DELETE FROM klient
WHERE klient.NrKlienta=2
```

MySQL zwrócił komunikat:

```
#1451 - Cannot delete or update a parent row: a foreign key constraint fails
(`obsluga_hotelu`.`rezerwacja`, CONSTRAINT `rezerwacja_ibfk_1` FOREIGN KEY (`FkKlient`)
REFERENCES `klient` (`NrKlienta`))
```

- klient o nr 2 posiada rezerwacje i nie mogą zostać usunięte jego dane.

```
DELETE FROM adres
WHERE IdAdres=1
```

MySQL zwrócił komunikat:

```
#1451 - Cannot delete or update a parent row: a foreign
    key constraint fails (`obsluga_hotelu`.`hotel`,
   CONSTRAINT `hotel_ibfk_1` FOREIGN KEY (`FkAdres`)
         REFERENCES `adres` (`IdAdres`))
```

- Do adresu o id=2 są przypisani klienci

```
DELETE FROM pokoj
WHERE NrPokoju=1
```

MySQL zwrócił komunikat:

```
#1451 - Cannot delete or update a parent row: a foreign
    key constraint fails (`obsluga_hotelu`.`pobyt`,
   CONSTRAINT `pobyt_ibfk_2` FOREIGN KEY (`FkPokoj`)
         REFERENCES `pokoj` (`NrPokoju`))
```

- W pokoju o nr 2 istnieją pobyty

Działanie Mechanizmów przetwarzania:

Triger „ustaw_dostepny”:

Przed operacją update:

NrPokoju	LiczbaMiejsc	Pietro	TypPokoju	CenaPokoju	DostepnyTeraz	FkNazwa
3	2	1	dwuosobowy	100	0	1

Po operacji update:

```
UPDATE `pobyt` SET `KoniecZameldowania` = '2019-05-08' WHERE `pobyt`.`NrPobytu` = 16;
```

NrPokoju	LiczbaMiejsc	Pietro	TypPokoju	CenaPokoju	DostepnyTeraz	FkNazwa
3	2	1	dwuosobowy	100	1	1

-po wymeldowaniu klienta z danego pokoju zmienia on swój status na dostępny.

```
INSERT INTO `pobyt` (`NrPobytu`, `PoczatekZameldowania`, `KoniecZameldowania`, `FkPokoj`,
`FkKlient`, `FkRezerwacja`) VALUES (NULL, '2019-05-08', '2019-05-08', '3', '12', '6');
```

NrPokoju	LiczbaMiejsc	Pietro	TypPokoju	CenaPokoju	DostepnyTeraz	FkNazwa
3	2	1	dwuosobowy	100	0	1

-Po wykonaniu operacji insert wraz z data wymeldowania pokój widnieje w bazie jako niedostępny, tymczasem nie jest zajęty.

Widok „historia_rezerwacji”:

Pokazano wiersze 0 - ... (Wykonanie zapytania trwało 0,0000 sekund(y).)

```
SELECT * FROM `historia_rezerwacji`
```

PoczatekRezerwacji	KoniecRezerwacji	ImieKlienta	ImieDrugie	NazwiskoKlienta
2019-04-25	2019-04-30	Tomasz	*NULL*	Kubat
2019-04-25	0000-00-00	Przemyslaw	Stanislaw	Kwiatkowski
2019-04-28	2019-05-05	Patryk	*NULL*	Dziatkiewicz
2019-04-28	2019-05-05	Janusz	*NULL*	Szczepaniak
2019-05-20	2019-05-25	Janusz	*NULL*	Szczepaniak
2019-05-11	2019-05-13	Janusz	*NULL*	Tracz
2019-05-16	2019-05-25	Józef	*NULL*	Poniatowski

Procedura „generuj_rachunek”:

Wiersz w tabeli „pokoj”:

NrPokoju	LiczbaMiejsc	Pietro	TypPokoju	CenaPokoju	DostepnyTeraz	FkNazwa
4	2	1	dwuosobowy	100	0	1

Wiersz w tabeli „pobyt”:

NrPobytu	PoczatekZameldowania	KoniecZameldowania	FkPokoj	FkKlient	FkRezerwacja
18	2019-05-05	2019-05-08	4	16	7

Wiersz w tabeli „klient”:

NrKlienta	ImieKlienta	ImieDrugie	NazwiskoKlienta	DataUrodzenia	Obywatelstwo	Email	FkAdres
16	Adam	*NULL*	Nawałka	*NULL*	Polska	*NULL*	21

Wywołanie procedury:

MySQL zwrócił pusty wynik (zero wierszy). (Wykonanie zapytania trwało 0,0000 sekund(y).)

```
call generuj_rachunek(16)
```

Efekt w postaci wygenerowanego wiersza w tabeli „rachunek”

NrRachunku	Oplata	CzasPobytu	FkKlient
7	300	3	16

W przypadku kiedy klient ma aktywny pobyt , wywołanie procedury się nie powiedzie, ponieważ nie możliwe jest obliczenie kosztu pobytu bez znanej daty zakończenia.

```
call generuj_rachunek(6)
```

MySQL zwrócił komunikat:

```
#1048 - Kolumna 'Oplata' nie może być null
```

Wszystkie powyższe operacje przeprowadzano na bazie danych której tablice miały średnią ilość rekordów w przedziale 50-150.