

Zespół Z01:

Kateryna Naumenko

Kacper Kilianek

Ivan Ryzhankow

Bazy Danych 1

Projekt "Hotel"

Celem naszego projektu było stworzenie bazy danych (wraz z funkcjami, sekwencjami itd.), która mogłaby być stosowana w hotelach, wraz z prostą aplikacją w języku programowania JAVA (nie łączymy ten projekt z przedmiotem PAP).

Projekt składa się z plików:

- 1) ddl_script.sql - tworzenie tabel
- 2) data_loader.sql - załadowanie danych
- 3) procedures.sql - funkcje, procedury, wyzwalacze itd.
- 4) tests.sql - testy

Oraz z folderu simple-emp-app-master, który zawiera aplikację w JAVA.

Baza danych posiada tabele:

- 1) ROOM_TYPES – tabela zawierająca informacje o typach pokoi,
- 2) ROOMS – tabela zawierająca ogólne informacje o pokojach w hotelu, tj. Numer pokoju czy jego dostępność,
- 3) AMENITIES – tabela zawierająca informacje o rodzajach udogodnień dla pokoi,
- 4) ROOM_AMENITY – tabela zawiera informacje o tym, jakie udogodnienia są w konkretnym pokoju
- 5) ADDITIONAL_SERVICES – tabela przedstawiająca rodzaje usług dodatkowych,
- 6) GUESTS – tabela zawierająca informacje o gościach hotelowych,
- 7) RESERVATIONS – tabela zawierająca informacje o rezerwacjach,
- 8) OPINIONS – tabela zawierająca informacje o wystawionych opiniach przez gości hotelowych,
- 9) ADDITIONAL_SERVICES_ORDERS – tabela zawierająca informacje o zamówieniach usług dodatkowych przez gości,
- 10) ROOM_RESERVATION – tabela dopasowująca pokoje do rezerwacji,
- 11) POSITIONS – tabela zawierająca informacje o stanowiskach pracowników,
- 12) EMPLOYEES – tabela zawierająca informacje o pracownikach,
- 13) PAYROLL – tabela zawierająca informacje o wypłatach pracowników.

```

classDiagram
    class AdditionalServices {
        # AS ID
        * NAME
        * DESCRIPTION
        * PRICE
    }
    class AdditionalServicesOrders {
        # ASO ID
        # ORDER DATE
    }
    class Opinions {
        # OPINION ID
        * OPINION
        * SCORE
        * OPINION DATE
    }
    class Guests {
        # GUEST ID
        * FIRST NAME
        * LAST NAME
        * ADDRESS
        * PHONE NUMBER
    }
    class RoomReservation {
        # RR ID
    }
    class Reservations {
        # RESERVATION ID
        * CHECK IN
        * CHECK OUT
        * DISCOUNT PERCENT
        * TOTAL PRICE
    }
    class Rooms {
        # ROOM NUMBER
        * NUMBER OF BEDS
        * PRICE PER NIGHT
        * ROOM AVAILABILITY
    }
    class RoomTypes {
        # RT ID
        * ROOM TYPE NAME
    }
    class RoomAmenity {
        # RA ID
    }
    class Amenities {
        # AMENITY ID
        * AMENITY NAME
    }
    class Positions {
        # POSITION ID
        * NAME
        * MIN SALARY
        * MAX SALARY
    }
    class Employees {
        # EMPLOYEE ID
        * FIRST NAME
        * LAST NAME
        * JOB TITLE
        * PHONE
    }
    class Payroll {
        # PAYROLL ID
        * SALARY
    }

    AdditionalServices "1" -- "*" AdditionalServicesOrders : Includes
    AdditionalServicesOrders "1" -- "*" AdditionalServices : Includes
    Opinions "1" -- "*" Guests : Is written by
    Guests "1" -- "*" Reservations : Makes
    Reservations "1" -- "*" Guests : Is made by
    RoomReservation "1" -- "*" Rooms : Includes
    Rooms "1" -- "*" RoomTypes : Includes
    Rooms "1" -- "*" RoomAmenity : Provides access to
    RoomAmenity "1" -- "*" Amenities : Includes
    Amenities "1" -- "*" RoomAmenity : Is Included
    Positions "1" -- "*" Employees : Is occupied by
    Employees "1" -- "*" Payroll : Receives
    Payroll "1" -- "*" Employees : Is paid
  
```

```

    erDiagram
        GUESTS ||--o{ RESERVATIONS : "has"
        GUESTS ||--o{ OPINIONS : "has"
        GUESTS ||--o{ ADDITIONAL_SERVICES : "has"
        ADDITIONAL_SERVICES ||--o{ ADDITIONAL_SERVICES_ORDERS : "has"
        ROOM_TYPES ||--o{ ROOMS : "has"
        ROOMS ||--o{ ROOM_RESERVATION : "has"
        ROOMS ||--o{ ROOM_AMENITY : "has"
        AMENITIES ||--o{ ROOM_AMENITY : "has"
        EMPLOYEES ||--o{ PAYROLL : "has"
        EMPLOYEES ||--o{ POSITIONS : "has"

        GUESTS {
            string FIRST_NAME
            string LAST_NAME
            string ADDRESS
            string PHONE_NUMBER
            number GUEST_ID PK
        }
        ADDITIONAL_SERVICES {
            string NAME
            string DESCRIPTION
            number PRICE
            number AS_ID PK
        }
        ROOM_TYPES {
            string ROOM_TYPE_NAME
            number RT_ID PK
        }
        POSITIONS {
            string NAME
            number MIN_SALARY
            number MAX_SALARY
            number POSITION_ID PK
        }
        RESERVATIONS {
            number GUEST_ID FK
            date CHECK_IN
            date CHECK_OUT
            number DISCOUNT_PERCENT
            number TOTAL_PRICE
            number RESERVATION_ID PK
        }
        OPINIONS {
            number GUEST_ID FK
            string OPINION
            number SCORE
            date OPINION_DATE
            number OPINIONS_ID PK
        }
        ADDITIONAL_SERVICES_ORDERS {
            number GUEST_ID FK
            number SERVICE_ID FK
            date ORDER_DATE
            number ASO_ID PK
        }
        ROOMS {
            number ROOM_TYPE_ID FK
            number NUMBER_OF_BEDS
            number PRICE_PER_NIGHT
            string ROOM_AVAILABILITY
            number ROOM_NUMBER PK
        }
        AMENITIES {
            string AMENITY_NAME
            number AMENITY_ID PK
        }
        EMPLOYEES {
            string FIRST_NAME
            string LAST_NAME
            string JOB_TITLE
            string PHONE
            number POSITION_ID FK
            number EMPLOYEE_ID PK
        }
        ROOM_RESERVATION {
            number RESERVATION_ID FK
            number ROOM_NUMBER FK
            number RR_ID PK
        }
        ROOM_AMENITY {
            number ROOM_NUMBER FK
            number AMENITY_ID FK
            number RA_ID PK
        }
        PAYROLL {
            number SALARY
            number EMPLOYEE_ID FK
            number PAYROLL_ID PK
        }
  
```

- 1) `ad_serv_price(g_id INTEGER)` - funkcja zwracająca koszt wszystkich usług dodatkowych
- 2) `get_total_price (reserv_id INTEGER)` - funkcja zwracająca ostateczny koszt zakwaterowania i obsługi dla danego gościa

- 1) update_pos (p_eid INTEGER, p_pid INTEGER) - procedura, która zmienia stanowisko pracownika
- 2) update_disc (reserv_id INTEGER) - procedura, która zmienia zniżkę w zależności od ilości usług dodatkowych
- 3) update_reservation_total_cost(reserv_id int) - procedura, która zmienia ostateczny koszt zakwaterowania dla podanej rezerwacji
- 4) get_guest_services(c_guest_id in number) - procedura, która zwraca wszystkie zamówione przez gościa dodatkowe usługi wraz z cenami

Wyzwalacze bazy danych:

- 1) reservation_cost_monitor - wyzwalacz, który po dodaniu albo zmianie rezerwacji oblicza nowy koszt zakwaterowania i zmienia ostateczną cenę w tabeli
- 2) payroll_salary_checker - wyzwalacz, który monitoruje czy płaca pracownika mieści się w zakresie przewidzianym przez stanowisko
- 3) update_room_availability - wyzwalacz, który po usunięciu albo dodaniu rezerwacji zmienia status pokoju na wolny/zajęty odpowiednio.

Aplikacja w JAVA została zrealizowana na podstawie projektu <https://gitlab-stud.elka.pw.edu.pl/aszmurlo/simple-emp-app>. Zmieniona została funkcja do sprawdzenia działania połączenia oraz dopracowana możliwość odczytu danych połączenia z pliku connection.properties. Program łączy się z bazą danych, wyświetla pracowników wraz z pensją, a potem zamyka połączenie.

Analiza: przez nas został stworzony pewny system, który reprezentuje działanie hotelu ze strony bazy danych. Wymagania projektowe zostały spełnione, jednak do rozwiązania komercyjnego ten projekt jeszcze się nie nadaje. Do projektu można dopisać dodatkowe wyzwalacze, funkcje itd., które ułatwią pracę z bazą danych dla pracowników, również wtedy należy zadbać o lepszą aplikację w JAVA, która pozwoli zakodować całą część programistyczną (selecty, polecenia ddl), żeby pracownik nie musiał pisać kodu, a wyłącznie wprowadzać odpowiednie dane. Ważnym zagadnieniem do rozwinięcia jest oczywiście kwestia bezpieczeństwa systemu, która była pominięta w ramach projektu (odporność na SQL injection na przykład). I oczywiście do rozwiązania komercyjnego przydałby się graficzny interfejs użytkownika.