Podstawowe założenia projektu

Cel

Projekt opiera się na stworzeniu bazy danych hotelu. Jest to baza złożona z 16 tabel, zawierająca informacje m.in. o rezerwacjach, pracownikach, pokojach oraz usługach do dyspozycji klientów. Baza obsługuje wiele aktywności, wyjaśnione dokładnie w opisie poszczególnych procedur/funkcji oraz wyzwalaczy.

Podstawowe założenia i możliwości:

-w tabelach takich jak: wyżywienie,stanowiska,informacje o rabatach,szczegóły usług, parking, rodzaj pokoju, pokoje, menu restauracji oraz zarobek dzienny, mamy wpisane dane od samego początku. Są one niezbędne do funkcjonowania całej bazy (np. musimy wiedzieć, jakie możliwe usługi dodatkowe można wykupić)

-przed dokonaniem rezerwacji w bazie muszą znajdować się dane klienta do którego przypisana zostanie rezerwacja.

-dane klienta wpisujemy tylko raz – po zakończeniu rezerwacji klient nadal pozostaje w bazie

-jeden klient może dokonać wielu rezerwacji niekoniecznie w różnych terminach

-przy wpisywaniu rezerwacji klient wybiera datę pobytu, rodzaj wyżywienia, pokój oraz ilość mieszkających osób – biorąc pod uwagę te parametry wyliczamy wstępną cenę pobytu.

-po stworzeniu rezerwacji klientowi przypisywane jest wolne miejsce parkingowe, wybrany pokój zmienia status na zajęty.

-podczas pobytu klient może wykupić dodatkowe dania w hotelowej restauracji oraz usługi, których ceny są automatycznie przeliczane przez ewentualny rabat oraz doliczane do całkowitego kosztu pobytu.

-w tabeli „Zarobek dzienny” zakładamy wcześniejsze istnienie (lub ręczne wpisywanie) dat, których przychód jest aktualizowany poprzez wywołanie funkcji podliczającej zarobek hotelu dla danego dnia. Funkcja ta również zwalnia pokoje oraz parkingi rezerwacji kończących się w podanym dniu.

-przy wykupieniu toru do kręgli, cena nie uwzględnia liczby osób grających – płacimy za 1 tor/godz

-zakładamy, że gdy klientowi należy się rabat to recepcjonista weryfikuje tę informację zaraz po dokonaniu rezerwacji i wpisuje odpowiedni rodzaj rabatu do tabeli „Rabaty”. Gdy rabat

jest aktywny, przy obliczaniu ceny dania/usługi do całkowitego kosztu, jest ona odpowiednio przeliczana.

-przyjmujemy, że w każdym momencie na kuchni znajduje się kucharz, który zajmuje się przygotowaniem dania lecz nie przechowujemy informacji, w jakich dniach/godzinach pracuje konkretny z nich.

- *Przyjęte ograniczenia:*

-każda rezerwacja otrzymuje tylko jedno miejsce parkingowe

-jako, że zwolnienie pokoju/parkingu z założenia uruchamiane jest pod koniec dnia, nie możliwe jest przypisanie tego samego pokoju/parkingu dwóm różnym rezerwacją jednego dnia (gdy np. jedna się skończy wcześniej niż zacznie druga)

-nie ma możliwości wyboru dwóch różnych rodzajów wyżywienia dla jednej rezerwacji - ustalany jest raz, dla każdej osoby ten sam.

-nie ma możliwości wcześniejszego zakończenia rezerwacji.

Opis stworzonych widoków oraz funkcji

Widoki:

- *VIEW WIDOK*

Dla każdego stanowiska wyświetla sumę premii w złotówkach pracowników na danym stanowisku. Stworzony głównie jako pomoc do wykonania jednej z funkcji.

Funkcje:

- *FUNC PodliczDzien*

Funkcja przyjmuje jeden parametr w postaci daty i zwraca sumę kosztów wszystkich rezerwacji kończących się w podanym dniu.

- *FUNC Ile_gości*

Zwraca liczbę gości w hotelu w dniu podanym jako parametr funkcji. Używana w funkcji zwracającej stopień zapełnienia hotelu (FUNC Stopien_zapelnienia)

- *FUNC Najlepsze_danie*

Zwraca nazwę dania, które było zamawiane najwięcej razy. Jeżeli jest kilka takich dań, wypisuje wszystkie.

- *FUNC wolnyParking*

Za pomocą kursorów wyszukuje Id pierwszego parkingu, który jest wolny i można przypisać go do rezerwacji.

- *FUNC Średnia_ocena_pokoju*

Zwraca średnią ocenę pokoju wyliczoną na podstawie opinii klientów wystawionych dla pokoju podanego parametrem.

- *FUNC Średnia_ocena*

Zwraca średnią ocenę hotelu - bierze pod uwagę wszystkie opinie klientów.

- *FUNC Najlepszy_Kucharz*

Zwraca Id_kucharza, który zrealizował najwięcej zamówień. Użyta do procedury przydzielającej premię kucharzowi (PROC Premia_dla_kucharza).

- *FUNC wolne_pokoje*

Zwraca tabelę pokoi, które nie są zajęte. Wykorzystywane przy tworzeniu rezerwacji.

- *FUNC Suma_premii*

Zwraca całkowity koszt w złotówkach ponoszony na rzecz comiesięcznej premii dla pracowników.

- *FUNC Stopien_zapelnienia*

Zwraca w (%) stopień zapełnienia hotelu, dzięki czemu możemy kontrolować przestrzeganie aktualnych obostrzeń COVID-owych.

Opis stworzonych procedur składowych

Procedury wstawiające dane:

- *PROC dopREZ*

Procedura dopisuje wiersze do tabeli rezerwacje zabezpieczając przed:

-podaniem id_klienta który nie istnieje w bazie. Błąd ten sugeruje najpierw uzupełnić dane klienta,

-wybraniem pokoju który nie istnieje,

-wprowadzeniem id pokoju który jest zajęty,

-podaniem pracownika obsługującego daną rezerwację który nie istnieje,

-wprowadzeniem daty wymeldowania która następuje wcześniej niż zameldowania,

-wybraniem wyżywienia które nie istnieje,

-wprowadzeniem id_pokoju który ma mniejszy zakres osób niż ilość osób na którą jest dokonywana rezerwacja.

Wyliczana jest odpowiednia cena całkowita zawierająca cenę pokoju oraz wyżywienia na cały pobyt. Cena ta może być aktualizowana poprzez np. dodatkowe usługi czy zamówienia dodatkowo płatnych dań w restauracji.

Na każdą rezerwację przypada 1 darmowe miejsce parkingowe. Wyszukiwanie pierwszego nie zajętego miejsca jest realizowane przez kursor.

Kolumna RodzajPłatności jest typem BIT zatem przyjmujemy, że płatność gotówką to 0,

a kartą 1.

- *PROC dopUSŁ*

Procedura uzupełnia tabelę Usługi zabezpieczając przed wprowadzeniem danych niepoprawnych. Sprawdzana jest:

-zgodność podanego Id_rezerwacji z odpowiadającym dla tabeli Rezerwacje,

-istnienie wybranej nazwy usługi,

-poprawność daty rozpoczęcia usługi oraz zakończenia. Wprowadzone zostało zabezpieczenie przed rozpoczęciem usługi po zakończeniu rezerwacji,

-czy usługa nie została wykupiona na więcej osób niż przewiduje dana rezerwacja.

Odpowiednio została podliczona kwota do zapłaty za skorzystanie z danej usługi. Cena ta jest automatycznie po wstawieniu wierszy do tabeli Usługi dodawana do ceny całkowitej za pobyt na daną rezerwację.

- *PROC dopPRAC*

Procedura uzupełnia tabelę Pracownicy zabezpieczając przed wprowadzeniem:

- niepoprawnej daty urodzenia względem daty zatrudnienia,

-stanowiska które nie istnieje,

-numeru PESEL który ma niepoprawną długość,

-osoby która już istnieje w bazie poprzez sprawdzenie nr. PESEL.

- *PROC dopKLIEN*

Procedura uzupełnia tabelę Klienci zabezpieczając przed wprowadzeniem:

-numeru PESEL który ma niepoprawną długość,

-osoby która już istnieje w bazie.

- *PROC dopRABAT*

Procedura dodaje rekordy do tabeli Rabaty. Tabela ta przechowuje kody przyznające zniżki dla danej rezerwacji na dodatkowe usługi lub zamówienia restauracji.

Dopisując rabaty jesteśmy zabezpieczani przed:

-podaniem rezerwacji która nie istnieje,

-wprowadzeniem kodu rabatu który nie istnieje.

- *PROC dopZamRes*

Procedura dodaje do tabeli Zamówienia Restauracji wszystkie dania które zostały kupione w restauracji hotelu. Nie uwzględniane są posiłki wykupione odgórnie i wydawane jako „wyżywienie.

Dodawane dane są zabezpieczane przed:

-brakiem podanej rezerwacji w tabeli Rezerwacje ,

-nie istnieniem wybranego dania,

-chęcią zakupienia dania po zakończeniu rezerwacji,

-podaniem nie istniejącego kucharza który przygotowywał dany posiłek.

Wyliczana jest zsumowana kwota za dane zamówienie, która później jest uwzględniana w cenie całkowitej dla rezerwacji.

- *PROC Dopisz_podliczenie*

Procedura aktualizuje tabelę Zarobek dzienny wyliczając przychód i dochód na dany dzień.

Wywołuję ona funkcję PodliczDzien() która będzie później opisana. Procedura jest uruchamiana na koniec każdego dnia i sprawia że możemy przechowywać zestawienie zarobków hotelu w czasie różnych dni. Również po jej wykonaniu gdy jakaś rezerwacja akurat tego dnia się kończy to zwalniane jest zajmowane przez nią miejsce parkingowe i pokój.

Opis stworzonych wyzwalaczy

- *TRIGGER zmienStanPokoju*

Wyzwalacz ten służy do zmiany stanu pokoju z 0 na 1 po wstawieniu wierszy do tabeli Rezerwacje.

Przy rezerwacji zawsze wybierany jest pokój którego stan musi przyjąć 1 po dokonaniu rezerwacji czyli zmienić status na zajęty.

- *TRIGGER zmienStanParkingu*

Wyzwalacz analogicznie do opisanego wyżej po wstawieniu wierszy do tabeli Rezerwacje zmienia Stan_rezerwacji z 0 na 1 dla odpowiednich miejsc parkingowych. Zatem ustawia je na zajęte.

- *TRIGGER odlicz_rabat_z_usługi*

Wyzwalacz ten po dodaniu wiersza do tabeli Usługi aktualizuję cenę całkowitą w tabeli Rezerwacje o kwotę usługi. Jednak jeśli dana rezerwacja posiada rabat na usługi to cena za usługę jest zmniejszana zgodnie z rabatem a dopiero później dopisywana do ceny całkowitej w tabeli Rezerwacje. Jednakże w tabeli Usługi widnieje cena nie uwzględniająca rabatu.

- *TRIGGER odlicz_rabat_z_zamowien*

Wyzwalacz po dodaniu zamówienia do tabeli Zamówienia restauracji aktualizuję cenę całkowitą w tabeli Rezerwacje o kwotę zamówienia. Jednak jeśli dana rezerwacja posiada rabat na restaurację to cena za dania jest zmniejszana zgodnie z rabatem a dopiero później dopisywana do ceny całkowitej w tabeli Rezerwacje oraz aktualizowana w tabeli Zamówienia restauracji.

Strategia pielęgnacji bazy danych (kopie zapasowe)

Została utworzona kopia bazy danych, aby po ewentualnej awarii być bezpiecznym czyli móc bez problemu przywrócić dane. Nawet jeśli dokonamy jakiś dużych zmian które

chcielibyśmy szybko cofnąć, a niekoniecznie jest to możliwe, to poprzez odtworzenie kopii można przywrócić wcześniejszy stan bazy danych.

Typowe zapytania

1) sprawdzanie poprawności danych w procedurze służącej do dopisywania danych do tabeli

```
IF NOT EXISTS (SELECT Id_rezerwacji FROM Rezerwacje
                    WHERE Id_rezerwacji = @Id)
BEGIN
RAISERROR('Rezerwacja o podanym id (%d) nie istnieje!',16,1,@Id)
RETURN(1)
END
```

2) zmienianie odpowiednich wartości w precedurach

```
UPDATE Pracownicy
SET Premia = Premia + 5
WHERE Id_pracownika = @ID
RETURN (0)
```

3) zwracanie wartości użytych do działania innych funkcji

```
BEGIN
      DECLARE @Osoby INT
      SET @Osoby = 0
            SET @Osoby = (SELECT SUM(Ilosć_osób) FROM Rezerwacje
                        WHERE Data_wymeldowania >= @Data AND
Data_zameldowania <= @Data)
      RETURN @Osoby
```

4) sprawdzanie warunku decydującego, czy wykonać operację

```
IF EXISTS (SELECT Id_rezerwacji FROM Rabaty R JOIN [Informacje o
rabatach] IOR ON R.Kod = IOR.Kod

WHERE (Id_rezerwacji = @Id AND Opis LIKE '%restauracji%'))
```