**Групповой проект**

**База данных для отеля "Гранд Кисловодск"**

Российские курорты набирают популярность! Не исключение и прекрасный город Кисловодск и в частности - знаменитый отель Гранд Кисловодск, сервису которого позавидуют любые европейские гостиницы Будапешта, Праги, Рима, Осло … перечислять можно бесконечно!

Теперь с помощью таких систем управления базами данных, как PostgreSQL, отель сможет вывести свой менеджмент на новый уровень.

Далее опишем программу, которую мы предлагаем Гранд Кисловодску для более эффективного управления.

**Кому нужна разрабатываемая программа (категории пользователей):**

1. Сотрудники отеля:

* Администраторы, которые занимаются управлением бронированиями и регистрацией гостей.
* Управляющие, которые контролируют работу отеля.
* Персонал, ответственный за обслуживание номеров и уборку.

1. Гости отеля, которые могут просматривать доступные номера, делать бронирования и проверять свои заказы.
2. Менеджеры по продажам, маркетингу и рекламе, которые анализируют данные для формирования стратегий (таргетинг и т.п.) привлечения клиентов и повышения продаж.

**Как она будет использоваться:**

Регистрация гостей

* Администраторы смогут регистрировать новых гостей, также как и гости сами смогут регистрироваться. База данных будет хранить их информацию

Управление бронированиями

* Пользователи смогут просматривать доступные номера, делать, изменять и отменять бронирования. Также как и администраторы смогут подтверждать, изменять и отменять бронирования.

Управление номерами:

* Управляющие смогут отслеживать занятость номеров и контролировать их состояние. Персонал сможет получать информацию о номерах, в которых необходимо убраться, что-либо заменить или починить. А также персонал сможет отмечать те номера и заявки, которые они выполнили. Гости смогут оставлять заявки на уборку и починку*.*

Отчётность:

* Менеджеры смогут генерировать отчеты по загрузке отеля (например, в разные периоды, сезоны и т.п.), доходам и другим необходимым показателям.

Обратная связь:

* Гости смогут оставлять отзывы о пребывании, что поможет улучшить качество услуг.

**Функциональные требования:**

Регистрация и управление аккаунтами гостей:

* Возможность добавления, изменения и удаления информации о гостях.
* Хранение истории бронирований и отзывов для каждого гостя.

Управление бронированиями:

* Создание, изменение и отмена бронирований.
* Поиск доступных номеров по датам, типам и количеству гостей.
* Подтверждение бронирования (например администратором)

Управление номерами:

* Учет состояния номеров (свободен, занят, на уборке и т.д.).
* Хранение информации о номерах (типы номеров и количество мест в номере)
* Возможность добавления новых номеров и изменения характеристик существующих.
* Возможность хранить и управлять заявками на уборку и починку для каждого номера

Отчётность и аналитика:

* Учет загруженности отеля, доходам, отзывам гостей и т.д.
* Анализ статистики бронирований и уровня удовлетворенности гостей.

Обратная связь:

* Возможность оставить отзыв о пребывании в отеле.
* Просмотр отзывов другими пользователями.
* Возможность просматривания отзывов менеджером с целью анализа работы персонала.

**Сущности и их атрибуты:**

Гости (Guests):

* GuestID (PK) – уникальный идентификатор гостя
* FullName – полное имя
* Email – адрес электронной почты
* PhoneNumber – номер телефона
* DateOfBirth – дата рождения
* RegistrationDate – дата регистрации в системе

Номера (Rooms):

* RoomID (PK) – уникальный идентификатор номера
* RoomType – тип номера (одноместный, двухместный, люкс и т.д.)
* Capacity – максимальная вместимость
* Price – цена за ночь
* Status – статус номера (свободен, занят, на уборке, на ремонте)

Бронирования (Bookings):

* BookingID (PK) – уникальный идентификатор бронирования
* GuestID (FK) – ссылка на гостя
* RoomID (FK) – ссылка на номер
* CheckInDate – дата заезда
* CheckOutDate – дата выезда
* BookingDate – дата бронирования
* Status – статус бронирования (подтверждено, отменено)

Отзывы (Reviews):

* ReviewID (PK) – уникальный идентификатор отзыва
* GuestID (FK) – ссылка на гостя
* RoomID (FK) – ссылка на номер
* Rating – оценка (1-5)
* Comment – текст отзыва
* ReviewDate – дата отзыва
* CheckInDate (FK) – дата заезда
* CheckOutDate (FK) – дата выезда

Персонал (Staff):

* StaffID (PK) – уникальный идентификатор сотрудника
* FullName – полное имя
* Position – должность (администратор, управляющий, уборщик и т.д.)
* PhoneNumber – номер телефона
* Email – адрес электронной почты

Заявки (Requests):

* RequestID - уникальный идентификатор заявки
* RequestType - тип заявки (уборка/починка/другое)
* GuestID (FK) - уникальный идентификатор гостя
* StaffID (FK) – уникальный идентификатор сотрудника (если заявка на плановую уборку, например)
* RoomID (FK) – уникальный идентификатор номера
* RequestDate - дата подачи заявки
* RequestStatus - статус заявки (подана/в обработке/в процессе/выполнена/отменена)

**Связи между сущностями:**

Гости могут иметь несколько бронирований, но каждое бронирование принадлежит только одному гостю (1:N).

Номера могут быть забронированы многими гостями (через бронирования), но каждое бронирование связано только с одним номером (1:N).

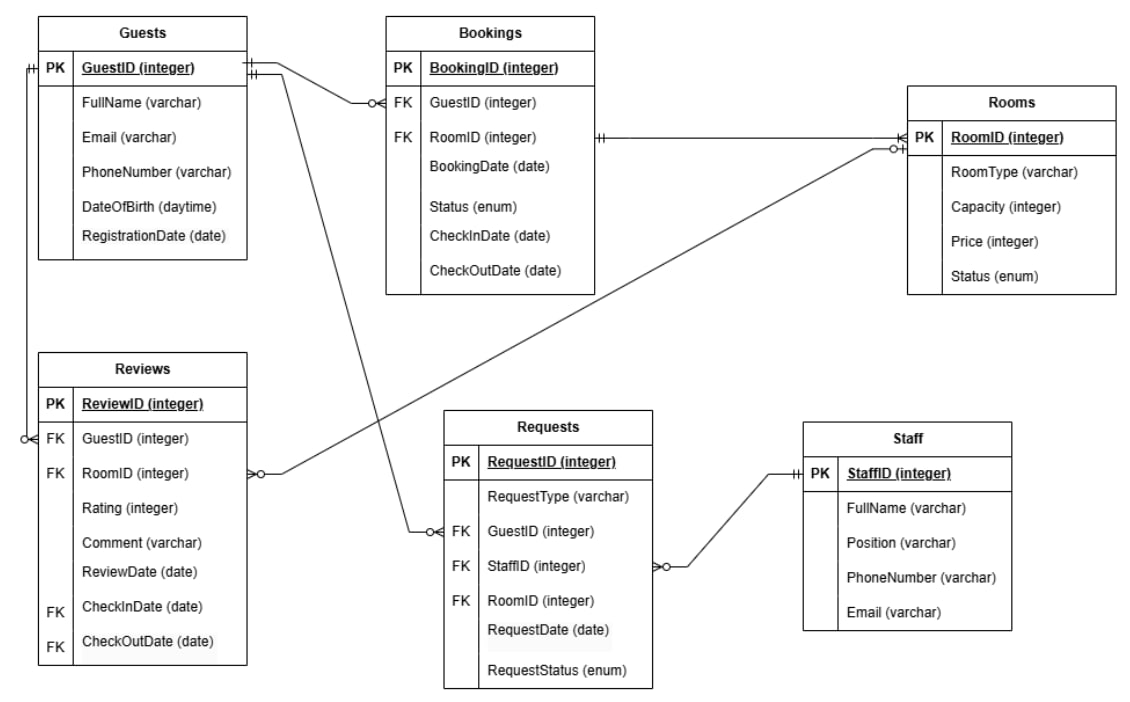
Гости могут оставить несколько отзывов, но каждый отзыв принадлежит только одному гостю (1:N).

Номера могут получить несколько отзывов, но каждый отзыв относится только к одному номеру (1:N).

Гости могут запросить какие-то услуги при желании, однако каждая заявка на услугу оформляется только на одного конкретного гостя (1:N).

За сотрудником может быть закреплено несколько заявок, либо ни одного, если у него выходной день, однако каждая заявка выполняется сотрудником единолично (1:N).

**ER-diagram:**



**Ограничения данных.**

– Email должен соответствовать формату электронной почты (к примеру, [guest1@mail.com](mailto:guest1@mail.com));

– PhoneNumber также должен соответствовать международному стандарту номера телефона – + (код страны) (XXX) XXX-XX-XX;

– DateOfBirth должна быть раньше даты заезда и соответствовать формату XX.XX.XXXX;

– Capacity должно быть целым положительным числом;

– Price – должно быть положительным числом;

– Status для Rooms и Bookings должен находиться в одном из состояний: {“Свободен”, “Занят”};

– Rating должен быть числом от 1 до 5;

– Position должен принимать значения из списка должностей: {“Администратор”, “Уборщик”, “Повар”, “Ремонтник”, “Водитель”};

– RequestType должен принимать значения из списка: {“Поломка”, “Уборка”, “Поездка”, “Пища”};

– RequestStatus должна принимать значения из списка: {“Подана”; “В обработке”, “Выполнена”}.

**Функциональные зависимости:**

Guests:

GuestID => FullName, Email, PhoneNumber, DateOfBirth, RegistrationDate (GuestID определяет остальные атрибуты сущности Guests)

Rooms:

RoomID => RoomType, Capacity, Price, Status (RoomID определяет тип номера, вместимость, цену и статус)

Bookings:

BookingID => GuestID, RoomID, CheckInDate, CheckOutDate, BookingDate, Status (BookingID определяет ID гостя, ID комнаты, дату заезда и дату выезда, дату бронирования и статус)

GuestID, RoomID, CheckInDate => BookingID (комбинация ID гостя, ID комнаты и даты заезда определяет ID бронирования)

Reviews:

ReviewID => GuestID, RoomID, Rating, Comment, ReviewDate, CheckInDate, CheckOutDate (ReviewID определяет ID гостя, ID комнаты, оценку, комментарий, дату отзыва, дату выезда)

GuestID, RoomID, ReviewDate => ReviewID (комбинация из ID гостя, ID комнаты, даты отзыва определяет ID отзыва)

Staff:

StaffID => FullName, Position, PhoneNumber, Email (StaffID определяет полное имя, должность, номер телефона и электронную почту сотрудника)

Requests:

RequestID => RequestType, GuestID, StaffID, RoomID, RequestDate, RequestStatus (RequestID определяет тип заявки, ID гостя, ID сотрудника, ID комнаты, дату заявки, статус заявки)

**Нормализация базы данных.**

Приведение к BCNF происходит разделением двух таблиц.

Во-первых, мы разделяем Bookings:

Bookings: BookingID (PK), GuestID (FK), RoomID (FK), CheckInDate, CheckOutDate, BookingDate, Status

BookingDetails: GuestID (FK), RoomID (FK), CheckInDate (FK), BookingID (FK, уникальный)

Во-вторых, разделяем таблицу Reviews:

Reviews: ReviewID (PK), GuestID (FK), RoomID (FK), Rating, Comment, ReviewDate

ReviewInfo: GuestID (FK), RoomID (FK), CheckInDate, CheckOutDate, ReviewID (FK, уникальный)

Итоговая структура базы данных выглядит таким образом:  
  
Guests: GuestID (PK), FullName, Email, PhoneNumber, DateOfBirth, RegistrationDate

Rooms: RoomID (PK), RoomType, Capacity, Price, Status

Bookings: BookingID (PK), GuestID (FK), RoomID (FK), CheckInDate, CheckOutDate, BookingDate, Status

BookingInfo: GuestID (FK), RoomID (FK), CheckInDate, BookingID (FK, уникальный)

Reviews: ReviewID (PK), GuestID (FK), RoomID (FK), Rating, Comment. ReviewDate

ReviewDetails: GuestID (FK), RoomID (FK), CheckInDate, CheckOutDate, ReviewID (FK, уникальный)

Staff: StaffID (PK), FullName, Position, PhoneNumber, Email

Requests: RequestID (PK), RequestType, GuestID (FK), StaffID (FK), RoomID (FK), RequestDate, RequestStatus

Гости (Guests):

GuestID (PK) – уникальный идентификатор гостя

FullName – полное имя

Email – адрес электронной почты

PhoneNumber – номер телефона

DateOfBirth – дата рождения

RegistrationDate – дата регистрации в системе

Номера (Rooms):

RoomID (PK) – уникальный идентификатор номера

RoomType – тип номера (одноместный, двухместный, люкс и т.д.)

Capacity – максимальная вместимость

Price – цена за ночь

Status – статус номера (свободен, занят, на уборке, на ремонте)

Бронирования (Bookings):

BookingID (PK) – уникальный идентификатор бронирования

GuestID (FK) – ссылка на гостя

RoomID (FK) – ссылка на номер

CheckInDate – дата заезда

CheckOutDate – дата выезда

BookingDate – дата бронирования

Status – статус бронирования (подтверждено, отменено)

BookingInfo:

GuestID (FK) – ссылка на гостя

RoomID (FK) – ссылка на номер

CheckInDate – дата заезд

BookingID (FK) – уникальный идентификатор бронирования

Отзывы (Reviews):

ReviewID (PK) – уникальный идентификатор отзыва

GuestID (FK) – ссылка на гостя

RoomID (FK) – ссылка на номер

Rating – оценка (1-5)

Comment – текст отзыва

ReviewDate – дата отзыва

ReviewDetails:

GuestID (FK) – ссылка на гостя

RoomID (FK) – ссылка на номер

CheckInDate – дата заезда

CheckOutDate – дата выезда

ReviewID (FK) – уникальный идентификатор отзыва

Персонал (Staff):

StaffID (PK) – уникальный идентификатор сотрудника

FullName – полное имя

Position – должность (администратор, управляющий, уборщик и т.д.)

PhoneNumber – номер телефона

Email – адрес электронной почты

Заявки (Requests):

RequestID - уникальный идентификатор заявки

RequestType - тип заявки (уборка/починка/другое)

GuestID (FK) - уникальный идентификатор гостя

StaffID (FK) – уникальный идентификатор сотрудника (если заявка на плановую уборку, например)

RoomID (FK) – уникальный идентификатор номера

RequestDate - дата подачи заявки

RequestStatus - статус заявки (подана/в обработке/в процессе/выполнена/отменена)

**Аномалии при недонормализованной базе.**

Аномалия обновления.

Если информация о датах заезда и выезда будет дублирована в таблице Reviews, то при изменении дат бронирования придется обновлять эту информацию во всех связанных отзывах. Это может привести к ошибкам.

Аномалия удаления.

Если гость удалит все свои бронирования, то могут быть удалены и связанные с ним отзывы.

Избыток данных.

В таблице Bookings дублируется информация о датах заезда и выезда, которая также присутствует в таблице Reviews.

**Создание таблиц в SQL:**

create table Guests(

GuestID INT primary key,

FullName VARCHAR(100) not null,

Email VARCHAR(100) unique,

PhoneNumber VARCHAR(20),

DateOfBirth DATE,

RegistrationDate DATE);

create table Rooms(

RoomID INT primary key,

RoomType VARCHAR(100) not null,

Capacity INT,

Price DECIMAL(10, 2) not null,

Status VARCHAR(100));

create table Bookings(

BookingID INT primary key,

GuestID INT,

RoomID INT,

CheckInDate DATE,

CheckOutDate DATE,

BookingDate DATE,

Status VARCHAR(100),

foreign key (GuestID) references Guests(GuestID),

foreign key (RoomID) references Rooms(RoomID));

create table BookingInfo(

GuestID INT,

RoomID INT,

CheckInDate DATE,

BookingID INT,

foreign key (GuestID) references Guests(GuestID),

foreign key (RoomID) references Rooms(RoomID),

foreign key (BookingID) references Bookings(BookingID));

create table Reviews(

ReviewID INT primary key,

GuestID INT,

RoomID INT,

Rating INT,

comment TEXT,

ReviewDate DATE,

foreign key (GuestID) references Guests(GuestID),

foreign key (RoomID) references Rooms(RoomID));

create table ReviewDetails(

GuestID INT,

RoomID INT,

CheckInDate DATE,

CheckOutDate DATE,

ReviewID INT,

foreign key (GuestID) references Guests(GuestID),

foreign key (RoomID) references Rooms(RoomID),

foreign key (ReviewID) references Reviews(ReviewID));

create table Staff(

StaffID INT primary key,

FullName VARCHAR(100) not NULL,

position VARCHAR(100),

PhoneNumber VARCHAR(25),

Email VARCHAR(100) UNIQUE);

create table Requests(

RequestID INT primary key,

RequestType VARCHAR(50),

GuestID INT,

StaffID INT,

RoomID INT,

RequestDate DATE,

RequestStatus VARCHAR(50),

foreign key (GuestID) references Guests(GuestID),

foreign key (StaffID) references Staff(StaffID),

foreign key (RoomID) references Rooms(RoomID));

**Запросы:**

Продемонстрируем, каким образом указанные в начале функциональные требования могут быть реализованы в созданной базе данных. Количество возможных запросов велико, однако для демонстрации напишем несколько различных запросов:

1) Добавление информации о госте:

insert into Guests (GuestID, FullName, Email, PhoneNumber, DateOfBirth, RegistrationDate)

values (1, 'Брейман Александр Давидович', 'breyman@gmail.com', '+ 7 (777) 777-77-77', '29.03.1972', current\_date);

2) Список свободных двухместных комнат, чтобы понимать, куда заселить новых гостей:

select RoomID, Price from Rooms

where RoomType = 'двухместный' and Status = 'свободен';

3) Информация о всех гостях, которые останавливались в номере 13 (вдруг мы нашли золотую сережку под ковром в номере 13 и хотим перезвонить всех и найти гостя, который ее потерял?):

select b.GuestID, b.CheckInDate, b.CheckOutDate, g.FullName, g.PhoneNumber

from Bookings b

join Guests g on b.GuestID = g.GuestID

where b.RoomID = 13;

4) Представим, что мы хотим сделать акцию и предоставить скидку всем клиентам, которые останавливались у нас минимум 2 раза. Выведем их id, имена и почты, куда и направим информацию об акции:

select g.GuestID, g.FullName, g.Email

from Guests g

join BookingInfo b on g.GuestID = b.GuestID

group by g.GuestID, g.FullName, g.Email

having count(b.BookingID) > 1;

5) Допустим, менеджер хочет посмотреть, какие отзывы оставляли клиенты номеров, в которых убирался сотрудник с id 17:

select r.ReviewID, r.comment

from Reviews r

join Requests re on r.RoomID = re.RoomID

where re.RequestType = 'уборка' and re.StaffID = 17;

6) Проанализируем удовлетворенность гостей по средней оценке за 12 месяцев:

select datepart(month, ReviewDate) MonthOfReview, avg(Rating) AverageRating

from Reviews

where ReviewDate >= date\_sub(current\_date, interval 12 month)

group by datepart(month, ReviewDate)

order by datepart(month, ReviewDate);

7) Посмотрим, какие сотрудники выполняли больше всего запросов (возможно, мы собираемся назначить им вознаграждение). Понятно, что у нас есть сотрудники, которые не выполняют requests, так как их работа заключается в чем-то другом. Они нас сейчас не интересуют, но не обязательно их отфильтровывать в этом запросе, потому что они в любом случае окажутся внизу рейтинга и не помешают нашей задаче:

select s.StaffID, s.FullName, s.position, count(r.RequestID) CountOfRequests

from Staff s

left join Requests r on s.StaffID = r.StaffID

group by s.StaffID, s.FullName, s.position

order by count(r.RequestID) desc

limit 30;

8) Допустим, мы хотим провести реновацию в отеле. Перед этим мы выведем список тех номеров, где чаще всего запрашивают ремонт. Их стоит проверить первыми — вероятно, эти номера больше всего нуждаются в реновации:

select RoomID, count(RequestID) CountOfRequests

from Requests

where RequestType = 'починка'

group by RoomID

order by count(RequestID) desc

limit 50;

9) Кроме того, в нашей базе данных потребуется группировка запросов в транзакции, чтобы никакая информация не потерялась из-за сбоя. Приведем пример группировки запросов в транзакции, который будет наиболее актуальным для нашей базы данных.

Это транзакция, которая обновляет информацию в таблицах с бронированием, комнатами и гостями. В противном случае (если не делать транзакции) может случиться так, что, например, в таблице с бронированиями бронь есть, а в таблице с комнатами написано, что комната свободна.

BEGIN

insert into Guests (GuestID, FullName, Email, PhoneNumber, DateOfBirth, RegistrationDate)

values (1, 'Брейман Александр Давидович', 'breyman@gmail.com', '+ 7 (777) 777-77-77', '29.03.1972', '13.12.2024');

update Rooms

set Status = 'занят'

where RoomID = 17;

insert into Bookings (BookingID, GuestID, RoomID, CheckInDate, CheckOutDate, BookingDate, Status)

values (1, 1, 17, '27.12.2024'', '08.01.2025'', '13.12.2024', 'подтверждено');

insert into BookingInfo (GuestID, RoomID, CheckInDate, CheckOutDate, BookingID)

values (1, 17, '27.12.2024', 1);

COMMIT;