ORACLE

Таблицы

1. Rooms

room\_number NUMBER(10) GENERATED AS IDENTITY(START WITH 1 INCREMENT BY 1)- уникальный идентификатор комнаты

room\_name (NVARCHAR2(50)) (NOT NULL) - название комнаты

room\_capacity (NUMBER(10)) (NOT NULL) - вместимость комнаты

room\_daily\_price (FLOAT(10)) (NOT NULL) - ежедневная стоимость проживания

room\_description (NVARCHAR2(200)) (NOT NULL) - описание комнаты

1. Guests

guest\_id NUMBER(10) GENERATED AS IDENTITY(START WITH 1 INCREMENT BY 1)- уникальный идентификатор гостя

guest\_email (NVARCHAR2(50)) (NOT NULL) - электронная почта гостя

guest\_name (NVARCHAR2(50)) (NOT NULL) - имя гостя

guest\_surname (NVARCHAR2(50)) (NOT NULL) - фамилия гостя

username (NVARCHAR2(50)) (NOT NULL, UNIQUE) - уникальное имя пользователя гостя

guest\_bdate (DATE) (NOT NULL) - дата рождения гостя

1. ServiceTypes

service\_type\_id NUMBER(10) GENERATED AS IDENTITY(START WITH 1 INCREMENT BY 1) - уникальный идентификатор типа услуги

service\_type\_name (NVARCHAR2(50)) (NOT NULL) - название типа услуги

service\_type\_description (NVARCHAR2(200)) (NOT NULL) - описание типа услуги

service\_type\_daily\_price (FLOAT(10)) (NOT NULL) - ежедневная стоимость типа услуги

1. Services

service\_id NUMBER(10) GENERATED AS IDENTITY(START WITH 1 INCREMENT BY 1)- уникальный идентификатор услуги

service\_type\_id (NUMBER(10)) (NOT NULL) - идентификатор типа услуги, FOREIGN KEY REFERENCES ServiceTypes(service\_type\_id)

service\_guest\_id (NUMBER(10)) (NOT NULL) - идентификатор гостя, который заказал услугу, FOREIGN KEY REFERENCES Guests(guest\_id)

service\_start\_date (DATE) (NOT NULL) - дата начала предоставления услуги

service\_end\_date (DATE) (NOT NULL) - дата окончания предоставления услуги

1. Booking

booking\_id NUMBER(10) GENERATED AS IDENTITY(START WITH 1 INCREMENT BY 1) - уникальный идентификатор брони

booking\_room\_id (NUMBER(10)) (NOT NULL) - идентификатор комнаты, на которую оформлена, FOREIGN KEY REFERENCES Rooms(room\_number)

**Представления**:

**BOOKING\_INFO\_VIEW**: Показывает детали брони. Зависимые объекты: таблица GUESTS, ROOMS.

CREATE OR REPLACE VIEW BOOKING\_INFO\_VIEW AS

SELECT

b.booking\_id,

b.booking\_room\_id,

r.room\_name,

g.guest\_id,

g.guest\_name,

g.guest\_surname,

b.booking\_start\_date,

b.booking\_end\_date,

b.booking\_status

FROM BOOKING b

JOIN GUESTS g ON b.booking\_guest\_id = g.guest\_id

JOIN ROOMS r ON b.booking\_room\_id = r.room\_number;

**GuestServiceInfo**: Показывает заказанного сервиса. Зависимые объекты: таблица SERVICES, SERVICE\_TYPES.

CREATE VIEW GuestServiceInfo AS

SELECT

g.guest\_id,

g.guest\_name,

g.guest\_surname,

s.service\_id,

st.service\_type\_name,

s.service\_start\_date,

s.service\_end\_date,

st.service\_type\_daily\_price

FROM GUESTS g

JOIN SERVICES s ON g.guest\_id = s.service\_guest\_id

JOIN SERVICE\_TYPES st ON s.service\_type\_id = st.service\_type\_id;

**Последовательности:**

**seq\_guests**: Генерирует уникальные идентификаторы для таблицы Guests. Нет зависимых объектов.

CREATE SEQUENCE seq\_guests

START WITH 1

INCREMENT BY 1

MAXVALUE 500;

**seq\_rooms**: Генерирует уникальные идентификаторы для таблицы Rooms. Нет зависимых объектов.

CREATE SEQUENCE seq\_rooms

START WITH 1

INCREMENT BY 1

MAXVALUE 600;

Процедуры:

**ADD\_ROOM**: Добавляет новую комнату в таблицу ROOMS. Нет зависимых объектов.

CREATE OR REPLACE PROCEDURE ADD\_ROOM(

p\_room\_name NVARCHAR2 DEFAULT NULL,

p\_room\_capacity NUMBER DEFAULT NULL,

p\_room\_daily\_price FLOAT DEFAULT NULL,

p\_room\_description NVARCHAR2 DEFAULT NULL

) AS

existing\_count NUMBER;

v\_room\_number NUMBER;

BEGIN

IF p\_room\_name IS NULL OR p\_room\_capacity IS NULL OR p\_room\_daily\_price IS NULL OR p\_room\_description IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20029, 'Все параметры должны быть заданы, а именно p\_room\_name, p\_room\_capacity, p\_room\_daily\_price, p\_room\_description.');

END IF;

IF p\_room\_capacity < 0 THEN

RAISE\_APPLICATION\_ERROR(-20006, 'Вместимость комнаты должна быть больше 0.');

END IF;

IF p\_room\_daily\_price <= 0 THEN

RAISE\_APPLICATION\_ERROR(-20007, 'Ежедневная стоимость комнаты должна быть больше 0.');

END IF;

SELECT COUNT(\*) INTO existing\_count FROM ROOMS

WHERE ROOM\_NAME = p\_room\_name AND ROOM\_CAPACITY = p\_room\_capacity;

IF existing\_count > 0 THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Комната с таким именем и вместимостью уже существует.');

END IF;

INSERT INTO ROOMS (room\_name, room\_capacity, room\_daily\_price, room\_description)

VALUES (p\_room\_name, p\_room\_capacity, p\_room\_daily\_price, p\_room\_description) returning room\_number into v\_room\_number;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Комната успешно создана. Номер комнаты: ' || v\_room\_number);

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Произошла ошибка: ' || SQLERRM);

ROLLBACK;

END ADD\_ROOM;

**UPDATE\_ROOM**: Обновляет информацию о комнате в таблицу ROOMS. Нет зависимых объектов.

CREATE OR REPLACE PROCEDURE UPDATE\_ROOM(

p\_room\_number NUMBER DEFAULT NULL,

p\_new\_room\_name NVARCHAR2 DEFAULT NULL,

p\_new\_room\_capacity NUMBER DEFAULT NULL,

p\_new\_room\_daily\_price FLOAT DEFAULT NULL,

p\_new\_room\_description NVARCHAR2 DEFAULT NULL

) AS

v\_existing\_count NUMBER;

v\_existing\_room ROOMS%ROWTYPE;

BEGIN

IF p\_room\_number IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20030, 'Номер комнаты должен быть задан обязательно.');

END IF;

-- Проверка наличия комнаты с указанным номером

SELECT \* INTO v\_existing\_room

FROM ROOMS WHERE room\_number = p\_room\_number;

IF v\_existing\_room.room\_number IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Комната с указанным номером не существует.');

END IF;

IF p\_new\_room\_capacity IS NOT NULL THEN

IF p\_new\_room\_capacity < 0 THEN

RAISE\_APPLICATION\_ERROR(-20006, 'Вместимость комнаты должна быть больше 0.');

END IF;

END IF;

IF p\_new\_room\_daily\_price IS NOT NULL THEN

IF p\_new\_room\_daily\_price <= 0 THEN

RAISE\_APPLICATION\_ERROR(-20007, 'Ежедневная стоимость комнаты должна быть больше 0.');

END IF;

END IF;

UPDATE ROOMS

SET

room\_name = COALESCE(p\_new\_room\_name, v\_existing\_room.room\_name),

room\_capacity = COALESCE(p\_new\_room\_capacity, v\_existing\_room.room\_capacity),

room\_daily\_price = COALESCE(p\_new\_room\_daily\_price, v\_existing\_room.room\_daily\_price),

room\_description = COALESCE(p\_new\_room\_description, v\_existing\_room.room\_description)

WHERE room\_number = p\_room\_number;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Комната успешно обновлена.');

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Произошла ошибка: ' || SQLERRM);

ROLLBACK;

END UPDATE\_ROOM;

**GET\_ROOM\_LIST**: Если не задан параметр идентификатора номера, выводит информацию о всех номера, иначе – о номере с переданным идентификатором.

CREATE OR REPLACE PROCEDURE GET\_ROOM\_LIST(p\_id NUMBER DEFAULT NULL) AS

v\_cursor SYS\_REFCURSOR;

v\_info ROOMS%ROWTYPE;

BEGIN

v\_cursor := Get\_Room\_Cursor(p\_id);

LOOP

FETCH v\_cursor INTO v\_info;

EXIT WHEN v\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Номер комнаты: ' || v\_info.ROOM\_NUMBER ||

', Название: ' || v\_info.ROOM\_NAME ||

', Вместимость: ' || v\_info.ROOM\_CAPACITY ||

', Суточная стоимость: ' || v\_info.ROOM\_DAILY\_PRICE ||

', Описание: ' || v\_info.ROOM\_DESCRIPTION);

END LOOP;

CLOSE v\_cursor;

END GET\_ROOM\_LIST;

**PRE\_BOOKING**: Добавляет новую бронь в таблицу BOOKING. Зависимые объекты: GUESTS, ROOMS

CREATE OR REPLACE PROCEDURE PRE\_BOOKING(

p\_room\_id NUMBER,

p\_guest\_id NUMBER,

p\_start\_date DATE,

p\_end\_date DATE

)

AS

v\_room\_exists NUMBER;

v\_guest\_exists NUMBER;

v\_room\_available NUMBER;

v\_booking\_id NUMBER;

BEGIN

-- Проверка существования гостя

SELECT COUNT(\*) INTO v\_guest\_exists FROM GUESTS WHERE GUEST\_ID = p\_guest\_id;

IF v\_guest\_exists = 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Гость с указанным ID не найден.');

END IF;

-- Проверка существования комнаты

SELECT COUNT(\*) INTO v\_room\_exists FROM ROOMS WHERE ROOM\_NUMBER = p\_room\_id;

IF v\_room\_exists = 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Номер с указанным ID не найден.');

END IF;

-- Проверка доступности комнаты в выбранные даты

SELECT COUNT(\*) INTO v\_room\_available

FROM BOOKING

WHERE BOOKING\_ROOM\_ID = p\_room\_id

AND (

(p\_start\_date BETWEEN BOOKING\_START\_DATE AND BOOKING\_END\_DATE)

OR (p\_end\_date BETWEEN BOOKING\_START\_DATE AND BOOKING\_END\_DATE)

OR (BOOKING\_START\_DATE BETWEEN p\_start\_date AND p\_end\_date)

OR (BOOKING\_END\_DATE BETWEEN p\_start\_date AND p\_end\_date)

);

IF v\_room\_available > 0 THEN

RAISE\_APPLICATION\_ERROR(-20013, 'Номер занят в выбранные даты.');

END IF;

INSERT INTO BOOKING (BOOKING\_ROOM\_ID, BOOKING\_GUEST\_ID, BOOKING\_START\_DATE, BOOKING\_END\_DATE, BOOKING\_STATUS)

VALUES (p\_room\_id, p\_guest\_id, p\_start\_date, p\_end\_date, 'BOOKED')

RETURNING BOOKING\_ID INTO v\_booking\_id;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Бронирование успешно добавлено. ID: '|| v\_booking\_id);

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Произошла ошибка: ' || SQLERRM);

ROLLBACK;

END PRE\_BOOKING;

Функции:

**DELETE\_ROOM**: Удаляет комнату из таблицы ROOMS. В случае успеха возвращает 1, иначе – 0. Нет зависимых объектов.

CREATE OR REPLACE FUNCTION DELETE\_ROOM(

p\_room\_number NUMBER

) RETURN NUMBER AS

v\_existing\_count NUMBER;

BEGIN

IF p\_room\_number IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20030, 'Номер комнаты должен быть задан обязательно.');

END IF;

-- Проверка наличия комнаты с указанным номером

SELECT COUNT(\*) INTO v\_existing\_count

FROM ROOMS WHERE room\_number = p\_room\_number;

IF v\_existing\_count = 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Комната с указанным номером не существует.');

END IF;

DELETE FROM ROOMS WHERE room\_number = p\_room\_number;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Комната успешно удалена.');

RETURN 1;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Произошла ошибка: ' || SQLERRM);

ROLLBACK;

RETURN 0;

END DELETE\_ROOM;

**Get\_Room\_Cursor**: Возращает курсор типа SYS\_REFCURSOR для таблицы ROOMS. Нет зависимых объектов.

CREATE OR REPLACE FUNCTION Get\_Room\_Cursor(p\_id NUMBER DEFAULT NULL) RETURN SYS\_REFCURSOR

AS

result\_cursor SYS\_REFCURSOR;

BEGIN

IF p\_id IS NOT NULL THEN

OPEN result\_cursor FOR

SELECT \* FROM ROOMS WHERE ROOM\_NUMBER = p\_id;

ELSE

OPEN result\_cursor FOR

SELECT \* FROM ROOMS;

END IF;

RETURN result\_cursor;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Произошла ошибка: ' || SQLERRM);

RETURN NULL;

END Get\_Room\_Cursor;

**Calculate\_Stay\_Cost**: Возвращает стоимость проживания в отеле. Зависимые объекты: BOOKING, SERVICES, SERVICE\_TYPES.

CREATE OR REPLACE FUNCTION Calculate\_Stay\_Cost(p\_booking\_id IN NUMBER) RETURN FLOAT AS

v\_total\_cost FLOAT := 0;

v\_current\_user\_id NUMBER;

v\_booking\_state NVARCHAR2(20);

BEGIN

-- Получаем информацию о брони

SELECT b.BOOKING\_GUEST\_ID, b.BOOKING\_STATUS

INTO v\_current\_user\_id, v\_booking\_state

FROM BOOKING b

WHERE b.BOOKING\_ID = p\_booking\_id;

IF v\_booking\_state != 'CONFIRMED' THEN

RAISE\_APPLICATION\_ERROR(-20036, 'Расчет стоимости доступен только для подтвержденных броней.');

END IF;

-- Рассчитываем стоимость проживания

SELECT

((b.BOOKING\_END\_DATE - b.BOOKING\_START\_DATE) \* r.ROOM\_DAILY\_PRICE +

NVL((SELECT SUM((s.SERVICE\_END\_DATE - s.SERVICE\_START\_DATE) \* st.SERVICE\_TYPE\_DAILY\_PRICE)

FROM SERVICES s

JOIN SERVICE\_TYPES st ON s.SERVICE\_TYPE\_ID = st.SERVICE\_TYPE\_ID

WHERE s.SERVICE\_GUEST\_ID = v\_current\_user\_id), 0)) AS TOTAL\_COST

INTO v\_total\_cost

FROM BOOKING b

JOIN ROOMS r ON b.BOOKING\_ROOM\_ID = r.ROOM\_NUMBER

WHERE b.BOOKING\_ID = p\_booking\_id;

RETURN v\_total\_cost;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Данные не найдены: ');

RETURN NULL;

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Произошла ошибка: ' || SQLERRM);

RETURN NULL;

END Calculate\_Stay\_Cost;

**Триггеры:**

**UPDATE\_GUEST\_TRIGGER:** При обновлении данных о гостях импортирует данные в xml-файл

create or replace trigger UPDATE\_GUEST\_TRIGGER

after insert or delete or update

on GUESTS

begin

DBMS\_OUTPUT.PUT\_LINE('Данные о гостях успешно обновлены');

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Произошла ошибка при экспорте гостей: ' || SQLERRM);

end;

MS SQL

**Таблицы:**

1. Rooms

room\_number (INT, GENERATED ALWAYS AS IDENTITY, PRIMARY KEY) - уникальный идентификатор комнаты

room\_name (NVARCHAR(50)) (NOT NULL) - название комнаты

room\_capacity (NUMBER(10)) (NOT NULL) - вместимость комнаты

room\_daily\_price (FLOAT(10)) (NOT NULL) - ежедневная стоимость проживания

room\_description (NVARCHAR(200)) (NOT NULL) - описание комнаты

1. Guests

guest\_id (INT, GENERATED ALWAYS AS IDENTITY, PRIMARY KEY) - уникальный идентификатор гостя

guest\_email (NVARCHAR(50)) (NOT NULL) - электронная почта гостя

guest\_name (NVARCHAR(50)) (NOT NULL) - имя гостя

guest\_surname (NVARCHAR(50)) (NOT NULL) - фамилия гостя

username (NVARCHAR(50)) (NOT NULL, UNIQUE) - уникальное имя пользователя гостя

guest\_bdate (DATE) (NOT NULL) - дата рождения гостя

1. ServiceTypes

service\_type\_id (INT, GENERATED ALWAYS AS IDENTITY, PRIMARY KEY) - уникальный идентификатор типа услуги

service\_type\_name (NVARCHAR(50)) (NOT NULL) - название типа услуги

service\_type\_description (NVARCHAR(200)) (NOT NULL) - описание типа услуги

service\_type\_daily\_price (FLOAT(10)) (NOT NULL) - ежедневная стоимость типа услуги

1. Services

service\_id (INT, GENERATED ALWAYS AS IDENTITY, PRIMARY KEY) - уникальный идентификатор услуги

service\_type\_id (NUMBER(10)) (NOT NULL, FOREIGN KEY REFERENCES ServiceTypes(service\_type\_id) ON DELETE CASCADE) - идентификатор типа услуги

service\_guest\_id (NUMBER(10)) (NOT NULL, FOREIGN KEY REFERENCES Guests(guest\_id) ON DELETE CASCADE) - идентификатор гостя, который заказал услугу

service\_start\_date (DATE) (NOT NULL) - дата начала предоставления услуги

service\_end\_date (DATE) (NOT NULL) - дата окончания предоставления услуги

1. Booking

booking\_id (INT, GENERATED ALWAYS AS IDENTITY, PRIMARY KEY) - уникальный идентификатор брони

booking\_room\_id (NUMBER(10)) (NOT NULL, FOREIGN KEY REFERENCES Rooms(room\_number) ON DELETE CASCADE) - идентификатор комнаты, на которую оформлена.

**Представления**:

**BOOKING\_INFO\_VIEW**: Показывает детали брони. Зависимые объекты: таблица GUESTS, ROOMS.

CREATE OR REPLACE VIEW BOOKING\_INFO\_VIEW AS

SELECT

b.booking\_id,

b.booking\_room\_id,

r.room\_name,

g.guest\_id,

g.guest\_name,

g.guest\_surname,

b.booking\_start\_date,

b.booking\_end\_date,

b.booking\_status

FROM BOOKING b

JOIN GUESTS g ON b.booking\_guest\_id = g.guest\_id

JOIN ROOMS r ON b.booking\_room\_id = r.room\_number;

**GuestServiceInfo**: Показывает заказанного сервиса. Зависимые объекты: таблица SERVICES, SERVICE\_TYPES.

CREATE VIEW GuestServiceInfo AS

SELECT

g.guest\_id,

g.guest\_name,

g.guest\_surname,

s.service\_id,

st.service\_type\_name,

s.service\_start\_date,

s.service\_end\_date,

st.service\_type\_daily\_price

FROM GUESTS g

JOIN SERVICES s ON g.guest\_id = s.service\_guest\_id

JOIN SERVICE\_TYPES st ON s.service\_type\_id = st.service\_type\_id;

Процедуры:

**ADD\_ROOM**: Добавляет новую комнату в таблицу ROOMS. Нет зависимых объектов.

CREATE OR ALTER PROCEDURE ADD\_ROOM

@p\_room\_number INT = NULL,

@p\_room\_name NVARCHAR(50) = NULL,

@p\_room\_capacity INT = NULL,

@p\_room\_daily\_price FLOAT = NULL,

@p\_room\_description NVARCHAR(200) = NULL

AS

BEGIN

DECLARE @existing\_count INT;

DECLARE @existing\_number\_count INT;

IF @p\_room\_name IS NULL OR @p\_room\_capacity IS NULL OR @p\_room\_daily\_price IS NULL OR @p\_room\_description IS NULL

BEGIN

THROW 50000, 'Все параметры должны быть заданы, а именно @p\_room\_name, @p\_room\_capacity, @p\_room\_daily\_price, @p\_room\_description.', 1;

END

IF @p\_room\_capacity < 0

BEGIN

THROW 50001, 'Вместимость комнаты должна быть больше 0.', 1;

END

IF @p\_room\_daily\_price <= 0

BEGIN

THROW 50002, 'Ежедневная стоимость комнаты должна быть больше 0.', 1;

END

SELECT @existing\_count = COUNT(\*)

FROM ROOMS WHERE ROOM\_NAME = @p\_room\_name AND ROOM\_CAPACITY = @p\_room\_capacity;

IF @existing\_count > 0

BEGIN

THROW 50003, 'Комната с таким именем и вместимостью уже существует.', 1;

END

SELECT @existing\_number\_count = COUNT(\*)

FROM ROOMS WHERE ROOM\_NUMBER = @p\_room\_number;

IF @existing\_number\_count > 0

BEGIN

THROW 50003, 'Комната с таким именем и вместимостью уже существует.', 1;

END

INSERT INTO ROOMS (room\_number, room\_name, room\_capacity, room\_daily\_price, room\_description)

VALUES (@p\_room\_number, @p\_room\_name, @p\_room\_capacity, @p\_room\_daily\_price, @p\_room\_description);

COMMIT;

PRINT 'Комната успешно создана. Номер комнаты: ' + CAST(@p\_room\_number AS NVARCHAR(10));

END;

**UPDATE\_ROOM**: Обновляет информацию о комнате в таблицу ROOMS. Нет зависимых объектов.

CREATE OR ALTER PROCEDURE UPDATE\_ROOM

@p\_room\_number INT = NULL,

@p\_new\_room\_name NVARCHAR(50) = NULL,

@p\_new\_room\_capacity INT = NULL,

@p\_new\_room\_daily\_price FLOAT = NULL,

@p\_new\_room\_description NVARCHAR(200) = NULL

AS

BEGIN

DECLARE @v\_existing\_count INT;

DECLARE @v\_existing\_room\_number INT;

DECLARE @v\_existing\_room\_name NVARCHAR(50);

DECLARE @v\_existing\_room\_capacity INT;

DECLARE @v\_existing\_room\_daily\_price FLOAT;

DECLARE @v\_existing\_room\_description NVARCHAR(200);

IF @p\_room\_number IS NULL

BEGIN

THROW 50004, 'Номер комнаты должен быть задан обязательно.', 1;

END

-- Проверка наличия комнаты с указанным номером

SELECT

@v\_existing\_room\_number = room\_number,

@v\_existing\_room\_name = room\_name,

@v\_existing\_room\_capacity = room\_capacity,

@v\_existing\_room\_daily\_price = room\_daily\_price,

@v\_existing\_room\_description = room\_description

FROM ROOMS

WHERE room\_number = @p\_room\_number;

IF @v\_existing\_room\_number IS NULL

BEGIN

THROW 50001, 'Комната с указанным номером не существует.', 1;

END

IF @p\_new\_room\_capacity IS NOT NULL

BEGIN

IF @p\_new\_room\_capacity < 0

BEGIN

THROW 50005, 'Вместимость комнаты должна быть больше 0.', 1;

END

END

IF @p\_new\_room\_daily\_price IS NOT NULL

BEGIN

IF @p\_new\_room\_daily\_price <= 0

BEGIN

THROW 50006, 'Ежедневная стоимость комнаты должна быть больше 0.', 1;

END

END

UPDATE ROOMS

SET

room\_name = COALESCE(@p\_new\_room\_name, @v\_existing\_room\_name),

room\_capacity = COALESCE(@p\_new\_room\_capacity, @v\_existing\_room\_capacity),

room\_daily\_price = COALESCE(@p\_new\_room\_daily\_price, @v\_existing\_room\_daily\_price),

room\_description = COALESCE(@p\_new\_room\_description, @v\_existing\_room\_description)

WHERE room\_number = @p\_room\_number;

COMMIT;

PRINT 'Комната успешно обновлена.';

END;

**GET\_ROOM\_LIST**: Если не задан параметр идентификатора номера, выводит информацию о всех номера, иначе – о номере с переданным идентификатором.

CREATE OR ALTER PROCEDURE GET\_ROOM\_LIST

@p\_id INT = NULL

AS

BEGIN

DECLARE @v\_room\_number INT,

@v\_room\_name NVARCHAR(50),

@v\_room\_capacity INT,

@v\_room\_daily\_price FLOAT,

@v\_room\_description NVARCHAR(200);

DECLARE room\_cursor CURSOR FOR

SELECT ROOM\_NUMBER, ROOM\_NAME, ROOM\_CAPACITY, ROOM\_DAILY\_PRICE, ROOM\_DESCRIPTION

FROM ROOMS

WHERE ROOM\_NUMBER = ISNULL(@p\_id, ROOM\_NUMBER);

OPEN room\_cursor;

FETCH NEXT FROM room\_cursor INTO @v\_room\_number, @v\_room\_name, @v\_room\_capacity, @v\_room\_daily\_price, @v\_room\_description;

WHILE @@FETCH\_STATUS = 0

BEGIN

PRINT 'Номер комнаты: ' + CAST(@v\_room\_number AS NVARCHAR(10)) +

', Название: ' + @v\_room\_name +

', Вместимость: ' + CAST(@v\_room\_capacity AS NVARCHAR(10)) +

', Суточная стоимость: ' + CAST(@v\_room\_daily\_price AS NVARCHAR(10)) +

', Описание: ' + @v\_room\_description;

FETCH NEXT FROM room\_cursor INTO @v\_room\_number, @v\_room\_name, @v\_room\_capacity, @v\_room\_daily\_price, @v\_room\_description;

END;

CLOSE room\_cursor;

END;

**PRE\_BOOKING**: Добавляет новую бронь в таблицу BOOKING. Зависимые объекты: GUESTS, ROOMS

CREATE OR ALTER PROCEDURE PRE\_BOOKING (

@p\_room\_id INT,

@p\_guest\_id INT,

@p\_start\_date DATE,

@p\_end\_date DATE

) AS

BEGIN

DECLARE @v\_room\_exists INT;

DECLARE @v\_guest\_exists INT;

DECLARE @v\_room\_available INT;

DECLARE @v\_booking\_id INT;

-- Проверка существования гостя

SELECT @v\_guest\_exists = COUNT(\*) FROM GUESTS WHERE GUEST\_ID = @p\_guest\_id;

IF @v\_guest\_exists = 0

BEGIN

THROW 51000, 'Гость с указанным ID не найден.', 1;

END;

-- Проверка существования комнаты

SELECT @v\_room\_exists = COUNT(\*) FROM ROOMS WHERE ROOM\_NUMBER = @p\_room\_id;

IF @v\_room\_exists = 0

BEGIN

THROW 51000, 'Номер с указанным ID не найден.', 1;

END;

-- Проверка доступности комнаты в выбранные даты

SELECT @v\_room\_available = COUNT(\*)

FROM BOOKING

WHERE BOOKING\_ROOM\_ID = @p\_room\_id

AND (

(@p\_start\_date BETWEEN BOOKING\_START\_DATE AND BOOKING\_END\_DATE)

OR (@p\_end\_date BETWEEN BOOKING\_START\_DATE AND BOOKING\_END\_DATE)

OR (BOOKING\_START\_DATE BETWEEN @p\_start\_date AND @p\_end\_date)

OR (BOOKING\_END\_DATE BETWEEN @p\_start\_date AND @p\_end\_date)

);

IF @v\_room\_available > 0

BEGIN

THROW 51000, 'Номер занят в выбранные даты.', 1;

END;

-- Вставка записи о бронировании

INSERT INTO BOOKING (BOOKING\_ROOM\_ID, BOOKING\_GUEST\_ID, BOOKING\_START\_DATE, BOOKING\_END\_DATE, BOOKING\_STATUS)

VALUES (@p\_room\_id, @p\_guest\_id, @p\_start\_date, @p\_end\_date, 'BOOKED');

SET @v\_booking\_id = SCOPE\_IDENTITY();

COMMIT;

PRINT 'Бронирование успешно добавлено. ID: ' + CAST(@v\_booking\_id AS NVARCHAR(10));

END;

**DELETE\_ROOM**: Удаляет комнату из таблицы ROOMS. В случае успеха возвращает 1, иначе – 0. Нет зависимых объектов.

CREATE OR ALTER PROCEDURE DELETE\_ROOM

@p\_room\_number INT

AS

BEGIN

DECLARE @v\_existing\_count INT;

IF @p\_room\_number IS NULL

BEGIN

THROW 50003, 'Номер комнаты должен быть задан обязательно.', 1;

END;

SELECT @v\_existing\_count = COUNT(\*)

FROM ROOMS WHERE room\_number = @p\_room\_number;

IF @v\_existing\_count = 0

BEGIN

THROW 50007, 'Комната с указанным номером не существует.', 1;

END;

DELETE FROM ROOMS WHERE room\_number = @p\_room\_number;

COMMIT;

PRINT 'Комната успешно удалена.';

END;

Функцкии:

**Calculate\_Stay\_Cost**: Возвращает стоимость проживания в отеле. Зависимые объекты: BOOKING, SERVICES, SERVICE\_TYPES.

CREATE OR ALTER FUNCTION Calculate\_Stay\_Cost

(@p\_booking\_id INT)

RETURNS FLOAT

AS

BEGIN

DECLARE @v\_total\_cost FLOAT = 0;

DECLARE @v\_current\_user\_id INT;

DECLARE @v\_booking\_state NVARCHAR(20);

-- Получаем информацию о брони

SELECT @v\_current\_user\_id = b.booking\_guest\_id,

@v\_booking\_state = b.booking\_status

FROM BOOKING b

WHERE b.booking\_id = @p\_booking\_id;

IF @v\_booking\_state != 'CONFIRMED'

BEGIN

RETURN NULL; -- или можно выбросить ошибку с помощью THROW

END;

-- стоимость проживания

SELECT @v\_total\_cost = DATEDIFF(day, b.booking\_start\_date, b.booking\_end\_date) \* r.room\_daily\_price +

COALESCE((SELECT SUM(DATEDIFF(day, s.service\_start\_date, s.service\_end\_date) \* st.service\_type\_daily\_price)

FROM SERVICES s

JOIN SERVICE\_TYPES st ON s.service\_type\_id = st.service\_type\_id

WHERE s.service\_guest\_id = @v\_current\_user\_id), 0)

FROM BOOKING b

JOIN ROOMS r ON b.booking\_room\_id = r.room\_number

WHERE b.booking\_id = @p\_booking\_id;

RETURN @v\_total\_cost;

END;

Триггеры:

**UPDATE\_GUEST\_TRIGGER:** При обновлении данных о гостях импортирует данные в xml-файл

CREATE TRIGGER UPDATE\_GUEST\_TRIGGER

ON GUESTS

AFTER INSERT, DELETE, UPDATE

AS

BEGIN

PRINT 'Данные о гостях успешно обновлены';

END;