Báo cáo Final Project

Mục lục

About

Đây là bản báo cáo cho project Cơ sở dữ liệu quản lý khách sạn của môn Thực hành cơ sở dữ liệu (IT3290)

Người thực hiện là nhóm 10 gồm

- Đặng Tiến Dũng - 20215011 (trưởng nhóm)
- Nghiêm Xuân Diện - 20215007
- Nguyễn Thành Đạt - 20215028

Giảng viên hướng dẫn: TS. Nguyễn Thị Oanh

Nêu vấn đề

1. Vấn đề

   Ngày nay, du lịch trở thành một phần quan trọng trong kinh tế dịch vụ và ngành khách sạn tất yếu cũng phát triển vô cùng. Ngoài việc lượng khách hàng ngày càng gia tăng, thì mô hình khách sạn cũng rất đa dạng, phức tạp. Do đó mà các chủ sở hữu khách sạn yêu cầu cần phải có một phần mềm quản lý để có tối ưu hóa quản trị, quản lý, giúp nâng cao tối đa tốc độ, hiệu quả công việc của nhân viên từ đó tăng trải nghiệm khách hàng.

2. Ý tưởng

   Xây dựng một hệ quản trị cơ sở dữ liệu quản lý khách sạn giúp nâng cao hiệu quả quản lý, tăng cường trải nghiệm khách hàng và tạo ra một môi trường kinh doanh khách sạn chuyên nghiệp. Giúp cho các bộ phận dễ dàng quản lý các tài nguyên như thông tin khách hàng, phòng, các đơn đặt, dễ dàng thống kê phân tích lượng dữ liệu đó để đưa ra những chiến dịch hợp lý, hiệu quả.

Phân tích

1. Các đối tượng trong hệ thống
   a. Quản lý khách hàng
      - Lễ tân
      - Chăm sóc khách hàng
   b. Quản lý phòng
2. Chức năng chính
   a. Lễ tân

- Đăng ký cho khách hàng
- Đặt phòng cho khách
- Check-in, check-out khách hàng
- Danh sách các voucher của khách hàng có thể áp dụng
- Danh sách các phòng và lọc theo yêu cầu
- Áp dụng các voucher, tính toán số tiền cuối cùng khách hàng phải trả và thanh toán
- Cập nhật trạng thái phòng

b. CSKH

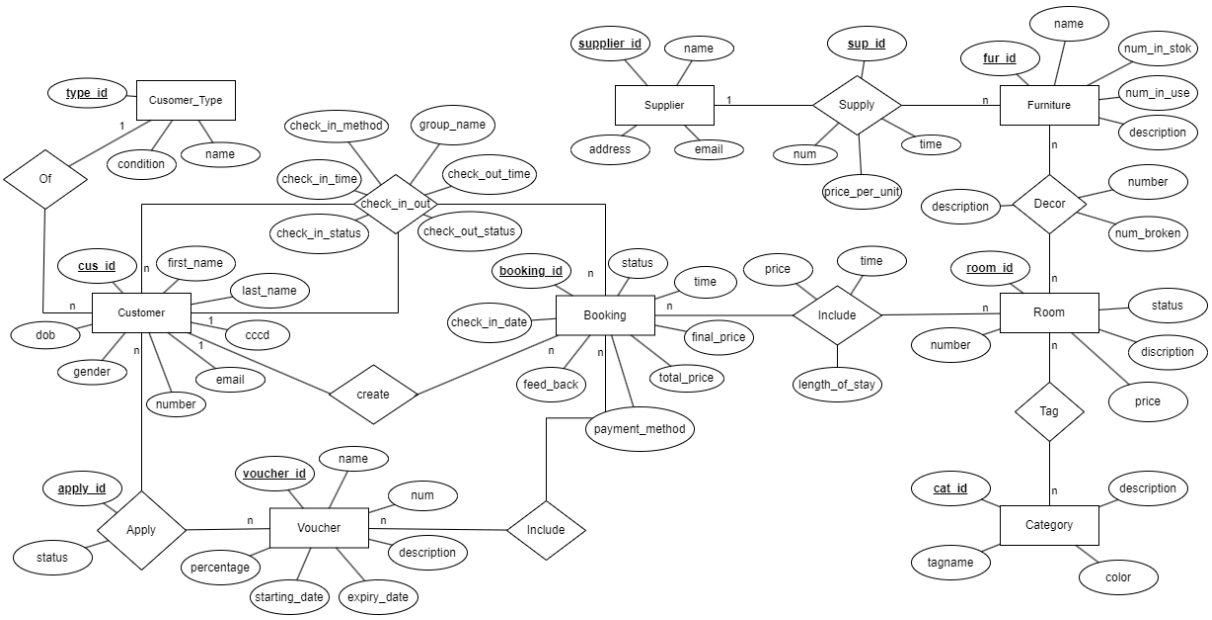
- Tạo, xóa voucher
- Tặng các voucher cho khách hàng
- Thống kê lượng khách hàng check-in, check-out mỗi tháng
- Hiển thị thông tin chi tiết về các voucher đã được sử dụng, đang được sử dụng hoặc sẽ được áp dụng trong thời gian tới
- Hiển thị thông tin cơ bản của khách hàng
- Hiển thị thông tin về từng đơn booking (xem nó đã được thanh toán hay chưa, đơn booking này khách sạn thu về được bao nhiêu,…)

c. Quản lý phòng

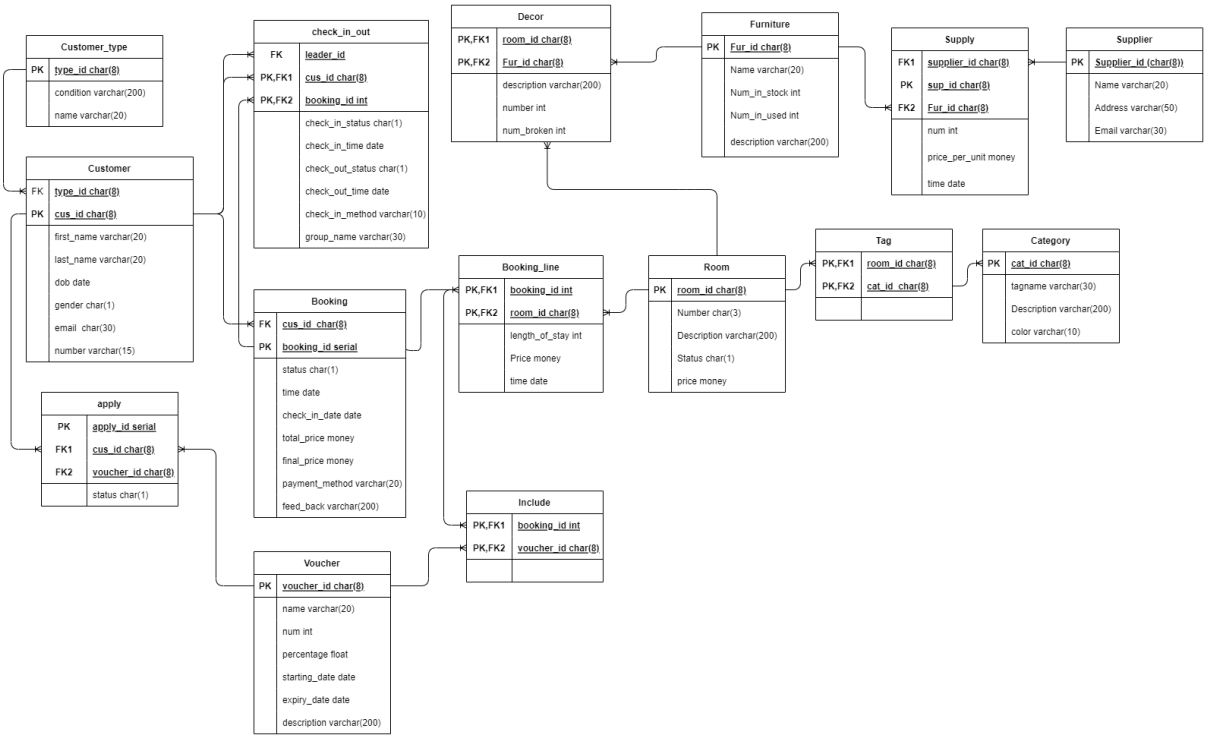
- Quản lý mua nội thất:
  - Thêm, Xóa, Sửa đơn hàng mua nội thất
  - Cập nhật số lượng nội thất trong kho
- Thống kê 3 nhà cung cấp bán nhiều hàng cho khách sạn nhất
- Tính tổng số tiền phải trả khi mua nội thất trong 1 tháng bất kỳ
- Decor phòng:
  - Kiểm tra số lượng nội thất
  - Thêm nội thất vào phòng, Bỏ nội thất ra khỏi phòng
  - Cập nhật số lượng nội thất
- Sửa chữa nội thất:
  - Kiểm tra số lượng nội thất
  - Sửa nội thất
  - Cập nhật số lượng nội thất
- Thống kê các nội thất có trong một phòng bất kỳ
- Thống kê thông tin về các phòng được sử dụng trong vòng 1 tháng
- Thống kê tỉ lệ phòng được thuê mỗi tháng

Thiết kế CSDL

1. Sơ đồ thực thể quan hệ



2. Các bảng theo mô hình quan hệ



3. Mô tả các bảng

a. customer(**cus_id**, *type_id*, first_name, last_name, dob, gender, email, number)

- Bảng lưu các thông tin cơ bản của khách hàng: loại khách, tên, ngày sinh, giới tính, email
- type_id: loại khách, mặc định là 00001111 (người bình thường)

b. customer_type(**type_id**, name, condition)

- Bảng lưu các thông tin về từng loại khách: tên và điều kiện kích hoạt

c. apply(**apply_id**, *cus_id, voucher_id*, status)
   - Bảng lưu thông tin về những lượt tặng voucher cho khách
   - status: trạng thái đã sử dụng hay chưa, mặc định là X (U là sử dụng rồi, X là chưa

d. voucher(**voucher_id**, name, num, percentage, starting_date, expiry_date, description)
   - Bảng lưu thông tin về các voucher: tên, số lượng phát hành, phần trăm giảm giá, ngày bắt đầu có hiệu lực và ngày hết hiệu lực

e. check_in_out(**cus_id, booking_id**, *leader_id*, check_in_status, check_in_time, check_out_status, check_out_time, check_in_method, group_name)
   - Bảng lưu thông tin về hoạt động check-in và check-out của khách hàng với 1 đơn
   - leader_id: id của khách hàng đã đặt đơn(trưởng đoàn)
   - check_in_status: trạng thái check-in của khách (I là đã check-in, X là chưa check-in), mặc định là X
   - check_out_status: trạng thái check-out của khách (O là đã check-out, X là chưa check-out), mặc định là X
   - check_in_time, check_out_time: ngày check-in và check-out, mặc định là ngày hôm nay (NOW()::date)
   - check_in_method: giấy tờ để check-in (cccd, gplx,...), mặc định là cccd

f. booking(**booking_id**, *cus_id*, status, time, check_in_date, total_price, final_price, payment_method, feed_back)
   - Lưu thông tin về 1 đơn hàng
   - status: trạng thái đơn (O là thanh toán rồi, X là chưa thanh toán), mặc định là X
   - time: thời gian đặt, mặc định là hôm nay
   - check_in_date: ngày đến check-in, mặc định là hôm nay
   - total_price: giá trước khi áp dụng voucher, mặc định là 0
   - final_price: giá sau khi áp dụng voucher, mặc định là 0
   - payment_method: phương tiện thanh toán (tiền mặt, thẻ,...), mặc định là cash (tiền mặt)
   - feed_back: đánh giá của khách hàng (nếu có)

g. booking_line(***booking_id, room_id***, length_of_stay, price, time)
   - Lưu thông tin về từng phòng trong 1 đơn
   - time: thời gian đặt phòng, mặc định là hôm nay
   - length_of_stay: khoảng thời gian thuê
   - price: giá khi thuê phòng trong khoảng thời gian trên (= time * length_of_stay), mặc định là 0

h. room(**room_id**, number, description, status, price)
   - Lưu thông tin về các phòng: số phòng, mô tả chung, trạng thái phòng và giá tiền thuê trong 1 ngày 1 đêm
   - status: trạng thái phòng (E là trống có thể sử dụng, U là có người thuê, X là không thể sử dụng), mặc định là E

i. tag(***room_id, cat_id***)
   - Lưu các phép gán thể loại phòng với phòng

j. category(**cat_id**, tagname, color, description)
   - Lưu các thể loại phòng: tên tag, màu, mô tả

k. decor(***room_id, fur_id***, number, num_broken, description)

- Lưu các bài trí nội thất: số lượng, số lượng bị hỏng, mô tả

l. furniture(**fur_id**, name, num_in_stock, num_in_used, description)

- Lưu các nội thất, vật dụng: tên, số lượng trong kho, số lượng đang sử dụng, mô tả

m. supply(**sup_id**, *supplier_id*, *fur_id*, num, price_per_unit, time)

- Lưu các cung cấp nội thất: thời gian mua, số lượng, đơn giá

n. supplier(**supplier_id**, name, address, email)

- Lưu các nhà cung cấp: tên, địa chỉ, email

Các câu truy vấn

1. Query

- Lượng khách hàng check_in theo từng tháng trong năm

```
SELECT EXTRACT (MONTH FROM check_in_time) AS month, COUNT(*) num_check_in FROM check_in_out
WHERE check_in_time <= '2023-12-31' AND check_in_time >= '2023-01-01'
GROUP BY (EXTRACT (MONTH FROM check_in_time))
ORDER BY COUNT(*) DESC
```

- Danh sách phòng đôi còn trống trên tầng 7, 8, 9

```
SELECT * FROM room
JOIN tag USING (room_id)
JOIN category USING (cat_id)
WHERE status = 'E' AND number SIMILAR TO '[7-9]%' AND tagname = 'DOUBLE';
```

- Danh sách các đoàn có trên 4 người

```
SELECT booking_id, leader_id, count(*) as num_member FROM check_in_out
WHERE check_in_time >= '2023-07-01' AND check_in_time <= '2023-07-31'
GROUP BY (booking_id, leader_id)
HAVING (count(*) >= 5);
```

- Danh sách các voucher mà khách hàng có thể sử dụng

```
SELECT * FROM voucher v
JOIN apply a USING (voucher_id)
JOIN customer c USING (cus_id)
WHERE c.cus_id = '00000001' AND status = 'X'
AND ((v.starting_date <= NOW()::date AND (v.expiry_date >= NOW()::date OR v.expiry_date IS NULL)) OR v.num = -1) ;
```

- Danh sách phòng có 2 tag Double và Sea View

```
SELECT r.* FROM room AS r
JOIN tag t USING(room_id)
JOIN category c USING (cat_id)
WHERE c.tagname IN ('Double', 'Sea View')
GROUP BY (r.*)
HAVING count(*) = 2;
```

- Đưa ra 3 nhà cung cấp bán nhiều hàng nhất cho khách sạn (theo tiền mua)

```
select name, sum(supply.num_in_order*supply.price_per_unit) as max_money
from supplier
right join supply using (supplier_id)
group by (supplier_id, name)
order by max_money DESC
LIMIT 3;
```

- Thống kê những phòng có nội thất cần sửa chữa

```
select room.number, furniture.name, decor.num_broken from room, decor, furniture
where (room.room_id = decor.room_id and decor.fur_id = furniture.fur_id and decor.num_broken > 0);
```

- Các loại nột thất có trong phòng bất kỳ

```
select room.room_id, room.number as room_number, furniture.name as fur_name, decor.number as fur_number, decor.num_broken
from room, decor, furniture
where ('103' = room.number and room.room_id = decor.room_id and decor.fur_id = furniture.fur_id);
```

- Hiển thị thông tin các phòng có thể được sử dụng và đang được sử dụng trong 1 tháng trong năm

```
SELECT DISTINCT r.room_id, r.number, r.description, r.status, r.price
FROM room r
LEFT JOIN booking_line bl ON r.room_id = bl.room_id
LEFT JOIN check_in_out cio ON r.room_id = cio.room_id
WHERE (bl.time >= '2023-07-01' AND bl.time < '2023-08-01')
   OR (cio.check_in_time >= '2023-07-01' AND cio.check_in_time < '2023-08-01')
   OR (cio.check_out_time >= '2023-07-01' AND cio.check_out_time < '2023-08-01')
   OR (cio.check_in_time < '2023-07-01' AND cio.check_out_time >= '2023-08-01')
   OR (cio.check_in_time < '2023-07-01' AND cio.check_out_time IS NULL);
```

- Hiển thị thông tin về những đơn hàng đã thanh toán

```
SELECT
 b.booking_id,
 c.first_name || ' ' || c.last_name AS customer_name,
 b.final_price
FROM booking b
INNER JOIN customer c ON b.cus_id = c.cus_id
WHERE b.status = 'O';
```

- Hiển thị thông tin về những voucher đã hết hạn và số lượng khách hàng đã được tặng voucher này

```
SELECT
 v.voucher_id,
 v.name,
 v.expiry_date,
 COUNT(a.voucher_id) AS num_uses
FROM voucher v
LEFT JOIN apply a ON v.voucher_id = a.voucher_id
WHERE v.expiry_date < NOW()::date
GROUP BY v.voucher_id, v.name, v.expiry_date;
```

- In ra tỉ lệ sử dụng phòng

```
SELECT
    COUNT(CASE WHEN status = 'U' THEN 1 END) / (COUNT(*) - COUNT(CASE WHEN status = 'X' THEN 1 END)) AS room_usage_rate
FROM
    room;
```

- In ra số lượng mỗi loại khách hàng của khách sạn

```
SELECT
    ct.type_id,
    ct.name,
    COUNT(c.cus_id) AS num_customers
FROM customer_type ct
LEFT JOIN customer c ON ct.type_id = c.type_id
GROUP BY ct.type_id, ct.name;
```

2. Function

- Áp dụng voucher cho 1 booking

```
CREATE OR REPLACE FUNCTION fnc_apply_voucher(in voucherid char(8), in bookingid integer) RETURNS void AS
$$
DECLARE
  leaderid char(8);
BEGIN
  SELECT cus_id INTO leaderid FROM booking
  WHERE booking_id = bookingid;
  if voucherid NOT IN (SELECT voucher_id FROM voucher v
    JOIN apply a USING (voucher_id)
    JOIN  customer c USING (cus_id)
    WHERE c.cus_id = leaderid AND status = 'X'
    AND (v.starting_date <= NOW()::date AND v.expiry_date >= NOW()::date OR v.num = -1) ) then
    RAISE NOTICE 'Voucher % not found', voucherid;
  return;
  end if;
  if NOT EXISTS (SELECT booking_id FROM booking WHERE booking_id = bookingid) then
    RAISE NOTICE 'Booking % not found', bookingid;
    return;
  end if;
  INSERT INTO include(voucher_id, booking_id) VALUES (voucherid, bookingid);
  UPDATE apply
  SET status = 'U'
  WHERE cus_id = leaderid AND voucher_id = voucherid;
END;
$$
LANGUAGE plpgsql;
```

- Check-in khách hàng

```
CREATE OR REPLACE FUNCTION fnc_check_in(in cusid char(8), in bookingid integer) RETURNS void AS
$$
DECLARE
    checkindate DATE;
    los INTEGER;
    leaderid char(8);
BEGIN
  if NOT EXISTS (SELECT 1 FROM customer WHERE cus_id = cusid) then
    RAISE NOTICE 'Customer % not found', cusid;
  return;
  end if;
  SELECT b.check_in_date INTO checkindate FROM booking b
  WHERE booking_id = bookingid;
  if checkindate IS NULL then
    RAISE NOTICE 'Booking % not found', bookingid;
    return;
  end if;
```

```
  if checkindate > NOW()::date then
    RAISE NOTICE 'Too soon. Coming later';
    return;
  end if;
  SELECT max(length_of_stay) INTO los FROM booking
   JOIN booking_line USING (booking_id)
   WHERE booking_id = bookingid;
  if checkindate + los <= NOW()::date then
    RAISE NOTICE 'Too late. Cant checkin';
    return;
  end if;
  SELECT cus_id INTO leaderid FROM customer
  JOIN booking USING (cus_id)
  WHERE booking_id = bookingid;
  INSERT INTO check_in_out(booking_id, cus_id, leader_id, check_in_status) VALUES (bookingid, cusid, leaderid, 'I');
  return;
END;
$$
LANGUAGE plpgsql;
```

- Check-out khách hàng

```
CREATE OR REPLACE FUNCTION fnc_check_out(in cusid char(8), in bookingid integer) RETURNS void AS
$$
DECLARE
  checkinstatus char(1);
BEGIN
  if NOT EXISTS (SELECT 1 FROM customer WHERE cus_id = cusid) then
    RAISE NOTICE 'Customer % not found', cusid;
    return;
  end if;
  SELECT check_in_status INTO checkinstatus FROM check_in_out
  WHERE booking_id = bookingid AND cus_id = cusid;
  if NOT EXISTS (SELECT 1 FROM booking WHERE booking_id = bookingid) then
    RAISE NOTICE 'Booking % not found', bookingid;
    return;
  end if;
  if checkinstatus = 'X' OR checkinstatus IS NULL then
    RAISE NOTICE 'You have not checked in before';
    return;
  end if;
  UPDATE check_in_out
  SET check_out_status = 'O'
  WHERE booking_id = bookingid AND cus_id = cusid;
END;

$$
LANGUAGE plpgsql;
```

- Tạo đơn

```
CREATE OR REPLACE FUNCTION fnc_cre_booking(in cusid char(8)) RETURNS void AS
$$
DECLARE type char(8);
BEGIN
  SELECT type_id INTO type FROM customer
  WHERE cus_id = cusid;
  if type IS NULL then
  RAISE NOTICE 'Customer % not found', cusid;
  return;
  end if;
  if type = '00001111' then
    RAISE NOTICE 'Provide CCCD pls';
  return;
  end if;
  INSERT INTO booking(cus_id) VALUES (cusid);
  return;
END;
$$
LANGUAGE plpgsql;
```

- Đặt phòng

```
CREATE OR REPLACE FUNCTION fnc_book_room(in bookingid INTEGER, in roomid char(8), in los INTEGER) RETURNS void AS
$$
DECLARE
  roomstatus char(1);
BEGIN
  if (los <= 0) then
    RAISE NOTICE 'length of stay cant be <= 0';
    return;
  end if;
  if NOT EXISTS (SELECT booking_id FROM booking WHERE booking_id = bookingid) then
    RAISE NOTICE 'Booking % not found', bookingid;
    return;
  end if;
  SELECT status INTO roomstatus FROM room WHERE room_id = roomid;
  if (roomstatus IS NULL) then
    RAISE NOTICE 'Room % not found', roomid;
    return;
  end if;
  if (roomstatus = 'U') then
    RAISE NOTICE 'Room % is using by someone else', roomid;
    return;
  end if;
  if (roomstatus = 'X') then
    RAISE NOTICE 'Room % is not available', roomid;
    return;
  end if;
  INSERT INTO booking_line(booking_id, room_id, length_of_stay) VALUES (bookingid, roomid, los);
  return;
END;
$$
LANGUAGE plpgsql;
```

- Tặng voucher cho khách hàng

```
CREATE OR REPLACE FUNCTION fnc_give_voucher(in voucherid char(8), in cusid char(8)) returns void AS
$$
DECLARE
  num_vch INTEGER;
BEGIN
  if NOT EXISTS (SELECT 1 FROM voucher WHERE voucher_id = voucherid ) then
    RAISE NOTICE 'Voucher % not found', voucherid;
  return;
  end if;
  if NOT EXISTS (SELECT 1 FROM customer WHERE cus_id = cusid ) then
    RAISE NOTICE 'Customer % not found', cusid;
  return;
  end if;
  SELECT num into num_vch FROM voucher
  WHERE voucher_id = voucherid;
  if (num_vch >= 0) then
    if num_vch = 0 then
      RAISE NOTICE 'The number of voucher % is zero', voucherid;
      return;
    end if;
    UPDATE voucher
    SET num = num - 1
    WHERE voucher_id = voucherid;
  end if;
  INSERT INTO apply(voucher_id, cus_id) VALUES(voucherid, cusid);
END;
$$
LANGUAGE plpgsql;
```

- Số tiền phải chi để mua nội thất trong 1 tháng

```
create or replace function total_money_in_a_month(in moth double precision, out total_money money) AS
$$
begin
    select into total_money sum(supply.num_in_order*supply.price_per_unit) from supply
    where moth = DATE_PART('month', supply.time);
end;
$$
language plpgsql;
```

- Tổng số nội thất đang cần sửa chữa

```
create or replace function fnc_total_fur_broken(out total_num_broken int) AS
$$
begin
    select into total_num_broken sum(num_broken) from decor;
end;
$$
language plpgsql;
```

- Tổng số nội thất đang sử dụng

```
create or replace function fnc_total_fur_use(out total_num_use int) AS
$$
begin
    select into total_num_use sum(num_in_use) from furniture;
end;
$$
language plpgsql;
```

- In ra thông tin của các phòng chưa được thuê và có giá tiền nhỏ hơn giá được nhập vào

```
CREATE OR REPLACE FUNCTION find_available_rooms_with_price_less_than(price_input NUMERIC)
RETURNS TABLE (
    room_id INT,
    room_number INT,
    description VARCHAR,
    status CHAR,
    price NUMERIC
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        room_id,
        number,
        description,
        status,
        price
    FROM
        room
    WHERE
        status = 'E'
        AND price < price_input
        AND NOT EXISTS (
            SELECT 1
            FROM booking_line bl
            WHERE bl.room_id = room.room_id
        );
END;
$$ LANGUAGE plpgsql;
```

- In ra các phòng chưa được sử dụng theo tagname được nhập vào

```
CREATE OR REPLACE FUNCTION find_rooms_by_tag(tagname_input TEXT)
RETURNS TABLE (room_id INT, number INT, description VARCHAR, status CHAR, price NUMERIC)
AS $$
BEGIN
    RETURN QUERY
    SELECT r.room_id, r.number, r.description, r.price
    FROM room r
    INNER JOIN tag t ON r.room_id = t.room_id
    INNER JOIN category c ON t.cat_id = c.cat_id
    WHERE r.status = 'E' AND c.tagname = tagname_input;
END;
$$
LANGUAGE plpgsql;
```

3. Trigger

- Tặng voucher cho khách hàng đặt lần đầu

```
CREATE FUNCTION trgfnc_defaut_give_voucher() RETURNS TRIGGER AS
$$
BEGIN
  INSERT INTO apply(voucher_id, cus_id) VALUES ('00000000', NEW.cus_id);
  RETURN NEW;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER trg_default_give_voucher
AFTER INSERT ON customer
FOR EACH ROW
EXECUTE PROCEDURE trgfnc_defaut_give_voucher();
```

- Gán loại khách hàng cho những khách hàng đến lần đầu

```
CREATE FUNCTION trgfnc_default_customer_type() RETURNS trigger AS
$$
BEGIN
  if NEW.cccd IS NOT NULL then
    UPDATE customer
    SET type_id = '00000000'
    WHERE cus_id = NEW.cus_id;
  end if;
  return NEW;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER trg_default_customer_type
AFTER INSERT ON customer
FOR EACH ROW
EXECUTE PROCEDURE trgfnc_default_customer_type();
```

- Tính giá tiền trên mỗi phòng trong 1 đơn

```
CREATE OR REPLACE FUNCTION trgfnc_cal_booking_line() RETURNS trigger AS
$$
DECLARE
  price_per_night MONEY;
BEGIN
  if (TG_OP = 'UPDATE') then
    if OLD.booking_id IS DISTINCT FROM NEW.booking_id then
      RAISE NOTICE 'cant change booking_id';
      RETURN NULL;
  end if;
    if OLD.price IS DISTINCT FROM NEW.price then
```

```
      return new;
    end if;
  end if;
  SELECT price INTO price_per_night FROM room
  WHERE room_id = NEW.room_id;
  UPDATE booking_line
  SET price = length_of_stay * price_per_night
  WHERE room_id = NEW.room_id AND booking_id = NEW.booking_id;
  RETURN NEW;
END;
$$
LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER trg_cal_booking_line
AFTER INSERT OR UPDATE ON booking_line
FOR EACH ROW
EXECUTE FUNCTION trgfnc_cal_booking_line();
```

- Tính giá tiền tổng cộng (trước khi áp dụng voucher)

```
CREATE OR REPLACE FUNCTION trgfnc_cal_total() RETURNS trigger AS
$$
BEGIN
  if (TG_OP = 'INSERT') then
    UPDATE booking
    SET total_price = total_price + NEW.price
    WHERE booking_id = NEW.booking_id;
    RETURN NEW;
  end if;
  if (TG_OP = 'UPDATE') then
    if NEW.booking_id IS DISTINCT FROM OLD.booking_id then
      RAISE NOTICE 'U cant change booking_id';
      RETURN NULL;
    end if;
    if NEW.length_of_stay <= 0 then
      RAISE NOTICE 'los cant <= 0';
      RETURN NULL;
    end if;
    UPDATE booking
    SET total_price = total_price + NEW.price - OLD.price
    WHERE booking_id = NEW.booking_id;
    RETURN NEW;
  end if;
  UPDATE booking
  SET total_price = total_price - OLD.price
  WHERE booking_id = OLD.booking_id;
  RETURN NULL;
END;
$$
LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER trg_cal_total_bkline_aftinsdel
AFTER INSERT OR DELETE ON booking_line
FOR EACH ROW
EXECUTE FUNCTION trgfnc_cal_total();

CREATE OR REPLACE TRIGGER trg_cal_total_bkline_befupd
BEFORE UPDATE ON booking_line
FOR EACH ROW
EXECUTE FUNCTION trgfnc_cal_total();
```

- Tính giá tiền cuối cùng (sau khi áp dụng voucher)

```
CREATE OR REPLACE FUNCTION trgfnc_cal_final() RETURNS trigger AS
$$
DECLARE
  percent float;
  final money;
  total money;
  bookingid int;
```

```
BEGIN
  if (TG_OP = 'INSERT' OR TG_OP = 'UPDATE') then
    SELECT total_price, booking_id INTO total, bookingid FROM booking
    WHERE booking_id = NEW.booking_id;
  else
    SELECT total_price, booking_id INTO total, bookingid FROM booking
    WHERE booking_id = OLD.booking_id;
  end if;
  final = total;
  for percent in (SELECT percentage FROM voucher
    JOIN include USING (voucher_id)
    WHERE booking_id = bookingid) loop
    final = final - total * percent / 100;
  end loop;
  UPDATE booking
  SET final_price = final
  WHERE booking_id = bookingid;
  RETURN NULL;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER trg_cal_final_include
AFTER INSERT OR DELETE ON include
FOR EACH ROW
EXECUTE FUNCTION trgfnc_cal_final();

CREATE TRIGGER trg_cal_final_booking
AFTER UPDATE ON booking
FOR EACH ROW
WHEN (NEW.total_price IS DISTINCT FROM OLD.total_price)
EXECUTE FUNCTION trgfnc_cal_final();
```

- Cập nhật số lượng nội thất sau khi đặt, sửa, hủy đơn

```
create or replace function tgfnc_af_supply()
returns trigger as
$$
begin
    if (TG_OP = 'INSERT')
    then
        update furniture
        set num_in_stock = num_in_stock + new.num_in_order
        where furniture.fur_id = new.fur_id;
        return new;

    elsif (TG_OP = 'DELETE')
    then
        update furniture
        set num_in_stock = num_in_stock - old.num_in_order
        where furniture.fur_id = old.fur_id;
        return old;

    elsif (TG_OP = 'UPDATE')
    then
        update furniture
        set num_in_stock = num_in_stock + new.num_in_order
        where furniture.fur_id = new.fur_id;

        update furniture
        set num_in_stock = num_in_stock - old.num_in_order
        where furniture.fur_id = old.fur_id;

        return new;
  end if;

  return null;
end;
$$
language plpgsql;

create trigger af_in_supply
```

```
after insert on supply
for each row
execute procedure tgfnc_af_supply();

create trigger af_de_supply
after delete on supply
for each row
execute procedure tgfnc_af_supply();

create trigger af_up_supply
after update on supply
for each row
execute procedure tgfnc_af_supply();
```

- Kiểm tra số lượng nội thất trước khi decor

```
create or replace function tgfnc_bf_decor()
returns trigger as
$$
declare id_id integer;
begin
    if (TG_OP = 'INSERT')
    then
        select into id_id num_in_stock from furniture
        where furniture.fur_id = new.fur_id;
        if (new.number > id_id)
        then
            raise Notice 'So luong khong du de decor!';
            return null;
        elsif (new.number < 0)
        then
            raise Notice 'So luong bi am! Hay nhap lai!';
            return null;
        end if;
        return new;

    elsif (TG_OP = 'UPDATE')
    then
        select into id_id num_in_stock from furniture
        where furniture.fur_id = new.fur_id;
        if (new.number - old.number > id_id)
        then
            raise Notice 'So luong khong du de decor!';
            return null;
        elsif (new.number < 0)
        then
            raise Notice 'So luong bi am! Hay nhap lai!';
            return null;
        end if;
        return new;
    end if;

    return null;
end;
$$
language plpgsql;

create trigger bf_in_decor
before insert on decor
for each row
execute procedure tgfnc_bf_decor();

create trigger bf_up_decor
before update on decor
for each row
execute procedure tgfnc_bf_decor();
```

- Cập nhật số lượng nội thất sau khi decor

```
create or replace function tgfnc_af_decor()
returns trigger as
$$
begin
    if (TG_OP = 'INSERT')
    then
        update furniture
        set num_in_stock = num_in_stock - new.number
        where furniture.fur_id = new.fur_id;

        update furniture
        set num_in_use = num_in_use + new.number
        where furniture.fur_id = new.fur_id;

        return new;

    elsif (TG_OP = 'UPDATE')
    then
        update furniture
        set num_in_stock = num_in_stock - (new.number - old.number)
        where furniture.fur_id = new.fur_id;

        update furniture
        set num_in_use = num_in_use + (new.number - old.number)
        where furniture.fur_id = new.fur_id;

        if(new.num_broken < old.num_broken)
        then
            update furniture
            set num_in_use = num_in_use - (old.num_broken - new.num_broken)
            where furniture.fur_id = new.fur_id;
        end  if;

        return new;
    elsif (TG_OP = 'DELETE')
    then
        update furniture
        set num_in_stock = num_in_stock + old.number
        where furniture.fur_id = old.fur_id;

        update furniture
        set num_in_use = num_in_use - old.number
        where furniture.fur_id = old.fur_id;

        return new;

    end if;

    return null;
end;
$$
language plpgsql;

create trigger af_in_decor
after insert on decor
for each row
execute procedure tgfnc_af_decor();

create trigger af_up_decor
after update on decor
for each row
execute procedure tgfnc_af_decor();

create trigger af_de_decor
after delete on decor
for each row
execute procedure tgfnc_af_decor();
```

- Kiểm tra số lượng num_broken nhập vào

```
create or replace function tgfnc_bf_decor_num_broken() returns trigger as
$$
begin
    if (new.num_broken < 0 or new.num_broken > old.number)
    then
        raise Notice 'So luong nhap bi loi, yeu cau nhap lai!';
        return null;
  end if;
    return new;
end;
$$
language plpgsql;

create trigger bf_up_decor_num_broken
before update on decor
for each row
when (new.num_broken is distinct from old.num_broken)
execute procedure tgfnc_bf_decor_num_broken();
```

- Cập nhật trạng thái phòng khi có nội thất bị hỏng và sau khi sửa chữa

```
create or replace function tgfnc_update_status_room() returns trigger as
$$
declare decor_room_id char(8);
begin
    for decor_room_id in (select room_id  from decor group by room_id having sum(num_broken)>0)
    loop
        update room
        set status = 'X'
        where room.room_id =decor_room_id;
    end loop;

    for decor_room_id in (select room_id  from decor group by room_id having sum(num_broken)=0)
    loop
        update room
        set status = 'E'
        where room.room_id =decor_room_id;
    end loop;
    return new;
end;
$$
language plpgsql;

create trigger af_up_decor_room
after update on decor
for each row
when (new.num_broken is distinct from old.num_broken)
execute procedure tgfnc_update_status_room();
```

- Cập nhật trạng thái phòng theo thời gian hiện tại

```
CREATE OR REPLACE FUNCTION update_room_status()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.check_in_date + NEW.length_of_stay >= NOW()::date AND NEW.check_in_date <= NOW()::date THEN
        UPDATE room
        SET status = 'U'
        WHERE room_id = (SELECT room_id FROM booking_line WHERE booking_id = NEW.booking_id);
    ELSE
        UPDATE room
        SET status = 'E'
        WHERE room_id = (SELECT room_id FROM booking_line WHERE booking_id = NEW.booking_id);
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER update_room_status_trigger
BEFORE UPDATE ON booking
FOR EACH ROW
EXECUTE FUNCTION update_room_status();

CREATE TRIGGER update_room_status_trigger
BEFORE UPDATE ON booking_line
FOR EACH ROW
EXECUTE FUNCTION update_room_status();
```

Hướng phát triển

1. Mở rộng mô hình để cho nhiều đối tượng sử dụng hơn
    - Quản lý nhân sự, hành chính
        - Thêm các thực thể, quan hệ như nhân viên, lịch, lương, login,...
        - Thêm các chức năng như xếp lịch, xếp role
    - Quản lý nhà xe:
        - Thêm các thực thể, quan hệ như xe, bãi đỗ xe, kiểu xe
        - Thêm các chức năng: gửi xe, lấy xe
2. Thêm các ràng buộc vào các thuộc tính
    - Các thuộc tính tiền, thuộc tính số ngày không thể âm
    - Kiểm tra các thuộc tính ngày check_in, check_out
3. Tạo 1 website có thể sử dụng database trên