

VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING



DATABASE SYSTEM LAB (CO2014)

RESORT MANAGEMENT

Class CC03

Instructor: Nguyen Thanh Binh
Trinh Son Lam 1852502

HO CHI MINH CITY, NOVEMBER 2022



Contents

1	Part1: Create Database System	2
1.1	Create database table and constrains	2
1.1.1	Branch	2
1.1.2	BranchPicture	2
1.1.3	Zone	2
1.1.4	RoomType	3
1.1.5	BedInfo	3
1.1.6	BranchHavRoomType	3
1.1.7	Room	4
1.1.8	SpicimenType	4
1.1.9	SpicTypeInRoomType	4
1.1.10	Material	5
1.1.11	Provider	5
1.1.12	ProviderMaterial	5
1.1.13	Customer	6
1.1.14	ServicePackage	6
1.1.15	BillService	6
1.1.16	TypeForDiscount	7
1.1.17	BillBooking	7
1.1.18	RentingRoom	7
1.1.19	PaymentBill	8
1.1.20	Company	8
1.1.21	Service	8
1.1.22	ServiceSpa	8
1.1.23	TpeSouvenir	9
1.1.24	BrandSouvenir	9
1.1.25	AreaPlace	9
1.1.26	ShopPicture	10
1.1.27	TimelineShop	10
1.2	Insert data to tables	11
2	Part2: Store Procedure/Function and Trigger	11
2.1	Store Procedure/Function	11
2.1.1	Procedure/Function for ServicePackage(GoiDichVu)	11
2.1.2	Procedure/Function for StatisticGuests	12
2.2	Triggers	13
2.2.1	Trigger to update values	13
3	Part3: Build an application	15
3.1	Create user	15
3.2	Some features	16
3.2.1	Login/Logout	16
3.2.2	Features	16

1 Part1: Create Database System

1.1 Create database table and constrains

1.1.1 Branch

```
CREATE TABLE Branch(  
    number_id SERIAL NOT NULL,  
    branch_id TEXT GENERATED ALWAYS AS ('CN' || number_id::text) STORED,  
    province TEXT,  
    address TEXT,  
    phone_num VARCHAR(12),  
    email TEXT,  
    PRIMARY KEY(branch_id)  
);
```

Figure 1: TABLE 1

1.1.2 BranchPicture

```
CREATE TABLE BranchPicture(  
    branch_id TEXT ,  
    link_image TEXT,  
    PRIMARY KEY(branch_id,link_image),  
    FOREIGN KEY(branch_id) REFERENCES Branch(branch_id)  
);
```

Figure 2: TABLE 2

1.1.3 Zone

```
CREATE TABLE Zone(  
    branch_id TEXT,  
    name_zone TEXT,  
    PRIMARY KEY(branch_id, name_zone),  
    FOREIGN KEY(branch_id) REFERENCES Branch(branch_id)  
);
```

Figure 3: TABLE 3

1.1.4 RoomType

```
CREATE TABLE RoomType(  
    roomtype_id SERIAL NOT NULL,  
    room_name TEXT,  
    area_square TEXT,  
    num_guests INT NOT NULL CHECK(num_guests > 0 AND num_guests < 11),  
    description TEXT,  
    PRIMARY KEY(roomtype_id)  
);
```

Figure 4: TABLE 4

1.1.5 BedInfo

```
CREATE TABLE BedInfo(  
    roomtype_id SERIAL,  
    bed_size NUMERIC(2,1),  
    quantity INT NOT NULL DEFAULT 1 CHECK(quantity > 0 AND quantity < 11),  
    PRIMARY KEY(roomtype_id, bed_size),  
    FOREIGN KEY(roomtype_id) REFERENCES RoomType(roomtype_id)  
);
```

Figure 5: TABLE 5

1.1.6 BranchHavRoomType

```
CREATE TABLE BranchHavRoomType(  
    roomtype_id SERIAL,  
    branch_id TEXT,  
    price INT NOT NULL,  
    PRIMARY KEY(roomtype_id, branch_id),  
    FOREIGN KEY(roomtype_id) REFERENCES RoomType(roomtype_id),  
    FOREIGN KEY(branch_id) REFERENCES Branch(branch_id)  
);
```

Figure 6: TABLE 6

1.1.7 Room

```
CREATE TABLE Room(  
    branch_id TEXT,  
    num_room VARCHAR(3),  
    roomtype_id SERIAL,  
    name_zone TEXT,  
    PRIMARY KEY(branch_id,num_room),  
    FOREIGN KEY (branch_id) REFERENCES Branch(branch_id),  
    FOREIGN KEY(branch_id,name_zone) REFERENCES Zone(branch_id,name_zone),  
    FOREIGN KEY(roomtype_id) REFERENCES RoomType(roomtype_id)  
);
```

Figure 7: TABLE 7

1.1.8 SpicimenType

```
CREATE TABLE SpicimenType(  
    spicimen_id VARCHAR(6) CHECK(spicimen_id ~ 'VT[0-9]{4}'),  
    spicimen_name TEXT,  
    PRIMARY KEY(spicimen_id)  
);
```

Figure 8: TABLE 8

1.1.9 SpicTypeInRoomType

```
CREATE TABLE SpicTypeInRoomType(  
    spicimen_id VARCHAR(6),  
    roomtype_id SERIAL,  
    quantity_spic INT NOT NULL DEFAULT 1 CHECK(quantity_spic > 0 AND quantity_spic < 21),  
    PRIMARY KEY(spicimen_id,roomtype_id),  
    FOREIGN KEY(spicimen_id) REFERENCES SpicimenType(spicimen_id),  
    FOREIGN KEY(roomtype_id) REFERENCES RoomType(roomtype_id)  
);
```

Figure 9: TABLE 9

1.1.10 Material

```
CREATE TABLE Material(  
    branch_id TEXT,  
    specimen_id VARCHAR(6),  
    num_material INT CHECK(num_material > 0),  
    status TEXT,  
    num_room VARCHAR(3),  
    PRIMARY KEY(branch_id,specimen_id,num_material),  
    FOREIGN KEY(branch_id) REFERENCES Branch(branch_id),  
    FOREIGN KEY(specimen_id) REFERENCES SpecimenType(specimen_id),  
    FOREIGN KEY(branch_id,num_room) REFERENCES Room(branch_id,num_room)  
);
```

Figure 10: TABLE 10

1.1.11 Provider

```
CREATE TABLE Provider(  
    provider_id VARCHAR(7) CHECK(provider_id ~ 'NCC[0-9]{4}'),  
    provider_name TEXT,  
    provider_email TEXT,  
    provider_address TEXT,  
    PRIMARY KEY(provider_id)  
);
```

Figure 11: TABLE 11

1.1.12 ProviderMaterial

```
CREATE TABLE ProviderMaterial(  
    provider_id VARCHAR(7),  
    specimen_id VARCHAR(6),  
    branch_id TEXT,  
    PRIMARY KEY(specimen_id,branch_id),  
    FOREIGN KEY(provider_id) REFERENCES Provider(provider_id),  
    FOREIGN KEY(specimen_id) REFERENCES SpecimenType(specimen_id),  
    FOREIGN KEY(branch_id) REFERENCES Branch(branch_id)  
);
```

Figure 12: TABLE 12

1.1.13 Customer

```
CREATE TABLE Customer(  
  customer_id VARCHAR(8) CHECK(customer_id ~ 'KH[0-9]{6}'),  
  cccd_cmnd VARCHAR(12) NOT NULL UNIQUE,  
  customer_name TEXT,  
  customer_phone TEXT UNIQUE,  
  customer_email TEXT UNIQUE,  
  username TEXT UNIQUE,  
  cus_password TEXT,  
  points INT NOT NULL DEFAULT 0 CHECK(points >= 0),  
  customer_type INT NOT NULL DEFAULT 1 CHECK(customer_type > 0 AND customer_type < 5),  
  PRIMARY KEY(customer_id)  
);
```

Figure 13: TABLE 13

1.1.14 ServicePackage

```
CREATE TABLE ServicePackage(  
  package_name TEXT,  
  num_days INT NOT NULL CHECK(num_days > 0 AND num_days < 101),  
  num_guests INT NOT NULL CHECK(num_guests > 0 AND num_guests < 11),  
  price INT NOT NULL,  
  PRIMARY KEY(package_name)  
);
```

Figure 14: TABLE 14

1.1.15 BillService

```
CREATE TABLE BillService(  
  customer_id VARCHAR(8),  
  package_name TEXT,  
  time_buying TIMESTAMP,  
  time_starting DATE CHECK(time_starting > time_buying),  
  total_price INT ,  
  PRIMARY KEY(customer_id,package_name,time_buying),  
  FOREIGN KEY(customer_id) REFERENCES Customer(customer_id),  
  FOREIGN KEY(package_name) REFERENCES ServicePackage(package_name)  
);
```

Figure 15: TABLE 15

1.1.16 TypeForDiscount

```
CREATE TABLE TypeForDiscount(  
    customer_type INT,  
    discount REAL,  
    point_type INT,  
    PRIMARY KEY(customer_type)  
);
```

Figure 16: TABLE 16

1.1.17 BillBooking

```
CREATE OR REPLACE FUNCTION Generate_Room_Booking_ID(timestamp,INT) RETURNS text AS  
$$  
    SELECT (to_char($1, 'DD') || to_char($1, 'MM') || to_char($1, 'YYYY'))||LPAD($2::text, 6, '0');  
$$ LANGUAGE sql IMMUTABLE;  
CREATE TABLE BillBooking(  
    book_id SERIAL NOT NULL,  
    booking_id VARCHAR(16) GENERATED ALWAYS AS ('DP' || Generate_Room_Booking_ID(time_booking,book_id)) STORED,  
    time_booking TIMESTAMP,  
    time_checkin DATE CHECK(time_checkin > time_booking),  
    time_checkout DATE CHECK(time_checkout > time_checkin),  
    bill_status INT CHECK(bill_status >= 0 AND bill_status <= 3),  
    bill_price INT NOT NULL DEFAULT 0,  
    customer_id VARCHAR(8),  
    package_name TEXT,  
    PRIMARY KEY(booking_id),  
    FOREIGN KEY(customer_id) REFERENCES Customer(customer_id),  
    FOREIGN KEY(package_name) REFERENCES ServicePackage(package_name)  
);
```

Figure 17: TABLE 17

1.1.18 RentingRoom

```
CREATE TABLE RentingRoom(  
    booking_id VARCHAR(16),  
    branch_id TEXT,  
    num_room VARCHAR(3),  
    PRIMARY KEY(booking_id,branch_id,num_room),  
    FOREIGN KEY(booking_id) REFERENCES BillBooking(booking_id),  
    FOREIGN KEY(branch_id,num_room) REFERENCES Room(branch_id,num_room)  
);
```

Figure 18: TABLE 18



1.1.19 PaymentBill

```
CREATE TABLE PaymentBill(  
    bill_num SERIAL NOT NULL,  
    bill_id VARCHAR(16) GENERATED ALWAYS AS ('HD'|| Generate_Room_Booking_ID(timer_checkout,bill_num)) STORED,  
    timer_checkin TIMESTAMP,  
    timer_checkout TIMESTAMP,  
    booking_id VARCHAR(16),  
    PRIMARY KEY(bill_id),  
    FOREIGN KEY(booking_id) REFERENCES BillBooking(booking_id)  
);
```

Figure 19: TABLE 19

1.1.20 Company

```
CREATE TABLE Company(  
    company_id VARCHAR(6) CHECK(company_id ~ 'DN[0-9]{4}'),  
    company_name TEXT,  
    PRIMARY KEY(company_id)  
);
```

Figure 20: TABLE 20

1.1.21 Service

```
CREATE TABLE Service(  
    service_id VARCHAR(6) CHECK(service_id ~ 'DV[RSCMB][0-9]{3}'),  
    service_type VARCHAR(1) CHECK(SUBSTRING(service_id,3,1) = service_type),  
    num_guests INT,  
    stype TEXT,  
    company_id VARCHAR(6),  
    PRIMARY KEY(service_id),  
    FOREIGN KEY(company_id) REFERENCES Company(company_id)  
);
```

Figure 21: TABLE 21

1.1.22 ServiceSpa

```
CREATE TABLE ServiceSpa(  
    service_id VARCHAR(6) CHECK(SUBSTRING(service_id,1,3) = 'DVS'),  
    spa_type TEXT,  
    PRIMARY KEY(service_id,spa_type),  
    FOREIGN KEY(service_id) REFERENCES Service(service_id)  
);
```

Figure 22: TABLE 22

1.1.23 TpeSouvenir

```
CREATE TABLE TypeSouvenir(  
    service_id VARCHAR(6) CHECK(SUBSTRING(service_id,1,3) = 'DVM'),  
    souv_type TEXT,  
    PRIMARY KEY(service_id,souv_type),  
    FOREIGN KEY(service_id) REFERENCES Service(service_id)  
);
```

Figure 23: TABLE 23

1.1.24 BrandSouvenir

```
CREATE TABLE BrandSouvenir(  
    service_id VARCHAR(6) CHECK(SUBSTRING(service_id,1,3) = 'DVM'),  
    souv_brand TEXT,  
    PRIMARY KEY(service_id,souv_brand),  
    FOREIGN KEY(service_id) REFERENCES Service(service_id)  
);
```

Figure 24: TABLE 24

1.1.25 AreaPlace

```
CREATE TABLE AreaPlace(  
    branch_id TEXT,  
    stt_num INT NOT NULL DEFAULT 1 CHECK( stt_num >= 1 AND stt_num <= 50),  
    area_length TEXT,  
    area_width TEXT,  
    area_price INT NOT NULL,  
    description TEXT,  
    service_id VARCHAR(6),  
    name_shop TEXT,  
    logo_shop TEXT,  
    PRIMARY KEY(branch_id,stt_num),  
    FOREIGN KEY(branch_id) REFERENCES Branch(branch_id),  
    FOREIGN KEY(service_id) REFERENCES Service(service_id)  
);
```

Figure 25: TABLE 25

1.1.26 ShopPicture

```
CREATE TABLE ShopPicture(  
    branch_id TEXT,  
    stt_num INT,  
    picture TEXT,  
    PRIMARY KEY(branch_id,stt_num,picture),  
    FOREIGN KEY(branch_id,stt_num) REFERENCES AreaPlace(branch_id,stt_num)  
);
```

Figure 26: TABLE 26

1.1.27 TimelineShop

```
CREATE TABLE TimelineShop(  
    branch_id TEXT,  
    stt_num INT,  
    time_starting TIME,  
    time_closing TIME,  
    PRIMARY KEY(branch_id,stt_num,time_starting),  
    FOREIGN KEY(branch_id,stt_num) REFERENCES AreaPlace(branch_id,stt_num)  
);
```

Figure 27: TABLE 27

Then when creating all of table we have a set of tables there:



Figure 28: TABLES



1.2 Insert data to tables

A data file have already create with name "insert_data" for adding them to database system. Then we have some data that provide for querying in the DBMS. There are some pictures when you query data

The screenshot shows the PGAdmin interface with a query executed: `SELECT * FROM Customer`. The result set contains 10 rows of customer data.

customer_id	code_cmr	customer_name	customer_phone	customer_email	username	cus_password	points	customer_type
1	KH000008	Nuno Alex	421321343454	nuno@gmail.com	nuno31	7412	7	1
2	KH000009	Panani Extra	526943443454	extra@gmail.com	panadol31	7411	39	2
3	KH000010	Hapacol Ex	032513443454	extra@gmail.com	hapacol31	96325	52	3
4	KH000004	Mario Vip	12345678901	vip@gmail.com	vip231	0987	136	4
5	KH000005	Alex Spel	433463464356	ssdqi@gmail.com	ssdq21	75445	53	3
6	KH000001	Maria Olala	839483943849	olala@gmail.com	olala231	122345	9	1
7	KH000006	Runan Tix	453453453533	runan@gmail.com	runan1	14212	20	1
8	KH000002	Alo Hih	354345453432	hihi@gmail.com	hihi1	32325	11	1
9	KH000003	Bu Haha	565769943849	haha@gmail.com	chaha2	15675	18	1
10	KH000007	Panamax R	124242423454	ri@gmail.com	raaa1	24323	4	1

Figure 29: Query from Customer table

pgAdmin

FileObjectToolsHelp

Browser

Operators

Procedures

Sequences

Tables (28)

areaplace

bedinfo

billbooking

billservice

branch

branchhavcomtype

branchpicture

brandsouvenir

company

customer

login_users

material

paymentbill

provider

providermaterial

rentingroom

room

roomtype

service

servicepackage

servicespa

shoppicture

spicmentype

spicypairroomtype

timeshopping

typediscount

typesouvenir

zone

Trigger Functions

Processespostgres/sMan...postgres/sMan...postgres/sMan...postgres/sMan...postgres/sMan...postgres/sManager@Hotel_DBMS

Query Query History

Scratch Pad

Data OutputMessagesNotifications

	book_id integer	booking_id PK character varying (16)	time_booking timestamp without time zone	time_checkin date	time_checkout date	bill_status integer	bill_price integer	customer_id character varying (8)	package_name text
1	4	DP05030202000004	2022-03-05 21:30:00	2022-03-06	2022-03-16	1	8000	KH000004	PeaceDays
2	5	DP18050202000005	2022-05-18 12:19:00	2022-05-25	2022-05-28	1	1800	KH000005	SweetHoliday
3	1	DP18050202000001	2022-05-18 06:19:00	2022-05-25	2022-05-28	1	2000	KH000001	SweetHoliday
4	6	DP05030202000006	2022-01-05 06:19:00	2022-01-06	2022-01-10	1	15000	KH000006	NormalDays
5	2	DP25080202000002	2022-08-25 10:31:00	2022-08-26	2022-08-29	1	1500	KH000002	ChillDays
6	3	DP18040202000003	2022-04-18 14:21:00	2022-04-20	2022-04-22	1	5000	KH000003	Romantic
7	7	DP10060202000007	2022-06-10 09:17:00	2022-06-12	2022-06-15	1	2000	KH000007	SweetHoliday
8	8	DP15070202000008	2022-07-15 10:13:00	2022-07-20	2022-07-22	1	7000	KH000008	AnotherDays
9	9	DP08090202000009	2022-09-08 11:19:00	2022-09-10	2022-09-12	1	7000	KH000009	AnotherDays
10	10	DP15110202000010	2022-11-15 15:14:00	2022-11-20	2022-11-21	1	1500	KH000010	ChillDays

Figure 30: Query from BillBooking

2 Part2: Store Procedure/Function and Trigger

2.1 Store Procedure/Function

2.1.1 Procedure/Function for ServicePackage(GoiDichVu)

```
1 CREATE OR REPLACE FUNCTION AddDate(_days INT, _datetime DATE) RETURNS DATE AS
2 $$
3 SELECT _datetime + _days;
```

```
4 $$ LANGUAGE sql IMMUTABLE;
5
6 CREATE OR REPLACE FUNCTION AddDateInt(_days INT, _datetime INT) RETURNS INT AS
7 $$
8     SELECT _datetime + _days;
9 $$ LANGUAGE sql IMMUTABLE;
10
11 CREATE TYPE mytype AS(
12     package_name TEXT,
13     num_guests INT,
14     time_starting DATE,
15     time_expiring DATE,
16     remaining_days INT
17 );
18 CREATE OR REPLACE FUNCTION PackageInfo(_customer_id TEXT)
19 RETURNS mytype
20 AS
21 $$
22 DECLARE tableshow mytype;
23 DECLARE tempvar INT;
24 DECLARE tempvar1 DATE;
25 DECLARE tempvar2 DATE;
26 BEGIN
27     tableshow.package_name = BillService.package_name FROM BillService WHERE
28     BillService.customer_id = _customer_id;
29     tableshow.num_guests = ServicePackage.num_guests FROM ServicePackage WHERE
30     ServicePackage.package_name = tableshow.package_name;
31     tableshow.time_starting = BillService.time_starting FROM BillService WHERE
32     BillService.customer_id = _customer_id ;
33     tableshow.time_expiring = (tableshow.time_starting+365);
34     tempvar= ServicePackage.num_days FROM ServicePackage WHERE ServicePackage.
35     package_name = tableshow.package_name;
36     tempvar1 = BillBooking.time_checkout FROM BillBooking WHERE BillBooking.
37     customer_id = _customer_id;
38     tempvar2 = BillBooking.time_checkin FROM BillBooking WHERE BillBooking.
39     customer_id = _customer_id;
40     IF(AddDate(365,tableshow.time_starting) - CURRENT_DATE -
41     AddDateInt(tempvar , - (tempvar1 - tempvar2))) > 0 THEN
42     tableshow.remaining_days = AddDateInt(tempvar, - (tempvar1 - tempvar2)) ;
43 ELSE
44     tableshow.remaining_days = AddDate(365,tableshow.time_starting) -
45     CURRENT_DATE ;
46 END IF;
47 RETURN tableshow;
48 END;
49 $$ LANGUAGE 'plpgsql';
```

To create function to view package service we need to have some information as follows:

- We need to have *package_name* from BillService(DonDichVu) where *customer_id* of BillService is the same with *customer_id* from input user.
- With *num_guests* we query from ServicePackage(GoiDichVu) where *package_name* of ServicePackage is the same one from BillService.
- With *time_starting* we query from BillService where *customer_id* of BillService is the same with *customer_id* from input user.
- With *time_expiring* we calculate with *time_starting* plus 365 days.
- With *remaining_days* we use formula with if the remaining days from *time_expiring* minus *current_time* minus usage days in package that is greater than 0, the remaining days is equal remaining usage days in package, otherwise is 0.

2.1.2 Procedure/Function for StatisticGuests

```
1 CREATE OR REPLACE FUNCTION TextDate(_year TEXT) RETURNS DATE AS
2 $$
```

```
3 SELECT TO_DATE(_year::TEXT,'YYYY');
4 $$ LANGUAGE sql IMMUTABLE;
5
6 CREATE OR REPLACE FUNCTION MonthDate(_month TIMESTAMP) RETURNS TEXT AS
7 $$
8     SELECT TO_CHAR(_month,'MM');
9 $$ LANGUAGE sql IMMUTABLE;
10
11 CREATE OR REPLACE FUNCTION StatisticGuests(_branch_id TEXT,_year TEXT)
12 RETURNS TABLE(
13     months TEXT,
14     sum_num_guests BIGINT
15 )
16 AS
17 $$
18 BEGIN
19     RETURN QUERY
20     SELECT
21         MonthDate(BillBooking.time_booking),
22         COUNT(BillBooking.booking_id)
23     FROM
24         RentingRoom JOIN PaymentBill ON RentingRoom.booking_id = PaymentBill.booking_id
25         JOIN BillBooking ON PaymentBill.booking_id = BillBooking.booking_id
26     WHERE
27         RentingRoom.branch_id = _branch_id AND DATE_TRUNC('year',BillBooking.
28         time_booking) = TextDate(_year)
29     GROUP BY MonthDate(BillBooking.time_booking);
30 END;
31 $$ LANGUAGE 'plpgsql';
```

To create function to view statistical guests we need to have some information as follows: We chose *month* and *sum_num_guests* from joining some tables as RentingRoom(PhongThue) with PaymentBill(HoaDonThanhToan) where *booking_id* is the same as well as with joining BillBooking where the condition is *branch_id* from RentingRoom and the year are the same with input from user by grouping follow month in a year.

2.2 Triggers

2.2.1 Trigger to update values

- UpdateSumServiceBill(TongTienGoiDichVu), we find *discount* from joining BillService(DonDichVu) to Customer through *customer_id* and joining with TypeForDiscount through *customer_type* where BillService has a new *customer_id* and then we use calculation as picture.

```
1 CREATE OR REPLACE FUNCTION UpdateSumServiceBill()
2 RETURNS TRIGGER
3 AS
4 $updateServiceBill$
5 BEGIN
6     UPDATE BillService
7     SET total_price = ServicePackage.price - ServicePackage.price * (SELECT
8         discount FROM BillService JOIN Customer
9         ON BillService.customer_id = Customer.customer_id
10        JOIN TypeForDiscount ON Customer.customer_type =
11        TypeForDiscount.customer_type
12        WHERE BillService.customer_id = NEW.customer_id LIMIT
13        1)
14     FROM ServicePackage
15     WHERE
16         ServicePackage.package_name = BillService.package_name AND BillService.
17         customer_id = NEW.customer_id;
18     RETURN NULL;
19 END;
20 $updateServiceBill$ LANGUAGE plpgsql;
21
22 CREATE OR REPLACE TRIGGER UpdateTotalServiceBill
23 AFTER INSERT
24 ON BillService
25 FOR EACH ROW
```

```
22 EXECUTE FUNCTION UpdateSumServiceBill();
23
```

- UpdateSumPayment(TongTienDonDatPhong), we need some information like *discount* from RentingRoom(PhongThue) to BillBooking(DonDatPhong) through *booking_id* and joining with Customer through *customer_id* where BillBooking has a new *booking_id* and then we calculate as formula as picture.

```
1 CREATE OR REPLACE FUNCTION UpdateSumPayment()
2 RETURNS TRIGGER
3 AS
4 $UpdateBillBooking$
5 BEGIN
6     UPDATE BillBooking
7     SET bill_price = BillBooking.bill_price - BillBooking.bill_price *(SELECT
8         discount FROM RentingRoom
9         JOIN BillBooking ON RentingRoom.booking_id = BillBooking.
10            booking_id
11            JOIN Customer ON Customer.customer_id = BillBooking.
12            customer_id
13            JOIN TypeForDiscount ON TypeForDiscount.customer_type =
14            Customer.customer_type
15            WHERE NEW.booking_id = BillBooking.booking_id LIMIT 1)
16 FROM RentingRoom
17 WHERE NEW.booking_id = BillBooking.booking_id;
18 RETURN NULL;
19 END;
20 $UpdateBillBooking$ LANGUAGE plpgsql;
21
22 CREATE OR REPLACE TRIGGER UpdateSumPaymentBill
23 AFTER INSERT
24 ON RentingRoom
25 FOR EACH ROW
26 EXECUTE FUNCTION UpdateSumPayment();
27
```

- UpdatePointCustomer(DiemCuaKhachHang), we need to find *bill_price* from joining BillBooking to Customer through *customer_id* where *bill_status* = 1 and BillBooking has a new *booking_id* and then divide 1000 and next take the result is plus initial point.

```
1 CREATE OR REPLACE FUNCTION UpdatePointCustomer()
2 RETURNS TRIGGER
3 AS
4 $UpdatCustomer$
5 BEGIN
6     UPDATE Customer
7     SET points = Customer.points + (SELECT bill_price FROM BillBooking JOIN
8         Customer ON
9         BillBooking.customer_id = Customer.customer_id
10        WHERE NEW.bill_status = 1 AND NEW.booking_id = BillBooking.
11        booking_id LIMIT 1)/1000
12 FROM BillBooking
13 WHERE NEW.customer_id = Customer.customer_id AND NEW.booking_id =
14        BillBooking.booking_id AND NEW.bill_status = 1;
15 RETURN NULL;
16 END;
17 $UpdatCustomer$ LANGUAGE plpgsql;
18
19 CREATE OR REPLACE TRIGGER UpdatePointClient
20 AFTER UPDATE
21 ON BillBooking
22 FOR EACH ROW
23 EXECUTE FUNCTION UpdatePointCustomer();
24
```

- UpdateTypeCustomer(LoaiKhachHang), we need to calculate the point of customer and then using it to comparison with some rules to update type of customers

```
1 CREATE OR REPLACE FUNCTION UpdateTypeCustomer()  
2 RETURNS TRIGGER  
3 AS  
4 $UpdateCustomer$  
5 BEGIN  
6     UPDATE Customer  
7     SET customer_type = (SELECT customer_type FROM TypeForDiscount WHERE NEW.  
8         points >= point_type  
9         ORDER BY customer_type DESC LIMIT 1)  
10    WHERE NEW.customer_id = Customer.customer_id;  
11    RETURN NULL;  
12 END;  
13 $UpdateCustomer$ LANGUAGE plpgsql;  
14  
15 CREATE OR REPLACE TRIGGER UpdateTypeClient  
16 AFTER UPDATE  
17 ON Customer  
18 FOR EACH ROW  
19 WHEN (NEW.points <> OLD.points)  
20 EXECUTE FUNCTION UpdateTypeCustomer();  
21
```

3 Part3: Build an application

A link video demo(backup in drive):

<https://drive.google.com/drive/folders/1cdX3ZRnXvM25c02UBAvyqK9pslph4xVD?usp=sharing>

3.1 Create user

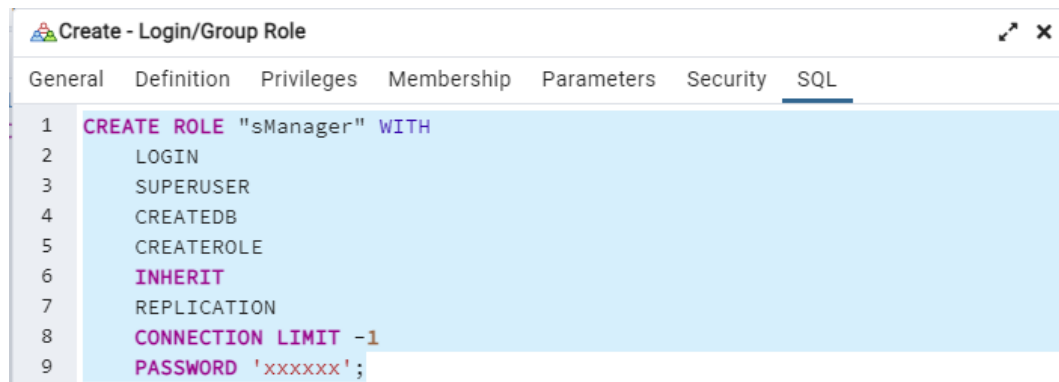
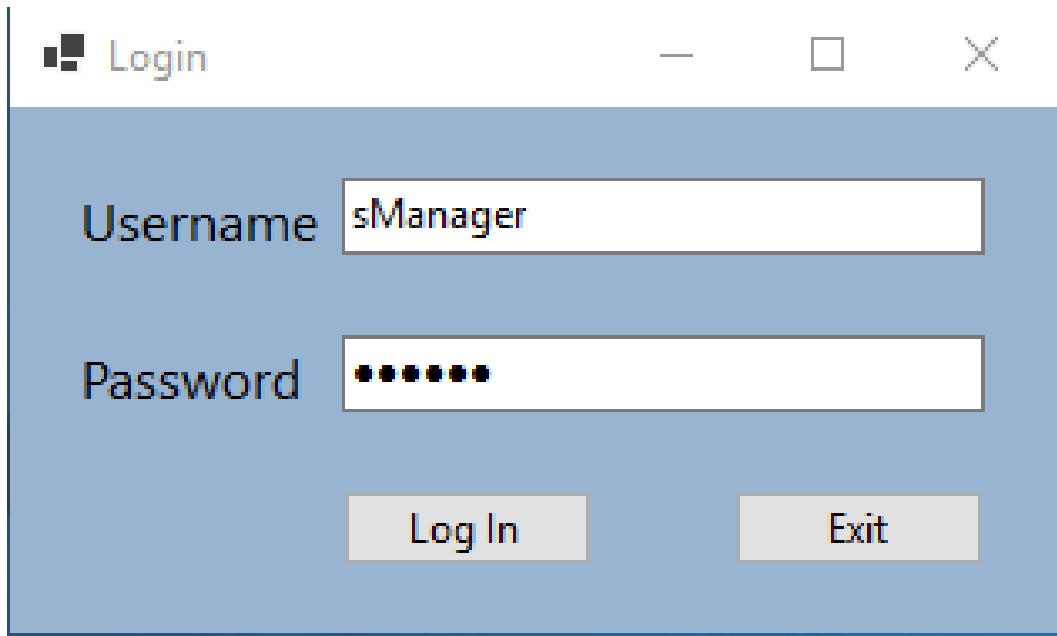


Figure 31: Create DBA user

3.2 Some features

3.2.1 Login/Logout

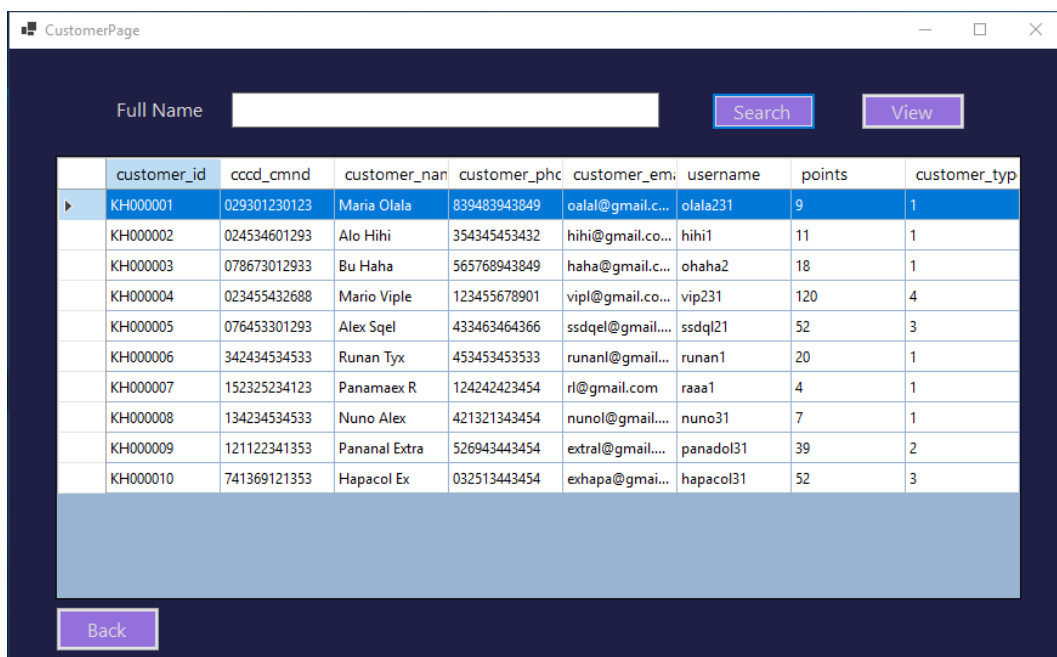


The image shows a login window titled "Login". It has a light blue background. There are two input fields: "Username" with the text "sManager" and "Password" with masked characters (dots). Below the fields are two buttons: "Log In" and "Exit".

Figure 32: Login/Logout UI

3.2.2 Features

- View information of customer



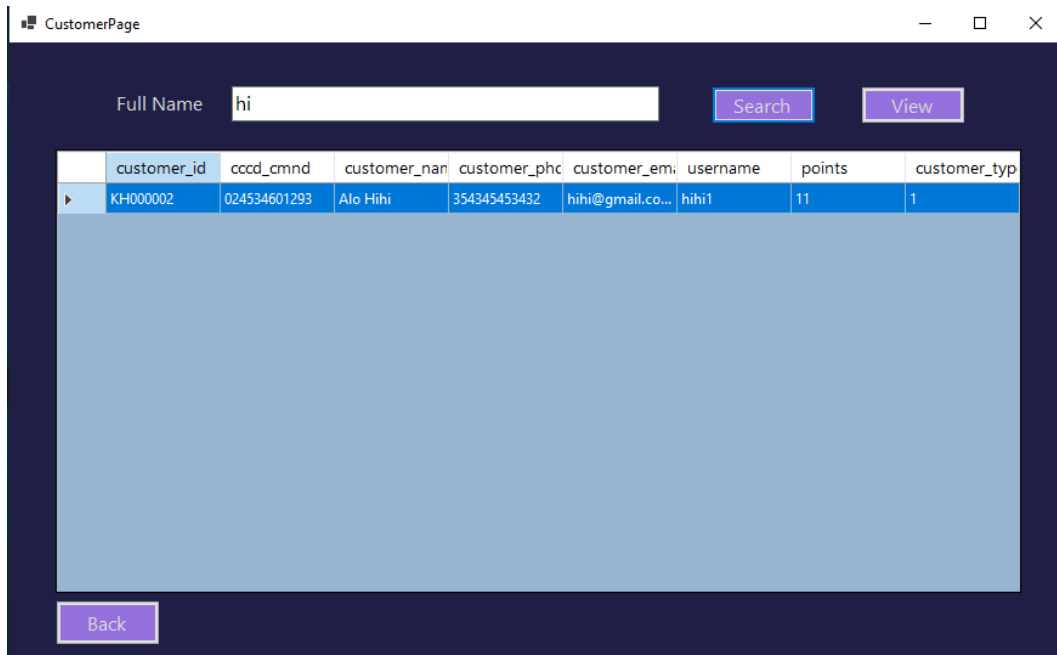
The image shows a "CustomerPage" window. It has a dark blue header with a search bar labeled "Full Name" and two buttons: "Search" and "View". Below the header is a table with customer information. The table has 9 columns: customer_id, cccd_cmnd, customer_name, customer_phone, customer_email, username, points, and customer_type. The first row is highlighted in blue.

	customer_id	cccd_cmnd	customer_name	customer_phone	customer_email	username	points	customer_type
▶	KH000001	029301230123	Maria Olala	839483943849	oalal@gmail.c...	olala231	9	1
	KH000002	024534601293	Alo Hihi	354345453432	hihi@gmail.co...	hihi1	11	1
	KH000003	078673012933	Bu Haha	565768943849	haha@gmail.c...	ohaha2	18	1
	KH000004	023455432688	Mario Viple	123455678901	vipl@gmail.co...	vip231	120	4
	KH000005	076453301293	Alex Sqel	433463464366	ssdqel@gmail....	ssdq121	52	3
	KH000006	342434534533	Runan Tyx	453453453533	runanl@gmail...	runan1	20	1
	KH000007	152325234123	Panamaex R	124242423454	rl@gmail.com	raaa1	4	1
	KH000008	134234534533	Nuno Alex	421321343454	nunol@gmail....	nuno31	7	1
	KH000009	121122341353	Pananal Extra	526943443454	extral@gmail....	panadol31	39	2
	KH000010	741369121353	Hapacol Ex	032513443454	exhapa@gmai...	hapacol31	52	3

At the bottom left of the window is a "Back" button.

Figure 33: Information customer

- Search information with some characters. And then view details a customer.

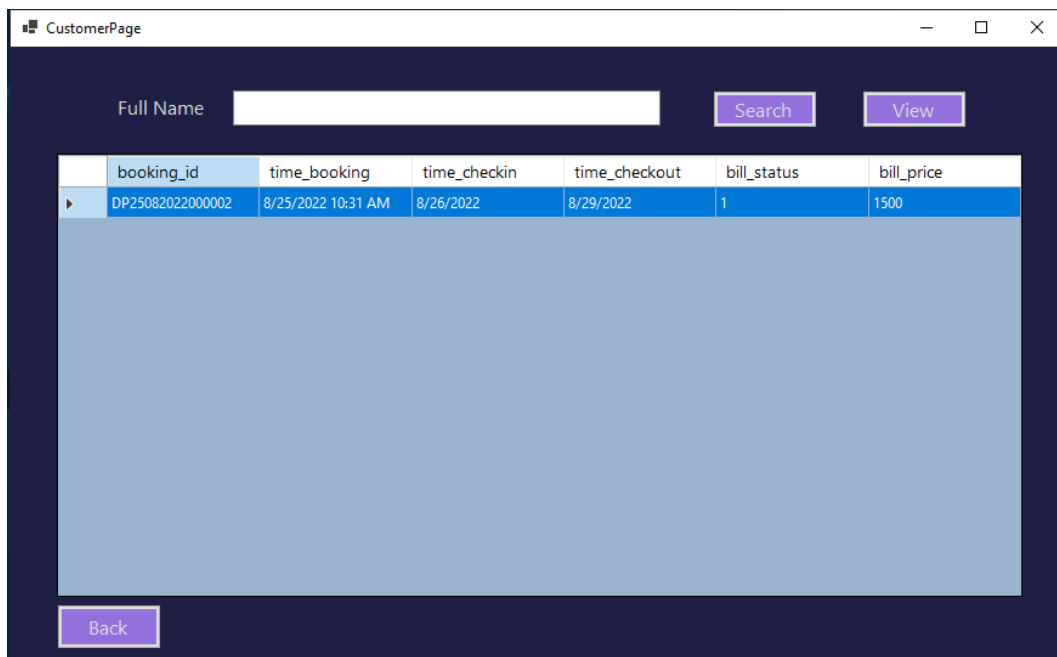


CustomerPage

Full Name

	customer_id	cccd_cmnd	customer_name	customer_phone	customer_email	username	points	customer_type
▶	KH000002	024534601293	Alo Hihi	354345453432	hihi@gmail.co...	hihi1	11	1

Figure 34: Search information



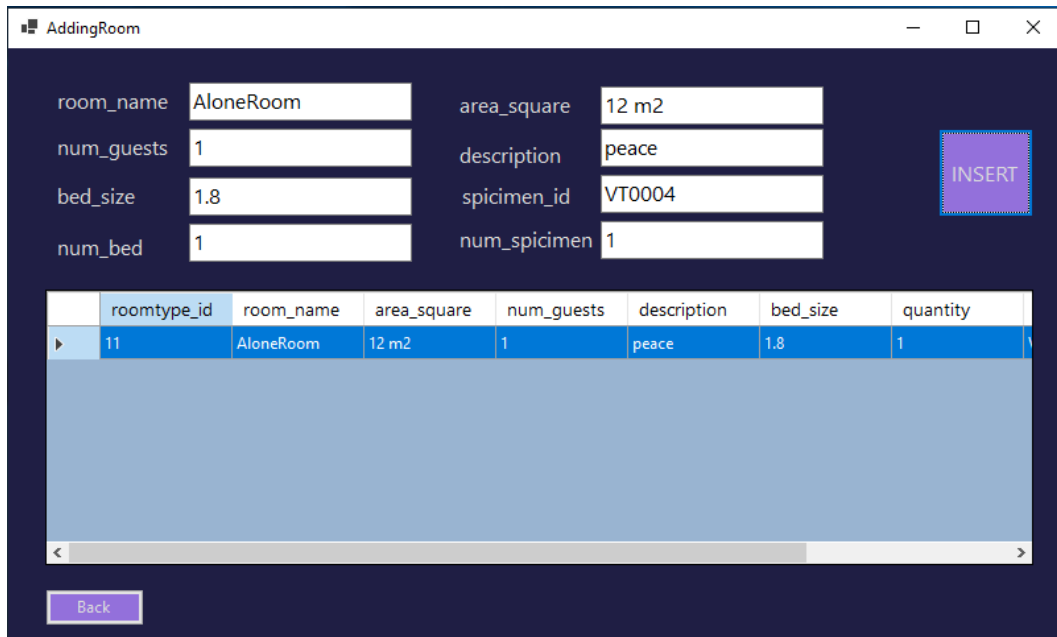
CustomerPage

Full Name

	booking_id	time_booking	time_checkin	time_checkout	bill_status	bill_price
▶	DP25082022000002	8/25/2022 10:31 AM	8/26/2022	8/29/2022	1	1500

Figure 35: View details

- Add new information of room type



AddingRoom

room_name: AloneRoom area_square: 12 m2

num_guests: 1 description: peace

bed_size: 1.8 spicimen_id: VT0004

num_bed: 1 num_spicimen: 1

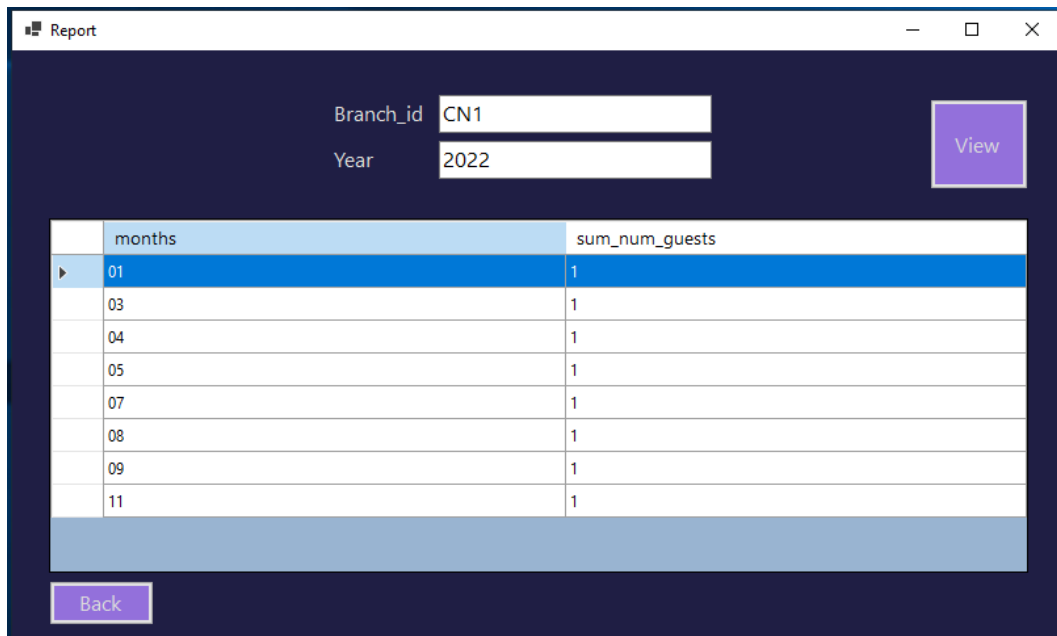
INSERT

	roomtype_id	room_name	area_square	num_guests	description	bed_size	quantity
▶	11	AloneRoom	12 m2	1	peace	1.8	1

Back

Figure 36: Add new information of room type

- View statistical guests in a branch in year



Report

Branch_id: CN1

Year: 2022

View

	months	sum_num_guests
▶	01	1
	03	1
	04	1
	05	1
	07	1
	08	1
	09	1
	11	1

Back

Figure 37: Add new information of room type

- In addition one feature for Admin to query on app



Query

select * from Customer

Query

	customer_i	cccd_cmnd	customer_i	customer_	customer_	username	cus_passw	points	customer_i
▶	KH000004	023455432...	Mario Viple	123455678...	vipl@gmai...	vip231	0987	120	4
	KH000005	076453301...	Alex Sqel	433463464...	ssdqel@g...	ssdq121	75445	52	3
	KH000001	029301230...	Maria Olala	839483943...	oalal@gm...	olala231	122345	9	1
	KH000006	342434534...	Runan Tyx	453453453...	runanl@g...	runan1	14212	20	1
	KH000002	024534601...	Alo Hihi	354345453...	hihi@gmai...	hihi1	32325	11	1
	KH000003	078673012...	Bu Haha	565768943...	haha@gm...	ohaha2	15675	18	1
	KH000007	152325234...	Panamaex R	124242423...	rl@gmail.c...	raaa1	24323	4	1
	KH000008	134234534...	Nuno Alex	421321343...	nunol@g...	nuno31	7412	7	1
	KH000009	121122341...	Pananal Ex...	526943443...	extral@gm...	panadol31	7411	39	2
	KH000010	741369121...	Hapacol Ex	032513443...	exhapa@g...	hapacol31	96325	52	3

Back

Figure 38: Query on app