CSI 2132 Project Report
Jared Lueck 8763876
Winter 2020
Rental Booking System

Programming Languages and DBMS

The programming language that was used for this language is python and some python libraries.. I used the postgres adapter psycopg2 to connect the application to postgres and execute all the queries.

DDLs

```
CREATE TABLE Branch (
        country varchar(20),
        primary key (country)
);

CREATE TABLE Rental_User (
        user_id int,
        email_address varchar(50) not null,
        unit_number int,
        street_number int,
        street varchar(20),
        city varchar(20) not null,
        province varchar(20) not null,
        country varchar(20) not null,
        firstname varchar(20) not null,
        middlename varchar(20),
        lastname varchar(20) not null,
        primary key (user_id),
        foreign key (country) references Branch(country)
);

CREATE TABLE Host (
        host_id int,
        email_address varchar(50) not null,
        unit_number int,
        street_number int,
        street varchar(20),
        city varchar(20) not null,
        province varchar(20) not null,
        country varchar(20) not null,
        firstname varchar(20) not null,
        middlename varchar(20),
        lastname varchar(20) not null,
        primary key (host_id),
        foreign key (host_id) references Rental_User(user_id)
                on delete cascade
);

CREATE TABLE Guest (
        guest_id int,
        email_address varchar(50) not null,
        unit_number int,
```

```
        street_number int,
        street varchar(20),
        city varchar(20) not null,
        province varchar(20) not null,
        country varchar(20) not null,
        firstname varchar(20) not null,
        middlename varchar(20),
        lastname varchar(20) not null,
        primary key (guest_id),
        foreign key (guest_id) references Rental_User(user_id)
                on delete cascade
);

CREATE TABLE Phonenumber (
        user_id int,
        phone_number varchar(20) not null,
        primary key (user_id, phone_number),
        foreign key (user_id) references Rental_User(user_id)
);

CREATE TABLE Property (
        property_id int,
        unit_number int,
        street_number int not null,
        street varchar(20) not null,
        city varchar(20) not null,
        province varchar(20),
        country varchar(20) not null,
        beds_number int not null,
        host_id int not null,
        foreign key (host_id) references Host(host_id)
                on delete cascade,
        primary key (property_id),
        foreign key (country) references Branch(country)

);

CREATE TABLE Payment (
        transaction_id int,
        transaction_type varchar(20),
        amount float not null,
        status varchar(20),
        host_id int not null,
        guest_id int not null,
        property_id int,
        primary key (transaction_id),
        foreign key (host_id) references Host(host_id),
        foreign key (guest_id) references Guest(guest_id),
        foreign key (property_id) references Property(property_id) on delete set null
);
```

```
CREATE TABLE Pricing (
        property_id int,
        rate float not null,
        guest_number int not null,
        property_type varchar(20),
        primary key (property_id),
        foreign key (property_id) references Property(property_id)
                on delete cascade
);

CREATE TABLE Rental_Agreement (
        agreement_id int,
        sign_date date not null,
        start_date date not null,
        end_date date not null constraint after_start check(end_date > start_date),
        host_id int not null,
        guest_id int not null,
        property_id int,
        foreign key (host_id) references Host(host_id)
                on delete cascade,
        foreign key (guest_id) references Guest(guest_id)
                on delete cascade,
        foreign key (property_id) references Property(property_id)
                on delete set null,
        primary key(agreement_id)
);

CREATE TABLE Review (
        review_id int,
        comment varchar(20) not null,
        review_date date not null,
        communication float,
        checkin float,
        cleanliness float,
        location float,
        property_id int,
        guest_id int,
        primary key (review_id),
        foreign key (property_id) references Property(property_id)
                on delete set null,
        foreign key (guest_id) references Guest(guest_id)
                on delete set null
);

CREATE TABLE Employee (
        employee_id int,
        firstname varchar(20) not null,
        lastname varchar(20) not null,
        email_address varchar(50) not null,
```

```
        position varchar(30),
        salary float,
        branch varchar(20) not null,
        foreign key (branch) references Branch(country),
        primary key (employee_id)
);

CREATE TABLE Manager (
        manager_id int,
        primary key (manager_id),
        foreign key (manager_id) references Employee(employee_id)
);

CREATE SEQUENCE property_sequence
        start 20
        increment 1;

CREATE OR REPLACE FUNCTION isValidUser(email varchar(50))
  RETURNS BOOLEAN AS $$
                BEGIN
                        PERFORM * FROM Rental_User U WHERE U.email_address = email;
                        IF FOUND THEN RETURN TRUE;
                        ELSE
                        RETURN FALSE;
                        END IF;
                END
        $$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION isValidEmployee(id int)
  RETURNS BOOLEAN AS $$
                BEGIN
                        PERFORM * FROM Employee E WHERE E.employee_id = id;
                        IF FOUND THEN RETURN TRUE;
                        ELSE
                        RETURN FALSE;
                        END IF;
                END
        $$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION isHost(id INT)
  RETURNS BOOLEAN AS $$
                BEGIN
                        PERFORM * FROM Host H WHERE H.host_id = id;
                        IF FOUND THEN RETURN TRUE;
                        ELSE
                        RETURN FALSE;
                        END IF;
                END
        $$ LANGUAGE plpgsql;
```

```
CREATE OR REPLACE FUNCTION new_property(user_id int, unit_number int, street_number int,
street varchar(20),  city varchar(20), province varchar(20), country varchar(20), beds_number int)
RETURNS int AS $$
	DECLARE property_id int;
	BEGIN
		property_id := nextval('property_sequence');
		IF NOT isHost(user_id) THEN
			INSERT INTO Host SELECT * FROM User U Where U.user_id = user_id;
		END IF;


		INSERT INTO Property VALUES(property_id, unit_number, street_number, street, city,
									province, country, beds_number,
user_id);

		RETURN property_id;
	END $$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION new_pricing(property_id int, rate float, guest_number int, property_type varchar(20))
	RETURNS void AS $$
		BEGIN
			INSERT INTO Pricing VALUES(property_id, rate, guest_number, property_type);
		END
	$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION is_occupied(id int)
	RETURNS VARCHAR as $$
	BEGIN
		PERFORM * FROM Property P INNER JOIN Rental_Agreement RA ON RA.property_id = P.property_id
		WHERE P.Property_id = id AND RA.start_date <= now() AND RA.end_date >= now();
		IF FOUND THEN RETURN 'OCCUPIED';
		ELSE RETURN 'UNOCCUPIED';
		END IF;
	END
	$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION construct_address(unit_num int, street_num int, street_name varchar)
	RETURNS VARCHAR as $$
	DECLARE address text;
	BEGIN
		IF NOT (unit_num is NULL) THEN SELECT concat(street_num, ' ', street_name, ' Unit ', unit_num) INTO
address;
		ELSE SELECT CONCAT(street_num, ' ', street_name) INTO address;
		END IF;
		RETURN address;
	END $$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION get_availability(prop_id int)
	RETURNS DATE AS $$
	DECLARE res date;
```

```
        BEGIN
        IF is_occupied(prop_id) = 'OCCUPIED' THEN SELECT MAX(end_date) FROM Property P
        INNER JOIN Rental_Agreement RA ON RA.property_id = P.property_id WHERE P.property_id = prop_id INTO res;
        ELSE SELECT now() into res;
        END IF;
        RETURN res;
END $$ LANGUAGE plpgsql;


CREATE OR REPLACE FUNCTION update_host_guest_table()
        RETURNS TRIGGER AS $update_info$
        BEGIN
        IF NOT OLD.email_address = NEW.email_address THEN
        UPDATE Host SET email_address = NEW.email_address WHERE Host.host_id= NEW.user_id;
        UPDATE Guest SET email_address = NEW.email_address WHERE Guest.guest_id= NEW.user_id;
        END IF;

        IF NOT OLD.unit_number = NEW.unit_number THEN
        UPDATE Host SET unit_number = NEW.unit_number WHERE Host.host_id= NEW.user_id;
        UPDATE Guest SET unit_number = NEW.unit_number WHERE Guest.guest_id= NEW.user_id;
        END IF;

        IF NOT OLD.street_number = NEW.street_number THEN
        UPDATE Host SET street_number = NEW.street_number WHERE Host.host_id= NEW.user_id;
        UPDATE Guest SET street_number = NEW.street_number WHERE Guest.guest_id= NEW.user_id;
        END IF;

        IF NOT OLD.street = NEW.street THEN
        UPDATE Host SET street = NEW.street WHERE Host.host_id= NEW.user_id;
        UPDATE Guest SET street = NEW.street WHERE Guest.guest_id= NEW.user_id;
        END IF;

        IF NOT OLD.city = NEW.city THEN
        UPDATE Host SET city = NEW.city WHERE Host.host_id= NEW.user_id;
        UPDATE Guest SET city = NEW.city WHERE Guest.guest_id= NEW.user_id;
        END IF;

        IF NOT OLD.province = NEW.province THEN
        UPDATE Host SET province = NEW.province WHERE Host.host_id= NEW.user_id;
        UPDATE Guest SET province = NEW.province WHERE Guest.guest_id= NEW.user_id;
        END IF;

        IF NOT OLD.firstname = NEW.firstname THEN
        UPDATE Host SET firstname = NEW.firstname WHERE Host.host_id= NEW.user_id;
        UPDATE Guest SET firstname = NEW.firstname WHERE Guest.guest_id= NEW.user_id;
        END IF;

        IF NOT OLD.middlename = NEW.middlename THEN
        UPDATE Host SET middlename = NEW.middlename WHERE Host.host_id= NEW.user_id;
        UPDATE Guest SET middlename = NEW.middlename WHERE Guest.guest_id= NEW.user_id;
```

```
        END IF;

        IF NOT OLD.lastname = NEW.lastname THEN
        UPDATE Host SET lastname = NEW.lastname WHERE Host.host_id= NEW.user_id;
        UPDATE Guest SET lastname = NEW.lastname WHERE Guest.guest_id= NEW.user_id;
        END IF;
        RETURN NEW;
END $update_info$ LANGUAGE plpgsql;

CREATE TRIGGER update_info BEFORE INSERT OR UPDATE ON Rental_User
   FOR EACH ROW EXECUTE PROCEDURE update_host_guest_table();
```

Installation

Python 3.8.2 was what I used for the development of this application. It is possible that older versions may work. Two packages are required. Tabular is for nicely printing results as tables. Psycopg2 is an adapter for Postgresql. You can use PIP to install them:

```
pip install psycopg2
```

```
pip install tabulate
```

Documentation for these 2 libraries:

https://pypi.org/project/tabulate/

https://pypi.org/project/psycopg2/

All the sql required for the application is inside of the SQL directory. This includes all the insert statements to populate the tables. The App directory contains python scripts, as well as a configuration file for the Database. Fill out the config.ini

```
[POSTGRESQL]
database = postgres
user = postgres
password = password
host = localhost
port = 5432
```

Figure 1: example of filled out config. Replace with correct values.
Alternatively, you can also put them in the connectDB.py file.

After filling out the database credentials, execute App/createDB.py. This will create all the tables, functions and triggers and then execute them.

You can now run the app. There are two interfaces: employeeCLI (for the employees) and UserCLI (For hosts and Guests).

Test Queries

Test Query 1

```
SELECT firstname, PR.property_type, PR.rate, sign_date, G.country, transaction_type, status FROM Rental_Agreement RA
        inner join Guest G
                on G.guest_id = RA.guest_id
        inner join Pricing Pri
                on Pri.property_id = RA.property_id
        inner join Pricing PR
                on PR.property_id = RA.property_id
        inner join Payment PA
                on PA.host_id = RA.host_id and PA.guest_id = RA.guest_id
                and PA.property_id = RA.property_id
        order by transaction_type asc, sign_date desc
```

Results for Test Query 1

firstname character varying (20)	property_type character varying (20)	rate double precision	sign_date date	country character varying (20)	transaction_type character varying (20)	status character varying (20)
Zena	Condo	45.07	2019-11-13	Netherlands	Cash	approved
Zena	Entire Place	719.03	2015-10-20	Netherlands	Cash	completed
Elliot	Apartment	4.14	2011-02-28	Singapore	Cash	approved
Hailey	Apartment	411.22	2007-04-29	Pitcairn Islands	Cash	pending
Elliot	Bungalow	263.33	2012-05-09	Singapore	Check	completed
Elliot	Apartment	4.14	2011-02-28	Singapore	Credit Card	approved
Elliott	Private Room	992.47	2011-10-15	Netherlands Antilles	Direct Debit	completed
Hailey	Apartment	411.22	2007-04-29	Pitcairn Islands	Direct Debit	approved

Test Query 2

```
CREATE VIEW GuestListView AS
SELECT * FROM Guest
ORDER country, guest_id
```

Results for Test Query 2

guest_id integer	email_address character varying (50)	country character varying (20)	country character varying (20)
31	lindgren.sienna@example.c...	British Indian Ocean	British Indian Ocean
110	landen95@example.net	Bulgaria	Bulgaria
3	ben.baker@gmail.com	Canada	Canada
4	theo.holland@gmail.com	Canada	Canada
5	david.chapmen@gmail.com	Canada	Canada
6	caden.koch@gmail.com	Canada	Canada
7	bruce.bright@hotmail.com	Canada	Canada
135	fay62@example.net	Chad	Chad
241	fritsch.leslie@example.net	Hong Kong	Hong Kong
252	shane82@example.net	Madagascar	Madagascar
300	hdenesik@example.org	Netherlands	Netherlands
393	veda36@example.org	Netherlands Antilles	Netherlands Antilles
400	weissnat.andy@example.net	Palestinian Territor	Palestinian Territor
422	kklein@example.net	Pitcairn Islands	Pitcairn Islands

Test Query 3

```
WITH pricing_info AS (SELECT RA.sign_date, RA.start_date, RA.end_date, PRI.rate, PRI.property_type, PRO.property_id,
RA.guest_id, RA.host_id FROM Rental_Agreement RA
        inner join Property PRO
                on PRO.property_id = RA.property_id
        inner join Pricing PRI
                on PRI.property_id = RA.property_id )

SELECT * FROM pricing_info PI
WHERE PI.rate = (SELECT MIN(rate) FROM pricing_info) AND
PI.end_date < DATE("now")
```

Results for Test Query 3

sign_date date	start_date date	end_date date	rate double precision	property_type character varying (20)	property_id integer	guest_id integer	host_id integer
2011-02-28	2011-03-10	2011-03-27	4.14	Apartment	10	450	252

Test Query 4

```
SELECT property_id, sign_date, start_date, end_date, country, rating FROM Rental_Agreement RA
        INNER JOIN Property PRO
                USING(property_id)
        INNER JOIN ( SELECT property_id,
                ROUND(CAST((AVG(communication + checkin + cleanliness + location)/4) AS NUMERIC), 2) AS rating
                FROM Review
                GROUP BY property_id) RAT
                USING(property_id)
        ORDER BY country, rating DESC
```

Results for Test Query 4

property_id integer	sign_date date	start_date date	end_date date	country character varying (20)	rating numeric
8	2019-11-10	2019-11-12	2019-11-14	British Indian Ocean	3.43
5	2019-11-13	2019-03-23	2019-12-29	Bulgaria	3.04
7	2011-10-15	2011-10-20	2011-10-25	Hong Kong	3.06
10	2011-02-28	2011-03-10	2011-03-27	Netherlands	3.24
9	2007-04-29	2007-04-30	2007-05-01	Netherlands Antilles	3.64
6	2015-10-20	2015-11-28	2015-04-29	Pitcairn Islands	3.50
0	2012-05-09	2012-05-10	2012-06-02	Singapore	2.96

Test Query 5

```
SELECT DISTINCT property_id, host_id, city, province, country FROM Property PRO
        WHERE NOT EXISTS (SELECT * FROM Rental_Agreement RA
                                WHERE RA.property_id = PRO.property_id)
```

Results for Test Query 5

	property_id [PK] integer	host_id integer	city character varying (20)	province character varying (20)	country character varying (20)
1	4	252	Collinsberg	Missouri	Madagascar
2	3	135	Lake Tiaraside	Yukon	Chad
3	1	2	Ottawa	Ontario	Canada
4	2	135	West Lois	Quebec	Palestinian Territor

Test Query 6

```
SELECT property_id, start_date as rental_start, end_date as rental_end, city, country, street FROM Rental_Agreement RA
        INNER JOIN Property PRO
                USING(property_id)
        WHERE extract(day from RA.start_date) = 10;
```

Results for Test Query 6

	property_id integer	rental_start date	rental_end date	city character varying (20)	country character varying (20)	street character varying (20)
1	0	2012-05-10	2012-06-02	Lake Billstad	Singapore	Locheland drive
2	10	2011-03-10	2011-03-27	Abagailberg	Netherlands	Rodeo drive

Test Query 7

```
WITH
managers AS (
        SELECT * FROM Employee e INNER JOIN Manager m ON m.manager_id = e.employee_id),
employees AS (
        SELECT * FROM Employee e1 left outer join Manager m1 on e1.employee_id = m1.manager_id  where manager_id
is NULL)
(Select * From employees where salary > 1500 order by employee_id)
UNION ALL
(Select * From managers where salary > 1500 order by employee_id);
```

Results for Test Query 7

employee_id integer	firstname character varying (20)	lastname character varying (20)	email_address character varying (50)	position character varying (30)	salary double precision	branch character varying (20)	manager_id integer
575	Tanner	Nicolas	mdach@gmail.com	Lawyer	22853.87	Netherlands Antilles	[null]
628	Rosamond	Koss	amueller@hotmail.com	Lawyer	20853.87	Netherlands	[null]
644	Leonora	Pollich	lily.stehr@gmail.com	Lawyer	26853.54	Netherlands	[null]
660	Elbert	Balistreri	adriana.rath@marks.com	HR Rep	26853.54	Pitcairn Islands	[null]
675	Orval	Hahn	carroll92@pollich.info	CR Representative	28853.54	Madagascar	[null]
867	Verla	Moen	eddie07@grady.com	Support Rep	22853.54	Palestinian Territor	[null]
907	Camron	Stanton	trey15@bogisich.com	Lawyer	66853.54	Netherlands Antilles	[null]
945	Elody	Rice	noe.maggio@yahoo.com	Lawyer	28853.54	Palestinian Territor	[null]
1	Jake	Evans	JakeEvans@rentals.com	Development Manager	50000	USA	1
2	Harrison	Cooke	HarrisonCooke@rentals.com	Product Manager	50000	Canada	2
3	Steve	Jobs	steverobs@rentals.com	Development Manager	100000	France	3
4	Miranda	Low	Miranda@rentals.com	Project Manager	23472	British Indian Ocean	4
5	Blake	Brown	Blake@rentals.com	HR Manager	88965.12	Chad	5
6	Jeff	Smith	Jeff@rentals.com	Finance Manager	77891.23	Hong Kong	6
7	Debra	Power	Debra@rentals.com	Project Manager	44567.53	Madagascar	7

Test Query 8

```
SELECT property_type, H.firstname as host_name, H.unit_number as street_number, H.street, amount, transaction_type  FROM
Property PRO
        INNER JOIN Payment PA
                on(PRO.property_id = PA.property_id and PRO.host_id = PA.host_id)
        INNER JOIN Host H
                ON (PRO.host_id = H.host_id )
        INNER JOIN Pricing PR
                ON(PRO.property_id = PR.property_id)
```

Results for Test Query 8

	property_type character varying (20)	host_name character varying (20)	street_number integer	street character varying (20)	amount double precision	transaction_type character varying (20)
1	Shared Space	Glenna	417	ut	123.45	Cash

Test Query 9

```
UPDATE Phonenumber
SET phone_number = '111-111-1111'
WHERE phone_number = '555-555-5555'
```

Results for Test Query 9

user_id [PK] integer	phone_number [PK] character varying (20)
1	613-823-4352
3	613-823-7791
31	134-559-0671x28720
110	916-694-5092x524

user_id [PK] integer	phone_number [PK] character varying (20)
1	555-555-5555
3	613-823-7791
31	134-559-0671x28720
110	916-694-5092x524

Test Query 10

```
CREATE FUNCTION FirstNameFirst(firstname varchar(20), lastname varchar(20))
        RETURNS varchar(50) AS $$
                SELECT CONCAT(firstname, ' ', lastname) as result
                $$ LANGUAGE SQL;
```

Results for Test Query 10

guest_id [PK] integer	firstnamefirst character varying
3	Ben Baker
4	Theo Holland
5	David Chapmen
6	Caden Koch
7	Bruce Bright
31	Shaniya Schmidt
110	Garfield Ferry
135	Javonte Cruickshank
241	Theresa Effertz
252	Aliza Nitzsche

Examples of DBA Queries

A database administrator can create and assign roles using pgAdmin4. The following are some examples of such queries:

```
CREATE VIEW property_pricing AS SELECT * FROM Property
INNER JOIN Pricing USING (property_id)
```

```
CREATE ROLE manager;

GRANT UPDATE ON Employee TO manager;

CREATE ROLE host_role;

GRANT UPDATE ON Property TO host_role;

REVOKE ALL ON manager FROM PUBLIC;
```