Index

Introduction

Restaurant Management System is a crucial tool for the hospitality industry that enables restaurants to manage their operations e ciently. It o ers numerous functionalities that help restaurants to streamline their operations and provide better services to their customers. In this project, we aim to develop a Restaurant Management System using SQL and PL/SQL. The system will provide features such as showing the menu, creating customer profiles, placing orders, giving tips, and more.

Requirement Analysis

To develop a successful Restaurant Management System, we need to analyze the requirements of the system. The following are the primary requirements for the system:

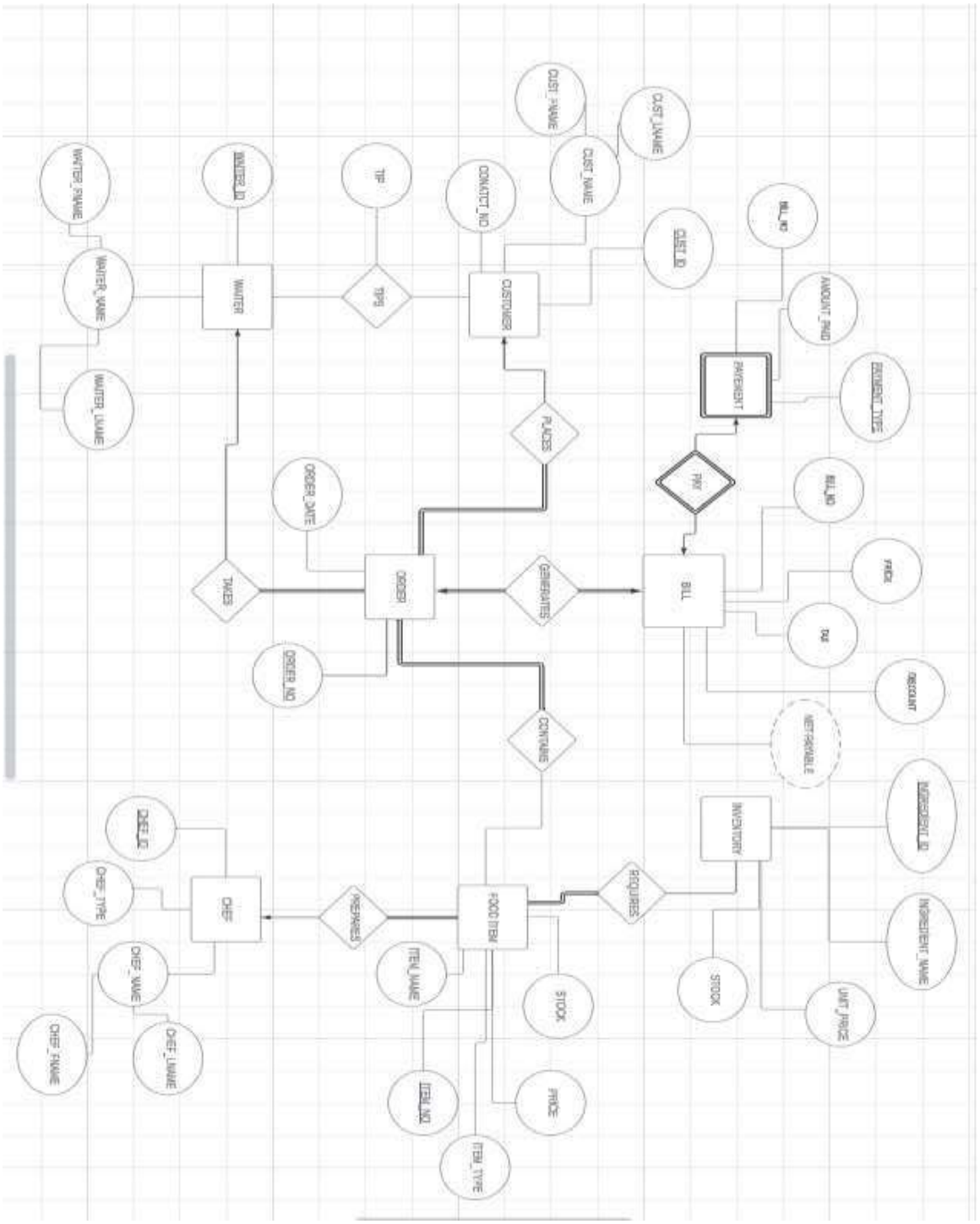
1. **Menu Management:** The system must have the ability to display the menu, including the items, their prices, and descriptions.
2. **Customer Management:** The system should allow the sta to create customer profiles and store their information, including their name, contact details,etc.
3. **Order Management:** The system should provide features to place orders and add more items to order.
4. **Bill and Tips Management:** The system should be able to generate bills and also allow customers to tip the waiter.

Software Requirements:

1. SQL
2. PL/SQL

3. Oracle Live SQL

ER Diagram



ER to Table

Relation 'Places'

Customer:- cust_id , cust_fname , cust_lname, contact_no Ord:- ord_no , ord_date , cust_id (FK)

Relation 'Takes'

Ord:- ord_no , ord_date , waiter_id (FK)
Waiter:- waiter_id , waiter_name , waiter_lname

Relation 'Tips'

Customer:- cust_id , cust_fname , cust_lname , contact_no
Waiter:- waiter_id , waiter_fname , waiter_lname
Tips:- cust_id (FK) , waiter_id (FK) , tip

Relation 'Prepares'

Food:- item_no , item_name , item_type , item_price , item_stock , chef_id (FK)
Chef:- chef_id , chef_fname , chef_lname , chef_type

Relation 'Generates'

Ord:- ord_no , ord_date
Bill:- bill_no , tot_price , tax , discount , net_payable , ord_no (FK)

Relation 'Contains'

Food:- item_no , item_name , item_type , item_price , item_stock
Contains:- item_no (FK) , ord_no (FK)

Ord:- ord_no , ord_date

Relation 'Requires'

Inventory:-ingredient_id, ingredient_name,unit_price,stock

Food_item:- item_no , item_name , item_type , item_price , item_stock

Relation 'pay'

Bill:- bill_no , tot_price , tax , discount , net_payable , ord_no (FK)

Payment:- amount_paid, payment_type, bill_no

Normalization

Customer Table (Already in 3NF):

cust_id	cust_fname	cust_lname	contact_no

Waiter Table (Already in 3NF)

waiter_id	waiter_fname	waiter_lname

Tips Table (Already in 3NF)

waiter_id (FK)	cust_id (FK)	tip

Order Table (Already in 3NF)

ord_no	ord_date	cust_id (FK)	waiter_id (FK)

Chef Table (Already in 3NF)

chef_id	chef_fname	chef_lname	chef_type

Food Table (in 2NF)

As item_no → item_type and item_type → chef_id

item_no	item_name	item_type	item_price	item_stock	chef_id (FK)

Breaking it into further tables Food Table:

item_no	item_name	item_type	item_price	item_stock

Prepares Table:

item_type	chef_id (FK)

Contains Table (Already in 3NF)

ord_no (FK)	item_no (FK)

Bill Table (Already in 3NF)

bill_no	tot_price	tax	discount	net_payable	ord_no (FK)

SQL

Creation Of Tables

```
create table waiter( waiter_id integer
    primary key, waiter_fname
    varchar(50) not null, waiter_lname
    varchar(50)
);

create table customer( cust_id
    integer primary key, cust_fname
    varchar(50) not null, cust_lname
    varchar(50), contact_no integer
);

create table tips( waiter_id integer references waiter(waiter_id),
   cust_id integer references customer(cust_id), tips integer not
   null
);

create table ord( ord_no integer primary key,
    ord_date date not null, cust_id integer
    references customer(cust_id), waiter_id integer
    references waiter(waiter_id)
);

create table chef( chef_id integer
    primary key, chef_fname
    varchar(50) not null,
    chef_lname varchar(50), chef_type
    varchar(50) not null
);
```

```
create table food( item_no integer
    primary key, item_name
    varchar(50) not null, item_type
    varchar(50) not null,
    item_price integer not null,
    item_stock integer
);

create table contains( ord_no integer references
    ord(ord_no), item_no integer references
    food(item_no)
);

create table prepares( item_type
    varchar(50) primary key, chef_id
    integer references chef(chef_id)
);

create table bill( bill_no
     integer primary key,
     tot_price integer not null,
     tax float default 5, discount
     integer default 0,
     net_payable float as
(tot_price+(tot_price*tax/100)-(tot_price*discount/100)), ord_no
     integer references ord(ord_no)
);


CREATE TABLE payment (
bill_no INTEGER REFERENCES
bill(bill_no),
  payment_type VARCHAR(50) NOT
```

```
    NULL,   amount_paid
INTEGER NOT NULL,
  CONSTRAINT pk_payment PRIMARY
    KEY (bill_no)
);

CREATE TABLE inventory (
ingredient_id INTEGER PRIMARY
    KEY,   ingredient_name
VARCHAR(50)
    NOT NULL,   unit_price
INTEGER NOT NULL,   stock
INTEGER NOT NULL
);
```

After Inserting Dummy Values Customer:

CUST_ID	CUST_FNAME	CUST_LNAME	CONTACT_NO
1	Alice	Brown	900000001
2	Bob	Green	900000002
3	Charlie	Blue	900000003

Waiter:

WAITER_ID	WAITER_FNAME	WAITER_LNAME
1	John	Doe
2	Jane	Smith
3	Bob	Johnson

Tips:

WAITER_ID	CUST_ID	TIPS
1	1	10
1	3	20

Food:

ITEM_NO	ITEM_NAME	ITEM_TYPE	ITEM_PRICE	ITEM_STOCK
2	French_Fries	Appetizer	50	100
3	Chocolate_cake	Dessert	80	30
1	Cheeseburger	Main Course	100	50

Chef:

CHEF_ID	CHEF_FNAME	CHEF_LNAME	CHEF_TYPE
1	John	Wick	Head_Chef
2	Sarah	Curry	Sous_Chef
3	Robert	Gun	Sous_Chef

Prepares:

ITEM_TYPE	CHEF_ID
Main_course	1
Appetizer	2
Dessert	3

Order:

ORD_NO	RD_DATE	CUST_ID	WAITER_ID
1	15-APR-24	1	1
2	16-APR-24	2	2
3	17-APR-24	3	1

Contains:

ORD_NO	ITEM_NO
1	1
1	2
2	2
2	3
3	1
3	3

PAYMENT:

BILL_NO	PAYMENT_TYPE	AMOUNT_PAID
3	Cash	230
1	Credit Card	180
2	UPI wallet	150

INVENTORY:

INGREDIENT_ID	INGREDIENT_NAME	UNIT_PRICE	STOCK
1	Beef Patty	50	20
2	Bun	10	30
3	French Fries (Frozen)	20	50
4	Chocolate	30	15
5	Flour	15	40

PL/SQL

1. show_menu procedure:

Displays the items and their prices using a cursor.

```
declare
cursor c1 is select item_name,item_price from food;
rec1 c1%rowtype;
procedure show_menu is
    begin
    open c1;
    loop
        fetch c1 into rec1;
        exit when c1%notfound;
        dbms_output.Put_line('Item :'||rec1.item_name||'  Price : ₹'||rec1.item_price);
    end loop;
    close c1;
end;

begin
show_menu;
end;

```

```
Statement processed.
Item :Cheeseburger  Price : ₹100
Item :French Fries  Price : ₹50
Item :Chocolate Cake  Price : ₹80
```

2. get_cust_id Function

If a customer already exists then returns the existing ID else creates a new customer and returns the new ID.

```
declare
id integer;
if_exists integer:=0;
function get_cust_id(fname in varchar,lname in varchar,contact in integer) return number is
    begin
        select count(*) into if_exists from customer where cust_fname=fname and cust_lname=lname and contact_no=contact;
        if if_exists>0 then
            select cust_id into id from customer where cust_fname=fname and cust_lname=lname and contact_no=contact;
            return(id);
        else
            select count(*)+1 into id from customer;
            insert into customer values(id,fname,lname,contact);
            return(id);
        end if;
    end;

begin
id:=get_cust_id('Blake','Ryan',900000004);
dbms_output.Put_line('Customer Id is ' || id);
end;
```

```
Statement processed.
Customer Id is 4
```

3. Placing Order

   a. in_stock trigger:

      Before Inserting the food items in the order it checks if the items are in stock if they are not in stock it will raise an error.

   b. after_order trigger:

      After Inserting the food items in the order it updates the stock of the items and decreases them accordingly.

   c. place_order function:

      This function inserts the items in the form of an array of item_no in the order and returns the order_no to the customer.

   d. add_order procedure:

      This function adds the items in the form of an array of item_no in the order of the given order_no.

```
create or replace trigger in_stock
before insert on contains for each row
declare
    stock integer;
begin
    select item_stock into stock from food where food.item_no=:new.item_no;
    if stock=0 then raise_application_error(-20000,'Out Of Stock');
    end if;
end;



create or replace trigger after_order
after insert on contains for each row
begin
    update food set item_stock=item_stock-1 where item_no=:new.item_no;
end;

```

```
Trigger created.
```

```
declare
type num_array is varray(50) of integer;
items num_array;
order_no integer;

function place_order(id in integer,items in num_array,wait_id in number) return integer is
begin
    select count(*)+1 into order_no from ord;
    insert into ord values(order_no,sysdate,id,wait_id);
    for i in 1..items.count loop
        insert into contains values(order_no,items(i));
    end loop;
    return (order_no);
end;

begin
items:=num_array(1,2);
order_no:=place_order(4,items,3);
dbms_output.Put_line('Your Order No is '||order_no);
end;

```

```
Statement processed.
Your Order No is 4
```

```
declare
type num_array is varray(50) of integer;
items num_array;
order_no integer;

procedure add_order(order_no in integer,items in num_array) is
begin
    for i in 1..items.count loop
        insert into contains values(order_no,items(i));
    end loop;
end;

begin
items:=num_array(3);
add_order(4,items);
end;
```

```
Statement processed.
```

After processing all the statements we can see in the contains table the order_no 4 has three items and stock has decreased by 1.

```
select c.ord_no,f.item_no,f.item_stock from food f, contains c where f.item_no=c.item_no and ord_no=4;
```

ORD_NO	ITEM_NO	ITEM_STOCK
4	1	49
4	2	99
4	3	29

4. Generating Bill

   a. display_bill trigger:

      It displays the bill along with the bill_no,ord_no,items, price, total price, discount, tax and the net_payable amount to the customer.

   b. generate_bill procedure:

It takes in the values order_no and any discount value and inserts the given values into the bill table.

```
create or replace trigger display_bill
after insert on bill for each row
begin
    dbms_output.Put_line('Total Price: ₹'||:new.tot_price);
    dbms_output.Put_line('Tax: '||:new.tax||' %');
    dbms_output.Put_line('Discount: '||:new.discount||' %');
    dbms_output.Put_line('Net Payable Amount: ₹'||:new.net_payable);
end;
```

```
Trigger created.
```

```
declare
cursor c2 (n integer) is select f.item_no,f.item_name,f.item_price,c.ord_no from food f,contains c where f.item_no=c.item_no and c.ord_no=n;
rec2 c2%rowtype;
total integer:=0;
b_no integer;

procedure generate_bill(order_no in integer, disc in float) is
begin
    open c2(order_no);
    select count(*)+1 into b_no from bill;
    dbms_output.Put_line('Bill No: '||b_no||' Order_no: '||order_no);
    loop
        fetch c2 into rec2;
        exit when c2%notfound;
        dbms_output.Put_line('Item: '||rec2.item_name||'  Price: ₹'|| rec2.item_price);
        total:=total+rec2.item_price;
    end loop;
    if (total>1000) then
        insert into bill(bill_no,tot_price,tax,discount,ord_no) values(b_no,total,10,disc,order_no);
    else
        insert into bill(bill_no,tot_price,discount,ord_no) values(b_no,total,disc,order_no);
    end if;
    close c2;
end;

begin
generate_bill(4,0);
end;
```

```
Statement processed.
Bill No: 4 Order_no: 4
Item: Cheeseburger  Price: ₹100
Item: French_Fries  Price: ₹50
Item: Chocolate_cake  Price: ₹80
Total Price: ₹230
Tax: 5 %
Discount: 0 %
Net Payable Amount ₹241.5
```

5. Tips:

   a. give_tip procedure:

   It takes in the value of the tip given by the customer to the denoted waiter.

   b. display_waiter_tip procedure:

   It displays the total tips collected by a specific waiter.

```
declare
procedure give_tip(id in integer, wait_id in integer, t in integer) is
begin
    insert into tips values(wait_id,id,t);
    dbms_output.Put_line('Waiter 3 Recieved ₹'||t||' tip');
end;

begin
give_tip(4,3,10);
end;
```

```
Statement processed.
Waiter 3 Recieved ₹10 tip
```

```
declare
tot integer;
procedure display_waiter_tip(wait_id in integer) is
begin
    select sum(tip) into tot from tips where waiter_id=wait_id;
    dbms_output.Put_line('Total tip for waiter id '||wait_id||' is ₹'||tot);
end;

begin
display_waiter_tip(3);
end;

```

```
Statement processed.
Total tip for waiter id 3 is ₹10
```

6. BILLS:

Calculates total bill of the customer using total price, taxes and discounts, and returns an integer.

```
CREATE OR REPLACE FUNCTION calculate_bill(p_bill_no INT) RETURN INT AS
  v_total_price INT;
  v_tax INT;
  v_discount INT;
BEGIN
  -- Retrieve the total price, tax, and discount for the given bill number
  SELECT tot_price, tax, discount INTO v_total_price, v_tax, v_discount
  FROM bill
  WHERE bill_no = p_bill_no;

  -- Calculate the final amount including tax and discount
  RETURN (v_total_price + v_tax - v_discount);
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RETURN NULL;
END calculate_bill;
/
```

```
/
DECLARE
  v_bill_no INT := 1; -- Provide the bill number for which you want to calculate the bill
  v_final_amount INT;
BEGIN
  v_final_amount := calculate_bill(v_bill_no);
  IF v_final_amount IS NOT NULL THEN
    DBMS_OUTPUT.PUT_LINE('Final Amount for Bill ' || v_bill_no || ': ' || v_final_amount);
  ELSE
    DBMS_OUTPUT.PUT_LINE('Invalid Bill Number.');
  END IF;
END;
/
```

```
Function created.


Statement processed.
Final Amount for Bill 1: 184
```

7. INVENTORY:

Check ingredient availability:- Checks the availability of ingredients required for the food_items ordered by the customers.

```
339 CREATE OR REPLACE PROCEDURE check_ingredient_availability(p_ingredient_id INT, p_quantity INT) AS
340   v_stock INT;
341 BEGIN
342   -- Retrieve the stock for the given ingredient id
343   SELECT stock INTO v_stock
344   FROM inventory
345   WHERE ingredient_id = p_ingredient_id;
346
347   -- Check if the required quantity is available
348   IF v_stock >= p_quantity THEN
349     DBMS_OUTPUT.PUT_LINE('Ingredient is available in sufficient quantity.');
350   ELSE
351     DBMS_OUTPUT.PUT_LINE('Insufficient quantity of ingredient.');
352   END IF;
353 EXCEPTION
354   WHEN NO_DATA_FOUND THEN
355     DBMS_OUTPUT.PUT_LINE('Ingredient not found.');
356 END check_ingredient_availability;
357 /
```

```
358
359 -- Calling the check_ingredient_availability procedure
360 DECLARE
361   v_ingredient_id INT := 1; -- Provide the ingredient ID you want to check
362   v_quantity_needed INT := 25; -- Provide the quantity you need
363 BEGIN
364   check_ingredient_availability(v_ingredient_id, v_quantity_needed);
365 END;
366 /
```

```
Procedure created.


Statement processed.
Insufficient quantity of ingredient.
```

Conclusion

In conclusion, the Restaurant Management System is an essential tool for the hospitality industry, and it provides numerous features to streamline operations and provide better services to customers. Developing a Restaurant Management System using SQL and PL/SQL is an excellent way to build a reliable, robust, and scalable system that meets the requirements of the restaurant industry.

However, while developing such a system, it is essential to consider the system requirements, external requirements, and hardware requirements to ensure that the system performs optimally and securely. By taking these requirements into account, the Restaurant Management System will be able to handle a large number of transactions and users, integrate with other third-party systems, be accessible from multiple devices and platforms, and comply with the relevant regulations and standards.

Moreover, the system will provide a seamless and intuitive user experience, making it easy for the restaurant staff to navigate and perform their tasks. The system will have a user-friendly interface with clear instructions and prompts, ensuring that the staff can use it with minimal training.

In summary, developing a Restaurant Management System using SQL and PL/SQL is an excellent investment for restaurants, as it will help them manage their operations more efficiently, reduce errors, and provide better services to their customers.

References

1. https://www.youtube.com/@parteekbhatia
2. Parteek Bhatia and Gurvinder Singh, Simplified Approach to DBMS.
3. Silverschatz A., Korth F. H. and Sudarshan S., Database System Concepts, Tata McGraw Hill (2010) 6th ed.