Course No: CSE 3110

Course Name: Database Systems Laboratory

Project Name: **COOK’S Management Database.**

**Submitted to:**

|  |  |
| --- | --- |
|  |  |
| Jarin Firose Moon  Lecturer  Department of Computer Science & Engineering  Khulna University of Engineering & Technology | **Md. Milon Islam**  **Lecturer**  **Department of Computer Science & Engineering**  **Khulna University of Engineering & Technology** |

**Submitted by:**

**Md. Azahar Alam**

Roll: **1507024**

Section: A

Group: A1

Department of Computer Science & Engineering

Khulna University of Engineering & Technology

Project Objectives:

1. To create a database for storing the information of cooks services to it’s customers.

2. To Create a database for managing the employee information and it’s personal information.

1. To collect information of various clients and store it’s information.
2. To store information of wages payment to employees and gross profit of a contract.
3. To create a database to ease the work of cook in storing and managing various data.

InTroDuction :

The project named ‘Cook’s Management Database’ is for the cooks who used to work on various social functionalities like Marriage party, Birthday party. This database is about to store the information of clients, employees, employee’s wages for each contract, the income of every contract and gross profit. It can delete or update the information of every employees and clients. It can manage the relationships between client and the cook, also between employees and the contractor.

Methodology :

In a database, there are DB objects such as table, view, synonyms, sequences etc. In this project there is 2 DB objects Table and Sequence. We have to Write codes to create the structure of this database. I am going to show the sample codes of creating the structure of the database.

# Table For create structure of DB:

In relational databases, and flat file databases, a table is a set of data elements (values) using a model of vertical columns (identifiable by name) and horizontal rows, the cell being the unit where a row and column intersect. A table has a specified number of columns, but can have any number of rows.

We can make the structure of database by DDL(Data Definition language). Data Definition Language (DDL) is a standard for commands that define the different structures in a database. DDL statements create, modify, and remove database objects such as tables, indexes, and users. Common DDL statements are CREATE, ALTER, and DROP.

|  |  |
| --- | --- |
|  |  |
| Table Creation | Create Table Table\_name (Column1 description, column2 description,…….., Key Constraints); |
| Drop Table | Drop Table Table\_name; |
| Add New Column | Alter Table Table\_name add Column\_name Column definition; |
| Drop Column | Alter table Table\_name drop column\_name ; |
| Sequence Creation | create sequence seq\_name  start with ‘Value’  increment by ‘Value’  maxvalue ‘Value’  nocycle  nocache ; |
| Drop Sequence | Drop sequence seq\_name; |
|  |  |

# Table for manipulating data:

We can manipulate data by DML. **DML** is abbreviation of Data Manipulation Language. It is used to retrieve, store, modify, delete, insert and update data in **database**. Examples: SELECT, UPDATE, INSERT statements. DDL is abbreviation of Data **Definition** Language. It is used to create and modify the structure of **database** objects in**database**.

|  |  |
| --- | --- |
|  |  |
| Insert | Insert into table\_name(column1,column2……) values(value of col1,value of col2,……); |
| Update | Update table\_name set column\_name=new value where conditions; |
| Delete | Delete from Table\_name where conditions; |
| Query | Select col\_name from table\_name where conditions; |
|  |  |

# Procedure:

**A procedure in Oracle**, **Oracle's** database language, PL/SQL, is made up of **stored procedures**, which build applications within **Oracle's** database. IT professionals use **stored** programs in **Oracle's** database to properly write and test code, and those programs become **stored procedures** once compiled.

Here is a sample format to build oracle storable procedure:

CREATE [OR REPLACE] PROCEDURE procedure\_name

[ (parameter [,parameter]) ]

IS

[declaration\_section]

BEGIN

executable\_section

[EXCEPTION

exception\_section]

END [procedure\_name];

# Functions:

A stored **function** (also called a user **function** or user-defined **function**) is a set of **PL/SQL** statements you can call by name. ... The call specification tells **Oracle Database** which Java method, or which named **function** in which shared library, to invoke when a call is made.

Here is a sample format to build oracle storable function:

CREATE [OR REPLACE] FUNCTION function\_name

[ (parameter [,parameter]) ]

RETURN return\_datatype

IS | AS

[declaration\_section]

BEGIN

executable\_section

[EXCEPTION

exception\_section]

END [function\_name];

# Trigger:

A **trigger** is a special type of stored procedure that automatically executes when an event occurs in the **database** server. DML **triggers** execute when a user tries to modify data through a data manipulation language (DML) event. DML events are INSERT, UPDATE, or DELETE statements on a table or view.

Here is a format for trigger creation.

CREATE [ OR REPLACE ] TRIGGER trigger\_name

BEFORE/AFTER INSERT/UPDATE/DELETE

ON table\_name

[ FOR EACH ROW ]

DECLARE

-- variable declarations

BEGIN

-- trigger code

EXCEPTION

WHEN ...

-- exception handling

END;

# File Operation:

In **Oracle PL**/**SQL**, UTL\_FILE is an **Oracle** supplied package which is used for **file operations** (read and write) in conjunction with the underlying operating system. UTL\_FILE works for both server and client machine systems. A directory has to be created on the server, which points to the target **file.**

|  |  |
| --- | --- |
|  |  |
| File Read | Before Load CSV file into database We need to create directory and seek for grant to read or write file in that directory. To Create directory & and grant the read write permission to the specified USER, we need to execute following commands as SYS or SYSTEM user. e:\mycsv\ must be a physical path on the disk.  SQL> create or replace directory DBDIR as 'e:\Database Project\';  Directory created.  SQL> grant read, write on directory DBDIR to A1507024;  Grant succeeded.  DECLARE  F UTL\_FILE.FILE\_TYPE;  --Other Variables.    BEGIN  F := UTL\_FILE.FOPEN (Directory, CSV File, 'Mode')  IF UTL\_FILE.IS\_OPEN(F) THEN  --Code for extract data from CSV file  --INSERT DATA    EXCEPTION    UTL\_FILE.FCLOSE(F);  END;  / |
| File Write | **To create a file, we need to create a directory and have the read write permission as**  **1)** create or replace directory MYCSV as '/home/oracle/mycsv';  **Note:** /home/oracle/mycsv has to be physical location on disk.  **2)** grant read, write on directory MYCSV to scott;  **Following is the pl/sql sample code to create CSV file**  DECLARE  --Declare a cursor and row type variable.  BEGIN    F := UTL\_FILE.FOPEN (Directory, CSV File, 'Mode')      --Code for Download the data into CSV file  UTL\_FILE.FCLOSE(F); END; / |
|  |  |

Description & Functionality :

In This section I’m going to discuss about the various components of my database.

# Tables:

There are 5 tables in the Database. Here is the functionality of every Table.

|  |  |
| --- | --- |
| Table Name | Functionality of the Table |
|  |  |
| Clients | Store The Information of the Clients. There 7 columns and 1 key constraints.  **Serial\_No** :Store serial of the rows.  **Name** :Store name of the clients  **Contact\_No:** Save contact number of the Clients.  **Email** Save email.  **Address**: Store address of clients.  **Services\_Provided** Services given to this clients.  **Suggested\_By :**The person who suggest this clients.  **primary key** : Serial\_No  **Sample Code:**  create table Clients(Serial\_No number(4) not null, Name varchar(15) not null, Contact\_No number(11) not null unique,Email varchar(24), Address varchar(20), Services\_Provided number(2) default 1, Suggested\_By varchar(20),primary key(Serial\_No)); |
| Services | This Table Store the information of the services that the cook given to his clients. There are 10 columns and 2 key constraints.  **Serial\_No** :Store serial of the rows.  **Booking\_Date :** Store the booking date of the service,  **Service\_Date** : Store the service date  **Client\_Id:** Store the Clients Identity.  **People** : Store the no of People attend the party.  **Community\_Center** :The name of the community center where the party will be arranged.  **Type:**  Store type of party (Marriage).  **Rate** : Per head service charge.  **Advance** : Money received in advance.  **primary key:** Serial\_No.  **foreign key :** Client\_Id References to the Clients Serial\_No .  **Sample Code:**  create table Services(Serial\_No number(4) not null, Booking\_Date date not null, Service\_Date date not null, Client\_Id Number(4) not null, People number(6) not null, Community\_Center varchar(15) , Type Varchar(12) not null, Rate number(2) not null, Advance number(6) default 0, primary key (Serial\_No), foreign key(Client\_Id) References Clients(Serial\_No) on delete cascade); |
| Employees | This table stores the Information of the employees who work under this cook or assist the cook. There are 10 columns and 1 key constraint in this table.  **Serial\_No** :Store serial of the rows  **ID** : Id of the employee.  **Employee\_Name** Name of the employee.  **Entry\_Date** Entry date of this employee.  **Category** Store the category of the employee.  **Gender:**  Store the Gender of the employee.  **Birthdate** : Store the Birthdate of the Employee.  **Age** :Store the age of the employee.  **Contact\_No** Store the contact no.  **Address:** Address of employee.  **Primary Key:** ID. |
| Wages | This table stores the information of the wages that is given to employees. Also store the Services for which the wage is given. There are 7 columns and 3 key constraints.  **Serial\_No** :Store serial of the rows  **Service\_Date** : Store the service date.  **Services\_ID:**The Id Of the service  **Clients\_Id** The Id of the clients.  **Employee\_ID**: Store the employee ID whom wage is given.  **wage** : The amount of wage given to the employee.  **Payment\_date** :Store the payment date.  **foreign keys:** Employee\_ID, Services\_ID, Clients\_ID. |
| Income | Store the information of income, total wage given, Gross profit of every service. There are 5 columns and No key constraint.  **Serial\_No** :Store serial of the rows  **Services\_ID:**The Id Of the service.  **Income:** Store the Income of every service.  **Wages\_given :** Total Wages given to employees.  **Gross\_Profit** Gross profit of this service. |
|  |  |

# Sequences:

There are 5 sequences in the database.

|  |  |
| --- | --- |
| Sequence | Functionality of the sequence |
|  |  |
| Clients\_serial | Store the serial no of every row of clients Table.  **Sample Code:**  create sequence Clients\_Serial  start with 1  increment by 1  maxvalue 9999  nocycle  nocache ;  Same Code for each sequence Creation (Name should be changed). |
| Services\_serial | Store the serial no of every row of Services Table. |
| Employees\_serial | Store the serial no of every row of Employees Table. |
| wages\_serial | Store the serial no of every row of Wages Table. |
| Income\_serial | Store the serial no of every row of Income Table. |

# Procedures:

There are 3 procedures in the database. Here is the functionality of every procedure.

|  |  |
| --- | --- |
| Procedures | Functionality of the procedures |
| setCategory | Set the category of every Employee.  **Sample Code:**  create or replace procedure setCategory  is  type namearray is varray(50) of employees.ID%type;  ID\_array namearray := namearray();  cnt number(2);  counter number(2);  dur number(6,2);  begin  select count(\*) into cnt from employees;  for counter in 1..cnt  loop  ID\_array.extend;  select ID into ID\_array(counter) from employees where serial\_no=counter;    end loop;  for counter in 1..cnt  loop  select (sysdate-entry\_date)/365 into dur from employees where ID=ID\_array(counter);  if(dur>10)then  update employees set Category='A' where ID=ID\_array(counter);  elsif(dur>5) then  update employees set Category='B' where ID=ID\_array(counter);  else  update employees set Category='C' where ID=ID\_array(counter);  end if;  end loop;    end; |
| setWage | Set the wage of employee basis on their Category.  **Sample Code:**  create or replace procedure setWage  as  type WArray is varray(50) of wages.employee\_ID%type;  wageArray WArray := WArray();  --cursor C is select Employee\_id from wages;  cnt number(2);  counter number(2);  cat varchar(10);  begin  select count(\*) into cnt from wages;  for counter in 1..cnt  loop  wageArray.extend;  select employee\_ID into wageArray(counter) from wages where serial\_no=counter;  end loop;  for counter in 1..cnt  loop  select category into cat from employees where ID=wageArray(counter);  if(cat='A')then  update wages set wage=1000 where employee\_ID=wageArray(counter);  elsif(cat='B') then  update wages set wage=750 where employee\_ID=wageArray(counter);  else  update wages set wage=600 where employee\_ID=wageArray(counter);  end if;  end loop;  end; |
| setBonus | Calculate Bonus for every employee.  **Sample Code:**  create or replace procedure setBonus  as    cursor C is select Employee\_id,sum(wage) as wg from wages group by Employee\_id;  cnt number(7,2);  counter number(2);  cat varchar(10);  begin    for EidBonus in c  loop  cnt:=EidBonus.wg;  cnt:=cnt/2;  dbms\_output.put\_line('Employee ID : '||EidBonus.employee\_ID||' With the Eid bonus : '||cnt);  end loop;  end; |

# Functions:

There are 2 functions in the data base.

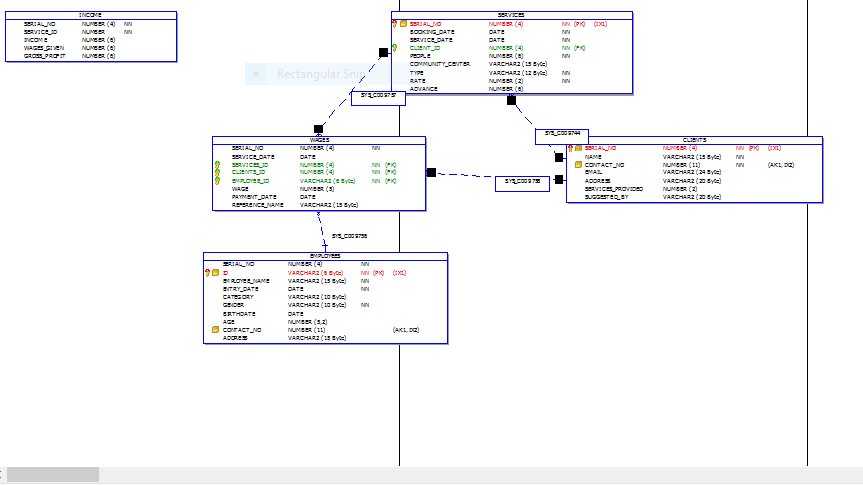
|  |  |
| --- | --- |
| Functions | Functionality of the Functions. |
| setIncome | Set the income of the services.  **Sample Code:**  create or replace function setIncome  (ppl\_in in number,r\_in in number)  return number  is  income number;  begin  income := ppl\_in\*r\_in;  return income;  end; |
| setGrossProfit | Set The gross profit of each service.  **Sample Code:**  create or replace function setGrossProfit  (inc number , id number)  return number  is  wage\_given number;  profit number;  begin  select sum(wage) into wage\_given from wages where services\_id=id;  update income set wages\_given=wage\_given where serial\_no=id;  profit := inc-wage\_given;  return profit;  end; |
|  |  |

E-R Diagram :

An **entity-relationship diagram** (**ERD**) is a data modeling technique that graphically illustrates an information system's entities and the relationships between those entities. An **ERD** is a conceptual and representational model of data used to represent the entity framework infrastructure.

**Here are some best practice tips for constructing an ERD:**

1. Identify the entities. The first step in making an **ERD** is to identify all of the entities you will use. ...
2. Identify relationships. Look at two entities, are they related? ...
3. Describe the relationship. How are the entities related? ...
4. Add attributes. ...
5. Complete the **diagram**.

****

Schema Diagram :

A database **schema** is the skeleton structure that represents the logical view of the entire database. A database **schema** defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of **schema diagrams**.

**To create a schema:**

1. In Object Explorer, expand the Databases folder.
2. Expand the database in which to create the new database schema.
3. Right-click the Security folder, point to New, and select Schema.
4. In the Schema - New dialog box, on the General page, enter a name for the new schema in the Schema name box.

|  |
| --- |
| **Clients** |
| Serial\_No |
| Name |
| Contact\_no |
| Email |
| Address |
| Services\_provided |
| Suggested\_by |

|  |
| --- |
| **Services** |
| Serial\_No |
| Booking\_date |
| Service\_date |
| Client\_ID |
| People |
| Community\_center |
| Type |
| Rate |
| Advance |

|  |
| --- |
| **Employees** |
| Serial\_no |
| ID |
| Name |
| Entry\_date |
| Category |
| Gender |
| Birthdate |
| Age |
| Contact\_No |
| Address |

|  |
| --- |
| **Wages** |
| Serial\_No |
| Services\_Date |
| Services\_Id |
| Clients\_ID |
| Employee\_ID |
| Wage |
| Payment\_name |
| Reference |

|  |
| --- |
| **Income** |
| Serial\_no |
| Services\_ID |
| Income |
| Wage\_given |
| Gross\_Profit |

# Figure: Schema Diagram of the database.

OutCome Of The Project:

This database will be a long time running as the services provided everytime. The DBMS is a central management system through which all the information can be distributed. As already said this will indicated about the required needed to take to lessen the information in certain service. As the technology progressing faster, this is a step that has to be taken by every cook as soon as possible. As information logs are getting bigger everyday and we are getting backdated every second. That’s why it should be immediate step that has to take. In future this database can be added in web application in the cook’s portal so that all the regional peoples can get connected and insert, update the perfect data to increase efficiency and working speed in personal perspectives.

# Academic outcome:

1. We can easily make the structure of database.

2. We can manipulate the raw data and also organize and manage it.

3. We can use functions, procedures, triggers to make the database dynamic.

4. This project help us to meet with various SQL commands and backend management.