NAME : SAMRIDHI GUPTA

USN : 1NH18CS168

SEM : 5

SECTION : C

CHAPTER 1

INTRODUCTION

1.1 COURSE OBJECTIVES

Python coding and database management are the most essential and highly demanding skills. It is vital for a software engineer to have a good hold on both of them. Python is a deciphered, elevated level and broadly useful programming language. Made by Guido van Rossum and first delivered in 1991, Python's plan reasoning stresses code comprehensibility with its prominent utilization of critical whitespace. Its language develops and object-arranged methodology plan to assist software engineers with composing clear, sensible code for little and huge scope ventures. Python is powerfully composed. It underpins various programming ideal models, including organized (especially, procedural), object-situated, and practical programming. Python is frequently portrayed as a "batteries included" language because of its complete standard library.

A data set is an assortment of data that is sorted out so it tends to be handily gotten to, oversaw and refreshed. PC data sets ordinarily contain collections of information records or documents, containing data about deals exchanges or communications with explicit clients. Normally, the information base chief gives clients the capacity to control read/compose access, indicate report age and dissect utilization. Information bases offer ACID (atomicity, consistency, separation and strength) consistence to ensure that information is steady and that exchanges are finished.

1.2 PROBLEM DEFINITION

For a successful and efficient running of any organization, there is a requirement of a user-friendly, transparent and a secured database. Designing a database with a GUI for keeping in track of electronic components supplied to the customer via an agency is the motivation of this project. The Spec Churner Business Tracker would be able to demonstrate the CRUD operations, i.e. create, read, update and delete data into the database through a Python GUI, designed using Tkinter. The electronic products like resistors, capacitors, transistors, etc. would make one of the strong entities. Products are manufactured by the original equipment manufacturer (OEM) and are shipped to various countries. Business agencies, which are well recognized by the Government of India, supply products to space organizations like our very own ISRO, which aid them in making of satellites and defence systems. Lastly, all these transactions aren't possible without payments and there is a must presence of banks. Henceforth, the database is going to be designed keeping in mind that all the business revolves around the supplier or the company. It's the government's effort to bring in more and more trade in the nation due to which, thousands of business opportunities are being generated. It's up to us that how we utilize these opportunities.

1.3 OUTCOMES OF THE PROJECT

After working on this project, I will be able to:

- code independently in Python language
- analyse and design a database for any organization
- connect Python to a database management system
- demonstrate CRUD operations, i.e. create, retrieve, update and delete information from a database
- work with Python libraries
- write queries into the database via an external connection
- design a graphical user interface using the Tkinter library of Python
- understand and apply concepts of the database like ER diagrams, keys, tables, constraints, etc.
- write queries for data query language (DQL), data manipulation language (DML), data definition language (DDL) and truncated control language (TCL)

CHAPTER 2

REQUIREMENT SPECIFICATION

2.1 HARDWARE REQUIREMENTS

- LAPTOP-2TUG730U
- HP Pavilion notebook
- Installed RAM- 16.0 GB
- Processor – Intel core i7, 8th gen-8750H

2.1 SOFTWARE REQUIREMENTS

- Anaconda Navigator
- Jupyter notebook with Python 3 kernel
- MySQL workbench 8.0 CE
- Python libraries

CHAPTER 3

IMPLEMENTATION FUNDAMENTALS

3.1 DATABASE FUNDAMENTALS

Python's standard library is broad, offering a wide scope of offices as demonstrated by the long chapter by chapter guide recorded underneath. The library contains worked in modules (written in C) that give admittance to framework usefulness, for example, record I/O that would somehow be unavailable to Python software engineers, just as modules written in Python that give normalized answers for some issues that happen in ordinary programming. A portion of these modules are unequivocally intended to empower and upgrade the transportability of Python programs by abstracting ceaselessly stage points of interest into stage impartial APIs.

All software applications interact with data, usually through a database management system. The various activities that can be performed in SQL using Python are:

- Connect to different database management systems with Python SQL libraries
- Interact with MySQL databases
- Perform common database queries using a Python application
- Develop applications across different databases using a Python script

Python offers various alternatives for creating GUI (Graphical User Interface). Out of all the GUI techniques, tkinter is the most normally utilized strategy. It is a standard Python interface to the Tk GUI toolbox dispatched with Python. Python with tkinter is the quickest and simplest approach to make the GUI applications. Making a GUI utilizing tkinter is a simple undertaking. To make a tkinter application:

- Bringing in the module – tkinter
- Make the principle window (holder)

- Add quite a few gadgets to the principle window
- Apply the occasion Trigger on the gadgets.

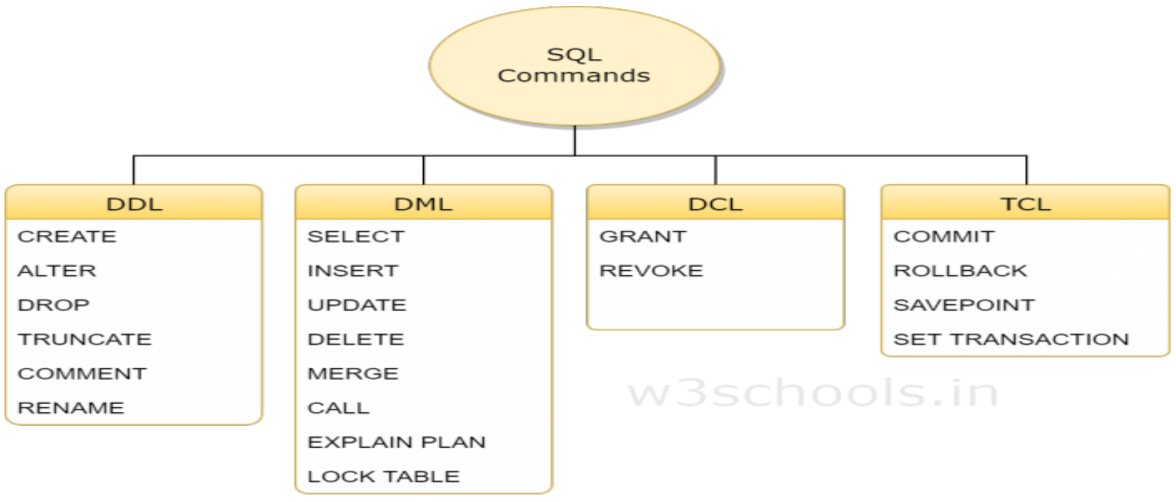
3.1.1 CONNECTING TO A DATABASE

Before you interact with any database through a Python SQL Library, you have to **connect** to that database. The library used here to connect to the MySQL database is the mysql.connector library. This helps in connecting the database to the Python via an external connection. The code used for connecting them is:

```
import mysql.connector as sql

mydb = sql.connect(

    host="localhost",

    user="root",

    passwd="****************",

    database="spec_churner",

    use_pure=True

)

print(mydb)
```

Line 1 of the code imports the mysql.connector library as sql. The “sql” acts likes a variable and it would enable in doing the same function as mysql.connector, without even writing it again and again. It can be placed by “sql” throughout the code.

- **Line 2** creates and object “mydb”, which encapsulates all the information required to connect to the database, namely the host, user, password and name of the database. “mydb” is used as a reference for performing any kind of queries into the database, and basically it acts like an instantiation for the connection. The TCP/IP port on which the MySQL server is listening (the default is 3306).
- **Line 3** specifies the host name of the database. The host name is simply the IP address of the MySQL server. It is important to note that the server of the database must be manually turned on, or else the code will throw an error saying that the “machine actively refused the connection”.
- **Lines 4, 5 and 6** take the user name, name of the database and the password for accessing it. You will be prompted to enter the password when MySQL Workbench attempts to establish the connection. MySQL Workbench can store the password in a vault.
- **Line 7** has a flag variable named use_pure, which is True by default because we want to establish a connection.



3.1.2 WRITING QUERIES FOR DATA DEFINITION LANGUAGE

DDL is short name of Data Definition Language, which manages information base compositions and depictions, of how the information ought to live in the information base. It is used to establish and modify the structure of objects in a database by dealing with descriptions of the database schema. Unlike data manipulation language (DML) commands that are used for data modification purposes, DDL commands are used for altering the database structure such as creating new tables or objects along with all their attributes (data type, table name, etc.). Given below are the DDL commands:

- CREATE - to make an information base and its articles like (table, list, sees, store method, capacity, and triggers)
- ALTER - modifies the structure of the current information base
- DELETE - erase objects from the information base
- TRUNCATE - eliminate all records from a table, including all spaces apportioned for the records are taken out
- COMMENT - add remarks to the information word reference
- RENAME - rename an item

Some of the queries used in the code are shown in the table below. They include queries used for creating tables for strong entities, their attributes along with their datatypes, alter commands and delete commands.

mycursor.execute('CREATE TABLE OEM (name VARCHAR(255), address VARCHAR(255), 0_code(int), contact bigint, price bigint)')

mycursor.exectue('CREATE TABLE COMPANY (name VARCHAR(255), address VARCHAR(255), state CHAR(20),

3.1.3 WRITING QUERIES FOR DATA MANIPULATION LANGUAGE

An information control language (DML) is a group of codes including orders allowing clients to control information in a data set. This control includes embeddings information into information base tables, recovering existing information, erasing information from existing tables and changing existing information. DML is generally fused in SQL information bases.

- SELECT: This command is used to retrieve rows from a table. The syntax is SELECT [column name(s)] from [table name] where [conditions]. SELECT is the most widely used DML command in SQL.
- UPDATE: This command modifies data of one or more records. An update command syntax is UPDATE [table name] SET [column name = value] where [condition].
- INSERT: This command adds one or more records to a database table. The insert command syntax is INSERT INTO [table name] [column(s)] VALUES [value(s)].
- DELETE: This command removes one or more records from a table according to specified conditions. Delete command syntax is DELETE FROM [table name] where [condition].

3.2 TKINTER FUNDAMENTALS

Tk/Tcl has for quite some time been a necessary piece of Python. It gives a powerful and stage free windowing toolbox, that is accessible to Python software engineers utilizing the tkinter bundle, and its augmentation, the tkinter.tix and the tkinter.ttk modules.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

- Import the *Tkinter* module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

```
•   %gui tk
•   from tkinter import *
•   import tkinter.messagebox as MessageBox
•   import mysql.connector as mysql
•   import sqlite3
•   import pandas as pd
•
•   '''def insert():
•       id_num = e_id.get()
•       company = e_company.get()
•       product = e_product.get()
•       country = e_country.get()
•
•       if(id_num=="" or company=="" or product=="" or country==""):
•           MessageBox.showinfo("Insert status", "All fields are required")
•       else:
•           cursor = mydb.cursor()
•           cursor.execute("Insert into tracker values ('"+id_num+"', '"+company+"',
    '"+product+"', '"+country+"')")
```

```
•         cursor.execute("commit")
•         MessgaeBox.showinfo("Insert Status", "Inserted Successfully")
•   '''
•
•   root = Tk()
•   root.geometry("700x500")
•   root.title("Spec Churner Business Tracker")
•
•   id_num = Label(root, text='Enter ID', font=('bold', 10))
•   id_num.place(x=20, y=30)
•
•   company = Label(root, text= 'Enter company name', font=('bold', 10))
•   company.place(x=20, y=60)
•
•   product = Label(root, text='Enter product name', font=('bold', 10))
•   product.place(x=20, y=90)
•
•   country = Label(root, text='Enter country', font=('bold', 10))
•   country.place(x=20, y=120)
•
•   ## creating entries for the user to enter values
•   e_id = Entry()
•   e_id.place(x=200, y=30)
•
•   e_company = Entry()
•   e_company.place(x=200, y=60)
•
•   e_product = Entry()
•   e_product.place(x=200, y=90)
•
```

- e_country = Entry()
- e_country.place(x=200, y=120)
-
- ## creating buttons for CRUD operations
- insert = Button(root, text="insert", font=("italic", 10), bg="white", command=insert)
- insert.place(x=20, y=140)
-
- delete = Button(root, text="delete", font=("italic", 10), bg="white", command=delete)
- delete.place(x=20, y=140)
-
- update = Button(root, text="update", font=("italic", 10), bg="white", command=update)
- update.place(x=20, y=140)
-
- get = Button(root, text="get", font=("italic", 10), bg="white", command=get)
- get.place(x=20, y=140)

3.3 CODING THE CRUD FUNCTIONS

In computer programming, create, read, update, and delete (CRUD) are the four essential elements of relentless storage. CRUD is likewise now and again used to portray UI shows that encourage seeing, looking, and evolving data, regularly utilizing PC based structures and reports. Together they make up the four essential tasks of capacity the board (direct control of the substance of capacity areas by clients) known as CRUD: create, read, update, and delete. CRUD activities are idempotent, implying that different utilizations of a similar activity have a similar impact on a capacity as a solitary application.

3.3.1 INSERT METHOD

The first function coded using the tkinter module is the insert() function, which is connected to the database from Jupyter notebook. When the user enters some details in the required fields of the tkinter GUI, this data is getting inserted into the database through the connection/ object which has been established in the code.

The GUI asks for the required fields in which data is supposed to be inserted, and it is done using the get method. Once data has been successfully inserted in the database, a dialogue box pops up showing the message “data inserted successfully”. If the user doesn’t insert data into all the fields, a message again pops up showing the message “all fields are required”.

```
def insert():

    id_num = e_id.get()

    company = e_company.get()

    product = e_product.get()

    country = e_country.get()


    if(id_num=="" or company=="" or product=="" or country==""):

        MessageBox.showinfo("Insert status", "All fields are required")

    else:

        mydb = sql.connect(host="localhost", user="root", passwd="Samridhigupta505",
database="tracker", use_pure=True)

        cursor = mydb.cursor()

        cursor.execute("Insert into tracker values ('"+id_num+"', '"+company+"',
'"+product+"', '"+country+"')")

        cursor.execute("commit")

        MessgaeBox.showinfo("Insert Status", "Inserted Successfully")

        mydb.close()
```

3.3.2 DELETE METHOD

The second method coded is the delete() method, which helps the user delete data from the database. The company ID is a compulsory field needed for the deletion of a record. If the user dosen’t enter the company ID, a dialogue box pops up saying, “ID is compulsory for deletion”. After a record is deleted, another message pops us saying, “data deleted successfully”.

```
def delete():


    if(e_id.get()==""):
        MessageBox.showinfo("Delete status", "Company ID is compulsory for delete")
    else:
        mydb = sql.connect(host="localhost", user="root", passwd="Samridhigupta505",
database="tracker", use_pure=True)
        cursor = mydb.cursor()
        cursor.execute("delete from tracker where id='"+e_id.get() +"'")
        cursor.execute("commit")


        e_id.delete(0, 'end')
        e_company.delete(0, 'end')
        e_product.delete(0, 'end')
        MessgaeBox.showinfo("Delete Status", "Deleted Successfully")
        mydb.close()
```

3.3.3 UPDATE METHOD

The update() method is used for updating or changing any of the fields of a record in the database. When the required changes are made, a message in a dialogue box pops us saying that “data updated successfully”.

```
def update():
    id_num = e_id.get()
    company = e_company.get()
    product = e_product.get()
    country = e_country.get()
    if(id_num=="" or company=="" or product=="" or country==""):
        MessageBox.showinfo("Update status", "All fields are required")
    else:
        mydb = sql.connect(host="localhost", user="root", passwd="Samridhigupta505", database="tracker", use_pure=True)
        cursor = mydb.cursor()
        cursor.execute("Update student set id='"+id_num+"', company'"+company+"', product='"+product+"' where id='"+id+"'")
        cursor.execute("commit")
         e_id.delete(0, 'end')
        e_company.delete(0, 'end')
        e_product.delete(0, 'end')
        MessgaeBox.showinfo("Update Status", "Updated Successfully")
        mydb.close()
```

3.3.4 GET AND SHOW METHODS

The get() method helps in fetching/ retrieving data required for any of the operations and the show method tracks the activities being done on the database like inserting data, deleting data or updating data. All the operations are tracked in the list box shown in the GUI.

```
def get():
  if(e_id.get()==""):
      MessageBox.showinfo("Delete status", "Company ID is compulsory for
delete")
   else:
      mydb = sql.connect(host="localhost", user="root",
passwd="Samridhigupta505", database="tracker", use_pure=True)
      cursor = mydb.cursor()
      cursor.execute("select * from tracker where id='"+e_id.get() +"'")
      rows = cursor.fetchall()

      for row in rows:
         e_company.insert(0, row[1])
         e_product.insert(0, row[1])
         e_country.insert(0, row[1])

      mydb.close()

def show():
   mydb = sql.connect(host="localhost", user="root",
passwd="Samridhigupta505", database="tracker", use_pure=True)

   cursor = mydb.cursor()

   cursor.execute("select * from execute")

   rows = cursor.fetchall()


   for row in rows:

      insertData = str(row[0])+ '     '+row[1]

      list.insert(list.size()+1, insertData)


   mydb.close()
```

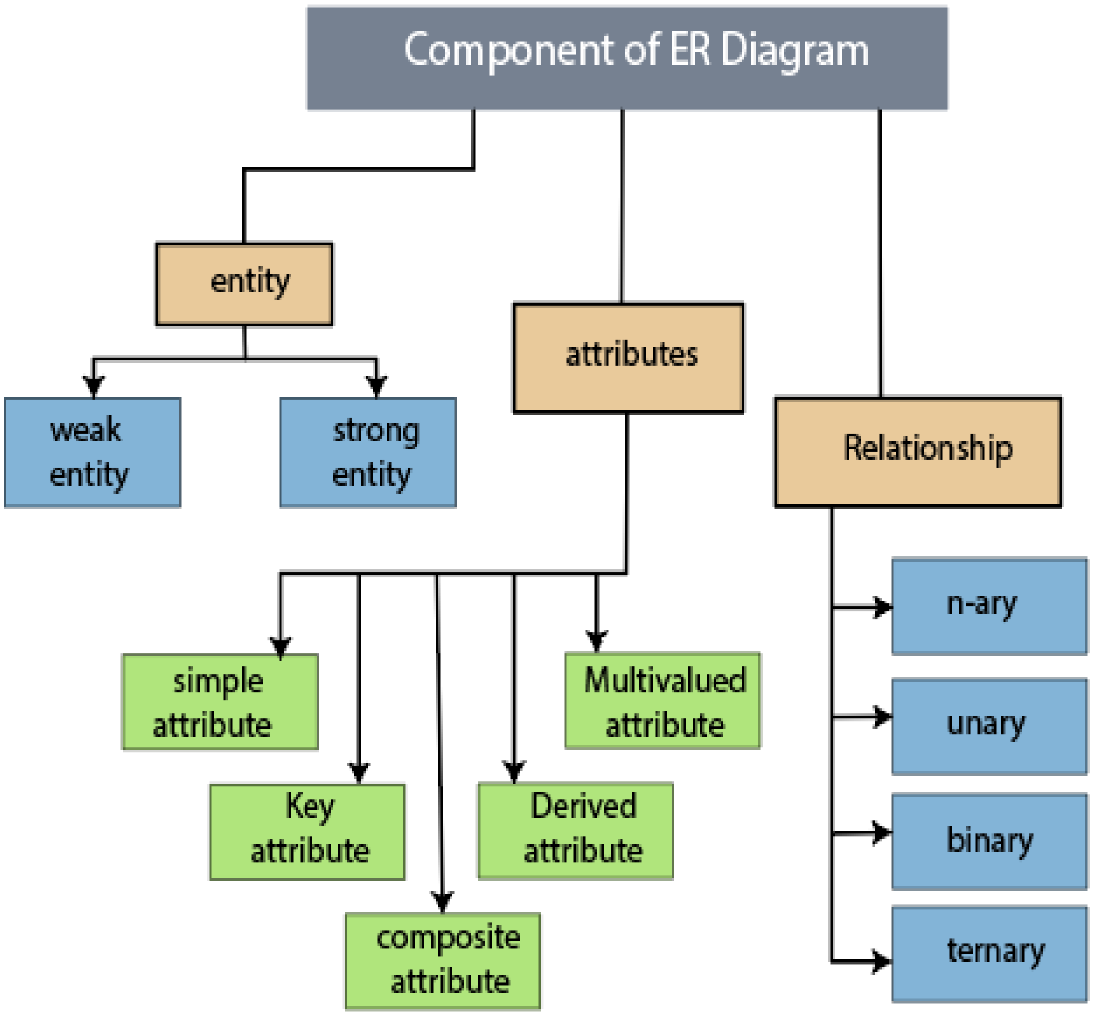
At the end of every method, it is very necessary to close the connection established with the database, so that there aren’t any external interruptions or interference.

CHAPTER 4

DATABASE DESIGN AND ER DIAGRAM

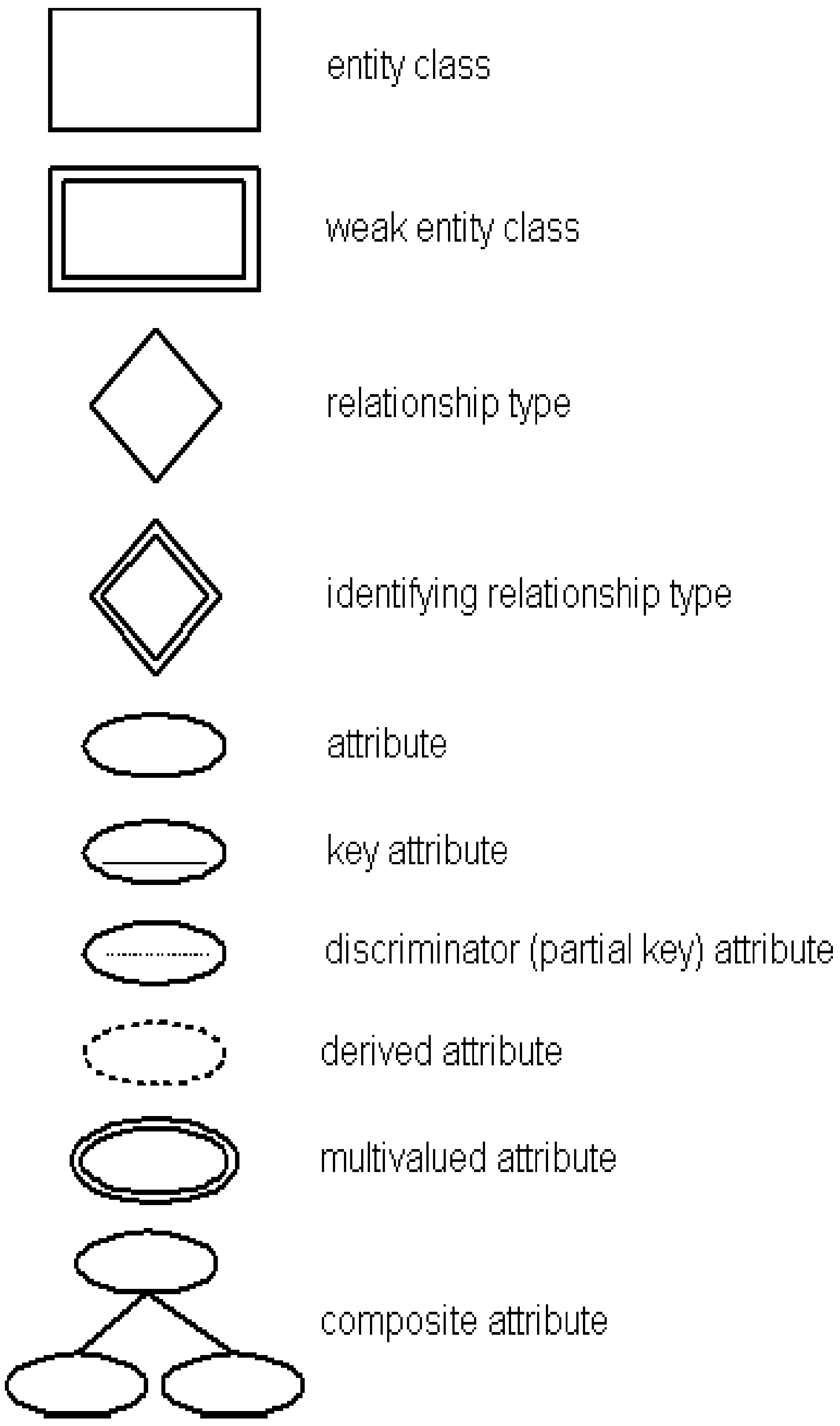
4.1 ER DIAGRAM FUNDAMENTALS

An ER diagram shows the relationship among substance sets. A substance set is a gathering of comparable elements and these elements can have credits. As far as DBMS, an element is a table or quality of a table in information base, so by indicating relationship among tables and their properties, ER graph shows the total legitimate structure of a data set.

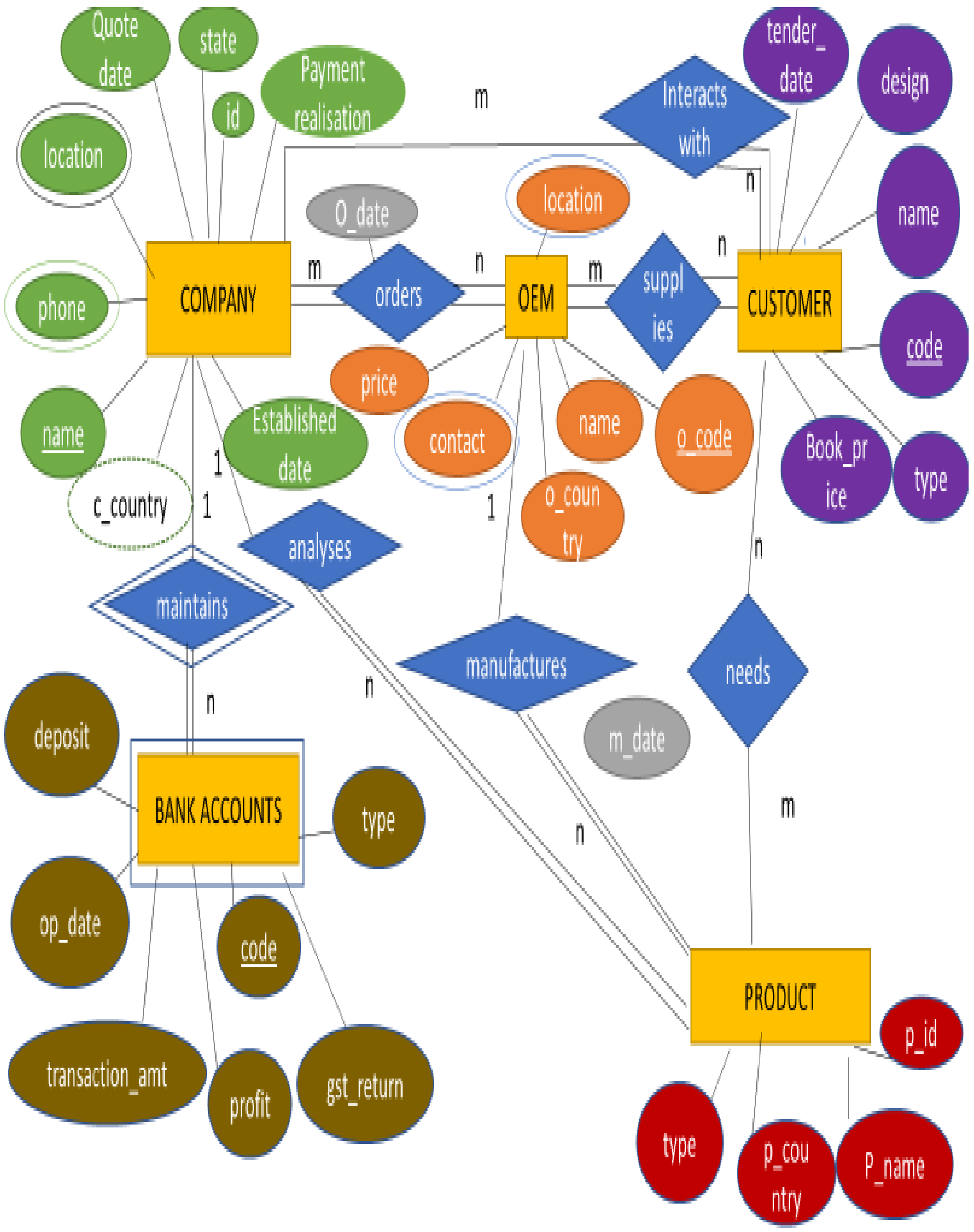


The geometric shapes listed below, explain the meaning of each shape in the ER diagram.

- **Rectangle :** shows the entities of database
- **Ellipse :** shows the attributes
- **Diamond :** shows relationship sets
- **Lines :** They link attributes to Entity Sets and Entity sets to Relationship Set
- **Double ellipse :** multivalued attributes
- **Double rectangle :** weak entities
- **Double line :** partial participation of an entity in the relationship set
- **Single line :** total participation of an entity in the relationship set
- **Dashed eclipse :** shows derived attributes


entity class
weak entity class
relationship type
identifying relationship type
attribute
key attribute
discriminator (partial key) attribute
derived attribute
multivalued attribute
composite attribute

4.2 ER DIAGRAM FOR SPEC CHURNER BUSINESS TRACKER



4.2.1 ENTITIES AND ATTRUBUTES

1. **COMPANY** :- This entity is the most prominent entity of the database. It has relationship with several other entities like the product, original equipment manufacturer and bank accounts. It's attributes are :-

- name and id : primary key
- location : multivalued attribute
- country : derived attribute
- state : country can be derived from the state attribute
- estd_date : established date
- pmnt_realisation : to check whether payment realization has been done or not
- phone : another multi valued attribute
- quote_date : to track the date when the quote for any tender was quoted

2. **OEM** :- The OEM or Original Equipment manufacturer connects with the company, product (as it manufacters the products to be supplied) and the customer. It's attributes are :-

- Price : price of the product to be given at
- name and o_code : primary keys
- country : name of the coutry to which the OEM belongs to
- location : multivalued attribute
- contact : multivalued attribute

3**. CUSTOMER** : The customer entity connects with the product and the OEM. It has an many or n:m relationship with both the OEM and product entities. It's attributes are :-

- Book_price : the price at which the customer opens the tender for any product purchase
- Code : code of the customer company
- Name : name of the customer company
- Design : specifications of the design that the customer has ordered
- Tender_date : the date at which the customer issues it's tenders
- Type : type of the organization i.e. governmental or private

4**. PRODUCT** :- The product entity connects with the OEM and the customer entities. It's attributes are :-

- Type : type of the product i.e. hardware, electronics, computer design, etc.
- Name : name of the product
- Qty : the quantity in which the product was manufactured or the number of pieces of that product
- P_country : the country from which the product has been supplied
- Code : primary key

5**. BANK ACCOUNT** : The bank account is a weak entity and is in total dependence on the company. It has the attributes :-

- Deposit : the amount deposited in the name of the company
- Op_date : opening date of the company account

- Gst_return : to check whether the GST return has been filed or not for a particular country
- Code : bank account code, it is the primary key
- Profit : the amount of profit the company has achieved
- Transaction_amount : the amount/ money withdrawn or deposited, total amount

6. **RELATIONS AND ATTRIBUTES** :

- Orders : The o_date attribute exists for the relation between the company and the original equipment manufacturer (OEM)
- Manufactures : the m_date attribute exists between the realtion between the OEM and the product entities
- Maintains : this relation exists between the company and bank account, it is a weak relation as it connects a strong entity to the weak entity
- Supplies : the OEM supplies products to the customer
- Interacts_with : the company interacts with the customer for knowing their requirements and the kind of products that they require. The company then gets them supplied from the OEM by ordering the products
- Needs : the customer needs the product
- Analyses : the company analyses the specifications of the products

CHAPTER 5

OUTPUTS

CHPATER 6

CONCLUSION

The Spec Churner Business Tracker is successfully able to perform CRUD operations on the data a user inserts into it. This data is reflected in the MySQL database created with the name of Spec Churner. The database also contains several tables been created to help track the company it's records of payment, tender payments, several dates, supplier, customer relations, relations with the original equipment manufacturer, relations with other countries and transactions with it's back accounts. The sole purpose of this project was to replicate the Graphical User Interface structure of any organization, which helps the users and makes it easy for them to store and warehouse their company data. This finds applications in the practical world and our daily usage, where we login using our credentials and save them somewhere. This data is being warehoused in a database, which is helping several organizations analyze their company's data. The Database is a powerful part of any organization, thus the project focuses more on the database queries than on the GUI application.

After the completion of this project, I have understood the usage of a database, designing a GUI, and the connection between the two in a thorough manner. I thank the college for giving me such a wonderful opportunity.

CHPATER 7

REFERENCES

LINKS

- https://www.youtube.com/watch?v=d56sf29tSzw&list=PLOgHibX8dosCc39SsNepgxzSmNEdVQrmn
- https://www.w3schools.in/mysql/ddl-dml-dcl/
- https://realpython.com/python-gui-tkinter/
- https://www.geeksforgeeks.org/jupyter-notebook-vs-python-idle/

BOOKS

- Fundamentals of database systems, 5th edition by Ramez Elmasri and Shankar B. Navathe