BACHELORS OF SCIENCE IN COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

IOST, TRIBHUVAN UNIVERSITY

Chardobato, Bhaktapur

Affiliated to Tribhuvan University

Lab Report

Advance Database

SUBMITTED BY

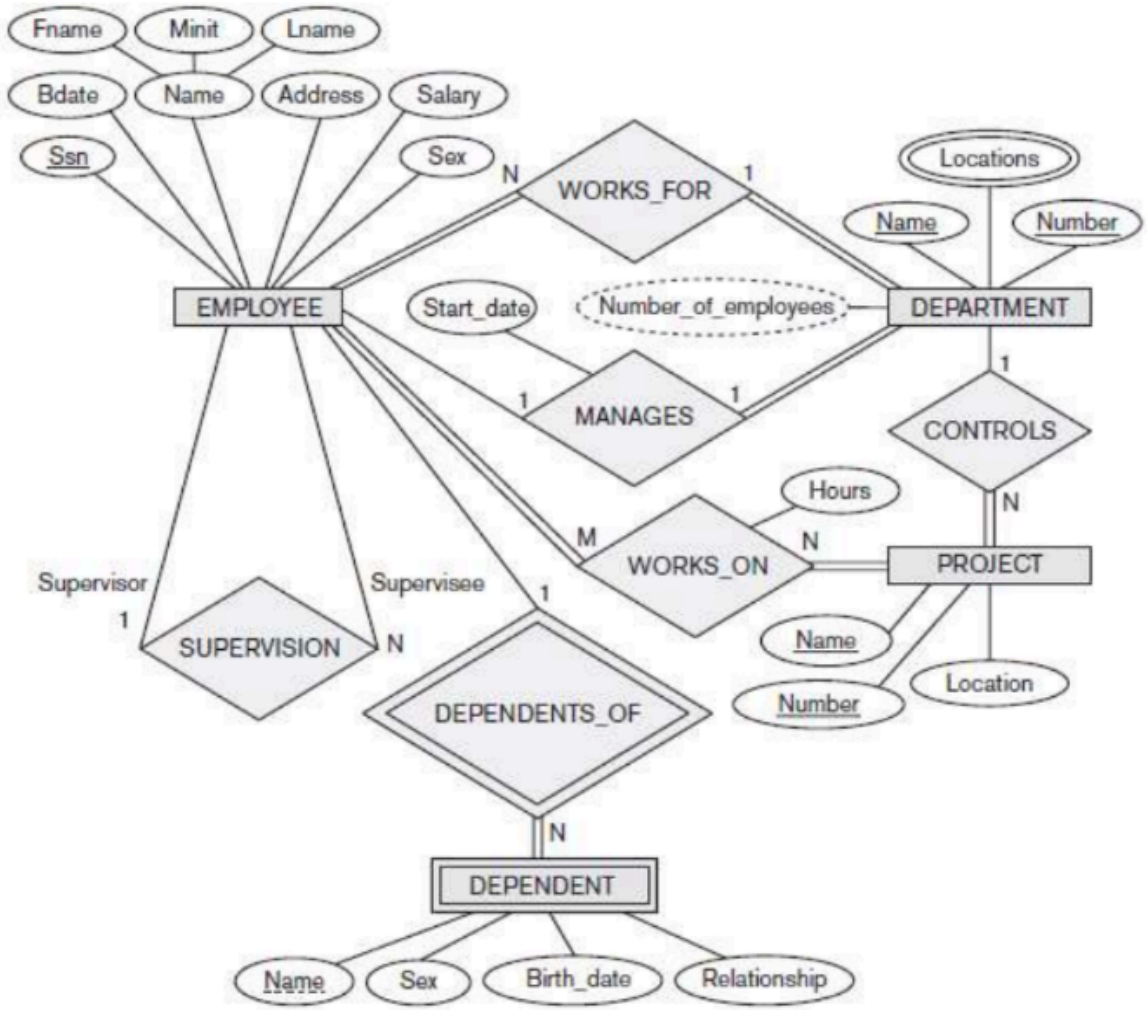
NAME: PARASH THAPA
SYMBOL NO: 24380
PROGRAM: CSIT 8th SEMESTER

SUBMITTED TO

SUMIT BIDARI

Lab-1:

CONVERTING THE GIVEN ER DIAGRAM TO RELATIONAL MODEL



SQL Queries for Each Table :

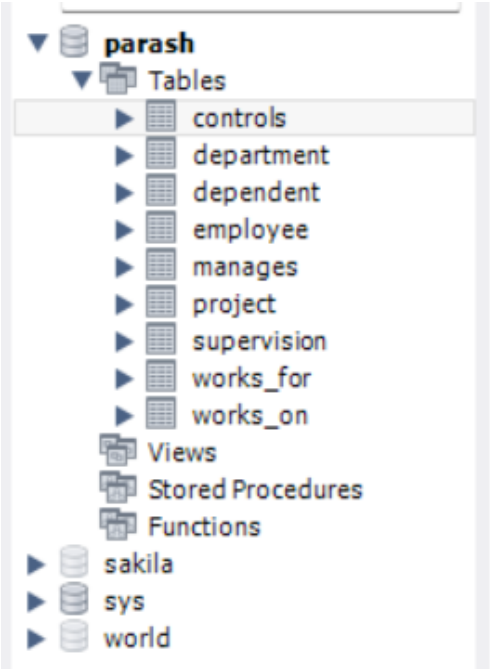
Employee :

```
CREATE TABLE DEPARTMENT (
   Number INT PRIMARY KEY,
   Name VARCHAR(255),
   Locations VARCHAR(255)
);

CREATE TABLE EMPLOYEE (
   Ssn INT PRIMARY KEY,
```

```
    Fname VARCHAR(255),
    Minit CHAR(1),
    Lname VARCHAR(255),
    Bdate DATE,
    Address VARCHAR(255),
    Sex CHAR(1),
    Salary DECIMAL(10, 2),
    Department_Number INT,
    FOREIGN KEY (Department_Number) REFERENCES DEPARTMENT(Number)
);

CREATE TABLE PROJECT (
    Number INT PRIMARY KEY,
    Name VARCHAR(255),
    Location VARCHAR(255),
    Department_Number INT,
    FOREIGN KEY (Department_Number) REFERENCES DEPARTMENT(Number)
);

CREATE TABLE WORKS_FOR (
    Employee_Ssn INT,
    Department_Number INT,
    Start_date DATE,
    PRIMARY KEY (Employee_Ssn, Department_Number),
    FOREIGN KEY (Employee_Ssn) REFERENCES EMPLOYEE(Ssn),
    FOREIGN KEY (Department_Number) REFERENCES DEPARTMENT(Number)
);

CREATE TABLE MANAGES (
    Employee_Ssn INT PRIMARY KEY,
    Department_Number INT,
    Start_date DATE,
    FOREIGN KEY (Employee_Ssn) REFERENCES EMPLOYEE(Ssn),
    FOREIGN KEY (Department_Number) REFERENCES DEPARTMENT(Number)
);

CREATE TABLE CONTROLS (
    Department_Number INT,
    Project_Number INT,
    PRIMARY KEY (Department_Number, Project_Number),
    FOREIGN KEY (Department_Number) REFERENCES DEPARTMENT(Number),
    FOREIGN KEY (Project_Number) REFERENCES PROJECT(Number)
);

CREATE TABLE WORKS_ON (
    Employee_Ssn INT,
```

```
    Project_Number INT,
    Hours DECIMAL(5, 2),
    PRIMARY KEY (Employee_Ssn, Project_Number),
    FOREIGN KEY (Employee_Ssn) REFERENCES EMPLOYEE(Ssn),
    FOREIGN KEY (Project_Number) REFERENCES PROJECT(Number)
);

CREATE TABLE DEPENDENT (
    Dependent_Ssn INT,
    Name VARCHAR(255),
    Sex CHAR(1),
    Birth_date DATE,
    Relationship VARCHAR(255),
    PRIMARY KEY (Dependent_Ssn, Name),
    FOREIGN KEY (Dependent_Ssn) REFERENCES EMPLOYEE(Ssn)
);


CREATE TABLE SUPERVISION (
    Supervisor_Ssn INT,
    Supervisee_Ssn INT,
    PRIMARY KEY (Supervisor_Ssn, Supervisee_Ssn),
    FOREIGN KEY (Supervisor_Ssn) REFERENCES EMPLOYEE(Ssn),
    FOREIGN KEY (Supervisee_Ssn) REFERENCES EMPLOYEE(Ssn)
);
```

OUTPUT :



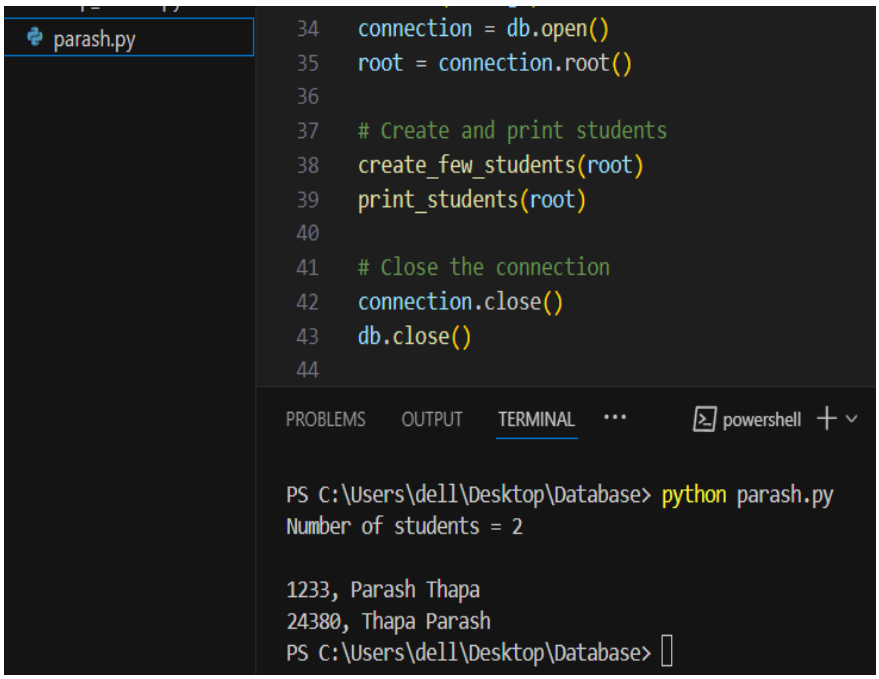
Lab-2:
Object Oriented Database Management System

SOURCE CODE:

```
from ZODB import DB, FileStorage
import transaction
from persistent import Persistent

# Define the Student class
class Student(Persistent):
    def __init__(self, sid, name):
        self.sid = sid
        self.name = name

    def __str__(self):
        return f"{self.sid}, {self.name}"

# Function to create a few students
def create_few_students(root):
    student1 = Student(1233, "Parash Thapa")
    student2 = Student(24395, "Thapa Parash")
    root['students'] = [student1, student2]
    transaction.commit()

# Function to print students
def print_students(root):
    if 'students' in root:
        students = root['students']
        print(f"Number of students = {len(students)}\n")
        for student in students:
            print(student)
    else:
        print("No students found.")

# Set up the database
storage = FileStorage.FileStorage('student.fs')
db = DB(storage)
```

```
connection = db.open()
root = connection.root()

# Create and print students
create_few_students(root)
print_students(root)

# Close the connection
connection.close()
db.close()
```

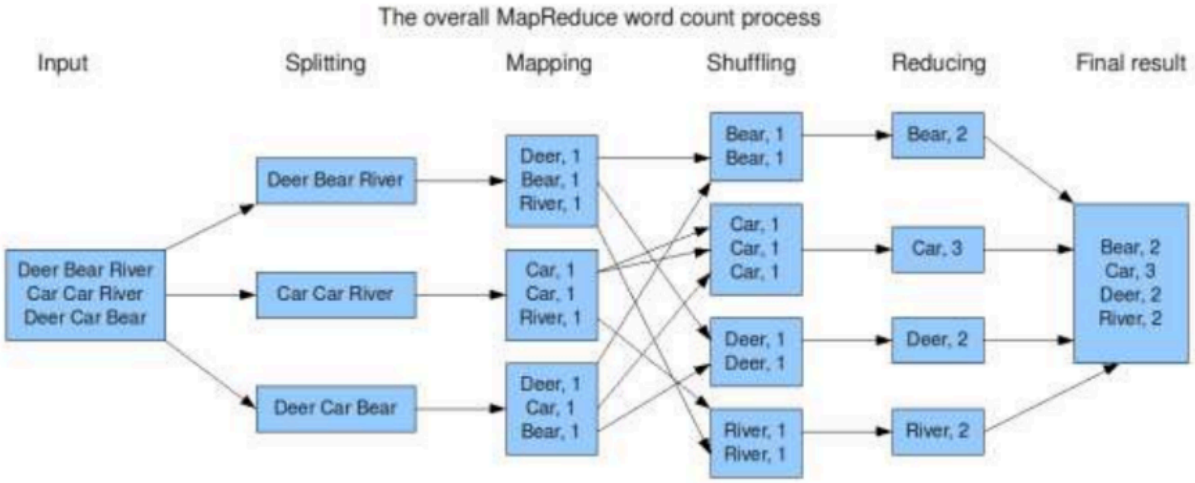
OUTPUT:



Lab-3:

MAP REDUCE

Question:



Create a class MapReduce and perform all the tasks as described above for the corpus you will be provided by the instructor.

SOURCE CODE:

```
class MapReduce:
    def __init__(self, input_content):
        self.input_content = input_content

    def split_and_map(self):
        # Split input content into lines and words
        lines = self.input_content.split("\n")
```

```
        word_pairs = []
        for line in lines:
            words = line.split()
            for word in words:
                word_pairs.append((word, 1))
        return word_pairs

    def shuffle_and_reduce(self, word_pairs):
        # Sort pairs by word (key)
        word_pairs.sort(key=lambda x: x[0])

        # Reduce: combine counts for each word
        result = {}
        current_word = None
        current_count = 0
        for word, count in word_pairs:
            if word == current_word:
                current_count += count
            else:
                if current_word:
                    result[current_word] = current_count
                current_word = word
                current_count = count
        if current_word:
            result[current_word] = current_count

        return result

if __name__ == "__main__":
    input_content = """Deer Bear River
Car Car River
Deer Car Bear"""

    mr = MapReduce(input_content)
    word_pairs = mr.split_and_map()
    word_counts = mr.shuffle_and_reduce(word_pairs)

    # Print the final word counts
    for word, count in word_counts.items():
        print(f"{word}: {count}")
    print("Interpreted by Parash Thapa 24380")
```

OUTPUT:

map_reduce.py

```python
mr = MapReduce(input_content)
word_pairs = mr.split_and_map()
word_counts = mr.shuffle_and_reduce(word_pairs)

# Print the final word counts
for word, count in word_counts.items():
    print(f"{word}: {count}")
print("Interpreted by Parash Thapa 24380")
```

PROBLEMS OUTPUT TERMINAL ··· powershell

```
PS C:\Users\dell\Desktop\Database> python map_reduce.py
Bear: 2
Car: 3
Deer: 2
River: 2
Interpreted by Parash Thapa 24380
PS C:\Users\dell\Desktop\Database>
```

Lab-4:

ACTIVE DATABASE CONCEPT

Question:

Assume the following tables
Product (BarCode, PName, Price, QuantityInStock)
Sale (SaleID, DeliveryAddress, CreditCard)
SaleItem (SaleID, BarCode, Quantity)

Create a trigger called updateAvailableQuantity that updates the quantity in stock in the Product table, for every product sold. The trigger should be executed after each insert operation on the SaleItem table: for the product with the given barcode (the one inserted into SaleItem), update

QUERY:

```
-- Create Product table
CREATE TABLE Product (
   BarCode INT PRIMARY KEY,
   PName VARCHAR(100),
   Price DECIMAL(10, 2),
   QuantityInStock INT
);

-- Create Sale table
CREATE TABLE Sale (
   SaleID INT PRIMARY KEY,
   DeliveryAddress VARCHAR(255),
   CreditCard VARCHAR(16)
);

-- Create SaleItem table
CREATE TABLE SaleItem (
   SaleID INT,
   BarCode INT,
   Quantity INT,
   FOREIGN KEY (SaleID) REFERENCES Sale(SaleID),
   FOREIGN KEY (BarCode) REFERENCES Product(BarCode)
);

INSERT INTO Product (BarCode, PName, Price, QuantityInStock) VALUES
(1001, 'Product A', 10.99, 50),
(1002, 'Product B', 15.49, 30),
(1003, 'Product C', 20.00, 20);
```

```
-- Insert sample data into Sale table
INSERT INTO Sale (SaleID, DeliveryAddress, CreditCard) VALUES
(1, '123 Main St, Anytown, USA', '1111222233334444'),
(2, '456 Elm St, Othertown, USA', '5555666677778888'),
(3, '789 Oak St, Thistown, USA', '9999000011112222');

-- Insert sample data into SaleItem table
INSERT INTO SaleItem (SaleID, BarCode, Quantity) VALUES
(1, 1001, 2),
(2, 1002, 1),
(3, 1003, 5);
-- Create the trigger
CREATE TRIGGER updateAvailableQuantity
AFTER INSERT ON SaleItem
FOR EACH ROW
   UPDATE Product
   SET QuantityInStock = QuantityInStock - NEW.Quantity
   WHERE BarCode = NEW.BarCode;

select * from saleitem;
select * from product;


INSERT INTO SaleItem (SaleID, BarCode, Quantity) VALUES
(1, 1001, 3);

select * from saleitem;
select * from product;
```

OUTPUT:

Before Update:

Result Grid | Filter Rows:

SaleID	BarCode	Quantity
1	1001	2
2	1002	1
3	1003	5

Result Grid | Filter Rows: | Edit:

BarCode	PName	Price	QuantityInStock
1001	Product A	10.99	50
1002	Product B	15.49	30
1003	Product C	20.00	20
NULL	NULL	NULL	NULL

After Update Trigger:

Result Grid | Filter Rows:

SaleID	BarCode	Quantity
1	1001	2
2	1002	1
3	1003	5
1	1001	3

Result Grid | Filter Rows: | Edit:

BarCode	PName	Price	QuantityInStock
1001	Product A	10.99	47
1002	Product B	15.49	30
1003	Product C	20.00	20
NULL	NULL	NULL	NULL

Lab-5:

DEDUCTIVE DATABASE CONCEPT

Question:

Given the following premises find the goal using Prolog

Premises

- All over-smart are stupid.
- Children of stupid are naughty.
- Hari is over smart.
- Ram is children of Hari.

Goal

- Is Ram naughty?

SOURCE CODE:

```
# Facts
oversmart_people = ["hari"]

# Stupid individuals are oversmart
def is_stupid(person):
    return person in oversmart_people

# Naughty individuals are stupid
def is_naughty(person):
    return is_stupid(person)

# Rules
child_of = {"ram": "hari"}

# Query
def is_ram_naughty():
    ram = "ram"
    return is_naughty(ram)

# Check if Ram is naughty
if is_ram_naughty():
    print("Ram is naughty!")
else:
    print("Ram is not naughty.")

print("Interpreted / Executed by Parash Thapa  24380")
```

OUTPUT:

parash.py

```python
    ram = "ram"
    return is_naughty(ram)

# Check if Ram is naughty
if is_ram_naughty():
    print("Ram is naughty!")
else:
    print("Ram is not naughty.")

print("Interpreted / Executed by Parash Thapa 24380")
```

PROBLEMS OUTPUT TERMINAL ··· powershell

```
PS C:\Users\dell\Desktop\Database> python parash.py
Ram is not naughty.
Interpreted / Executed by Parash Thapa 24380
PS C:\Users\dell\Desktop\Database>
```