Database Final Project

Requirements

This project has two components:

Written Report Instructions

In this component, you are going to look around to find the business you are interested in. It might be a hardware store in the corner, an ice cream shop, a street vendor, an online discussion board, anything you want to record information and play with. You are going to follow exactly the same steps we did in Module 4:

1. Think carefully about the business, describe a clear background story of the business.
2. Make sufficient assumptions, and create the ER model.
3. From the ER model, create the ERD.
4. Convert the ERD to a relational model,
5. Normalize the relational model to 3NF.
6. Finalize the relational model in 3NF for further implementation.

You should show the clear step-by-step development of your relational model, which demonstrates your understanding of designing relational databases, and your capability of doing it.

WRITTEN REPORT

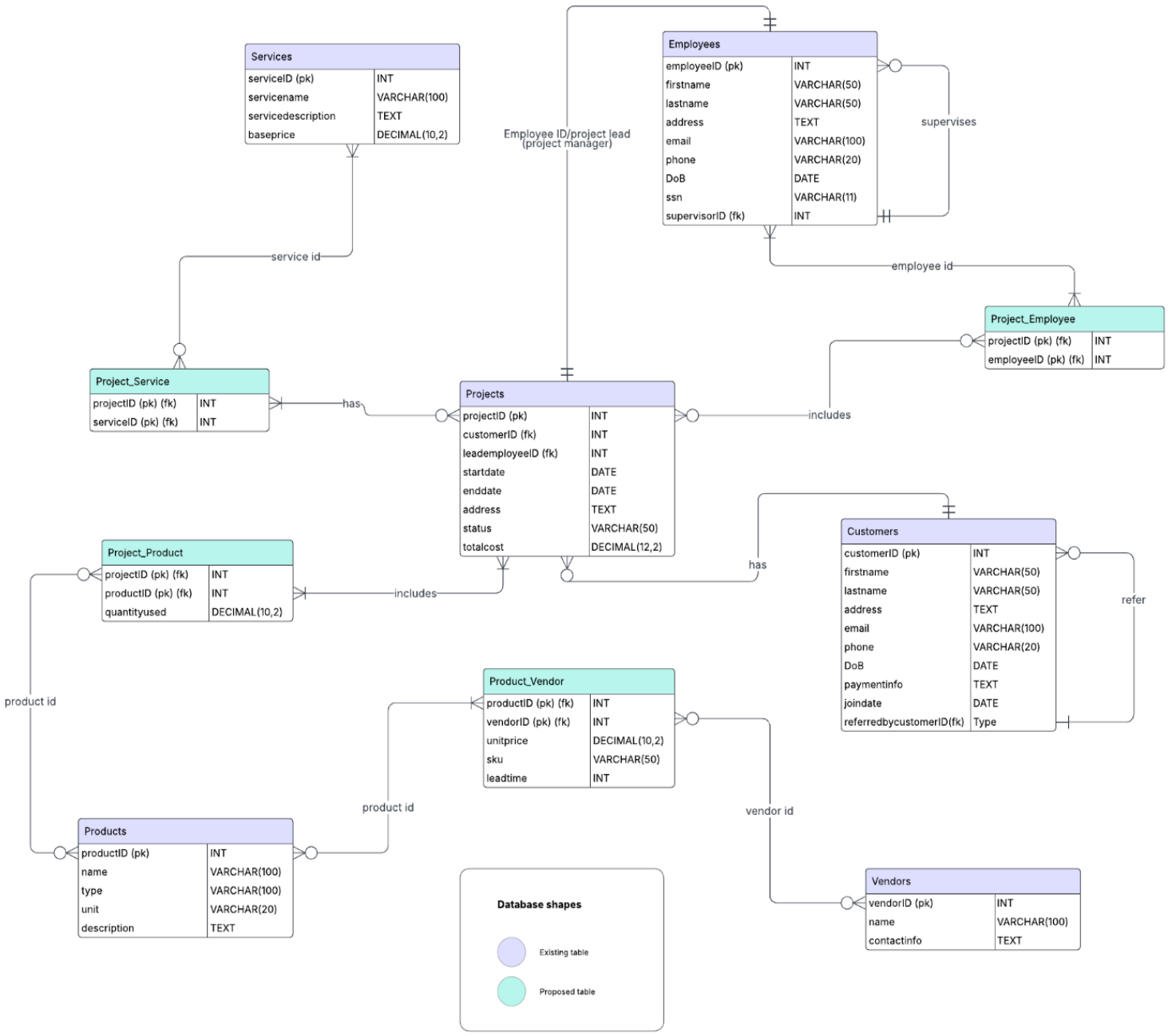
1. **Business:**
   - Home improvement business. They offer high-quality renovations, repairs, and upgrades to private homes. This business may involve multiple services, projects staffed by employees from various trades, and they have a customer referral program. They offer kitchen and bathroom remodeling, interior and exterior painting, flooring installations, roofing and siding repairs, window and door replacement, deck and patio construction, drywall installation and repair, plumbing and electrical updates, basement finishing and waterproofing, and general handyman services

2. **Entities model** (bold identifiers)**:**
   - Customers(**customerID**, firstname, lastname, address, email, phone, DoB, paymentinfo, joindate, referredbycustid)
   - Projects(**projectID**, customerID, startdate, enddate, address, status, totalcost, leademployeeID)
   - Employees(**employeeID**, firstname, lastname, ssn, address, email, phone, DoB, supervisorID)
   - Services(**serviceID**, servicename, serviceprice, servicedescription,)
   - Products(**productID**, totalprice, unit, descirption)
   - Vendors(**vendorID**, contactinfo)

   Relationships:

   - **Customers** may have 0+projects. May refer 0+ customers.
   - **Projects** belong to only 1 customer. May involve 0+ employees. Involves 1+ service. Is managed by only 1 employee. Contains 0+ products.
   - **Employees** may work on 0+ projects. Has only 1 supervisor. May supervise 0+ employees.
   - **Services** belong to 0+ projects.
   - **Products** required by 0+ projects. Can have 1+ vendor.
   - **Vendors** supplies 0+ products

3. ERD:


Services
serviceID (pk) INT
servicename VARCHAR(100)
servicedescription TEXT
baseprice DECIMAL(10,2)
Employees
employeeID (pk) INT
firstname VARCHAR(50)
lastname VARCHAR(50)
address TEXT
email VARCHAR(100)
phone VARCHAR(20)
DoB DATE
ssn VARCHAR(11)
supervisorID (fk) INT
supervises
Employee ID/project lead (project manager)
service id
employee id
Project_Employee
projectID (pk) (fk) INT
employeeID (pk) (fk) INT
Project_Service
projectID (pk) (fk) INT
serviceID (pk) (fk) INT
has
Projects
projectID (pk) INT
customerID (fk) INT
leademployeeID (fk) INT
startdate DATE
enddate DATE
address TEXT
status VARCHAR(50)
totalcost DECIMAL(12,2)
includes
Customers
customerID (pk) INT
firstname VARCHAR(50)
lastname VARCHAR(50)
address TEXT
email VARCHAR(100)
phone VARCHAR(20)
DoB DATE
paymentinfo TEXT
joindate DATE
referredbycustomerID(fk) Type
refer
has
Project_Product
projectID (pk) (fk) INT
productID (pk) (fk) INT
quantityused DECIMAL(10,2)
includes
product id
Product_Vendor
productID (pk) (fk) INT
vendorID (pk) (fk) INT
unitprice DECIMAL(10,2)
sku VARCHAR(50)
leadtime INT
product id
vendor id
Products
productID (pk) INT
name VARCHAR(100)
type VARCHAR(100)
unit VARCHAR(20)
description TEXT
Vendors
vendorID (pk) INT
name VARCHAR(100)
contactinfo TEXT
Database shapes
Existing table
Proposed table

4. **Relational Model**
   - Customers(customerID, firstname, lastname, address, email, phone, DoB, paymentinfo, joindate, referredbycustomerID(fk))
   - Projects(projectID, customerID(fk), leademployeeID, startdate, enddate, address, status, totalcost)
   - Employees(employeeID, firstname, lastname, ssn, address, email, phone, DoB, supervisorID(fk))
   - Services(serviceID, servicename, servicedescription, baseprice)
   - Products(productID , name, type, unit, description)
   - Vendors(vendorID, name, contactinfo)
   - Product_Vendor(productID(fk), VendorID(fk), unitprice, sku, leadtime)
   - Project_Employee(projectID(fk), employeeID(fk))
   - Project_Service(projectID(fk), ServiceID(fk))
   - Project_product(projectID(fk), productID(fk), quantityused)

5. Normalization

Customers: all atomic, non-composite keys therefore no partial dependencies, no transitive dependencies. This is 3NF.
Employees: all atomic, no partial dependencies, no transitive dependencies. This is 3NF.
Projects: all atomic, non-composite keys therefore no partial dependencies, no transitive dependencies. This is 3NF.
Services: all atomic, non-composite keys therefore no partial dependencies, no transitive dependencies. This is 3NF.
Products: all atomic, non-composite keys therefore no partial dependencies, no transitive dependencies. This is 3NF.
Vendors: all atomic, non-composite keys therefore no partial dependencies, no transitive dependencies. This is 3NF.
Product_Vendors: all atomic, composite depend on full combination therefore no partial dependencies, no transitive dependencies. This is 3NF.
Project_employees: all atomic, composite depend on full combination therefore no partial dependencies, no transitive dependencies. This is 3NF.
Project_Service: all atomic, composite depend on full combination therefore no partial dependencies, no transitive dependencies. This is 3NF.
Project_Product: all atomic, composite depend on full combination therefore no partial dependencies, no transitive dependencies. This is 3NF.

6. Finalization:

a. Customers(customerID, firstname, lastname, address, email, phone, DoB, paymentinfo, joindate, referredbycustomerID(fk))
   FD: customerID → all other attributes
b. Projects(projectID, customerID(fk), leademployeeID, startdate, enddate, address, status, totalcost)
   FD: projectID → all other attributes
c. Employees(employeeID, firstname, lastname, ssn, address, email, phone, DoB, supervisorID(fk))
   FD: employeeID → all other attributes
d. Services(serviceID, servicename, servicedescription, baseprice)
   FD: serviceID → all other attributes
e. Products(productID , name, type, unit, description)
   FD: productID → all other attributes
f. Vendors(vendorID, name, contactinfo)
   FD: CustomerID → all other attributes
g. Product_Vendor(productID(fk), VendorID(fk), unitprice, sku, leadtime)
   FD: Full Composite PK→ all other attributes
h. Project_Employee(projectID(fk), employeeID(fk))
   FD: Composite PK, no non-key attributes
i. Project_Service(projectID(fk), ServiceID(fk))
   FD: Composite PK, no non-key attributes
j. Project_product(projectID(fk), productID(fk), quantityused)
   FD: Full Composite PK→ all other attributes

Below is the finalized SQL schema implementation

```
-- Customers Table
CREATE TABLE Customers (
customerID INT PRIMARY KEY,
firstname VARCHAR(50),
lastname VARCHAR(50),
address TEXT,
email VARCHAR(100),
phone VARCHAR(20),
dob DATE,
paymentinfo TEXT,
joindate DATE,
referredbycustomerID INT,
FOREIGN KEY (referredbycustomerID) REFERENCES Customers(customerID)
);


-- Employees Table
CREATE TABLE Employees (
employeeID INT PRIMARY KEY,
firstname VARCHAR(50),
lastname VARCHAR(50),
ssn VARCHAR(11),
address TEXT,
email VARCHAR(100),
phone VARCHAR(20),
dob DATE,
supervisorID INT,
FOREIGN KEY (supervisorID) REFERENCES Employees(employeeID)
);
```

```
-- Projects Table
CREATE TABLE Projects (
projectID INT PRIMARY KEY,
customerID INT,
leademployeeID INT,
startdate DATE,
enddate DATE,
address TEXT,
status VARCHAR(50),
totalcost DECIMAL(12,2),
FOREIGN KEY (customerID) REFERENCES Customers(customerID),
FOREIGN KEY (leademployeeID) REFERENCES Employees(employeeID)
);

-- Services Table
CREATE TABLE Services (
serviceID INT PRIMARY KEY,
servicename VARCHAR(100),
servicedescription TEXT,
baseprice DECIMAL(10,2)
);

-- Products Table
CREATE TABLE Products (
productID INT PRIMARY KEY,
name VARCHAR(100),
type VARCHAR(50),
unit VARCHAR(20),
description TEXT
);
```

```
-- Vendors Table
CREATE TABLE Vendors (
vendorID INT PRIMARY KEY,
name VARCHAR(100),
contactinfo TEXT
);


-- Product_Vendor Table
CREATE TABLE Product_Vendor (
productID INT,
vendorID INT,
unitprice DECIMAL(10,2),
sku VARCHAR(50),
leadtime INT,
PRIMARY KEY (productID, vendorID),
FOREIGN KEY (productID) REFERENCES Products(productID),
FOREIGN KEY (vendorID) REFERENCES Vendors(vendorID)
);


-- Project_Employee Table
CREATE TABLE Project_Employee (
projectID INT,
employeeID INT,
PRIMARY KEY (projectID, employeeID),
FOREIGN KEY (projectID) REFERENCES Projects(projectID),
FOREIGN KEY (employeeID) REFERENCES Employees(employeeID)
);
```

```
-- Project_Service Table
CREATE TABLE Project_Service (
projectID INT,
serviceID INT,
PRIMARY KEY (projectID, serviceID),
FOREIGN KEY (projectID) REFERENCES Projects(projectID),
FOREIGN KEY (serviceID) REFERENCES Services(serviceID)
);

-- Project_Product Table
CREATE TABLE Project_Product (
projectID INT,
productID INT,
quantityused DECIMAL(10,2),
PRIMARY KEY (projectID, productID),
FOREIGN KEY (projectID) REFERENCES Projects(projectID),
FOREIGN KEY (productID) REFERENCES Products(productID)
);
```