DBMS LAB 08 TASKS

Prepared by:
Mohammad Anas Jawad
Lecturer, IUT CSE

Department of Computer Science and Engineering
Islamic University of Technology
July 31, 2021

1 A FEW POINTS ABOUT VIEWS

I mentioned in my lecture video that views cannot be updated and they cannot be used to modify the parent table they are generated from. This statement is true only for complex views. In case you are working with simple views, they can be used to update/modify the parent table. That is, any change you perform in your simple view, will be reflected in your parent table.

For understanding the difference between simple and complex views, check out the following:

SIMPLE VIEW	COMPLEX VIEW
Contains only one single base table or is created from only one table.	Contains more than one base tables or is created from more than one tables.
We cannot use group functions like MAX(), COUNT(), etc.	We can use group functions.
Does not contain groups of data.	It can contain groups of data.
DML operations could be performed through a simple view.	DML operations could not always be performed through a complex view.
INSERT, DELETE and UPDATE are directly possible on a simple view.	We cannot apply INSERT, DELETE and UPDATE on complex view directly.
Simple view does not contain group by, distinct, pseudocolumn like rownum, columns defiend by expressions.	It can contain group by, distinct, pseudocolumn like rownum, columns defiend by expressions.
Does not include NOT NULL columns from base tables.	NOT NULL columns that are not selected by simple view can be included in complex view.

Figure 1: Difference between simple and complex views [Source: GeeksForGeeks]

2 SOME FAQS

1. How can I delete a table if it is used by other tables in referential integrity constraints?

   ANS: `DROP TABLE table_name CASCADE CONSTRAINTS;`

2. What type of privileges can I grant to roles and users?

   ANS: Visit the following link to get a good idea about the types of privileges you can grant to roles and users.

   GRANT Statement - Oracle Documentation

3. How do I reference a single attribute as a foreign key in Table B if it is part of a composite primary key in Table A?

   ANS: : You can't. You have to use all the attributes used in the composite primary key of A to reference A. Example:

   ```
   CONSTRAINT EXAMPLE_FK1 FOREIGN KEY(attr_1,attr_2) REFERENCES A(attr_1,attr_2)
   ```

4. How do I know if I should use ON DELETE CASCADE for a particular foreign key or not?

   ANS: Unless mentioned in the scenario, you'll mostly have to rely on your common sense. However, there are some cases where it's difficult to understand whether you should or not; don't worry about them too much.

5. Will this solution sheet solve all my problems?

   ANS: No, but if used properly, they will help you in your never-ending quest of mastering the arts of DBMS and SQL operations. :)

SCENARIO 1

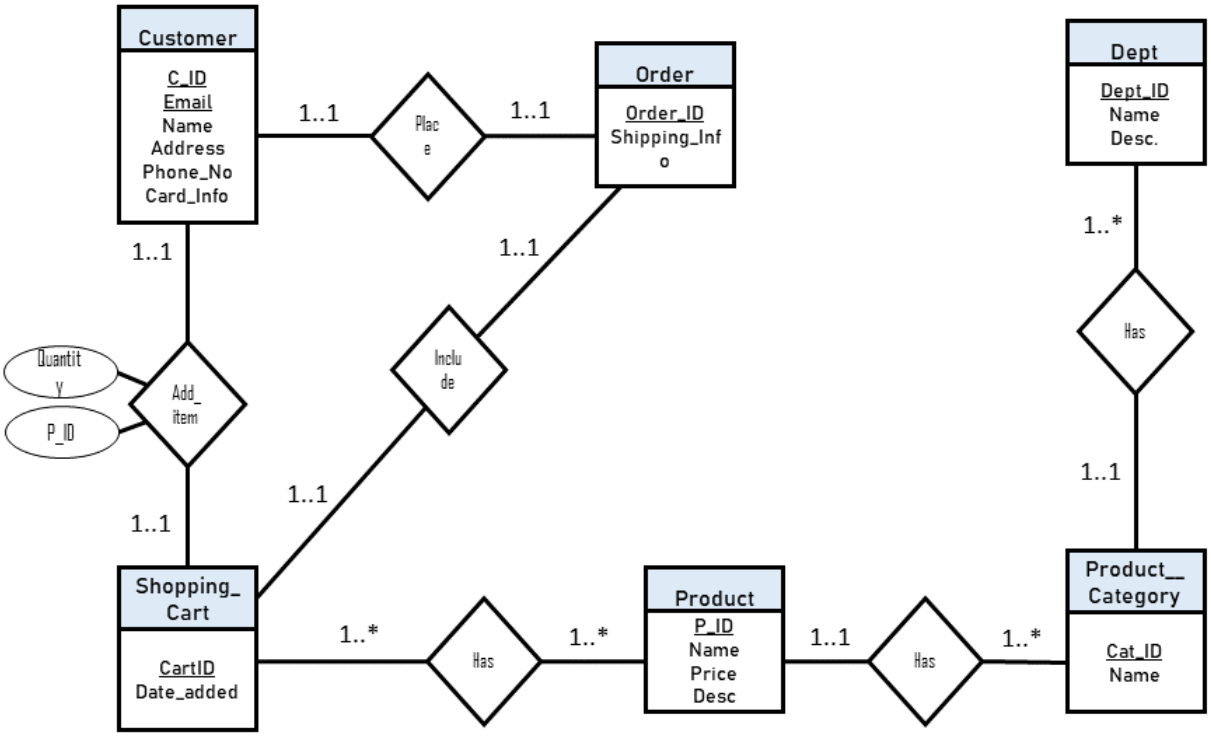

Figure 2: Partial ERD of a Shopping Cart Project

1. After determining the type of relationship that exist between the entities (i.e. one-to-one, one-to-many or many-to-many), create all the tables that can be found from this ERD.

   Use the cascading delete clause for all the foreign keys in your tables. If there's any foreign key that should not have cascading delete, write down a comment explaining your reasoning.

   [Hint: There will be at least 7 tables in total.]

2. Create a view named Shopping_Cart_Details that gives information about all the prod-

ucts included in the shopping cart. This view should have the attributes Cart ID, Date added, Product ID, Name and Price of the product.
From this view, create another view named Product_Purchase_Total_1 that contains the date the shopping carts have been added on and the total number of times a certain product has been purchased on that day.

3. Create a view named Customer_Purchase_History that includes all the information of the customer, the product ID and the quantity of the purchased product.

4. Three categories of users have been identified so far for this database:

    - A customer will be able to view his information from the database.
    - A salesperson will be able to modify the shopping cart and order details of the customer.
    - The manager will be able to update the customer information, modify department, product category and product information and obviously view all this information.

    Now, create three roles for these three categories of users and grant them the appropriate privileges.

Solution

```
--Task 1

CREATE TABLE CUSTOMER
(
C_ID INT,
EMAIL VARCHAR2(30),
NAME VARCHAR2(20),
ADDRESS VARCHAR2(20),
PHONE_NO INT,
CARD_INFO INT,
CONSTRAINT CUSTOMER_PK PRIMARY KEY (C_ID,EMAIL)
);

CREATE TABLE SHOPPING_CART
(
CART_ID VARCHAR2(20) PRIMARY KEY,
DATE_ADDED DATE,
QUANTITY INT,
P_ID INT,
C_ID INT,
EMAIL VARCHAR2(30),
CONSTRAINT CART_FK_1 FOREIGN KEY(C_ID, EMAIL) REFERENCES CUSTOMER(C_ID,
    EMAIL) ON DELETE CASCADE,
CONSTRAINT CART_FK_2 FOREIGN KEY(P_ID) REFERENCES PRODUCT(P_ID)
--no cascade for fk2 because removing a product shouldn't delete the
   whole shopping cart entry
);

CREATE TABLE ORDERS
(
ORDER_ID VARCHAR2(20) PRIMARY KEY,
SHIPPING_INFO VARCHAR2(20),
C_ID INT,
```

```
EMAIL VARCHAR2(30),
CART_ID VARCHAR2(20),
CONSTRAINT ORDER_FK_1 FOREIGN KEY(C_ID, EMAIL) REFERENCES CUSTOMER(C_ID, EMAIL) ON DELETE CASCADE,
CONSTRAINT ORDER_FK_2 FOREIGN KEY(CART_ID) REFERENCES SHOPPING_CART(CART_ID) ON DELETE CASCADE
);

CREATE TABLE PRODUCT
(
P_ID INT PRIMARY KEY,
NAME VARCHAR2(20),
PRICE NUMBER(5,2),
CAT_ID VARCHAR2(20),
CONSTRAINT PRODUCT_FK FOREIGN KEY(CAT_ID) REFERENCES PRODUCT_CATEGORY(CAT_ID)
--no cascading here as deleting a product category shouldn't remove the product
);

CREATE TABLE PRODUCT_CATEGORY
(
CAT_ID VARCHAR2(20) PRIMARY KEY,
NAME VARCHAR2(20),
DEPT_ID VARCHAR2(20),
CONSTRAINT PRODUCT_CATEGORY_FK FOREIGN KEY(DEPT_ID) REFERENCES DEPT(DEPT_ID)
--no cascading here as deleting a dept shouldn't remove the entire product category
);
CREATE TABLE DEPT
(
DEPT_ID VARCHAR2(20) PRIMARY KEY,
NAME VARCHAR2(20),
DESCr VARCHAR2(30)
```

```
);

CREATE TABLE SHOPPING_CART_HAS_PRODUCT
(
CART_ID VARCHAR2(20),
P_ID INT,
CONSTRAINT SHOPPING_CART_HAS_PRODUCT_PK PRIMARY KEY(CART_ID, P_ID),
CONSTRAINT SHOPPING_CART_HAS_PRODUCT_FK_1 FOREIGN KEY(CART_ID)
    REFERENCES SHOPPING_CART(CART_ID) ON DELETE CASCADE,
CONSTRAINT SHOPPING_CART_HAS_PRODUCT_FK_2 FOREIGN KEY(P_ID) REFERENCES
    PRODUCT(P_ID)
--no cascade on the P_ID foreign key as removing a product shouldn't
    delete the whole entry from the shopping_cart_has_product table
);

--Task 2


CREATE OR REPLACE VIEW SHOPPING_CART_DETAILS AS
SELECT SHOPPING_CART_HAS_PRODUCT.CART_ID, SHOPPING_CART.DATE_ADDED,
    SHOPPING_CART_HAS_PRODUCT.P_ID, PRODUCT.NAME, PRODUCT.PRICE FROM
    SHOPPING_CART_HAS_PRODUCT, SHOPPING_CART, PRODUCT
WHERE SHOPPING_CART_HAS_PRODUCT.P_ID = PRODUCT.P_ID
AND SHOPPING_CART_HAS_PRODUCT.CART_ID = SHOPPING_CART.CART_ID;

CREATE OR REPLACE VIEW PRODUCT_PURCHASE_TOTAL_1 AS
SELECT DATE_ADDED,P_ID, COUNT(*) AS QUANTITY FROM SHOPPING_CART_DETAILS
GROUP BY DATE_ADDED, P_ID;

CREATE OR REPLACE VIEW PRODUCT_PURCHASE_TOTAL_2 AS
SELECT DATE_ADDED, COUNT(*) AS TOT_PURCHASED FROM SHOPPING_CART_DETAILS
GROUP BY DATE_ADDED;


--Task 3
```

```
91
92
93  CREATE OR REPLACE VIEW CUSTOMER_PURCHASE_HISTORY AS
94  SELECT CUSTOMER.C_ID, CUSTOMER.EMAIL, CUSTOMER.NAME, CUSTOMER.ADDRESS,
        CUSTOMER.PHONE_NO, CUSTOMER.CARD_INFO, SHOPPING_CART.P_ID,
        SHOPPING_CART.QUANTITY FROM CUSTOMER, SHOPPING_CART
95  WHERE CUSTOMER.C_ID = SHOPPING_CART.C_ID;
96
97  --Task 4
98
99  CREATE ROLE ROLE_CUSTOMER;
100 CREATE ROLE ROLE_SALESPERSON;
101 CREATE ROLE ROLE_MANAGER;
102
103 GRANT SELECT ON CUSTOMER TO ROLE_CUSTOMER;
104
105 GRANT SELECT ON SHOPPING_CART TO ROLE_SALESPERSON;
106 GRANT INSERT ON SHOPPING_CART TO ROLE_SALESPERSON;
107 GRANT UPDATE ON SHOPPING_CART TO ROLE_SALESPERSON;
108 GRANT DELETE ON SHOPPING_CART TO ROLE_SALESPERSON;
109 GRANT SELECT ON ORDERS TO ROLE_SALESPERSON;
110 GRANT INSERT ON ORDERS TO ROLE_SALESPERSON;
111 GRANT UPDATE ON ORDERS TO ROLE_SALESPERSON;
112 GRANT DELETE ON ORDERS TO ROLE_SALESPERSON;
113
114 GRANT ROLE_CUSTOMER TO ROLE_MANAGER;
115 GRANT INSERT ON CUSTOMER TO ROLE_MANAGER;
116 GRANT UPDATE ON CUSTOMER TO ROLE_MANAGER;
117 GRANT DELETE ON CUSTOMER TO ROLE_MANAGER;
118
119 GRANT SELECT ON DEPT TO ROLE_MANAGER
120 GRANT INSERT ON DEPT TO ROLE_MANAGER;
121 GRANT UPDATE ON DEPT TO ROLE_MANAGER;
122 GRANT DELETE ON DEPT TO ROLE_MANAGER;
123
```

```
124 GRANT SELECT ON PRODUCT_CATEGORY TO ROLE_MANAGER
125 GRANT INSERT ON PRODUCT_CATEGORY TO ROLE_MANAGER;
126 GRANT UPDATE ON PRODUCT_CATEGORY TO ROLE_MANAGER;
127 GRANT DELETE ON PRODUCT_CATEGORY TO ROLE_MANAGER;
128
129 GRANT SELECT ON PRODUCT TO ROLE_MANAGER
130 GRANT INSERT ON PRODUCT TO ROLE_MANAGER;
131 GRANT UPDATE ON PRODUCT TO ROLE_MANAGER;
132 GRANT DELETE ON PRODUCT TO ROLE_MANAGER;
```

SCENARIO 2

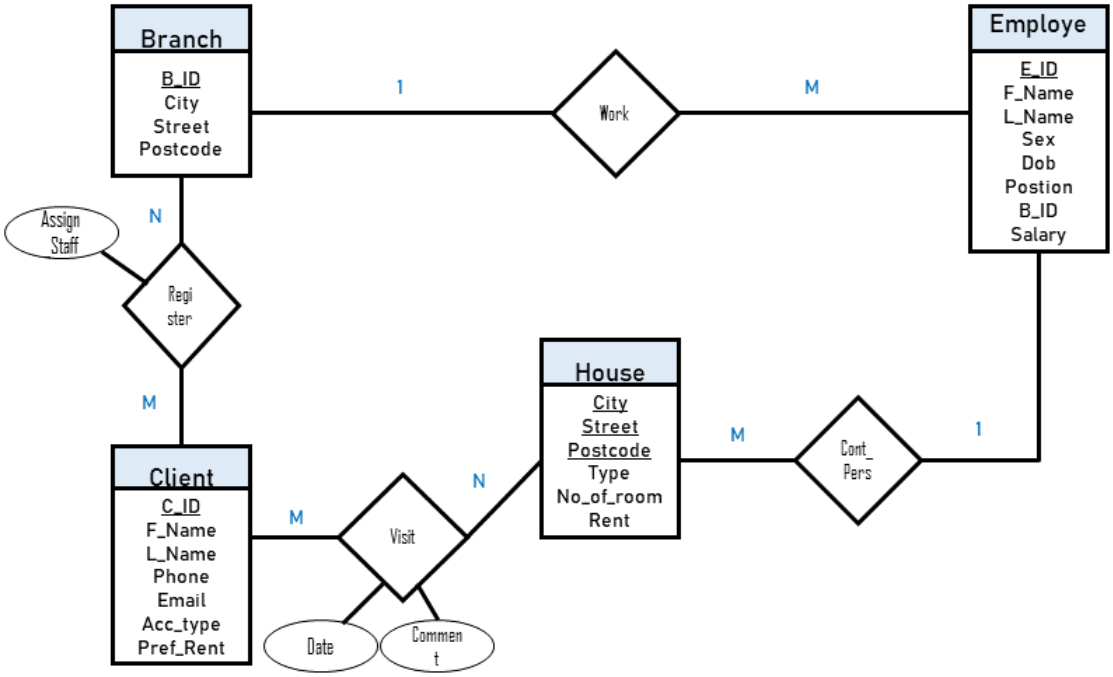

Figure 3: Partial ERD of the 'Bhalo Basha Chai' project

1. Create all the tables that can be found from this ERD.
   [Hint: There will be at least 6 tables in total.]
   Use the cascading delete clause for all the foreign keys in your tables. If there's any foreign key that should not have cascading delete, write down a comment explaining your reasoning.

2. Create a view named Visiting_List that displays which customer has visited which house and on which day. Your view should contain the attributes C_ID, First Name, Last Name and phone number of the customer, the visiting date and the address of the

visited house.

From this view, create another view named HOUSE_VISIT_HISTORY that contains the number of times a house has been visited.

3. Create a view named Employee_List that displays all the branches and the employees that work under each branch. Your view should have the attributes B_ID, City, E_ID, First name of employee and his position in the company.

4. There are three types of users in this database: Customers, Staff and Admin. Customers should be able to view their own information. Staff can view and modify both his and the customer's information. The Admin can modify Branch and House information, and at the same time can do everything the staff and customers can do.

   Create three separate roles for the three categories of users.

Solution

```
--Task 1

CREATE TABLE BRANCH
(
  B_ID VARCHAR2(20),
  CITY VARCHAR2(20),
  STREET VARCHAR2(20),
  POSTCODE INT,
  CONSTRAINT BRANCH_PK PRIMARY KEY(B_ID)
);

CREATE TABLE EMPLOYEE
(
  E_ID VARCHAR2(20),
  F_NAME VARCHAR2(20),
  L_NAME VARCHAR2(20),
  SEX VARCHAR2(5),
  DOB DATE,
  POSITION VARCHAR2(20),
  SALARY NUMBER(10,2),
  WORK_B_ID VARCHAR2(20),
  CONSTRAINT EMPLOYEE_PK PRIMARY KEY(E_ID),
  CONSTRAINT EMPLOYEE_FK1 FOREIGN KEY(WORK_B_ID) REFERENCES BRANCH(B_ID
   )
--no cascading here as removing a certain branch shouldn't fire its
   employees
);

CREATE TABLE HOUSE
(
  CITY VARCHAR2(20),
  STREET VARCHAR2(20),
  POSTCODE INT,
```

```
  TYPE VARCHAR2(10),
  NO_OF_ROOM INT,
  RENT NUMBER(10,2),
  CONTACT_E_ID VARCHAR2(20),
  CONSTRAINT HOUSE_PK PRIMARY KEY(CITY, STREET, POSTCODE),
  CONSTRAINT HOUSE_FK1 FOREIGN KEY(CONTACT_E_ID) REFERENCES EMPLOYEE(
   E_ID)
--no cascading here as firing an employee shouldn't remove a property
);

CREATE TABLE CLIENT
(
  C_ID VARCHAR2(20),
  F_NAME VARCHAR2(20),
  L_NAME VARCHAR2(20),
  PHONE INT,
  EMAIL VARCHAR2(30),
  ACC_TYPE VARCHAR2(20),
  PREF_RENT NUMBER(10,2),
  CONSTRAINT CLIENT_PK PRIMARY KEY(C_ID)
);

CREATE TABLE CLIENT_REGISTER_BRANCH
(
  C_ID VARCHAR2(20),
  B_ID VARCHAR2(20),
  ASSIGNED_STAFF VARCHAR2(20),
  CONSTRAINT CLIENT_REGISTER_BRANCH_PK PRIMARY KEY(C_ID,B_ID),
  CONSTRAINT CLIENT_REGISTER_BRANCH_FK1 FOREIGN KEY(C_ID) REFERENCES
   CLIENT(C_ID) ON DELETE CASCADE,
  CONSTRAINT CLIENT_REGISTER_BRANCH_FK2 FOREIGN KEY(B_ID) REFERENCES
   BRANCH(B_ID),
--no cascading for fk2 as removing a branch shouldn't remove whether a
   client has registered or not
  CONSTRAINT CLIENT_REGISTER_BRANCH_FK3 FOREIGN KEY(ASSIGNED_STAFF)
```

```
    REFERENCES EMPLOYEE(E_ID)
--no cascading for fk3 as removing an employee shouldn't remove whether
    a client has registered at a branch or not
);

CREATE TABLE CLIENT_VISIT_HOUSE
(
  C_ID VARCHAR2(20),
  HOUSE_CITY VARCHAR2(20),
  HOUSE_STREET VARCHAR2(20),
  HOUSE_POSTCODE INT,
  DATETIME DATE,
  COMMENTS VARCHAR2(50),
  CONSTRAINT CLIENT_VISIT_HOUSE_PK PRIMARY KEY(C_ID, HOUSE_CITY,
    HOUSE_STREET, HOUSE_POSTCODE),
  CONSTRAINT CLIENT_VISIT_HOUSE_FK1 FOREIGN KEY(C_ID) REFERENCES CLIENT
    (C_ID) ON DELETE CASCADE,
  CONSTRAINT CLIENT_VISIT_HOUSE_FK2 FOREIGN KEY(HOUSE_CITY,
    HOUSE_STREET, HOUSE_POSTCODE) REFERENCES HOUSE(CITY, STREET,
    POSTCODE) ON DELETE CASCADE
);


--Task 2

CREATE OR REPLACE VIEW VISITING_LIST AS
SELECT CLIENT.C_ID, CLIENT.F_NAME, CLIENT.L_NAME, CLIENT.PHONE,
    CLIENT_VISIT_HOUSE.DATETIME AS VISITDATE, CLIENT_VISIT_HOUSE.
    HOUSE_CITY, CLIENT_VISIT_HOUSE.HOUSE_STREET, CLIENT_VISIT_HOUSE.
    HOUSE_POSTCODE FROM CLIENT, CLIENT_VISIT_HOUSE
WHERE CLIENT.C_ID = CLIENT_VISIT_HOUSE.C_ID;

CREATE OR REPLACE VIEW HOUSE_VISIT_HISTORY AS
SELECT HOUSE_CITY, HOUSE_STREET, HOUSE_POSTCODE , COUNT(*) AS
    TOT_VISITED_TIMES FROM VISITING_LIST
```

```
88 GROUP BY HOUSE_CITY, HOUSE_STREET, HOUSE_POSTCODE;
89
90 --Task 3
91
92 CREATE OR REPLACE VIEW EMPLOYEE_LIST AS
93 SELECT BRANCH.B_ID, BRANCH.CITY, EMPLOYEE.E_ID, EMPLOYEE.F_NAME,
       EMPLOYEE.POSITION FROM BRANCH, EMPLOYEE
94 WHERE EMPLOYEE.WORK_B_ID=BRANCH.B_ID;
```

SCENARIO 3

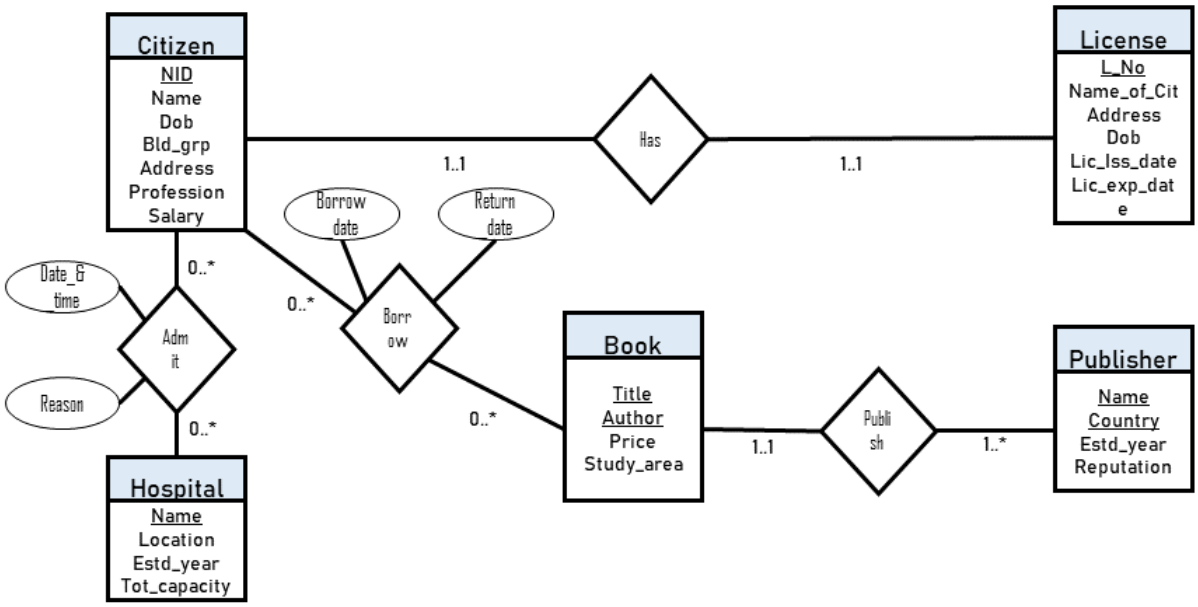

Figure 4: Partial ERD of the BD Government's digitalized sectors

1. Create all the tables that can be found from this ERD.
   [Hint: There will be at least 7 tables in total.]
   Use the cascading delete clause for all the foreign keys in your tables. If there's any foreign key that should not have cascading delete, write down a comment explaining your reasoning.

2. Create a view named Borrow_Record that displays the list of Citizens that have borrowed books. Your view should contain the attributes NID, citizen name, title, author, study area of the book, publisher name and country. From this view, create another view

named Citizen_Borrowal_Count that contains NID of a citizen and the total number of times he has borrowed a book.

3. Create a view named License_Holder_Info that gives information about each citizen holding a license number. Your view should contain the attributes License number, License issue and expiry date, Name and Profession of citizen.

4. Three categories of users have been identified so far for this database.

    - The Library Admin will be able to view and modify book and publisher records.
    - The Hospital Clerk will be able to view the Citizen IDs admitted to the Hospital.
    - And a regular citizen will be able to view his citizen and driving license information.

    Now, create three separate roles and grant them the appropriate privileges for these three categories of users.

Solution

```
--Task 1

CREATE TABLE CITIZEN
(
  NID INT,
  NAME VARCHAR2(30),
  DOB DATE,
  BLD_GRP VARCHAR2(5),
  ADDRESS VARCHAR2(30),
  PROFESSION VARCHAR2(20),
  SALARY NUMBER(10,2),
  CONSTRAINT CITIZEN_PK PRIMARY KEY(NID)
);

CREATE TABLE LICENSE
(
  L_NO VARCHAR2(30),
  NAME_OF_CIT VARCHAR2(30),
  ADDRESS VARCHAR2(30),
  DOB DATE,
  LIC_ISSUE_DATE DATE,
  LIC_EXP_DATE DATE,
  NID INT,
  CONSTRAINT LICENSE_PK PRIMARY KEY(L_NO),
  CONSTRAINT LICENSE_FK1 FOREIGN KEY(NID) REFERENCES CITIZEN(NID) ON
    DELETE CASCADE
);


CREATE TABLE PUBLISHER
(
  NAME VARCHAR2(30),
  COUNTRY VARCHAR2(20),
```

```
  ESTD_YEAR DATE,
  REPUTATION VARCHAR2(20),
  CONSTRAINT PUBLISHER_PK PRIMARY KEY(NAME, COUNTRY)
);

CREATE TABLE BOOK
(
  TITLE VARCHAR2(30),
  AUTHOR VARCHAR2(30),
  PRICE NUMBER(5,2),
  STUDY_AREA VARCHAR2(20),
  PUBLISHER_NAME VARCHAR2(30),
  PUBLISHER_COUNTRY VARCHAR2(20),
  CONSTRAINT BOOK_PK PRIMARY KEY(TITLE, AUTHOR),
  CONSTRAINT BOOK_FK1 FOREIGN KEY(PUBLISHER_NAME, PUBLISHER_COUNTRY)
   REFERENCES PUBLISHER(NAME, COUNTRY)
--no cascading for fk1 here, as a book shouldn't be removed if its
   publisher suddenly stopped publishing
);

CREATE TABLE HOSPITAL
(
  NAME VARCHAR2(30),
  LOCATION VARCHAR2(30),
  ESTD_YEAR DATE,
  TOTAL_CAPACITY INT,
  CONSTRAINT HOSPITAL_PK PRIMARY KEY(NAME)
);

CREATE TABLE CITIZEN_BORROW_BOOK
(
  NID INT,
  BOOK_TITLE VARCHAR2(30),
  BOOK_AUTHOR VARCHAR2(30),
  BORROW_DATE DATE,
```

```
  RETURN_DATE DATE,
  CONSTRAINT CITIZEN_BORROW_BOOK_PK PRIMARY KEY(NID, BOOK_TITLE,
   BOOK_AUTHOR),
  CONSTRAINT CITIZEN_BORROW_BOOK_FK1 FOREIGN KEY(NID) REFERENCES
   CITIZEN(NID) ON DELETE CASCADE,
  CONSTRAINT CITIZEN_BORROW_BOOK_FK2 FOREIGN KEY(BOOK_TITLE,
   BOOK_AUTHOR) REFERENCES BOOK(TITLE, AUTHOR) ON DELETE CASCADE
);

CREATE TABLE CITIZEN_ADMIT_HOSPITAL
(
  NID INT,
  HOSP_NAME VARCHAR2(30),
  DATEANDTIME DATE,
  REASON VARCHAR2(30),
  CONSTRAINT CITIZEN_ADMIT_HOSPITAL_PK PRIMARY KEY(NID, HOSP_NAME),
  CONSTRAINT CITIZEN_ADMIT_HOSPITAL_FK1 FOREIGN KEY(NID) REFERENCES
   CITIZEN(NID),
--no cascading for fk1 here as the hospital should still keep track of
   its admissions even if a citizen were to disappear/go boom
  CONSTRAINT CITIZEN_ADMIT_HOSPITAL_FK2 FOREIGN KEY(HOSP_NAME)
   REFERENCES HOSPITAL(NAME)
--no cascading for fk2 as a citizen may want to keep track of which
   hospitals he went to even if they no longer exist
);

--Task 2

CREATE OR REPLACE VIEW BORROW_RECORD AS
SELECT CITIZEN.NID, CITIZEN.NAME, BOOK.TITLE, BOOK.AUTHOR, BOOK.
   STUDY_AREA, BOOK.PUBLISHER_NAME, BOOK.PUBLISHER_COUNTRY FROM CITIZEN
   , BOOK, CITIZEN_BORROW_BOOK
WHERE CITIZEN.NID = CITIZEN_BORROW_BOOK.NID AND BOOK.TITLE =
   CITIZEN_BORROW_BOOK.BOOK_TITLE AND BOOK.AUTHOR = CITIZEN_BORROW_BOOK
   .BOOK_AUTHOR;
```

```
90
91 CREATE OR REPLACE VIEW CITIZEN_BORROWAL_COUNT AS
92 SELECT NID, COUNT(*) AS TOT_BORROWALS FROM BORROW_RECORD
93 GROUP BY NID;
94
95 --Task 3
96
97 CREATE OR REPLACE VIEW LICENSE_HOLDER_INFO AS
98 SELECT LICENSE.L_NO, LICENSE.LIC_ISSUE_DATE, LICENSE.LIC_EXP_DATE,
      CITIZEN.NAME, CITIZEN.PROFESSION FROM LICENSE, CITIZEN
99 WHERE LICENSE.NID = CITIZEN.NID;
```

SCENARIO 4

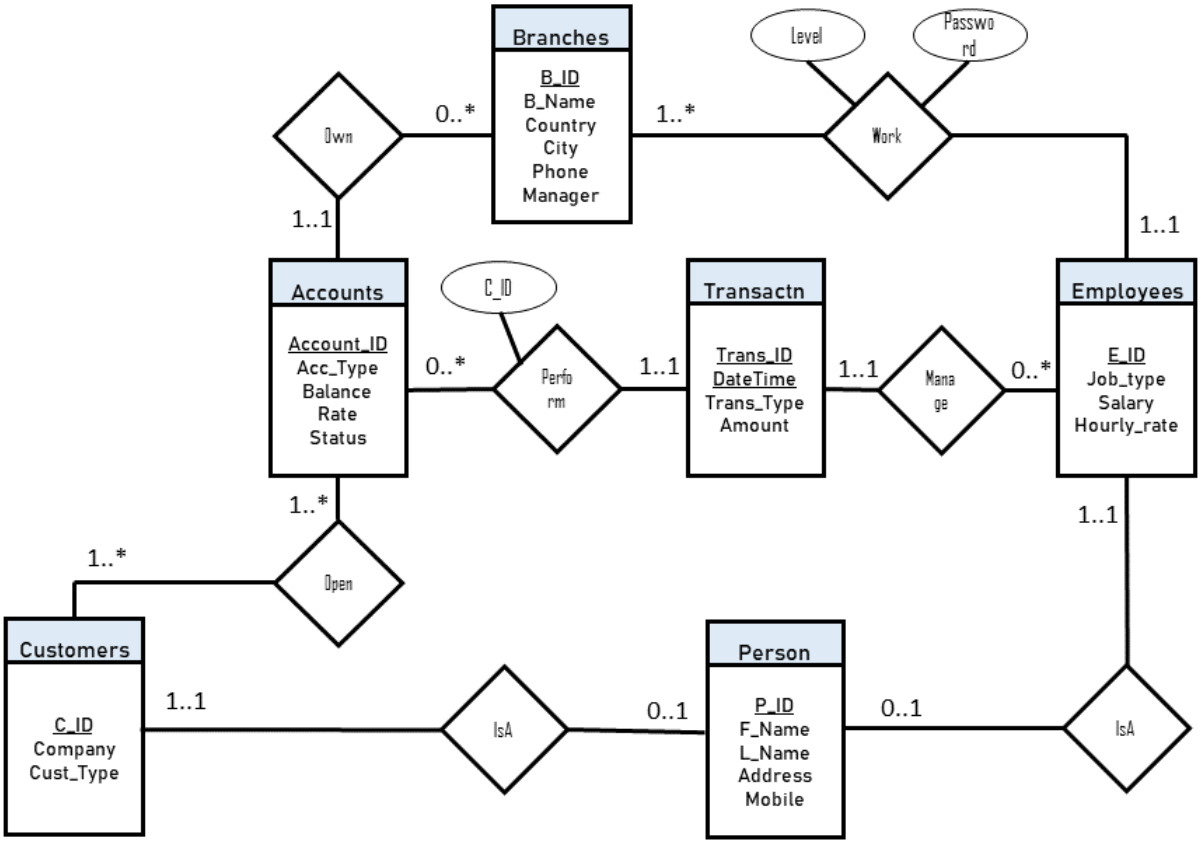

Figure 5: Partial ERD of a Banking System

1. After determining the type of relationship that exist between the entities (i.e. one-to-one, one-to-many or many-to-many), create all the tables that can be found from this ERD.

   [Hint: There will be at least 7 tables in total.]

   Use the cascading delete clause for all the foreign keys in your tables. If there's any foreign key that should not have cascading delete, write down a comment explaining your reasoning.

2. Create a view named Customer_Account_Details that gives information about cus-

tomers and their accounts. This view should have the attributes C_ID, Customer type, Account ID, Account type, Balance and Rate.
From this view, create another view named Customer_Balance_Total that contains the customer ID and the total balance of the customer across all his accounts.

3. Create a view named Customer_Transaction_History that includes the C_ID, Account ID, DateTime and amount of transaction.

4. Three categories of users have been identified so far for this database:

   - A customer will be able to view his account information from the database.
   - An accountant will be able to modify the transaction record and account information of a customer.
   - The manager will be able to modify the customer, employee and branch information and obviously, view all this information.

   Now, create three roles for these three categories of users and grant them the appropriate privileges.

Solution

```
--TASK 1

CREATE TABLE PERSON
(
  P_ID VARCHAR2(30),
  F_NAME VARCHAR2(20),
  L_NAME VARCHAR2(20),
  ADDRESS VARCHAR2(50),
  MOBILE INT,
  CONSTRAINT PERSON_PK PRIMARY KEY(P_ID)
);

CREATE TABLE CUSTOMERS
(
  C_ID VARCHAR2(30),
  COMPANY VARCHAR2(20),
  CUST_TYPE VARCHAR2(10),
  P_ID VARCHAR2(30),
  CONSTRAINT CUSTOMERS_PK PRIMARY KEY(C_ID),
  CONSTRAINT CUSTOMERS_FK1 FOREIGN KEY(P_ID) REFERENCES PERSON(P_ID) ON
    DELETE CASCADE
);

CREATE TABLE BRANCHES
(
  B_ID VARCHAR2(30),
  B_NAME VARCHAR2(30),
  SALARY NUMBER(10,3),
  COUNTRY VARCHAR2(30),
  CITY VARCHAR2(30),
  PHONE INT,
  MANAGER VARCHAR2(30),
  CONSTRAINT BRANCHES_PK PRIMARY KEY(B_ID)
```

```
);


CREATE TABLE EMPLOYEES
(
  E_ID VARCHAR2(30),
  JOB_TYPE VARCHAR2(10),
  SALARY NUMBER(10,3),
  HOURLY_RATE NUMBER(10,3),
  P_ID VARCHAR2(30),
  B_ID VARCHAR2(30),
  LEVEL_NO INT,
  PASSWORD VARCHAR2(30),
  CONSTRAINT EMPLOYEES_PK PRIMARY KEY(E_ID),
  CONSTRAINT EMPLOYEES_FK1 FOREIGN KEY(P_ID) REFERENCES PERSON(P_ID) ON
    DELETE CASCADE,
  CONSTRAINT EMPLOYEES_FK2 FOREIGN KEY(B_ID)REFERENCES BRANCHES(B_ID)
--no cascading for fk2 as removing a branch shouldn't fire its
   employees
);

CREATE TABLE ACCOUNTS
(
  ACCOUNT_ID VARCHAR2(30),
  ACC_TYPE VARCHAR2(10),
  BALANCE NUMBER(10,3),
  RATE NUMBER(10,3),
  STATUS VARCHAR2(30),
  B_ID VARCHAR2(30),
  CONSTRAINT ACCOUNTS_PK PRIMARY KEY(ACCOUNT_ID),
  CONSTRAINT ACCOUNTS_FK1 FOREIGN KEY(B_ID) REFERENCES BRANCHES(B_ID)
--no cascading for fk1 as removing a branch shouldn't delete its
   accounts
);

```

```
65 CREATE TABLE TRANSACTIONS
66 (
67   TRANS_ID INT,
68   DATETIME DATE,
69   TRANSTYPE VARCHAR2(20),
70   AMOUNT NUMBER(10,2),
71   ACCOUNT_ID VARCHAR2(30),
72   C_ID VARCHAR2(30),
73   E_ID VARCHAR2(30),
74   CONSTRAINT TRANSACTIONS_PK PRIMARY KEY(TRANS_ID,DATETIME),
75   CONSTRAINT TRANSACTIONS_FK1 FOREIGN KEY(ACCOUNT_ID) REFERENCES
      ACCOUNTS(ACCOUNT_ID) ON DELETE CASCADE,
76   CONSTRAINT TRANSACTIONS_FK2 FOREIGN KEY(C_ID) REFERENCES CUSTOMERS(
      C_ID),
77 --no cascading for fk2 as an account may have more than one customer
      and thus removing one customer shouldn't remove the transaction
      record
78   CONSTRAINT TRANSACTIONS_FK3 FOREIGN KEY(E_ID) REFERENCES EMPLOYEES(
      E_ID)
79 --no cascading for fk3 as removing an employee shouldn't affect the
      transactions he has managed
80 );
81
82 CREATE TABLE CUSTOMERS_OPEN_ACCOUNTS
83 (
84   C_ID VARCHAR2(30),
85   ACCOUNT_ID VARCHAR2(30),
86   CONSTRAINT CUSTOMERS_OPEN_ACCOUNTS_PK PRIMARY KEY(C_ID, ACCOUNT_ID),
87   CONSTRAINT CUSTOMERS_OPEN_ACCOUNTS_FK1 FOREIGN KEY(C_ID) REFERENCES
      CUSTOMERS(C_ID) ON DELETE CASCADE,
88   CONSTRAINT CUSTOMERS_OPEN_ACCOUNTS_FK2 FOREIGN KEY(ACCOUNT_ID)
      REFERENCES ACCOUNTS(ACCOUNT_ID) ON DELETE CASCADE
89 );
90
91
```

```
92 --TASK 2
93
94 CREATE OR REPLACE VIEW CUSTOMER_ACCOUNT_DETAILS AS
95 SELECT CUSTOMERS.C_ID, CUSTOMERS.CUST_TYPE, ACCOUNTS.ACCOUNT_ID,
       ACCOUNTS.ACC_TYPE, ACCOUNTS.BALANCE, ACCOUNTS.RATE FROM CUSTOMERS,
       ACCOUNTS, CUSTOMERS_OPEN_ACCOUNTS
96 WHERE CUSTOMERS.C_ID=CUSTOMERS_OPEN_ACCOUNTS.C_ID AND ACCOUNTS.
       ACCOUNT_ID=CUSTOMERS_OPEN_ACCOUNTS.ACCOUNT_ID;
97
98 CREATE OR REPLACE VIEW CUSTOMER_BALANCE_TOTAL AS
99 SELECT C_ID, SUM(BALANCE) AS TOT_BALANCE FROM CUSTOMER_ACCOUNT_DETAILS
100 GROUP BY C_ID;
101
102 --TASK 3
103
104 CREATE OR REPLACE VIEW CUSTOMER_TRANSACTION_HISTORY AS
105 SELECT TRANSACTIONS.C_ID, ACCOUNTS.ACCOUNT_ID, TRANSACTIONS.DATETIME,
       TRANSACTIONS.AMOUNT FROM ACCOUNTS, TRANSACTIONS
106 WHERE ACCOUNTS.ACCOUNT_ID = TRANSACTIONS.ACCOUNT_ID;
```