به نام خدا

گزارش آز پایگاه داده	هفته ی ششم – بخش اول
سید پوریا احمدی	۹۷۲۳۰۰۲
محمدمهدی هجرتی	۹۷۲۳۱۰۰

سوال اول)

برای ساختن جداول با توجه به اینکه رابطه ی loan-branch یک به چند می باشد پس لازم است در سمت loan کلید خارجی branch اضافه شود و جدول branch بدون هیچ تغییری ساخته شود.

```
10
11  CREATE TABLE BRANCH (
12      branch_name VARCHAR(64),
13      branch_city VARCHAR(64),
14      assets INT,
15      PRIMARY KEY (branch_name)
16  );
17
```

رابطه ی cust-banker نیز یک به چند می باشد پس employee بدون تغییر ساخته می شود و کلید خارجی از آن به سمت رابطه ی چند تایی که customer می باشد برده می شود.

```
18
19  CREATE TABLE EMPLOYEE (
20      employee_id INT,
21      employee_name VARCHAR(64),
22      dependent_name VARCHAR(64),
23      telephone_number VARCHAR(20),
24      starting_date DATE,
25      employment_length INT,
26      PRIMARY KEY (employee_id),
27  );
28
```

```sql

CREATE TABLE CUSTOMER (
    customer_id INT,
    customer_name VARCHAR(64),
    customer_street VARCHAR(64),
    customer_city VARCHAR(64),
    employee_id INT,
    PRIMARY KEY (customer_id),
    FOREIGN KEY (employee_id)
        REFERENCES EMPLOYEE (employee_id)
);

```

از طرف دیگر با توجه به یک به یک بودن رابطه ی borrower بین هر کدام از customer یا loan می توانیم کلید خارجی را قرار دهیم که ما این کار را برای loan انجام داده ایم.

پس در جدول loan یک کلید خارجی مربوط به branch قرار گرفته است و یک کلید customer نیز الان افزوده می شود.

```sql

CREATE TABLE LOAN (
    loan_number INT,
    amount INT,
    branch_name VARCHAR(64),
    customer_id INT,
    PRIMARY KEY (loan_number),
    FOREIGN KEY (branch_name)
        REFERENCES BRANCH (branch_name),
    FOREIGN KEY (customer_id)
        REFERENCES CUSTOMER (customer_id)
);

```

با توجه به یک به چند بودن رابطه ی loan-payment در سمت چندتایی رابطه یعنی payment کلید افزوده می شود.

```sql

CREATE TABLE PAYMENT (
    payment_number INT,
    payment_date DATE,
    payment_amount INT,
    loan_number INT,
    PRIMARY KEY (payment_number),
    FOREIGN KEY (loan_number)
        REFERENCES LOAN (loan_number)
);

```

در نهایت رابطه ی depositor نیز یک رابطه ی یک به یک بین account و customer ایجاد می کند پس تغییری در جداول ایجاد نمی شود. اما به دلیل اینکه خود رابطه ی depositor دارای یک entity است پس یک جدول مجزا برای آن ساخته می شود.

```sql

CREATE TABLE DEPOSITOR (
    customer_id INT,
    account_number INT,
    access_date DATE,
    PRIMARY KEY (customer_id, account_number),
    FOREIGN KEY (customer_id)
        REFERENCES CUSTOMER (customer_id),
    FOREIGN KEY (account_number)
        REFERENCES ACCOUNT (account_number)
);
```

در جدول account نیز یک رابطه ی ارث بری وجود دارد به این صورت که دو نوع checking و saving علاوه بر داشتن ویژگی های account, دارای ویژگی های مختص به خودشان نیز هستند پس دو جدول مجزا نیز برای آن ها تشکیل می دهیم.

```sql

CREATE TABLE ACCOUNT (
    account_number INT,
    balance INT,
    customer_id INT,
    PRIMARY KEY (account_number),
);

CREATE TABLE CHECKING_ACCOUNT (
    account_number INT,
    overdraft_amount INT,
    PRIMARY KEY (account_number),
    FOREIGN KEY (account_number)
        REFERENCES ACCOUNT (account_number)
);


CREATE TABLE SAVING_ACCOUNT (
    account_number INT,
    interest_rate INT,
    PRIMARY KEY (account_number),
    FOREIGN KEY (account_number)
        REFERENCES ACCOUNT (account_number)
);

```

در ادامه داده های نمونه را به صورت زیر وارد پایگاه داده ی ساخته شده می کنیم.

```sql
INSERT INTO BRANCH (branch_name, branch_city, assets)
VALUES ('one', 'Tehran', '12000'),
       ('two', 'Tehran', '13000'),
       ('three', 'mashhad', '15000');

INSERT INTO EMPLOYEE (employee_id, employee_name, dependent_name, telephone_number, starting_date, employment_length)
VALUES (1, 'Ali', 'Ali2', '123456', '2022-03-01', 2),
        (2, 'Hasan', 'Hasan2', '234567', '2022-05-08', 1),
        (3, 'Abbas', 'Abbas2', '345678', '2022-02-28', 4);

INSERT INTO CUSTOMER (customer_id, customer_name, customer_street, customer_city, employee_id)
VALUES (1, 'Ali', 'S12', 'Tehran', '1'),
        (2, 'Hasan', 'S32', 'Tehran', '1'),
        (3, 'Abbas', 'S32', 'Tehran', '2'),
        (4, 'Ali2', 'S32', 'Tehran', '3');

INSERT INTO LOAN (loan_number, amount, branch_name, customer_id)
VALUES (1, 100, 'one', 2),
        (2, 250, 'three', 1);

INSERT INTO PAYMENT (payment_number, payment_date, payment_amount, loan_number)
VALUES (1, '2022-05-08', 500 , 1),
        (2, '2022-06-8', 600, 1);

INSERT INTO ACCOUNT (account_number, balance, customer_id)
VALUES (1, 500, 1),
        (2, 200, 2),
        (3, 800, 3),
        (4, 700, 4);

INSERT INTO CHECKING_ACCOUNT (account_number, overdraft_amount)
VALUES (1, 100),
        (2, 200);

INSERT INTO SAVING_ACCOUNT (account_number, interest_rate)
VALUES (3, 500),
        (4, 600);

INSERT INTO DEPOSITOR (customer_id, account_number, access_date)
VALUES (1, 1, '2022-01-02'),
        (2, 2, '2022-02-22'),
        (4, 3, '2022-03-13');
```

سوال دوم)

الف)

همانطور که مشخص است یک view ساخته شده که در آن که نام و id مشتری به همراه نام شعبه و مقداروام دریافتی را نشان میدهد فقط با توجه به اینکه اطلاعات در یک جدول نیست از inner join استفاده شده است و دو جدول customer و loan را روی customer_id جوین کرده ایم و در نهایت آن را چاپ کرده ایم.

```sql
CREATE VIEW first_view
AS SELECT CUSTOMER.customer_id , CUSTOMER.customer_name, LOAN.branch_name , LOAN.amount as amount_of_loan
FROM CUSTOMER inner join LOAN on Customer.customer_id = LOAN.customer_id  ;

select * from first_view;
```

ب)

در این سوال شماره حساب و میزان سود حساب هایی را که سال آنها بعد از ۲۰۰۹ باشد را نشان داده ایم ((year(access_date سال را چک میکند که با شرط whereچک شده است) و با توجه به اینکه اطلاعات در سه جدول متفاوت بود دو بار از inner join استفاده شده است.

```
select ACCOUNT.account_number ,interest_rate  from
DEPOSITOR inner join account on DEPOSITOR.account_number    = ACCOUNT.account_number
inner join saving_account on saving_account.account_number = ACCOUNT.account_number
where YEAR(access_date)>2009
```

ج)

در این سوال شماره باز پرداخت وام هایی را که شعبه آنها تهران بوده است را نشان داده ایم. با شرط where شهر را چک کرده ایم و با توجه به اینکه اطلاعات در سه جدول متفاوت بود دو بار از inner join استفاده شده است.

```
select payment_number from
PAYMENT inner join LOAN on PAYMENT.loan_number = LOAN.loan_number
inner join BRANCH on BRANCH.branch_name = LOAN.branch_name where branch_city ='tehran';
```