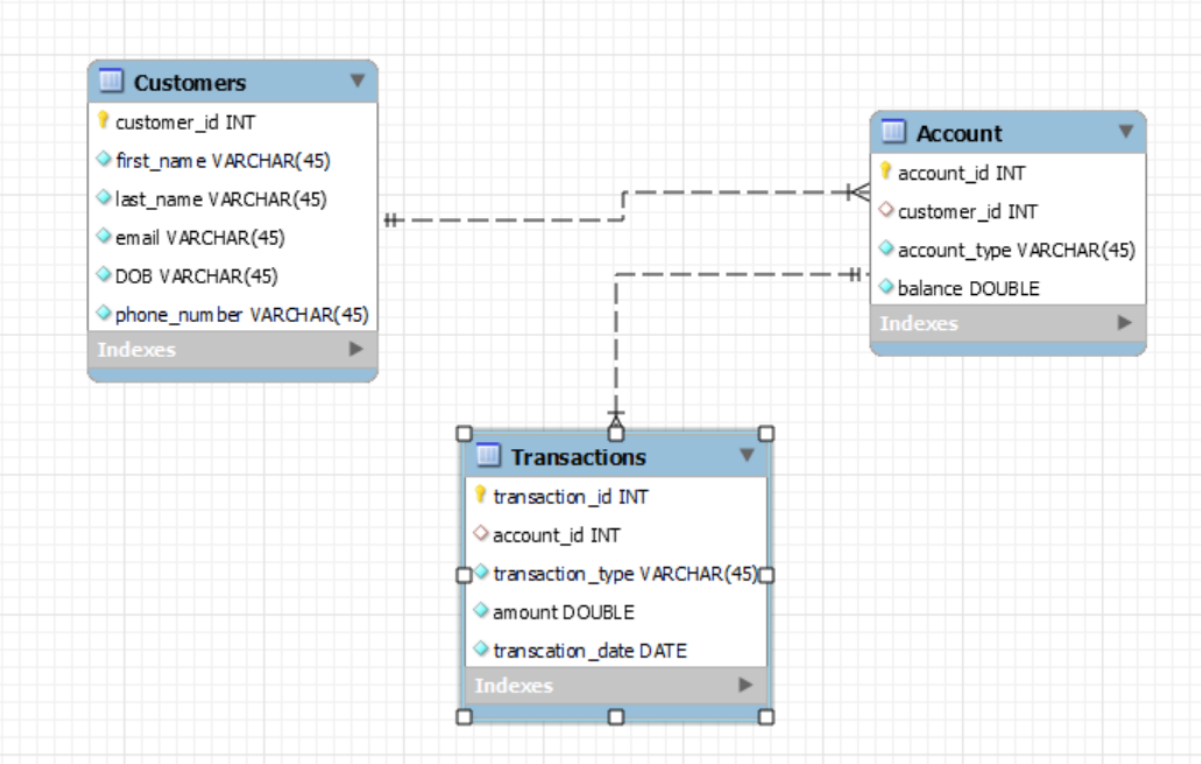

-- MySQL Script generated by MySQL Workbench

-- Tue Mar 5 08:28:08 2024

-- Model: New Model   Version: 1.0

-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;

SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;

SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ER
ROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- ---

-- Schema BankingSystem

-- ---

```
-- -----------------------------------------------------
-- Schema BankingSystem
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `BankingSystem` DEFAULT CHARACTER SET utf8 ;
USE `BankingSystem` ;

-- -----------------------------------------------------
-- Table `BankingSystem`.`Customers`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BankingSystem`.`Customers` (
  `customer_id` INT NOT NULL AUTO_INCREMENT,
  `first_name` VARCHAR(45) NOT NULL,
  `last_name` VARCHAR(45) NOT NULL,
  `email` VARCHAR(45) NOT NULL,
  `DOB` VARCHAR(45) NOT NULL,
  `phone_number` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`customer_id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `BankingSystem`.`Account`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BankingSystem`.`Account` (
  `account_id` INT NOT NULL AUTO_INCREMENT,
  `customer_id` INT NULL,
  `account_type` VARCHAR(45) NOT NULL,
  `balance` DOUBLE NOT NULL,
  PRIMARY KEY (`account_id`),
  INDEX `customer_id_idx` (`customer_id` ASC),
  CONSTRAINT `customer_id`
```

```
    FOREIGN KEY (`customer_id`)
    REFERENCES `BankingSystem`.`Customers` (`customer_id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `BankingSystem`.`Transactions`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BankingSystem`.`Transactions` (
  `transaction_id` INT NOT NULL AUTO_INCREMENT,
  `account_id` INT NULL,
  `transaction_type` VARCHAR(45) NOT NULL,
  `amount` DOUBLE NOT NULL,
  `transcation_date` DATE NOT NULL,
  PRIMARY KEY (`transaction_id`),
  INDEX `account_id_idx` (`account_id` ASC),
  CONSTRAINT `account_id`
    FOREIGN KEY (`account_id`)
    REFERENCES `BankingSystem`.`Account` (`account_id`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

insert into customers(first_name,last_name,dob,email,phone_number) values
```

```
('anand','raju','2002-03-21','anand@gmail.com','9988776655'),
('pavan','kalyan','2001-02-10','kalyan@gmail.com','9977665544'),
('taraka','rama rao','2002-11-15','ramarao@gmail.com','8877665544'),
('abhi','kalyan','2003-03-15','abhi@gmail.com','9977665544'),
('raja','ram','2004-10-15','raja@gmail.com','8877665544');

insert into account(account_type,balance,customer_id) values
('savings',50000,1) ,
('current',120000,2) ,
('zero_balance',100000,3),
('current',150000,1) ,
('savings',30000,5);

insert into transactions(transaction_type,amount,transcation_date,account_id)
values
('deposit', 10000, '2024-02-01',1),
('withdrawal', 5000, '2024-02-02',1),
('deposit', 20000, '2024-02-02',2),
('withdrawal', 8000, '2024-02-02',3),
('transfer', 20000, '2024-02-01',4),
('transfer', 7000, '2024-02-05',5);

-- Tasks 2: Select, Where, Between, AND, LIKE:
-- 1. Insert at least 10 sample records into each of the following tables.
--  • Customers
--  • Accounts
--  • Transactions

-- done above

-- 2. Write SQL queries for the following tasks:
```

-- 1. Write a SQL query to retrieve the name, account type and email of all customers.

```
select first_name,email from customers;
```

-- 2. Write a SQL query to list all transaction corresponding customer.

```
select c.customer_id,c.first_name,a.account_id,a.account_type,t.transaction_id,t.transcation_date
from customers c join account a
on c.customer_id=a.customer_id
join transactions t
on a.account_id=t.account_id
order by c.customer_id;
```

-- 3. Write a SQL query to increase the balance of a specific account by a certain amount.

```
update account set balance=balance+1000 where account_id=2;
```

-- 4. Write a SQL query to Combine first and last names of customers as a full_name.

```
select  concat(first_name,' ',last_name) as full_name from customers;
```

-- 5. Write a SQL query to remove accounts with a balance of zero where the account type is savings.

```
delete from account where balance=0 and account_type='savings';
```

-- 6. Write a SQL query to Find customers living in a specific city.

```
select customer_name,city from customers where city='hyderabad';
```

-- 7. Write a SQL query to Get the account balance for a specific account.

```
select account_id,balance from account where account_id=1;

-- 8. Write a SQL query to List all current accounts with a balance greater than $1,000.

select account_id from account where account_type='current' and balance>1000;

-- 9. Write a SQL query to Retrieve all transactions for a specific account.

select a.account_id,t.transaction_id,t.transaction_type,t.amount
from account a join  transactions t
on a.account_id=t.account_id
where a.account_id=1;

-- 10. Write a SQL query to Calculate the interest accrued on savings accounts based on a given
interest rate.

-- there are no rate of interest and time columns

-- 11. Write a SQL query to Identify accounts where the balance is less than a specified overdraft
limit.

select account_id,balance from account where balance<100000;

-- 12. Write a SQL query to Find customers not living in a specific city.

select * from customers where city not in ('hyderabad');

-- Tasks 3: Aggregate functions, Having, Order By, GroupBy and Joins:

-- 1. Write a SQL query to Find the average account balance for all customers.
```

```
select avg(balance) from account;

-- 2. Write a SQL query to Retrieve the top 10 highest account balances.

select * from account order by balance desc limit 0,10;

-- 3. Write a SQL query to Calculate Total Deposits for All Customers in specific date.

select sum(amount) from transactions where transaction_type='deposit' and
year(transcation_date)=2024 and month(transcation_date)<4;

-- 4. Write a SQL query to Find the Oldest and Newest Customers.

select c.customer_id,c.first_name,a.account_id,t.transcation_date
from customers c join account a
on c.customer_id=a.customer_id
join transactions t
on a.account_id=t.account_id
where t.transcation_date=(select max(transcation_date) from transactions)
 or t.transcation_date=(select min(transcation_date) from transactions)
order by t.transcation_date;

-- 5. Write a SQL query to Retrieve transaction details along with the account type.

select t.transaction_id,t.account_id,t.transaction_type,a.account_type
from account a join transactions t
on a.account_id=t.account_id;

-- 6. Write a SQL query to Get a list of customers along with their account details.

select c.customer_id,concat(c.first_name,c.last_name) as
full_name,a.account_id,a.balance,a.account_type
```

```
from customers c join account a
on c.customer_id=a.customer_id;

-- 7. Write a SQL query to Retrieve transaction details along with customer information for a specific account.

select c.customer_id,concat(c.first_name,c.last_name) as full_name,
a.account_id,t.transaction_id,t.transaction_type,t.amount,t.transcation_date
from customers c join account a
on c.customer_id=a.customer_id
join transactions t
on a.account_id=t.account_id
where a.account_id=2;

-- 8. Write a SQL query to Identify customers who have more than one account.

select c.customer_id,concat(c.first_name,c.last_name) as full_name,count(a.account_id) as no_of_accounts
from customers c join  account a
on c.customer_id=a.customer_id
group by c.customer_id
having no_of_accounts>1;

-- 9. Write a SQL query to Calculate the difference in transaction amounts between deposits and withdrawals.

select
 abs((select sum(amount) from transactions where transaction_type='deposit' )-(select sum(amount) from transactions where transaction_type='withdrawal'))
as difference;

-- 10. Write a SQL query to Calculate the average daily balance for each account over a specified period.
```

```
select day(transcation_date),avg(amount)
from transactions
group by day(transcation_date);

-- 11. Calculate the total balance for each account type.

select account_type,sum(balance) from account group by account_type;

-- 12. Identify accounts with the highest number of transactions order by descending order.

select account_id,count(transaction_id) as number_of_transactions
from transactions
group by account_id order by number_of_transactions desc;

-- 13. List customers with high aggregate account balances, along with their account types.

select c.first_name,a.account_type,a.balance
from customers c join account a
on c.customer_id=a.customer_id
where a.balance>(select avg(balance) from account);

-- 14. Identify and list duplicate transactions based on transaction amount, date, and account.

select * from transactions group by amount;
select * from transactions group by transcation_date;
select * from transactions group by account_id;

select * from transactions group by amount,transcation_date,account_id;
```

-- Tasks 4: Subquery and its type:

-- 1. Retrieve the customer(s) with the highest account balance.

-- considering single account

```
select concat(c.first_name,c.last_name) as full_name,a.balance
from customers c join account a
on c.customer_id=a.customer_id
 where a.balance=(select max(balance) from account);
```

-- 2. Calculate the average account balance for customers who have more than one account.

```
select concat(c.first_name,c.last_name) as full_name,avg(a.balance) as
average_balance,count(a.account_id) as num_of_accounts
from customers c join account a
on c.customer_id=a.customer_id
group by c.customer_id
having num_of_accounts>1;
```

-- 3. Retrieve accounts with transactions whose amounts exceed the average transaction amount.

```
select a.account_id,t.transaction_id,sum(t.amount) as transactions_amount
from account a join transactions t
on a.account_id=t.account_id
group by a.account_id
having transactions_amount>(select avg(amount) from transactions);
```

-- 4. Identify customers who have no recorded transactions.

```
select customer_id,account_id from account where account_id not in (select account_id from
transactions);
```

-- 5. Calculate the total balance of accounts with no recorded transactions.

```
select sum(balance) as total_balance
from account
where account_id not in (select account_id from transactions);
```

-- 6. Retrieve transactions for accounts with the lowest balance.

```
select a.account_id,t.transaction_id,t.transcation_date,t.transaction_type,t.amount
from account a join transactions t
on a.account_id=t.account_id
where a.balance=(select min(balance) from account);
```

-- 7. Identify customers who have accounts of multiple types.

```
select concat(c.first_name,c.last_name) as customer_name,count(c.customer_id) as
num_of_accounts
from customers c join account a
on c.customer_id=a.customer_id
group by c.customer_id
having num_of_accounts>1;
```

-- 8. Calculate the percentage of each account type out of the total number of accounts.

```
select account_type,count(account_type)*100/(select count(*) from account) as percentage
from account
group by account_type;
```

-- 9. Retrieve all transactions for a customer with a given customer_id.

```
select *
from transactions where account_id in
(select account_id from account where customer_id in
                                    (select customer_id from customers where customer_id=1)); --
nested query


select c.customer_id,concat(c.first_name,c.last_name) as full_name,a.account_id,a.account_type,
t.transaction_id,t.transcation_date,t.transaction_type,t.amount
from customers c join account a
on c.customer_id=a.customer_id
join transactions t
on a.account_id=t.account_id
where c.customer_id=1;  -- through join




-- 10. Calculate the total balance for each account type, including a subquery within the SELECT
clause.


select account_type as type,(select sum(balance) from account where account_type=type)
from account
group by account_type;
```