CSCI 5408

DATA MANAGEMENT AND WAREHOUSING

LAB-3: TRANSACTIONS AND DISTRIBUTED DATABASE

Table of Contents

1 : Design a banking application database

Create database

create database lab3;

Create Customer details Table

CREATE TABLE customer_details (
id int NOT NULL AUTO_INCREMENT,
name varchar(30) NOT NULL,
address varchar(255) NOT NULL,
email varchar(40) NOT NULL,
phoneNumber int(10) NOT NULL,
PRIMARY KEY (id)
);

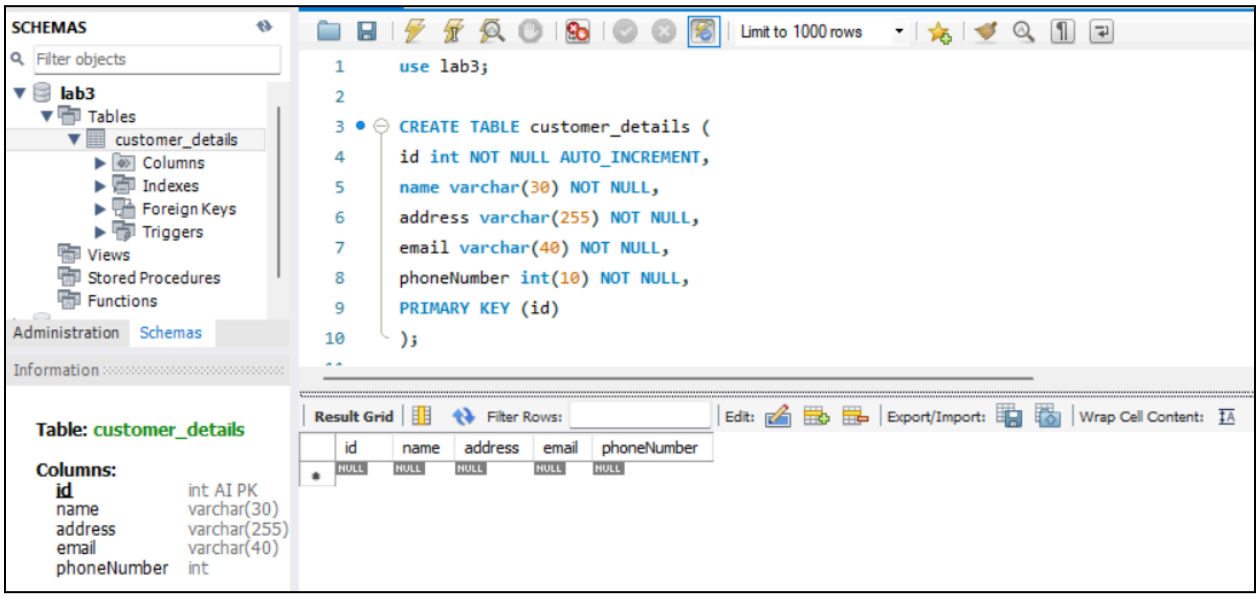


Figure 1: Create customer_details Table

Create account Table

CREATE TABLE account (
accountNumber int(14) NOT NULL,

balance int NOT NULL,
id int,
PRIMARY KEY (accountNumber),
FOREIGN KEY (id) REFERENCES customer_details(id)
);

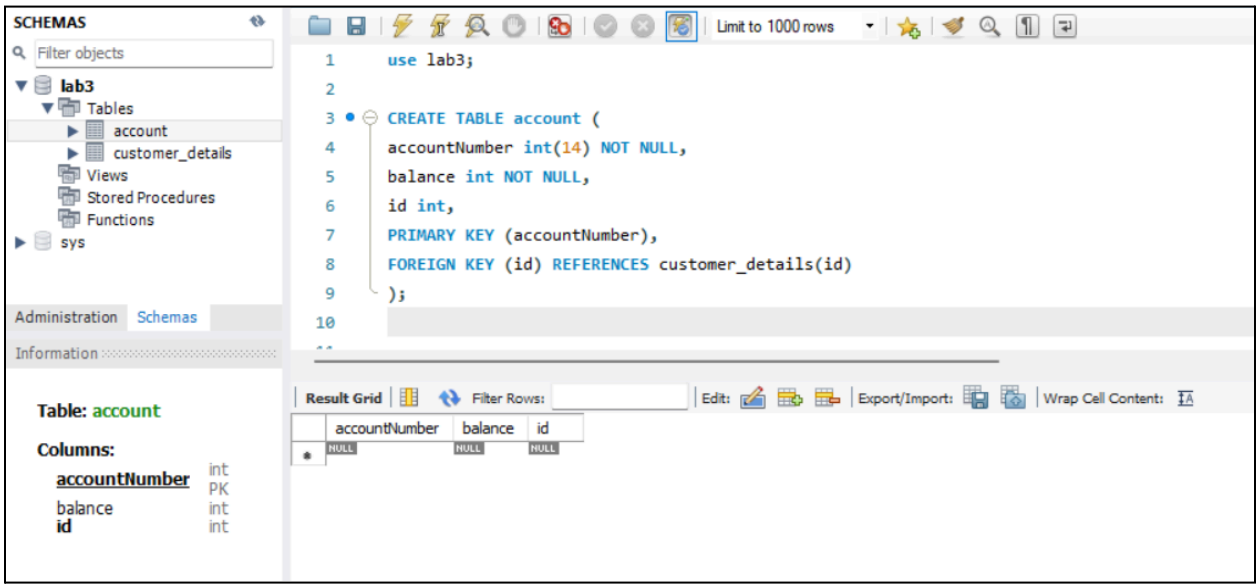


Figure 2: Create account Table

Create account_transfer_details Table

CREATE TABLE account_transfer_details (
id int NOT NULL AUTO_INCREMENT,
senderAccountNumber int(14) NOT NULL,
receiverAccountNumber int(14) NOT NULL,
transactionDate DATETIME Not NULL,
transactionstatus ENUM ('waiting', 'accepted', 'declined'),
PRIMARY KEY (id),
FOREIGN KEY (senderAccountNumber) REFERENCES account(accountNumber),
FOREIGN KEY (receiverAccountNumber) REFERENCES account(accountNumber)
);

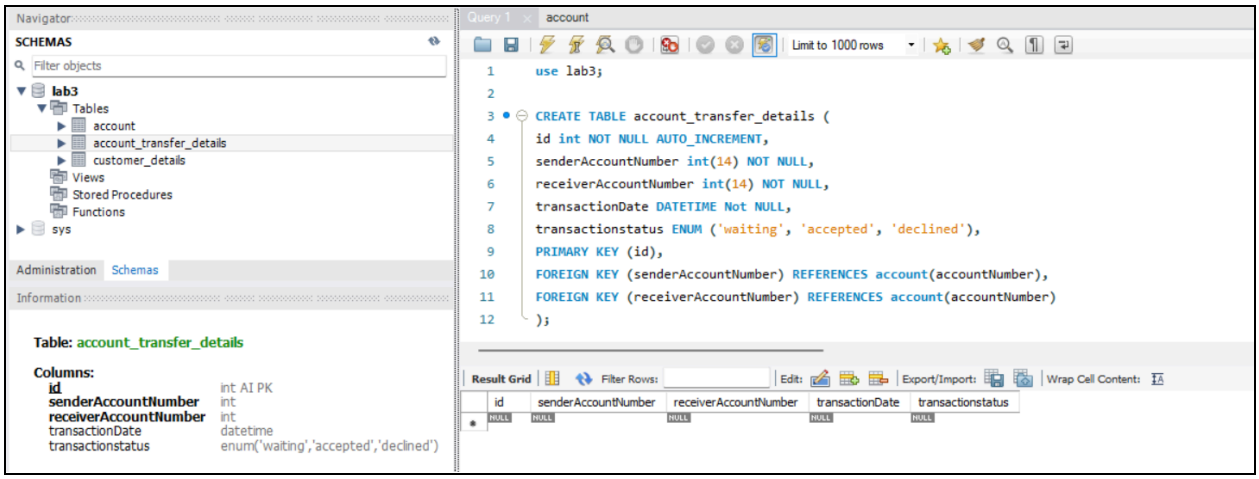


Figure 3: Create account_transfer_details *Table*

2 : Inserting Dummy Data

Inserting in customer_details Table : -

INSERT INTO customer_details (name, address, email, phoneNumber) VALUES ('Jay', 'Quinpool Halifax', 'jay@gmail.com', 215236475);

INSERT INTO customer_details (name, address, email, phoneNumber) VALUES ('Khush', '1881 Brunswick Street', 'khush@gmail.com', 541535870);

INSERT INTO customer_details (name, address, email, phoneNumber) VALUES ('Tanuj', 'Quinpool Halifax', 'tanuj@gmail.com', 785421523);

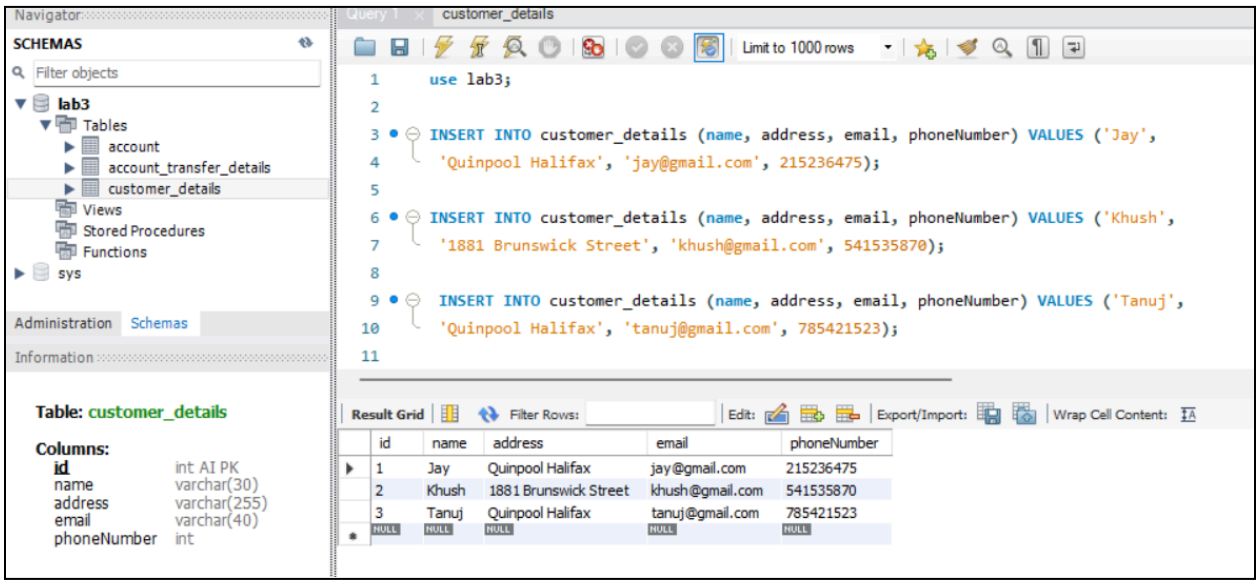


Figure 4: Insert data into customer_details Table

Inserting in account Table: -

INSERT INTO account (accountNumber, balance, id) VALUES (542635214, 254, 1);
INSERT INTO account (accountNumber, balance, id) VALUES (748541263, 985, 2);
INSERT INTO account (accountNumber, balance, id) VALUES (632541526, 985, 3);

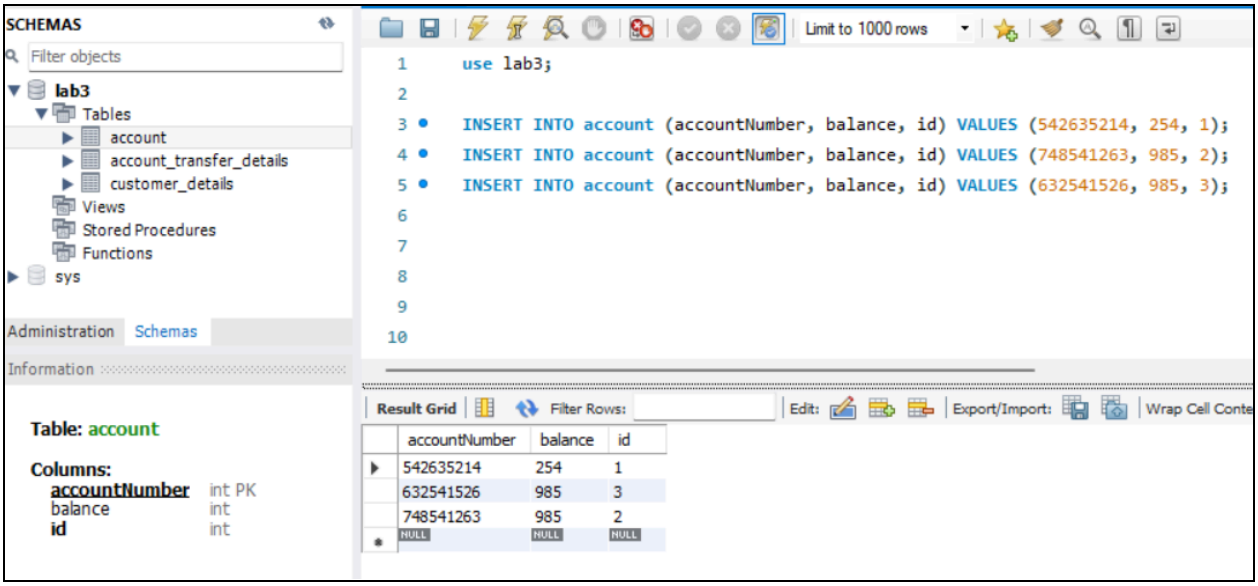


Figure 5: Insert data into account Table

3 : Transactions

3(A) - Transaction “Accepted” state : -

```
SET AUTOCOMMIT=0;

START TRANSACTION;

UPDATE account SET balance = balance - 100
WHERE accountNumber = 542635214;

INSERT INTO account_transfer_details (id, senderAccountNumber, receiverAccountNumber,
transactionDate, transactionstatus)
VALUES (1, 542635214, 748541263, now(), 'waiting');

-- TRANSACTION SUCCESS ASSUMPTION --

UPDATE account
SET balance = balance + 100
WHERE accountNumber = 748541263;

UPDATE account_transfer_details
SET transactionstatus = 'accepted'
WHERE id = 1;

COMMIT;
```

Before

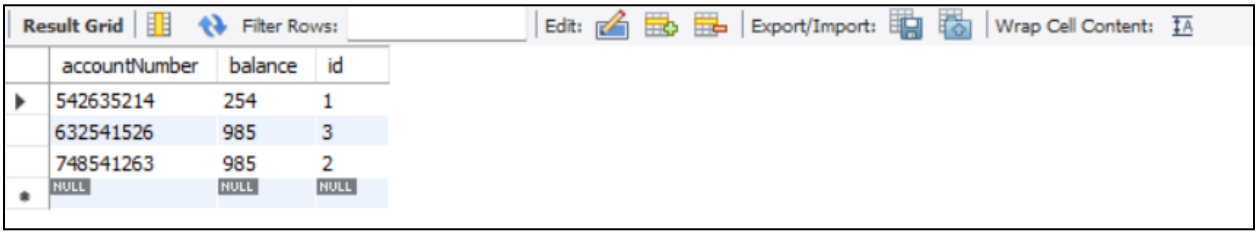
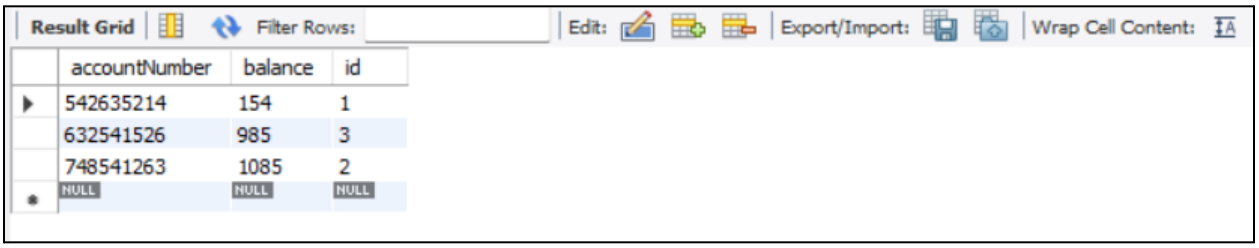


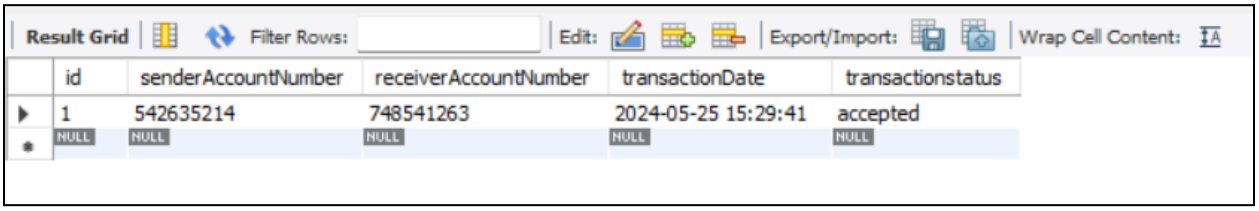
Figure 6: Before Success Transaction account table.

After



accountNumber	balance	id
542635214	154	1
632541526	985	3
748541263	1085	2
NULL	NULL	NULL

Figure 7: After Success Transaction account Table



id	senderAccountNumber	receiverAccountNumber	transactionDate	transactionstatus
1	542635214	748541263	2024-05-25 15:29:41	accepted
NULL	NULL	NULL	NULL	NULL

Figure 8: After Success Transaction account_transfer_details *Table*

Explanation: -

- Firstly, I have added values like how much amount we wanted to debit, the sender and receiver account number, and one predefined transfer_id which is 1 in this case.
- After starting the transaction with START TRANSACTION, the debit operation took place with 100 debit amounts and inserted records to the account_transfer_details table with the same parameters and state as ‘WAITING’.
- Here, We have assumed that the transaction has successfully passed the security check, so I have updated the account table of the receiver with a credit operation of the same amount that debited from the sender
- Finally, updated status to ACCEPTED of the account_transfer_details table.

3(B) - Transaction “Declined” state : -

```
SET AUTOCOMMIT=0;

START TRANSACTION;

UPDATE account SET balance = balance - 200
WHERE accountNumber = 542635214;

INSERT INTO account_transfer_details (id, senderAccountNumber, receiverAccountNumber,
transactionDate, transactionstatus)
VALUES (2, 542635214, 748541263, now(), 'waiting');

SAVEPOINT Debit_Success;

-- TRANSACTION FAILED --

ROLLBACK TO Debit_Success;

UPDATE account SET balance = balance + 200
WHERE accountNumber = 542635214;

UPDATE account_transfer_details SET transactionstatus = 'declined'
WHERE id = 2;

COMMIT;
```

Before

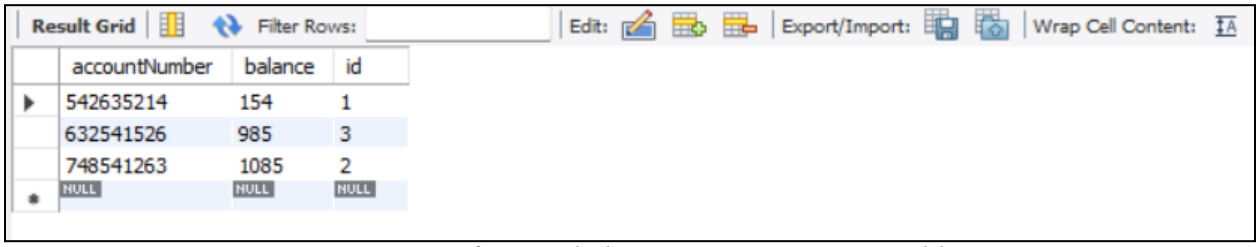
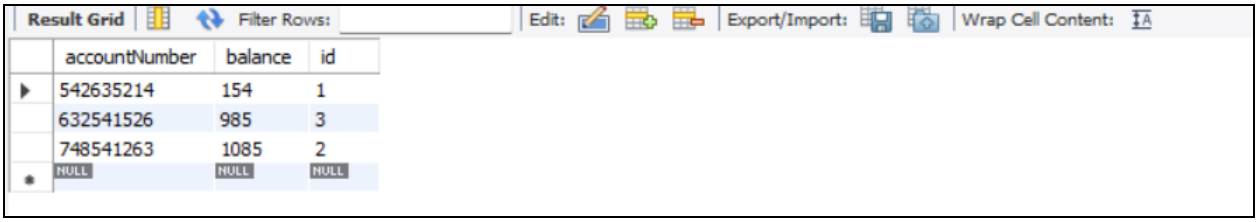


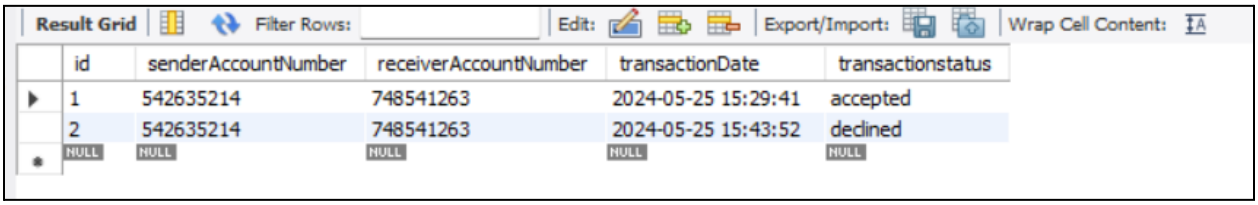
Figure 9: Before Failed Transaction account Table

After



accountNumber	balance	id
542635214	154	1
632541526	985	3
748541263	1085	2
NULL	NULL	NULL

Figure 10: After Failed Transaction account Table



id	senderAccountNumber	receiverAccountNumber	transactionDate	transactionstatus
1	542635214	748541263	2024-05-25 15:29:41	accepted
2	542635214	748541263	2024-05-25 15:43:52	declined
NULL	NULL	NULL	NULL	NULL

Figure 11: After Failed Transaction account_transfer_details Table

Explanation: -

- Here we have added one savepoint before the assumption of the transaction failed.
- Rollback to the previous savepoint is Debit_Success in our case, which means if the transaction has failed, it will go to the previous state where we have added ‘waiting state of account_transfer_details.
- Lastly, I have added the declined status of the transaction in account_transfer_details.