# Project Report

1. Project Proposal:
2. Significance of bank database manage system:

With the rapid progress of the economies of the society and fast development of the internet, today’s banking companies are facing the challenges of fierce competition, accommodating more complex requirements from the customers, providing more financial services and managing the explosion of data efficiently and effectively. To address these challenges successfully, a powerful and efficient bank database manage system(DBMS) is indispensable and significant. A good bank DBMS can provide the banking companies with following benefits:

First, the powerful bank DBMS can provide more functions and services to the customers such as providing to customer various account report accurately and efficiently, thus improve the competence of the bank company.

Second, the powerful bank DBMS can guarantee the normal and efficient operation of modern bank companies by managing the large and complicate data of bank, this can’t be achieved without efficient database manage system.

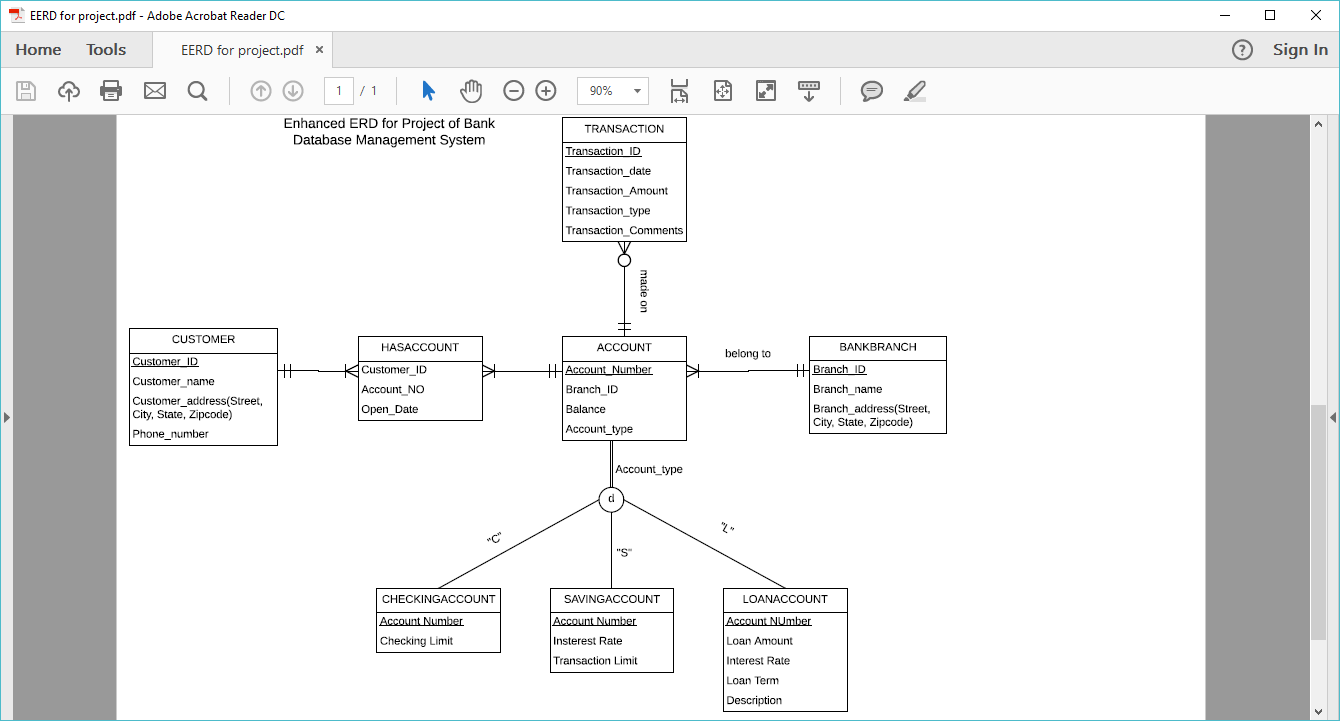
Third, a good bank DBMS can provide high rate of security for customer financial status, such as strict account accessing control, automatic implementation of some financial functions, automatic data collection and update of the database, all of these effectively prevent bank from lost caused by personal mistake and reduce the cost of bank operation and hence increase the profit of the back.

1. The bank DBMS involves several sections:
2. custom-oriented functions, including deposit, withdraw, make transfer and pay loan;
3. administrative functions: add/drop customers, open/close account;
4. reporting functions: print monthly statements, yearly income statements, loan payment schedule
5. EERD of the Bank DBMS

The bank DBMS include the following entities:

1. Customer: collecting the customer information such as name, address and contact information
2. Account: store the information about the account, including the account number, account balance, account type and bank branch of account.
3. Checking account: is subtype of account with unique attribute of checking limit(maximum amount of money for each transaction)
4. Saving account: subtype of account with attributes of account number, interest rate and transaction limit(the limit on times of transactions each month )
5. Loan account: subtype of account, which fields include account number, loan amount, loan term, interest rate and description on the loan.
6. Bank branch: including the fields of branch ID, branch name, branch address.
7. Transaction: recording the each transaction information, such as transaction ID, transaction date, amount, type (deposit, withdraw, transfer) and comments on the transaction.

The relations among different entities are described in the following enhance entity relation diagram:



1. In this EERD, each customer could have more than one hasaccount record (such as one customer has one checking account, one saving account) but at least one hasaccount record; each hasaccount must have one and only one customer.
2. Each Account could have more than one hasaccount record(one account has more than one account holder such as joint account), but at least one hasaccount record(means each account has at least one account holder), each hasaccount has and only has one account associated with it.
3. The account entity contains the common attributes of subtype accounts, such as account number, balance, branch ID and account type. There are three subtypes of account, checking account, saving account and loan account. Each subtype of account has their own unique attributes, such as loan term and loan amount for loan account.
4. Each account only belongs to one bank branch, and each bank branch has at least one account
5. Each account may has more than one transactions and each transaction is made on only one account
6. Conceptual Design of the system

CUSTOMER

|  |  |  |  |
| --- | --- | --- | --- |
| Customer\_ID | Customer\_Name | Customer\_Address | Phone\_Num |

HASACCOUNT

|  |  |  |
| --- | --- | --- |
| Customer\_ID | Acct\_No | Open\_Date |

|  |  |  |  |
| --- | --- | --- | --- |
| Acct\_No | Balance | Acct\_Type | Branch\_ID  ACCOUNT |

BANKBRANCH

|  |  |  |
| --- | --- | --- |
| Branch\_ID | Branch\_Name | Branch\_Address |

CHECKINGACCOUNT

|  |  |
| --- | --- |
| Acct\_No | Check\_Limit |

SAVINGACCOUNT

|  |  |  |
| --- | --- | --- |
| Acct\_No | Interest\_Rate | Trans\_Limit |

LOANACCOUNT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Acct\_No | Loan\_Amt | Interest\_Rate | Term | Description |

TRANSACTION

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Trans\_ID | Acct\_No | Trans\_Date | Trans\_Type | Trans\_Amnt | Trans\_Comts |

1. Physical Design of the Bank DBMS
2. Customer table:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Cust\_ID | Lname | Fname | Street | City | State | Zipcode | Tele\_No |

Fields definition:

|  |  |  |  |
| --- | --- | --- | --- |
| Fields | Data Type | Range of value | Constrains |
| Cust\_ID | Int | 8 | Primary key, Not null, |
| Lname | Varchar | 25 | Not null |
| Fname | Varchar | 25 | Not null |
| Street | Varchar | 30 |  |
| City | Varchar | 20 |  |
| State | Varchar | 2 |  |
| Zipcode | Varchar | 10 |  |
| Tele\_No | Varchar | 14 |  |

1. HasAccount table:

|  |  |  |
| --- | --- | --- |
| Cust\_ID | Acct\_No | Open\_Date |

Fields definition:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Data Type | Range | Constrains |
| Cust\_ID | Int | 8 | Primary Key, not null, reference to CUSTOMER(Cust\_ID) |
| Acct\_no | Int | 8 | Primary Key, not null, reference to ACCOUNT(Acct\_NO) |
| Open\_Date | Date |  | Default value: current date |

1. Account table:

|  |  |  |  |
| --- | --- | --- | --- |
| Acct\_No | Balance | Branch\_ID | Acct\_Type |

Fields definition:

|  |  |  |  |
| --- | --- | --- | --- |
| Fields | Data Type | Range | Constraints |
| Acct\_No | Int | 8 | Primary Key, not null |
| Balance | Numeric | 10 with 2 decimal digit | Default value:0.00 |
| Branch\_ID | Int | 5 | Foreign Key, reference to Table Branch(Branch\_ID) |
| Acct\_Type | Varchar | 2 | Not null, check in {‘C’,’S’,’L’} |

1. BankBranch table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Branch\_ID | Name | Street | City | State | Zipcode |

Fields definition

|  |  |  |  |
| --- | --- | --- | --- |
| Fields | Data Type | Range | Constraints |
| Branch\_ID | Int | 5 | Primary Key, not null |
| Name | Varchar | 25 | Not null |
| Street | Varchar | 30 |  |
| City | Varchar | 20 |  |
| State | Varchar | 2 |  |
| Zipcode | Varchar | 14 |  |

1. CheckingAccount table:

|  |  |
| --- | --- |
| Acct\_No | Check\_Limit |

Fields Definition:

|  |  |  |  |
| --- | --- | --- | --- |
| Fields | Data Type | Range | Constraints |
| Acct\_No | Int | 8 | Primary Key, not null |
| Check\_Limit | Numeric | 10 with 2decimal digits | Default value:0.00 |

1. SavingAccount table:

|  |  |  |
| --- | --- | --- |
| Acct\_No | Interest\_Rate | Trans\_Limit |

Fields definition:

|  |  |  |  |
| --- | --- | --- | --- |
| Fields | Data Type | Range | Constraints |
| Acct\_No | Int | 8 | Primary Key, not null, foreign key, refere to table Account(Acct\_No) |
| Insterest\_Rate | Numeric | 6 with 2decimal digits | Default value:0.00% |
| Trans\_Limit | Int | 2 | Default value: 0 |

1. LoanAccount table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Acct\_No | Loan\_Amt | Interest\_Rate | Term | Loan\_Description |

Fields definition:

|  |  |  |  |
| --- | --- | --- | --- |
| Fields | Data Type | Range | Constraints |
| Acct\_No | Int | 8 | Primary Key, not null |
| Loan\_Amt | Numeric | 15 with 2 decimal digits | Default value: 0.00 |
| Term | Numeric | 5 with 1 decimal digits | Default value:0.0 |
| Interest\_Rate | Numeric | 6 with 2 decimal digits | Default value:0.00 |
| Loan\_description | Varchar | 40 |  |

1. Transaction table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Trans\_ID | Acct\_No | Trans\_Date | Trans\_Type | Trans\_Amnt | Trans\_Comts |

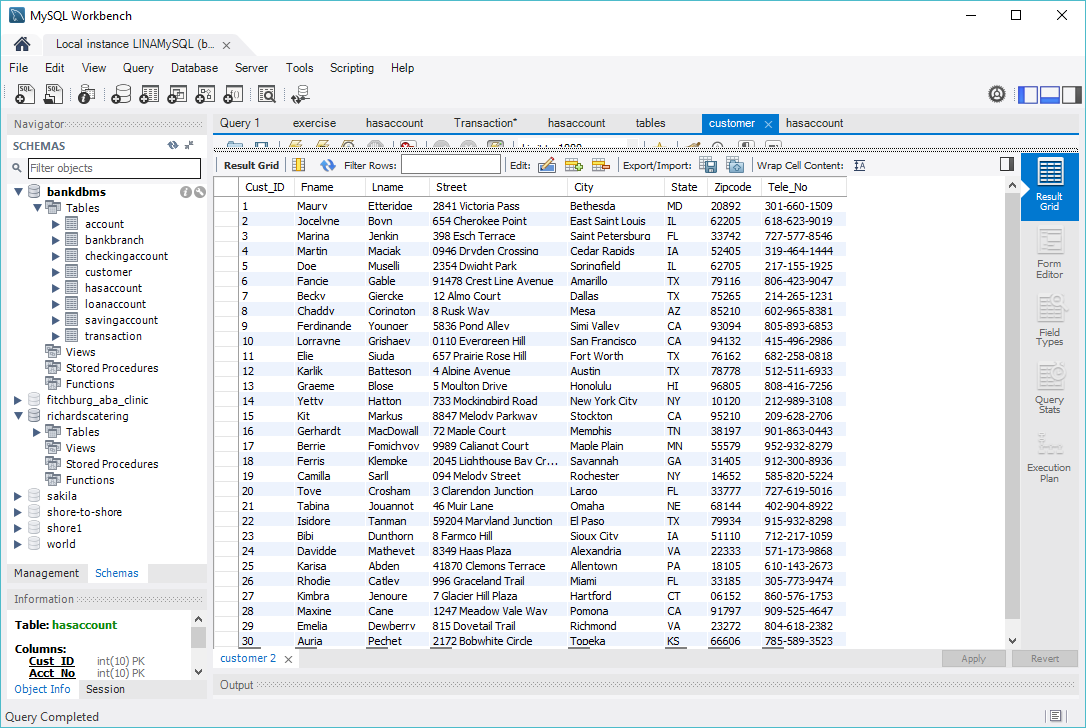
Fields definition:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Data Type | Range | Constraints |
| Trans\_ID | Int | 8 | Primary key, not null |
| Acct\_No | Int | 8 | Foreign key, reference table Aaccount(Acct\_No), not null |
| Trans\_Date | Date |  | Default value: current date |
| Trans\_Type | Varchar | 10 | Check in{‘Transfer’, ‘Deposit’,’Withdraw’} |
| Trans\_Amnt | Numberic | 8 with 2 decimal digits | Defualt value:0.00 |
| Trans\_Comts | Varchar | 40 |  |

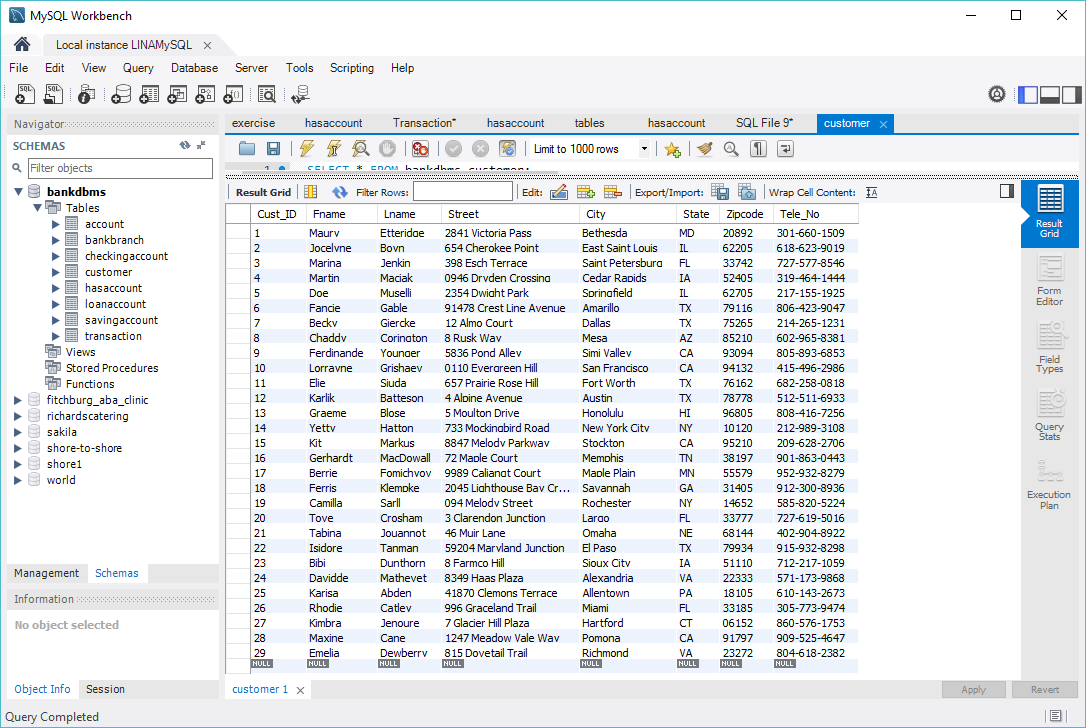
1. Implementation of the system
2. Drop a customer:

Query: delete from customer where Fname='Auria' and Lname='Pechet';

Before drop an existing customer:

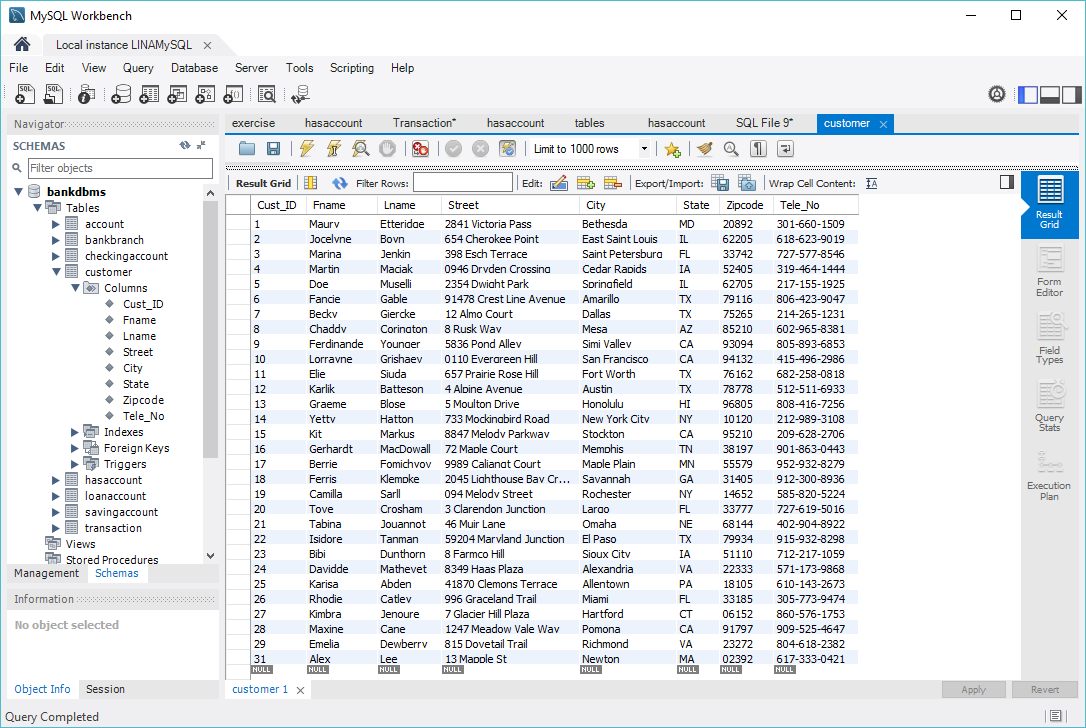


After drop an existing customer:



1. Add a new customer:

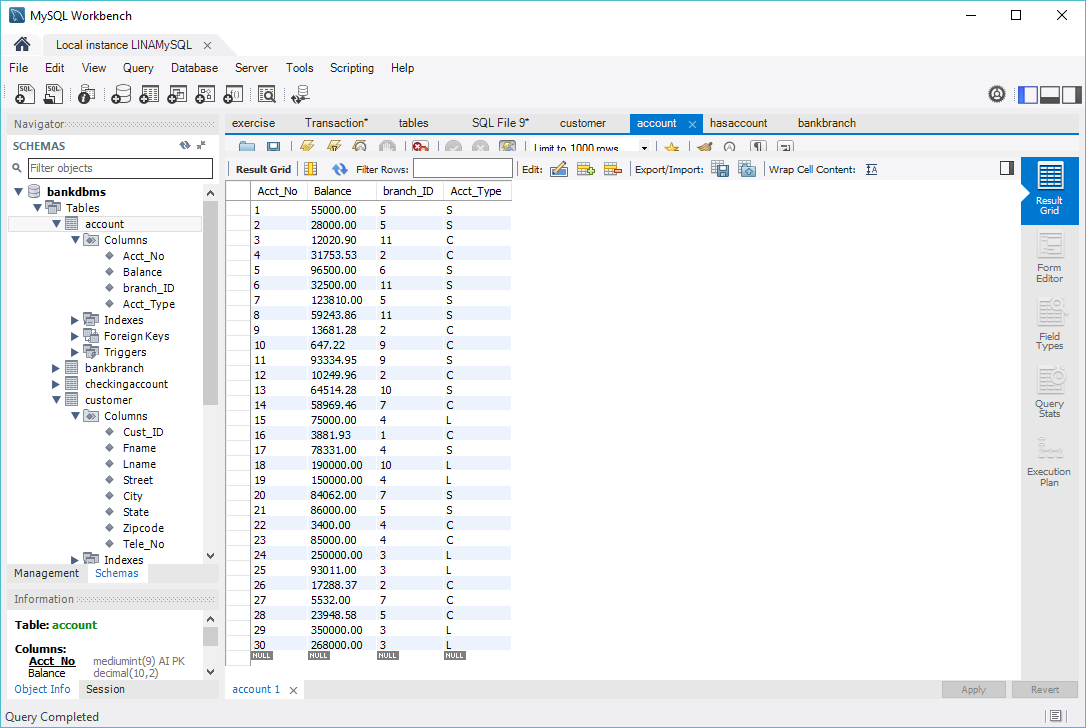
Query: insert into customer(fname, lname, Street, City, State, Zipcode, Tele\_No) values ('Alex','Lee','13 Mapple St', 'Newton', 'MA', '02392','617-333-0421');

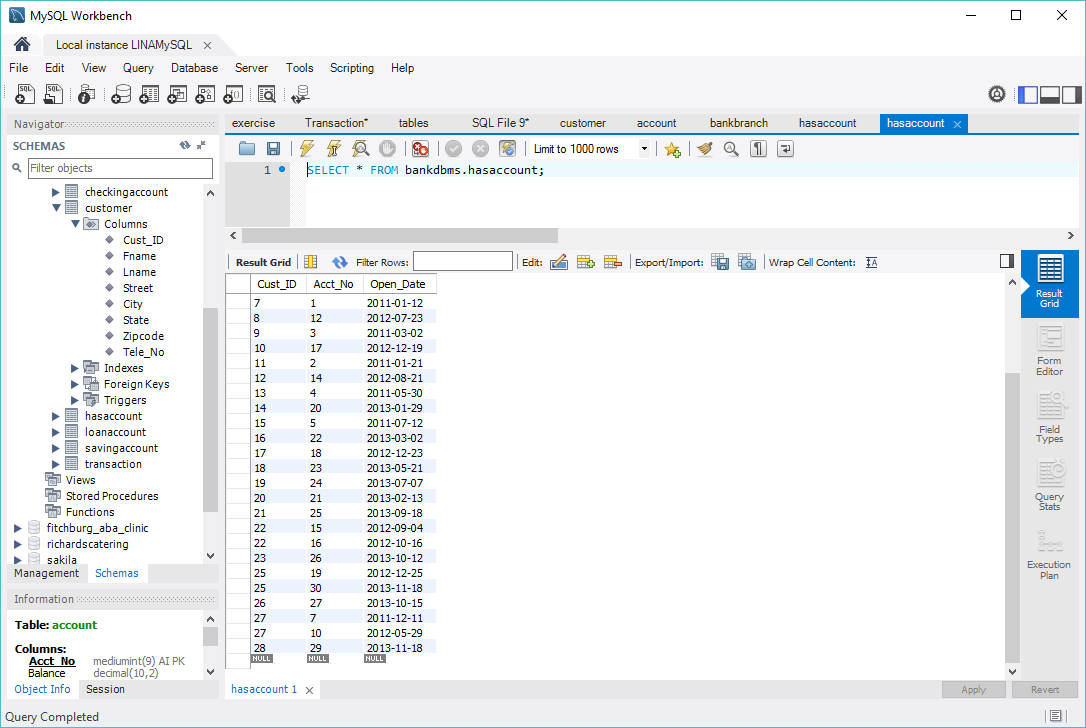


1. Open an account

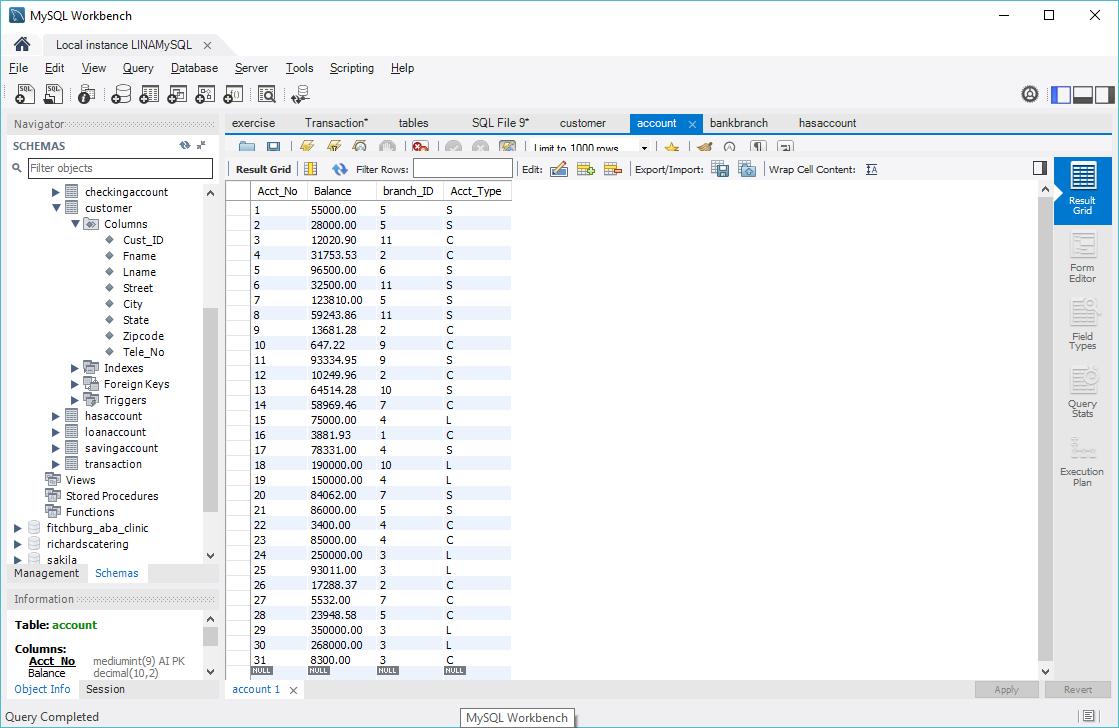
Query: insert into account(Acct\_No, Balance, Branch\_ID, Acct\_Type) values(31, 8300, 3, 'C');

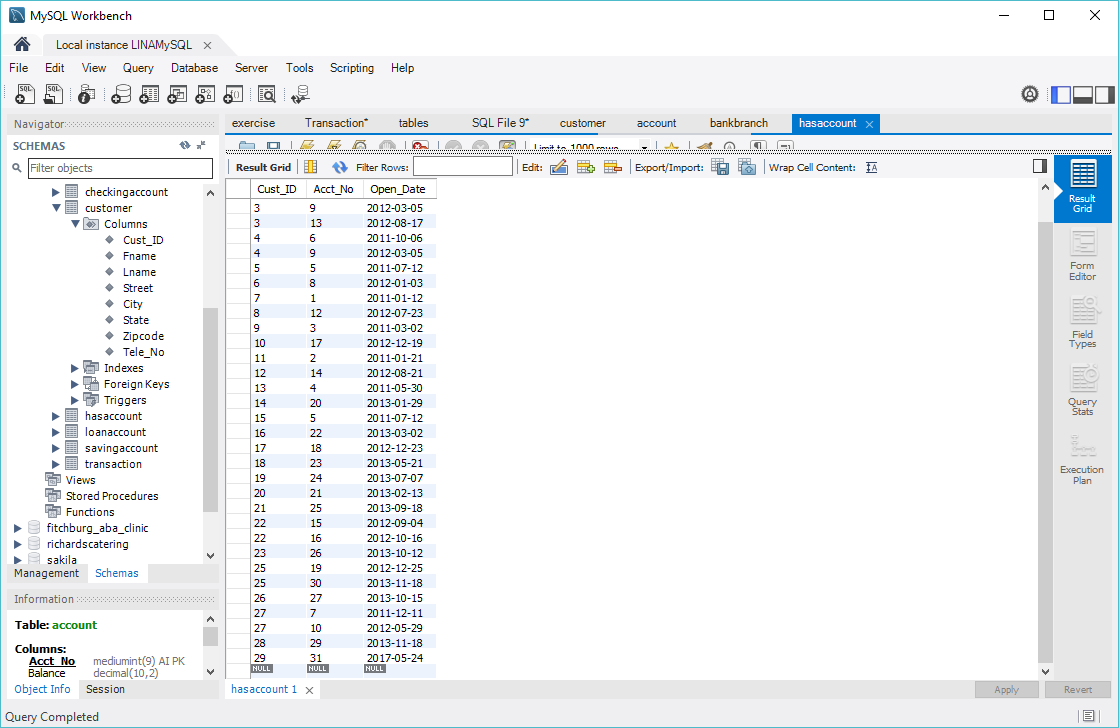
insert into hasaccount(Cust\_ID, Acct\_No, Open\_Date) values(29, 31, '2017-05-24');

before open account:



After opening account:



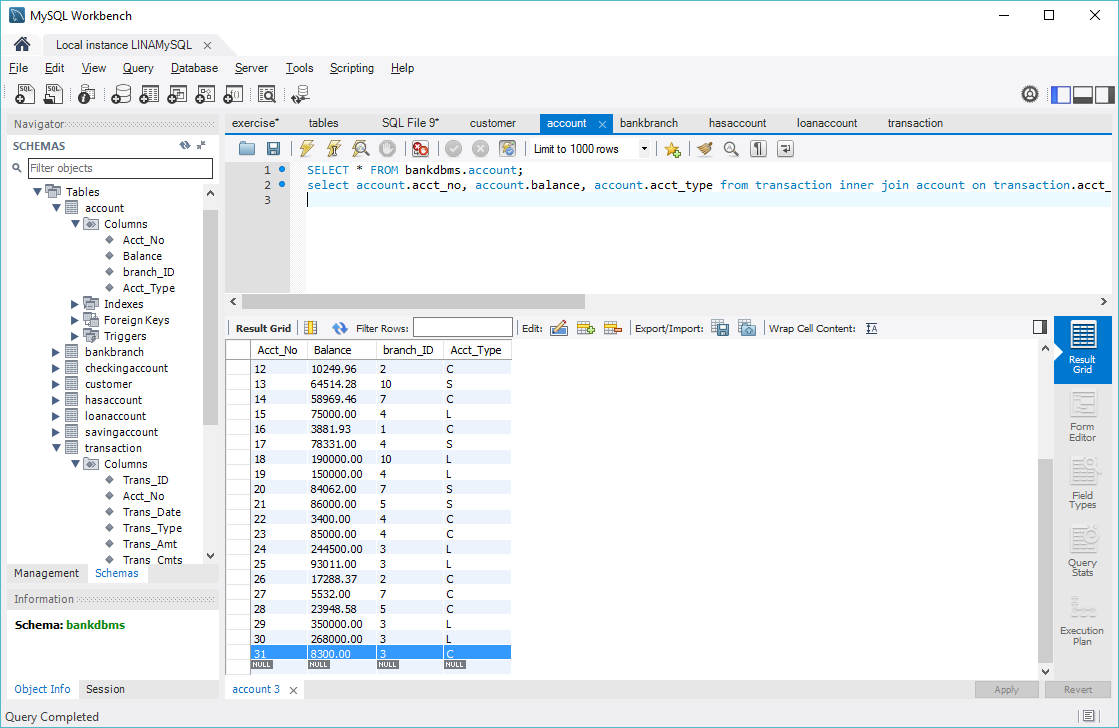


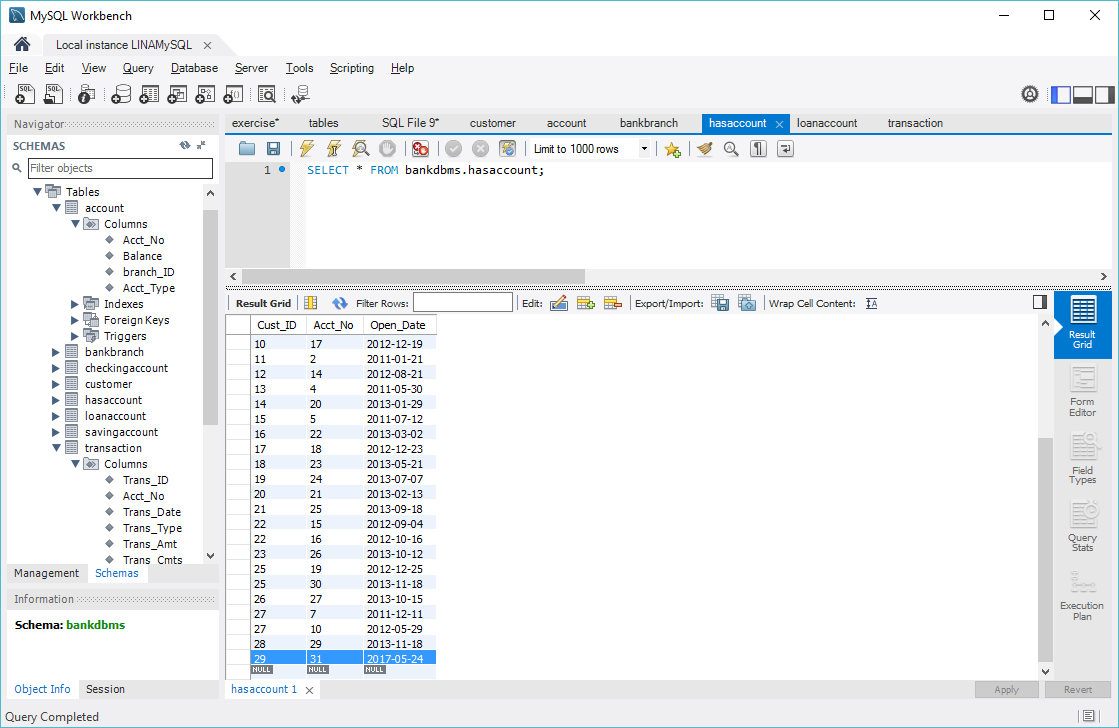
1. Close an account:

Queries: delete from account where Acct\_No=31;

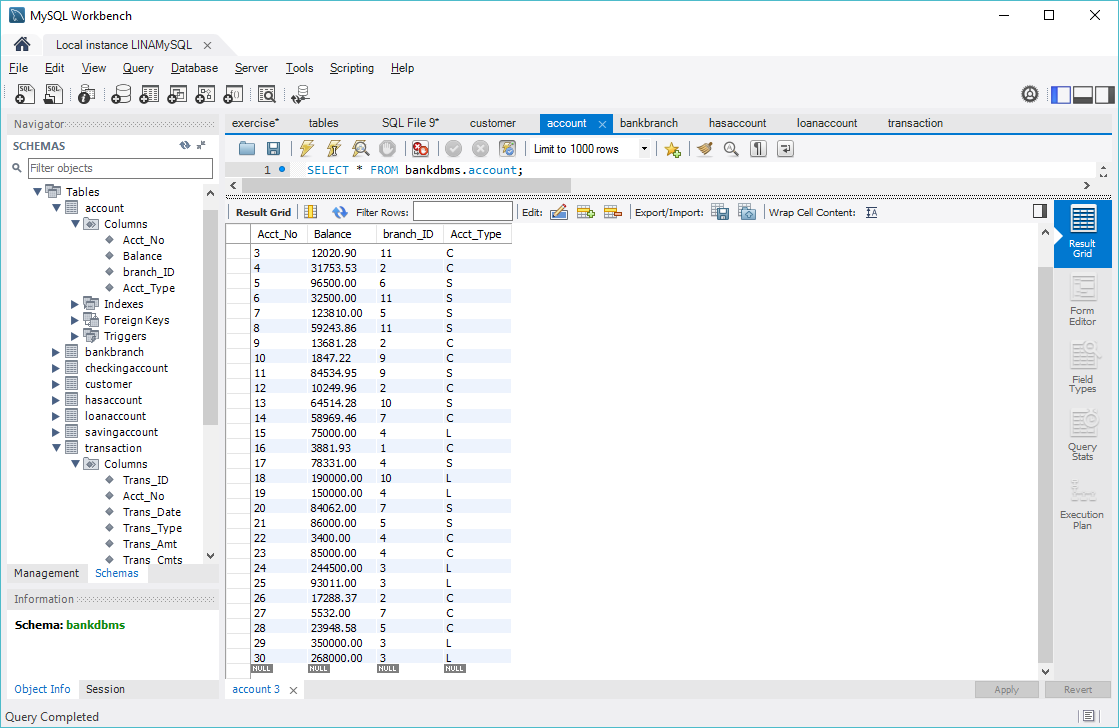
delete from hasaccount where cust\_ID=29 and acct\_No=31;

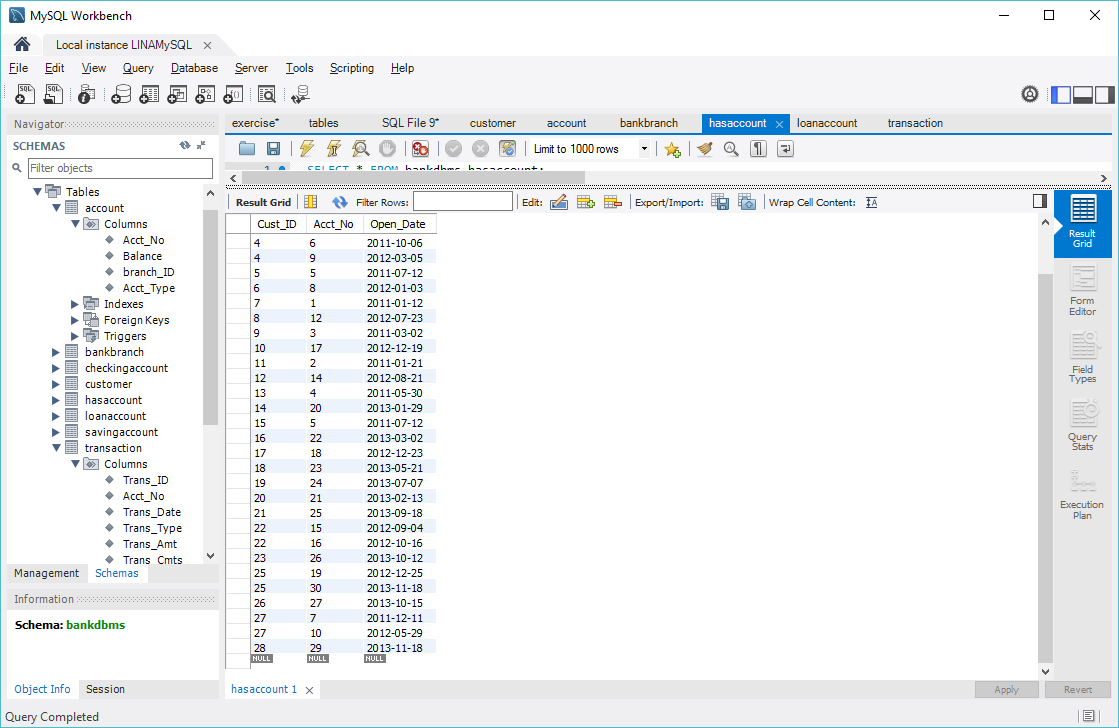
before closing an account:





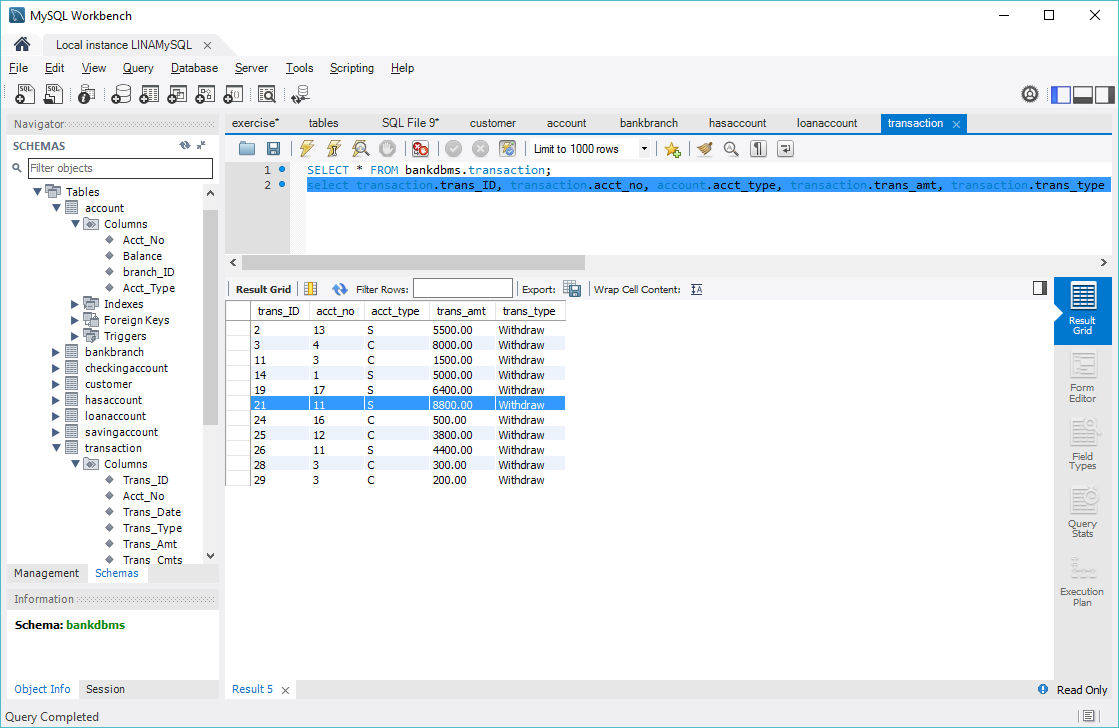
After closing an account:





1. Withdraw Transaction:

For example, implement the withdraw transaction with transaction ID 21:

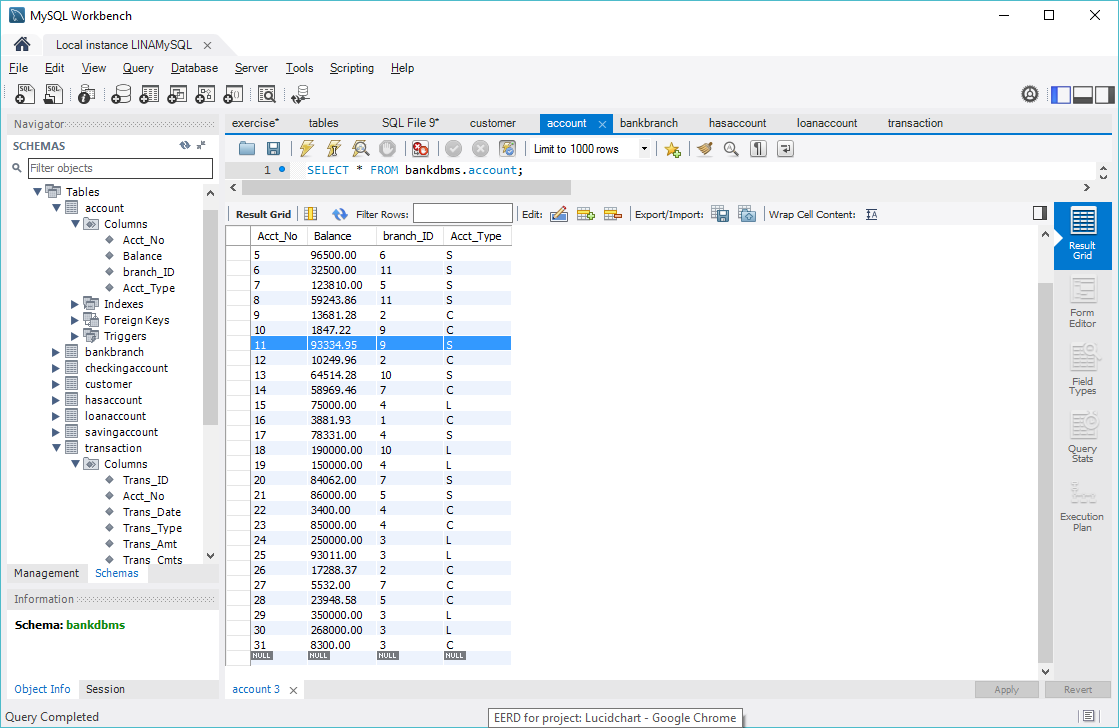


Queries: 1)select account.acct\_no,account.balance-transaction.trans\_amt as newbalance from transaction inner join account

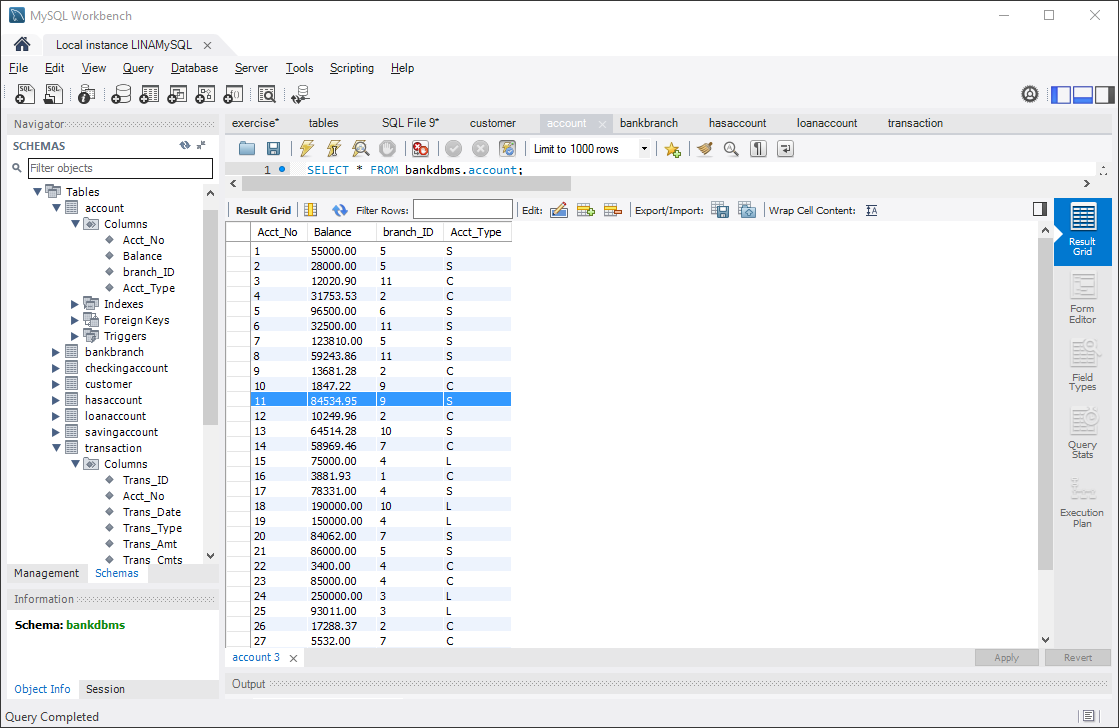
on transaction.Acct\_No=account.Acct\_No and transaction.Trans\_ID=21;

2)update account set balance=84534.95 where account.Acct\_No=11;

Before withdraw:

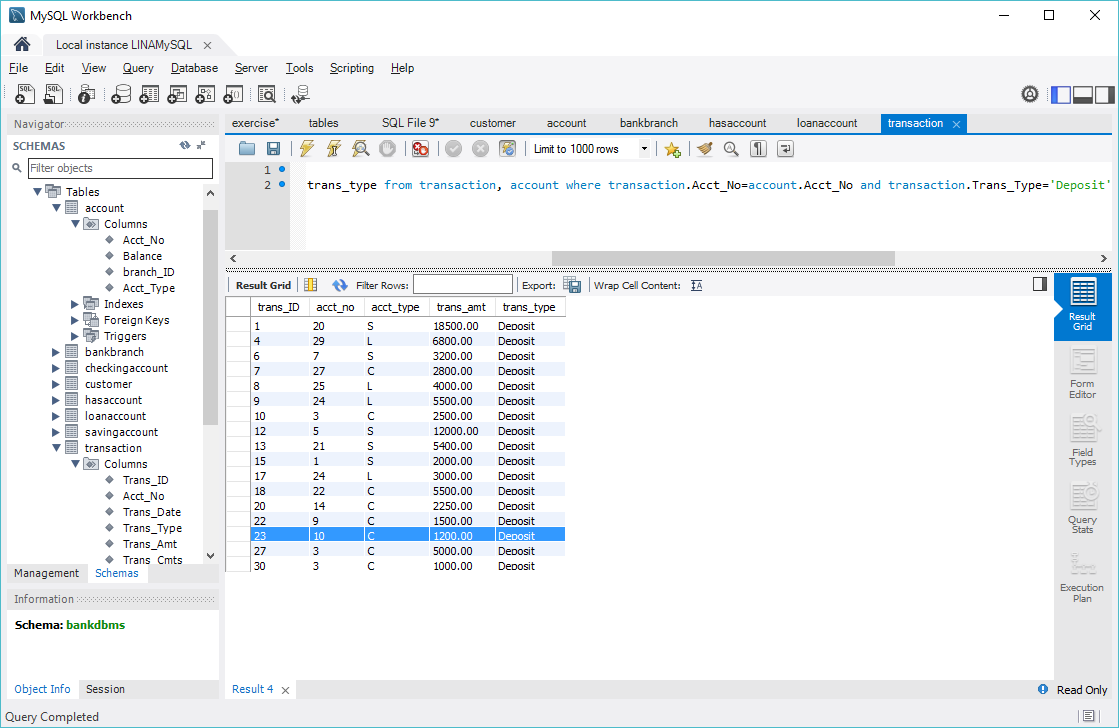


After withdraw:



1. Deposit Transaction:

Suppose implement an deposit transaction with transaction ID of 23



Queries :

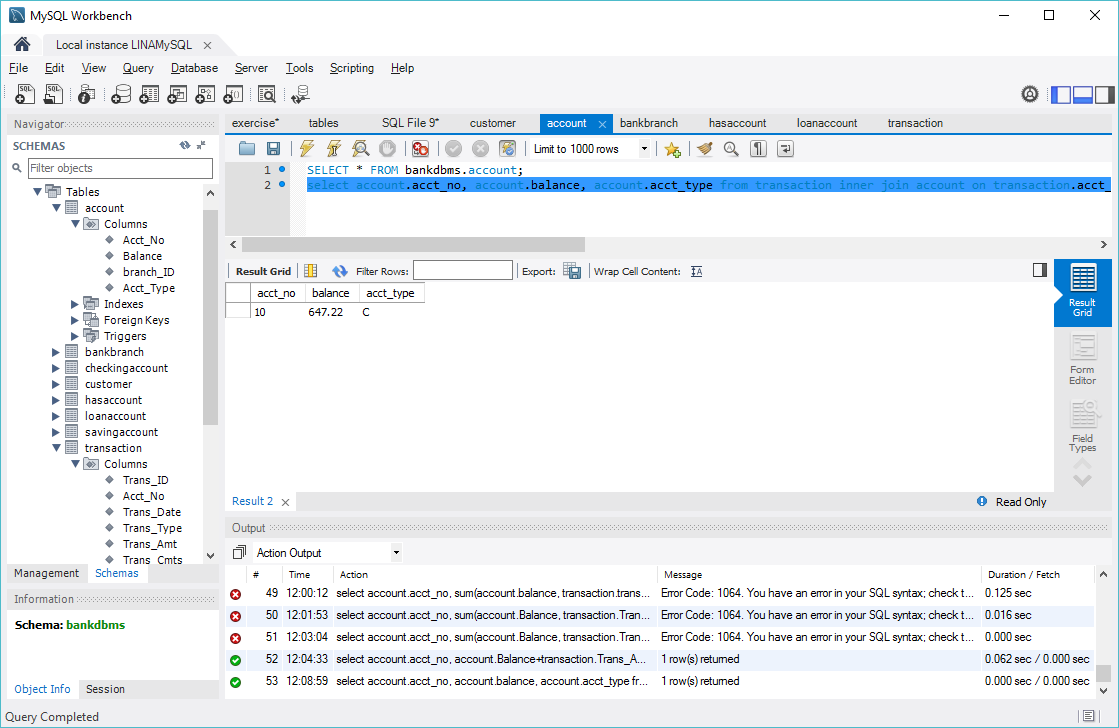
1. select account.acct\_no, account.Balance+transaction.Trans\_Amt as newbalance

from transaction, account

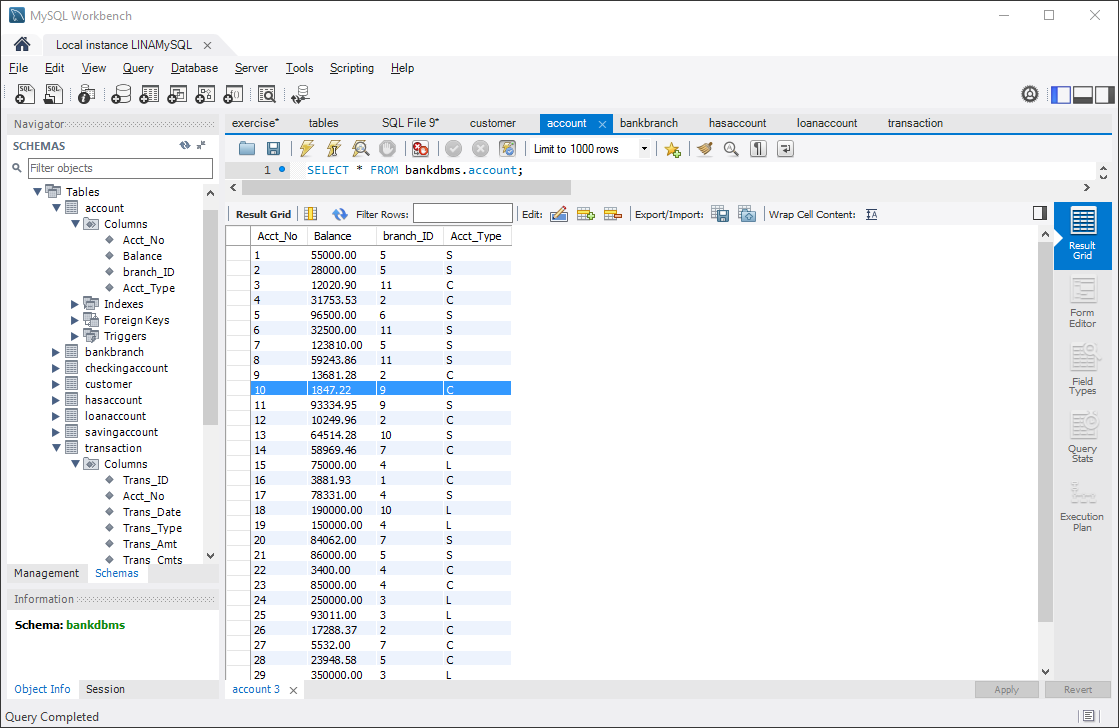
where transaction.acct\_no=account.acct\_no and transaction.trans\_id=23;

1. update account set balance=1847.22 where acct\_no=10;

before deposit transaction:

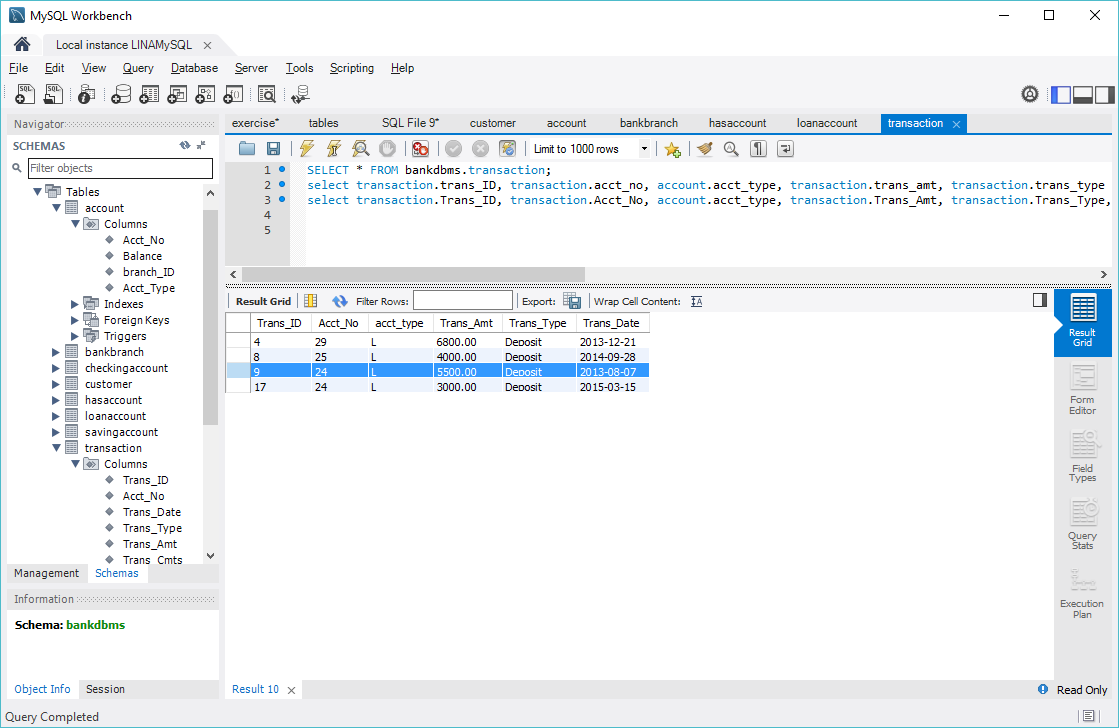


After deposit transaction:



1. Loan Payment Transaction:

Suppose implement deposit (make a loan payment) transaction (transaction ID: 9)on a loan account:

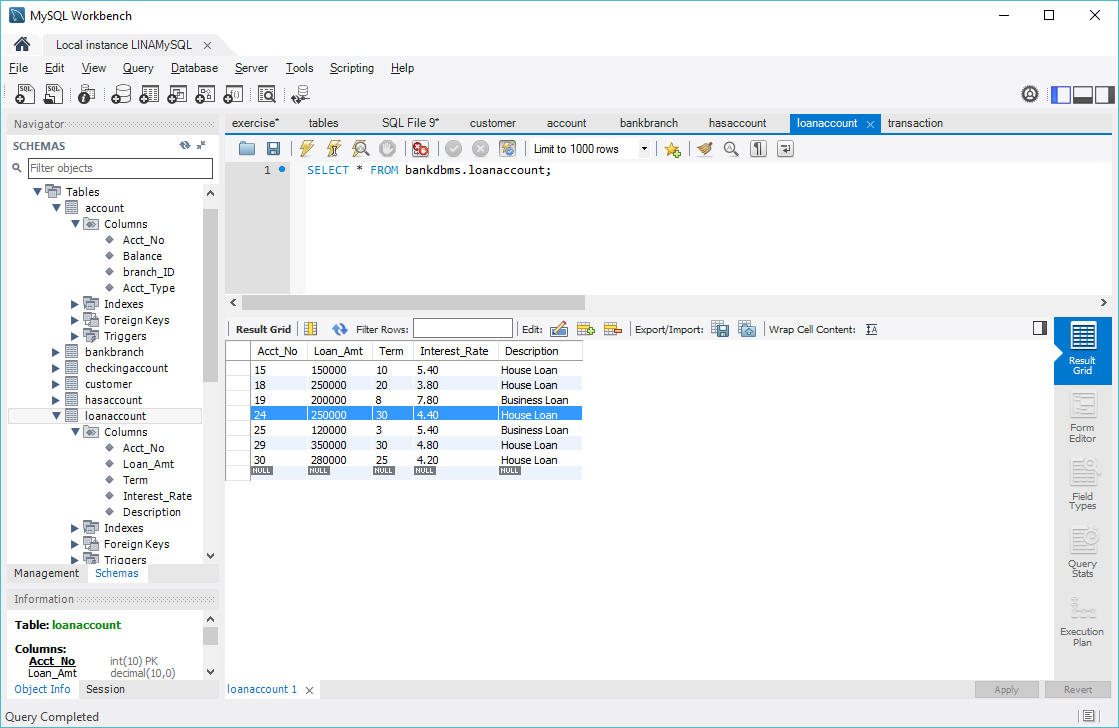


Queries :

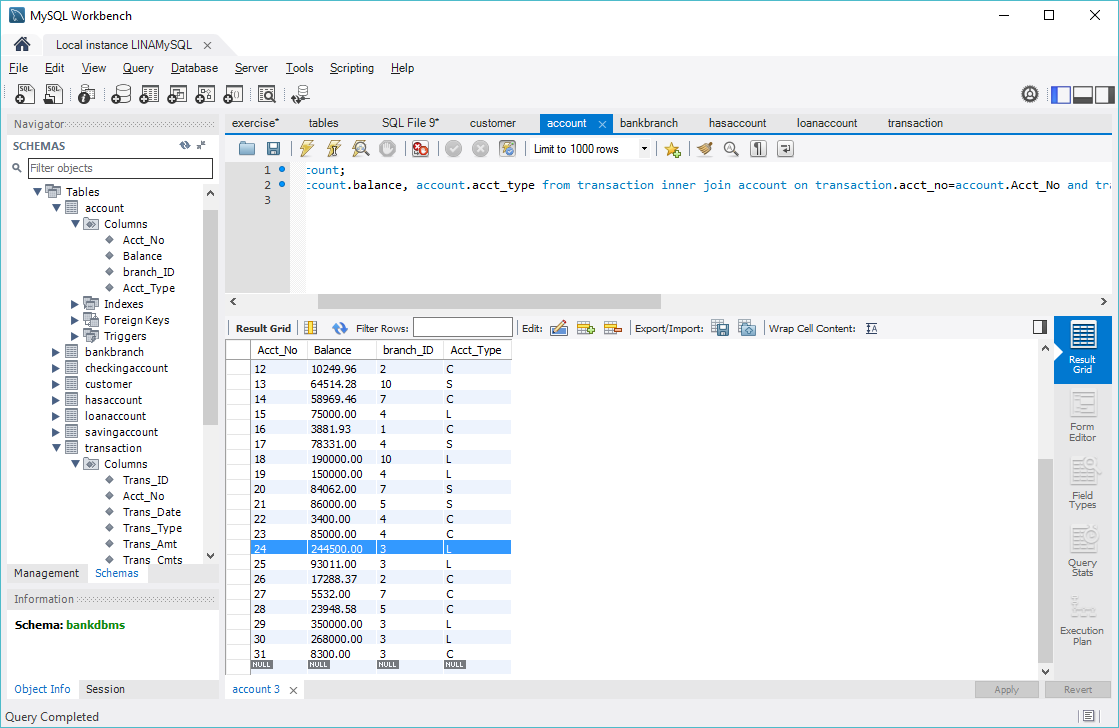
1)select account.acct\_no, account.balance-transaction.Trans\_Amt as newbalance from transaction inner join account on transaction.acct\_no=account.acct\_no and transaction.trans\_ID=9;

2)update account set balance=244500.00 where acct\_no=24;

Before loan payment:

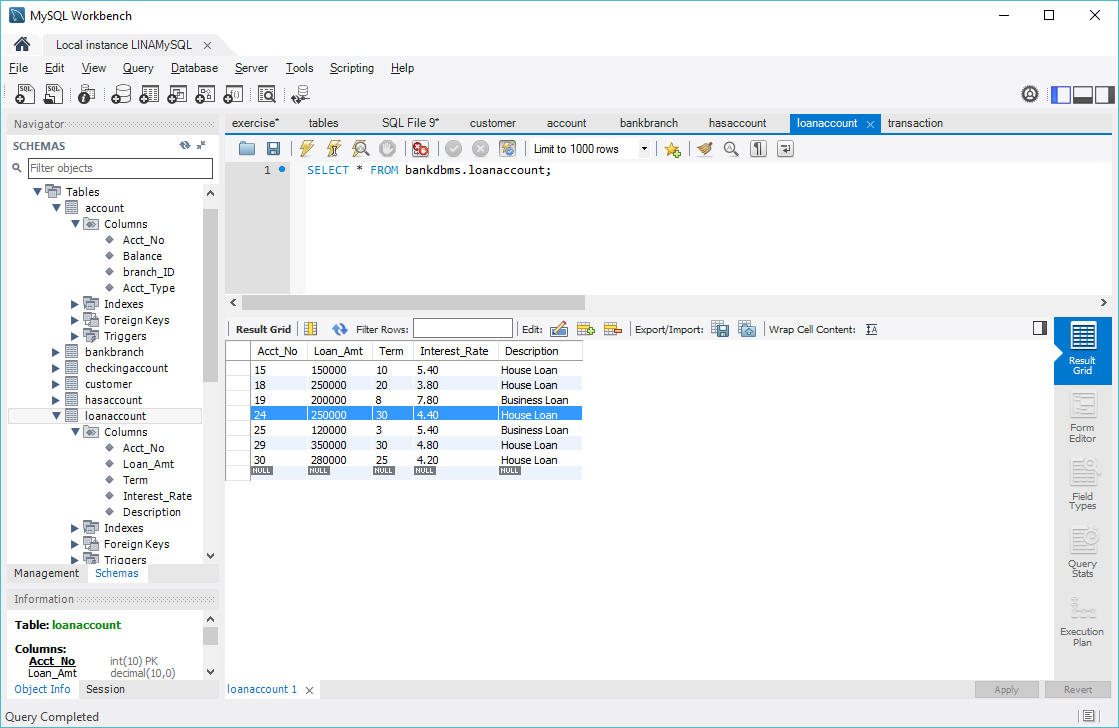


After loan payment of transaction 9:



1. Make payment schedule for loan Account:

Suppose make a payment schedule for the customer with the loan account 24:



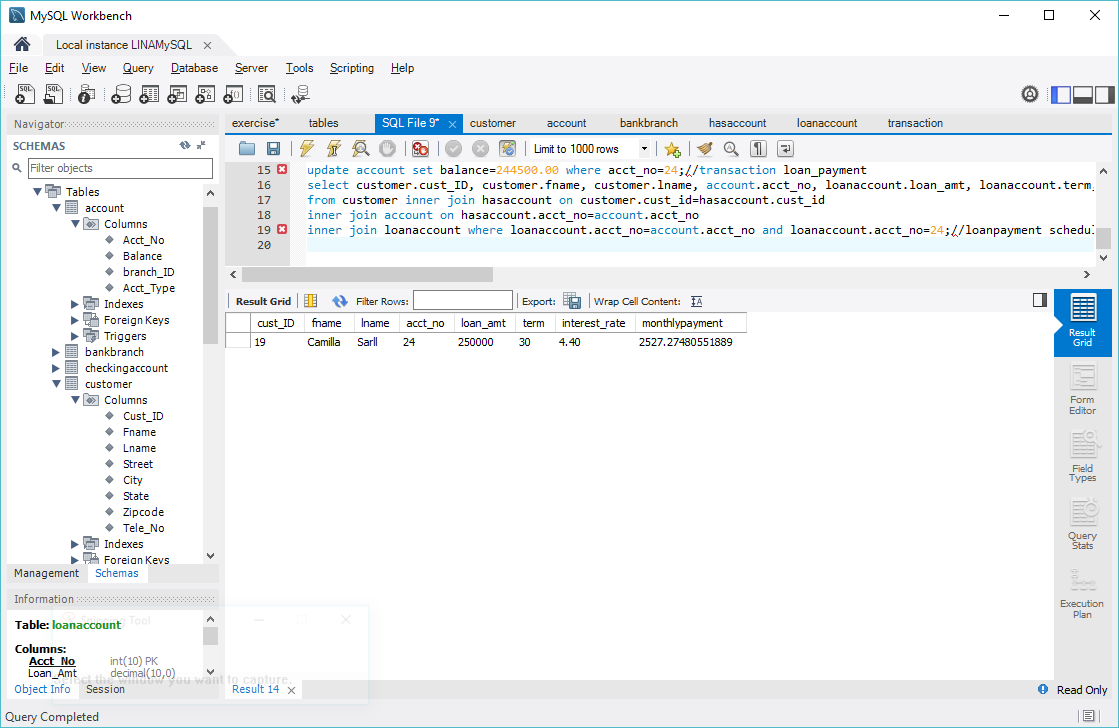
Query: select customer.cust\_ID, customer.fname, customer.lname, account.acct\_no, loanaccount.loan\_amt, loanaccount.term, loanaccount.interest\_rate, loanaccount.loan\_amt\*pow((1+loanaccount.interest\_rate\*0.01),loanaccount.term)/(loanaccount.term\*12) as monthlypayment

from customer inner join hasaccount on customer.cust\_id=hasaccount.cust\_id

inner join account on hasaccount.acct\_no=account.acct\_no

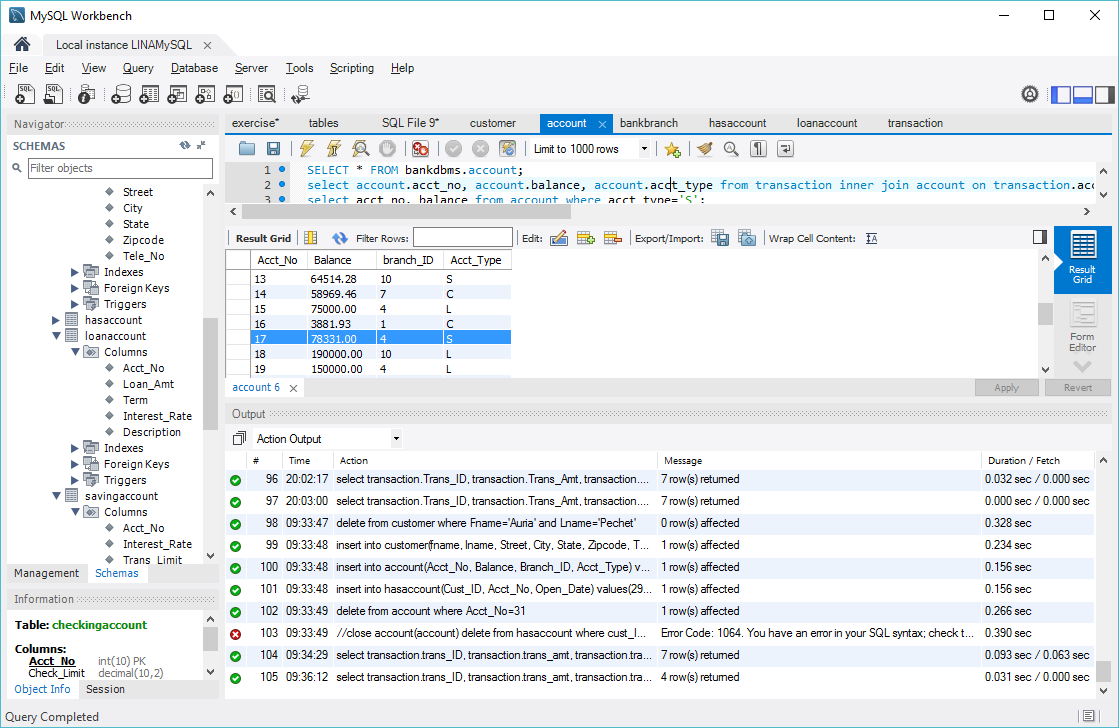
inner join loanaccount where loanaccount.acct\_no=account.acct\_no and loanaccount.acct\_no=24;

result:



1. Calculate yearly interest for saving account:

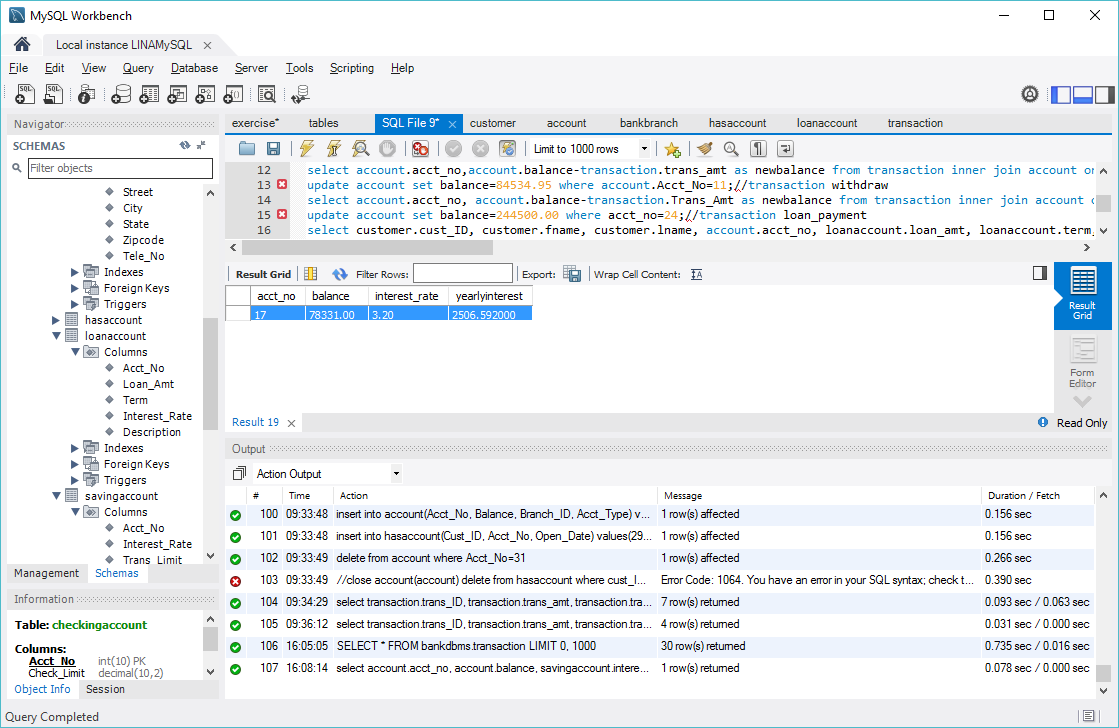
Suppose we need to know the yearly interest income of the saving account 17



Query: select account.acct\_no, account.balance, savingaccount.interest\_rate, account.balance\*savingaccount.interest\_rate\*0.01 as yearlyinterest

from account inner join savingaccount on account.acct\_no=savingaccount.acct\_no and account.acct\_no=17;

result:



1. List monthly transactions of an account:

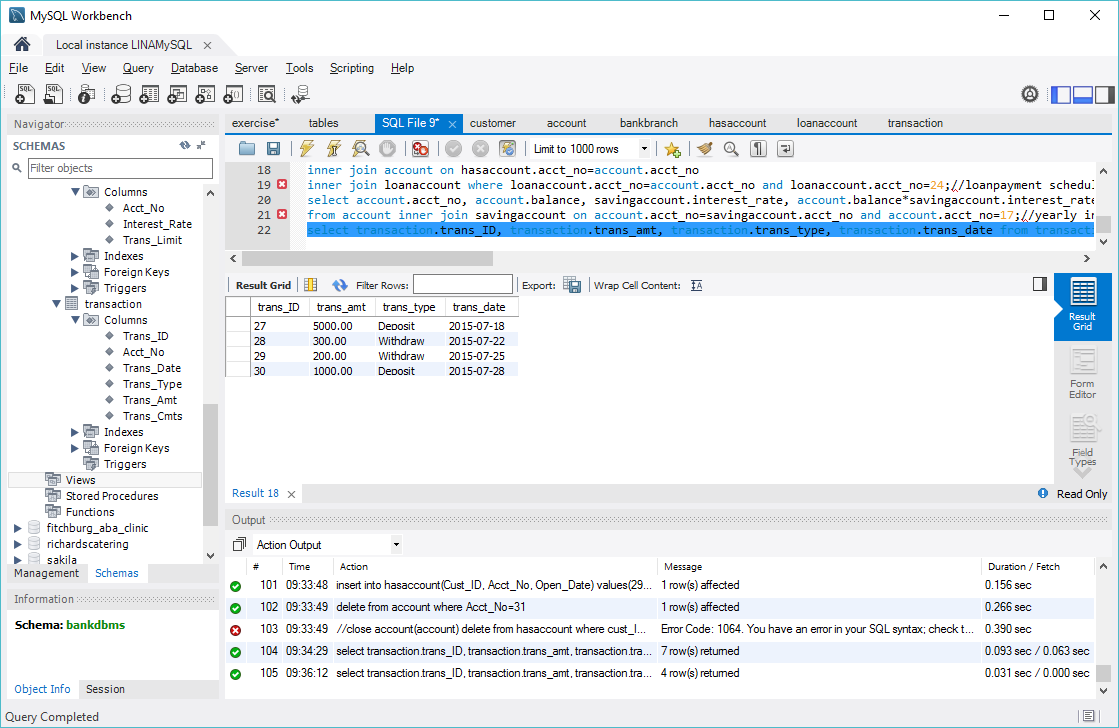
Suppose the customer Ferdinande Young wants to have a report on the transactions on his account during July 2015;

Query: select transaction.trans\_ID, transaction.trans\_amt, transaction.trans\_type, transaction.trans\_date from customer inner join hasaccount on customer.Cust\_ID=hasaccount.Cust\_ID and customer.lname='Younger'

inner join account on hasaccount.Acct\_No=account.Acct\_No

inner join transaction on transaction.acct\_no=account.acct\_no and transaction.trans\_date>'2015-07-01' and transaction.trans\_date<'2015-07-31'

Result:



1. Conclusion

In this project, a bank DBMS is designed, first the basic entities of system are proposed and their fields are analyzed during the project proposal; then the enhanced entity relationship diagram(EERD) of the system are provided to illustrate the relationships among the entities in the system.

Based on the provide EERD of the system, the relationships (tables) for each entity, the fields of each entity and the relationships among tables are carefully designed during phase of conceptual design.

In the phase of physical design of the system, the data type, range of variation and property of each field of the relations are given.

The last period of system design implements some basic functions of system with mySQL queries, such as add/drop customer, open/close account, make loan payment schedule for the customer, generate monthly transaction report for the customer, calculate the yearly interest for saving account.

The further work would focus on the interface design of system to collect the information and data by interaction with customer and implement the previous functions automatically; there are some other functions that should be explored to improve the power of the system.