

Lab Session 6: Advanced SQL

In this lab, we are going to try out more complicated queries. As usual, we will use our simple bank database as an example.

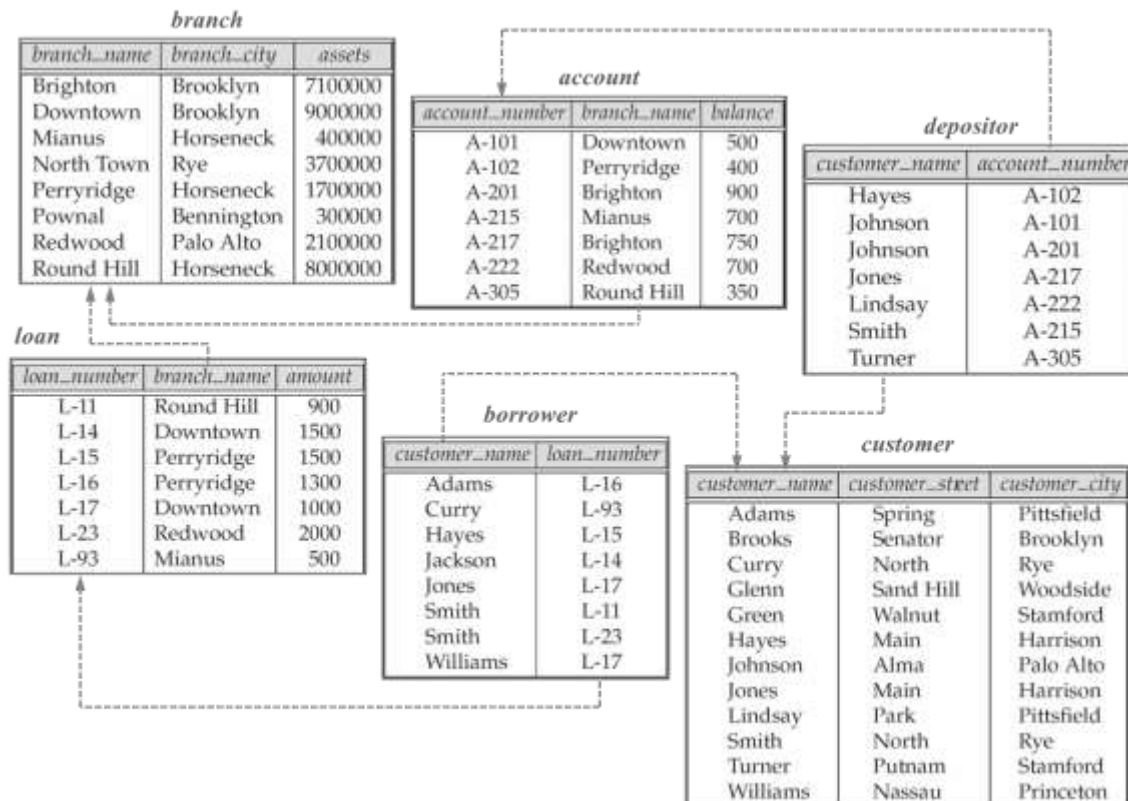


Figure 1. Bank database

Part I: Customers with accounts in every branch of Brooklyn

In the slides of this course, we have seen how to find the customers who have accounts in every branch in the city of Brooklyn.

1. Write a query to show the branches in the city of Brooklyn. (Show the branch names.)
2. Expand the previous query to show the accounts that belong to those branches. (Show branch names and account numbers.)
3. Expand the previous query to show also the customers who own those accounts. (Show branch names, account numbers, and customer names.)

4. In the result of the previous query, can you find any customer who has accounts in every branch of Brooklyn?
5. Execute the following query to find those customers directly:

```
select distinct customer_name
from depositor as d
where not exists (
    select branch_name
    from branch as b
    where branch_city = 'Brooklyn'
    and branch_name not in (
        select branch_name
        from account as a, depositor as d2
        where a.account_number = d2.account_number
        and d2.customer_name = d.customer_name));
```

6. The query above returns a customer from table depositor (d) if and only if the customer obeys a certain condition. Can you explain this condition in your own words?
(Hint: the third **select** in the query returns the branches where that customer has accounts.)
7. The query in step 5 has a double negation with **not exists** and **not in**. Rewrite the query using **not exists** twice (i.e. replace the **not in** by a second **not exists**).

Part II: Customers with accounts in every branch of the city where they live

We are now interested in finding the customers who have accounts in every branch of the same city where they live.

8. Can you formulate this question using a double negation?
(Hint: if a customer has accounts in every branch of the city where he/she lives, then we cannot find a branch in that city where the customer does not have an account.)
9. Change the query from step 5 so that, instead of selecting the branches in the city of Brooklyn, you select the branches in the same city where the customer lives.
(Hint: in the first **select**, you may want to join the tables depositor and customer so that you can access the customer city in the second **select**.)

When you run the query, you may get the following result:

```

+-----+
| customer_name |
+-----+
| Hayes         |
| Jones         |
| Lindsay       |
| Turner        |
+-----+

```

10. Run the following query to check where these customers live, and where they have accounts:

```

select c.customer_name, c.customer_city, b.branch_city
from customer as c, depositor as d, account as a, branch as b
where c.customer_name = d.customer_name
      and d.account_number = a.account_number
      and a.branch_name = b.branch_name
      and c.customer_name in ('Hayes', 'Jones', 'Lindsay', 'Turner');

```

In the result of this query, you will see that the cities (customer_city and branch_city) do not match. Then how is it possible that those customers appear in the result of the previous query? What is going on?

(Hint: in the database, you will find that there no branches in the cities where these customers live, so we cannot find a branch in that city where the customer does not have an account.)

11. Improve the query you wrote in step 9 to make sure that you get the expected result:

```
Empty set (0.00 sec)
```

i.e. there are no customers who have accounts in every branch in the same city where they live.

(Hint: you have to enforce another condition, which is that there must exist branches in the city where the customer lives.)

Part III: Date and time

In the slides of this course, we have seen how to work with dates, times, and timestamps.

12. Use the following instructions to create a database about movies and tickets in a movie theater:

```
create table movie (  
    title varchar(255),  
    year year,  
    length time,  
    primary key(title, year) );  
create table ticket (  
    number integer,  
    showdate date,  
    showtime time,  
    title varchar(255),  
    year year,  
    primary key(number),  
    foreign key(title, year) references movie(title, year) );
```

13. Use the following instructions to insert some records in these tables:

```
insert into movie values ('Back to the Future', 1985, '01:56:00');  
insert into movie values ('Back to the Future II', 1989, '01:48:00');  
insert into movie values ('Back to the Future III', 1990, '01:58:00');  
insert into ticket values (123456, '2015-10-26', '18:25:00', 'Back to the Future', 1985);  
insert into ticket values (123457, '2015-10-26', '21:15:00', 'Back to the Future', 1985);  
insert into ticket values (123458, '2015-10-27', '00:05:00', 'Back to the Future', 1985);
```

14. Run the following query to discover how much time you have left before each of these movie sessions:

```
select showdate, showtime,  
        datediff(showdate, current_date) as daysuntil,  
        timediff(showtime, current_time) as timeuntil  
from ticket;
```

15. Run the following query to determine at what time the movie will end, including 20 minutes of trailers and advertisements:

```
select t.title, t.showtime as begins,  
        addtime(addtime(t.showtime, m.length), sec_to_time(20*60)) as ends  
from ticket as t, movie as m  
where t.title = m.title and t.year = m.year;
```

16. Check out the documentation for these and other date and time functions:

<https://dev.mysql.com/doc/refman/5.1/en/date-and-time-functions.html>