KOREA UNIVERSITY
DATAX LAB

# Chapter 18 - Lab

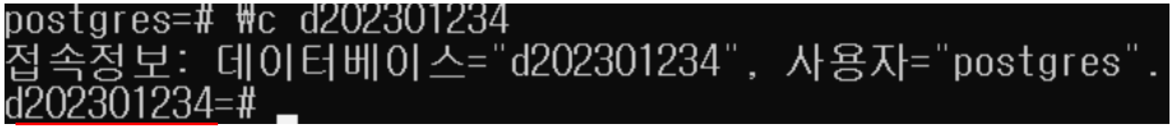
# Concurrency Control

## Deadlock

- Two (or more) transactions each hold locks that the other wants
  - For example, if transaction 1 acquires an exclusive lock on table A and then tries to acquire an exclusive lock on table B, while transaction 2 has already exclusive-lock table B and now wants an exclusive lock on table A, then neither on can proceed.
- PostgreSQL automatically detects deadlock situations and resolves them
  - Aborting one of the transactions involved, allowing the other(s) to complete

## Lab Setup

- Execute PostgreSQL SQL Shell (psql) and login your database
  - Server [localhost]: Press the enter key
  - Database [postgres]: Press the enter key
  - Port [5432]: Press the enter key
  - Username [postgres]: Press the enter key
  - Password for user postgres: Type your own password
  - \c d{StudentID}

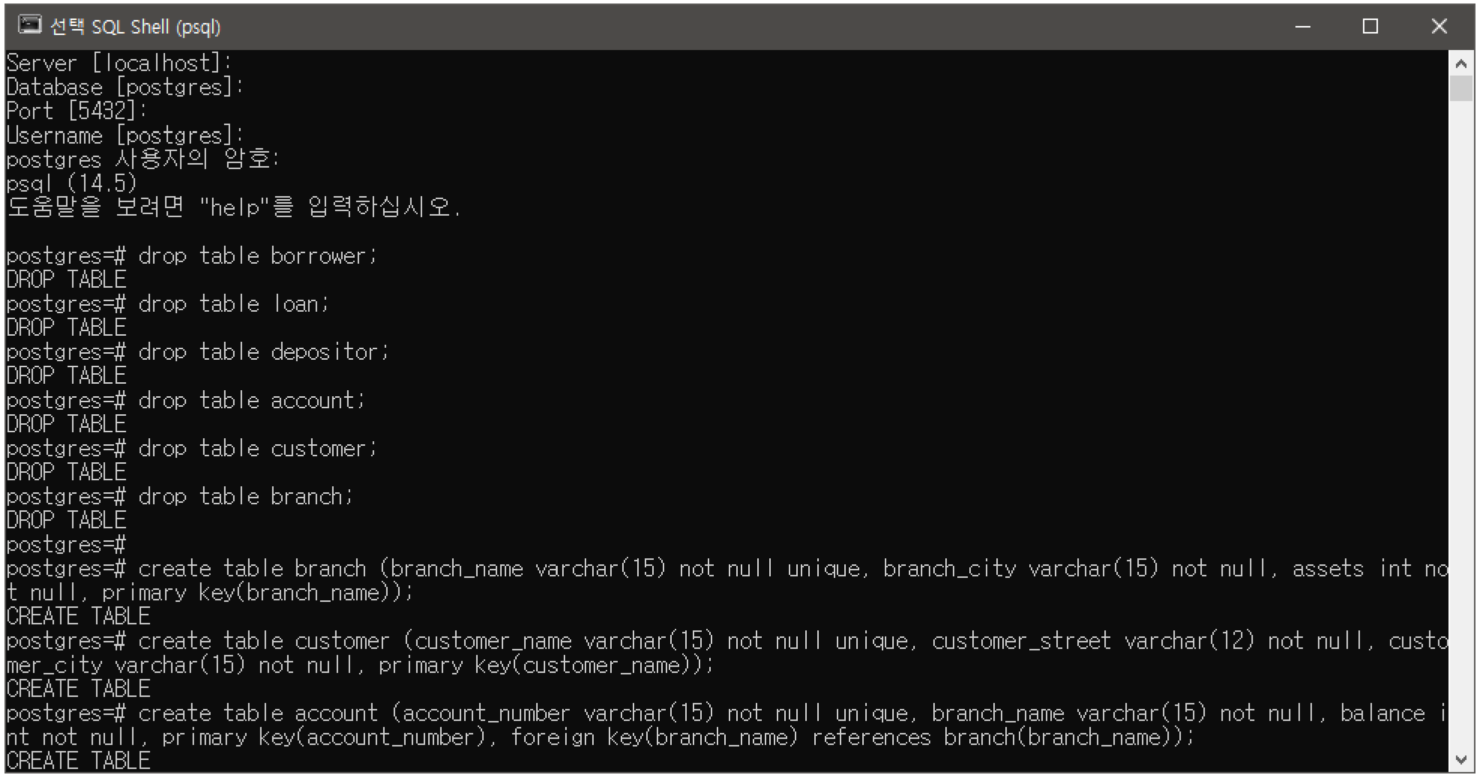


```
postgres=# ₩c d202301234
접속정보: 데이터베이스="d202301234", 사용자="postgres".
d202301234=# _
```

Your answers must be displayed along with your student ID.

# Lab Setup

- Download the “bank.txt” file from blackboard
- Copy & paste all the contents in the “bank.txt” file on PostgreSQL
  - If you want to reset database, just copy & paste again



```
Server [localhost]:
Database [postgres]:
Port [5432]:
Username [postgres]:
postgres 사용자의 암호:
psql (14.5)
도움말을 보려면 "help"를 입력하십시오.

postgres=# drop table borrower;
DROP TABLE
postgres=# drop table loan;
DROP TABLE
postgres=# drop table depositor;
DROP TABLE
postgres=# drop table account;
DROP TABLE
postgres=# drop table customer;
DROP TABLE
postgres=# drop table branch;
DROP TABLE
postgres=#
postgres=# create table branch (branch_name varchar(15) not null unique, branch_city varchar(15) not null, assets int no
t null, primary key(branch_name));
CREATE TABLE
postgres=# create table customer (customer_name varchar(15) not null unique, customer_street varchar(12) not null, custo
mer_city varchar(15) not null, primary key(customer_name));
CREATE TABLE
postgres=# create table account (account_number varchar(15) not null unique, branch_name varchar(15) not null, balance i
nt not null, primary key(account_number), foreign key(branch_name) references branch(branch_name));
CREATE TABLE
```

# “bank” Database Schema

- branch (branch_name, branch_city, assets)
- customer (customer_name, customer_street, customer_city)
- account (account_number, branch_name, balance)
- depositor (customer_name, account_number)
- loan (loan_number, branch_name, amount)
- borrower (customer_name, loan_number)

※ Be careful regarding the primary-key and foreign-key constraints!

(e.g. No customers have the same name, ….)

## Exercise 1

a. Generate a deadlock from one table
    - Row-level lock

b. Prevent a deadlock from one table using ‘LOCK TABLE’ statement

# Exercise 2

a. Generate a deadlock from two tables
    - Row-level lock
b. Generate a deadlock from two tables with ‘LOCK TABLE’ statement
    - Table-level lock
c. Use ‘LOCK TABLE’ statement to implement Two-Phase Locking from two tables

## Homework

- Complete today’s practice exercises
- Write your queries and take screenshots of execution results
- Submit your report on blackboard
  - 10:29:59, December 17th, 2024
  - **Only PDF files** are accepted
  - **No late submission**

# End of Lab