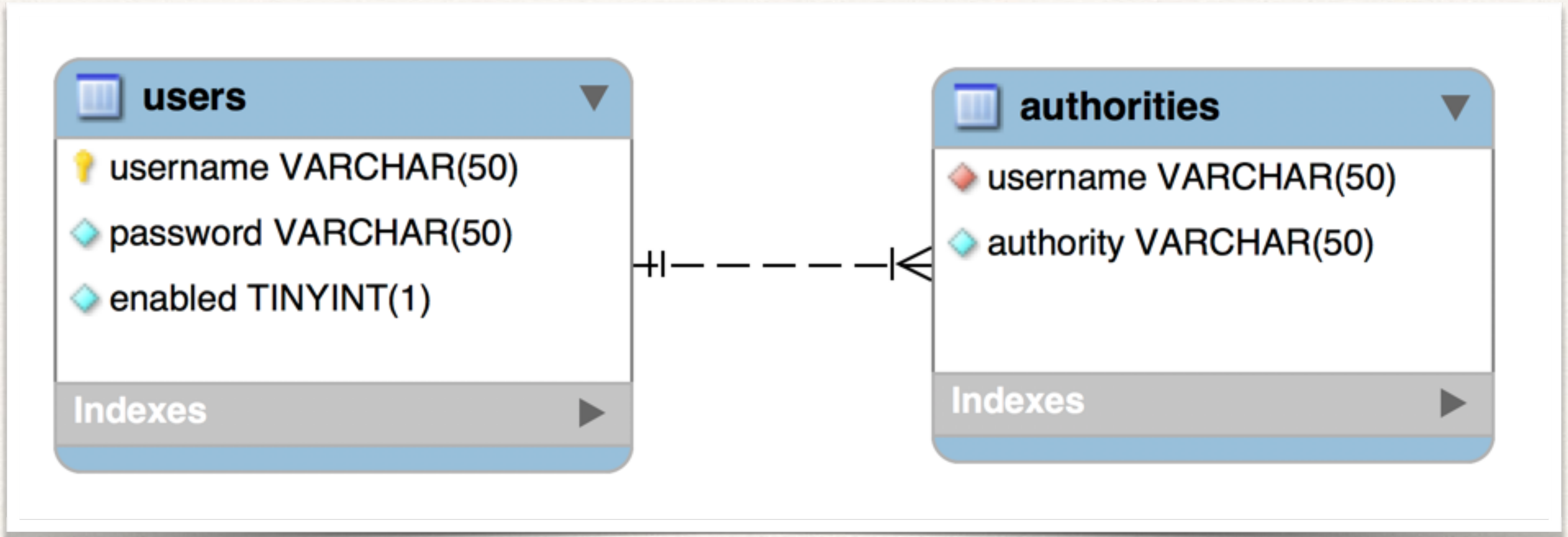
# Spring Security Custom Tables

## Default Spring Security Database Schema

# Default Spring Security Database Schema



## Custom Tables

## Custom Tables

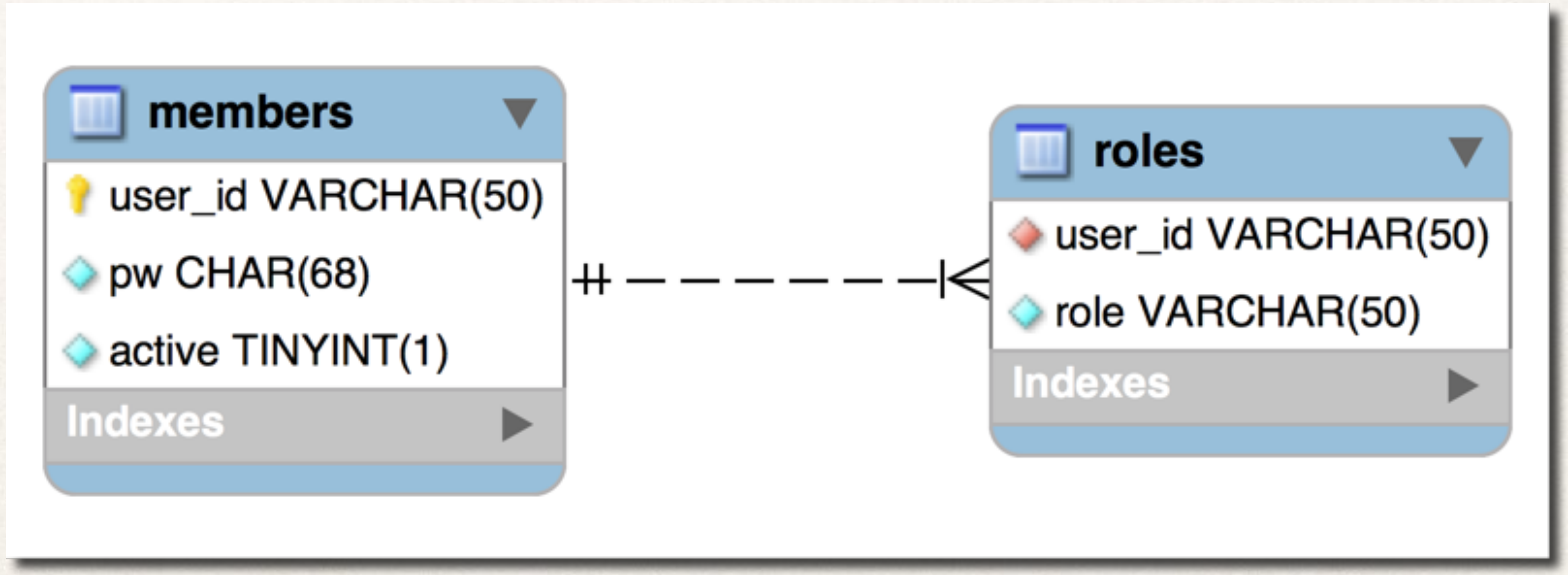
- What if we have our own custom tables?

## Custom Tables

- What if we have our own custom tables?
- Our own custom column names?

# Custom Tables

- What if we have our own custom tables?
- Our own custom column names?



# For Security Schema Customization

- Tell Spring how to query your custom tables
- Provide query to find user by user name
- Provide query to find authorities / roles by user name

## Development Process

Step-By-Step

# Development Process

Step-By-Step

1. Create our custom tables with SQL

# Development Process

Step-By-Step

1. Create our custom tables with SQL

2. Update JDBC properties file to point to new database schema

# Development Process

Step-By-Step

1. Create our custom tables with SQL

2. Update JDBC properties file to point to new database schema

3. Update Spring Security Configuration

## Development Process

Step-By-Step

1. Create our custom tables with SQL

2. Update JDBC properties file to point to new database schema

3. Update Spring Security Configuration
    - Provide query to find user by user name

## Development Process

Step-By-Step

1. Create our custom tables with SQL

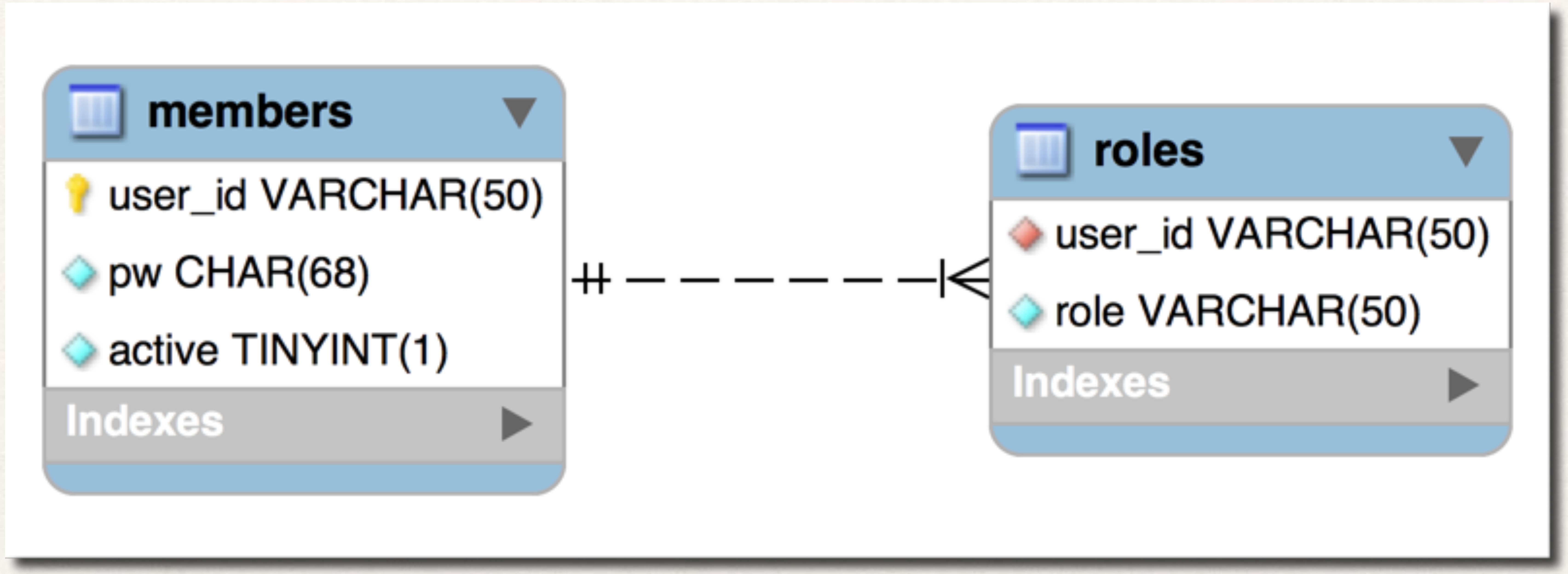
2. Update JDBC properties file to point to new database schema

3. Update Spring Security Configuration

    - Provide query to find user by user name

    - Provide query to find authorities / roles by user name

# Step 1: Create our custom tables with SQL



## Step 2: Update JDBC properties file to point to new database schema

**File: src/main/resources/persistence-mysql.properties**

```
#
# JDBC connection properties
#
jdbc.driver=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/spring_security_demo_bcrypt_custom?useSSL=false
jdbc.user=springstudent
jdbc.password=springstudent

#
# Connection pool properties
#
connection.pool.initialPoolSize=5
connection.pool.minPoolSize=5
connection.pool.maxPoolSize=20
connection.pool.maxIdleTime=3000
```

## Step 2: Update JDBC properties file to point to new database schema

**File: src/main/resources/persistence-mysql.properties**

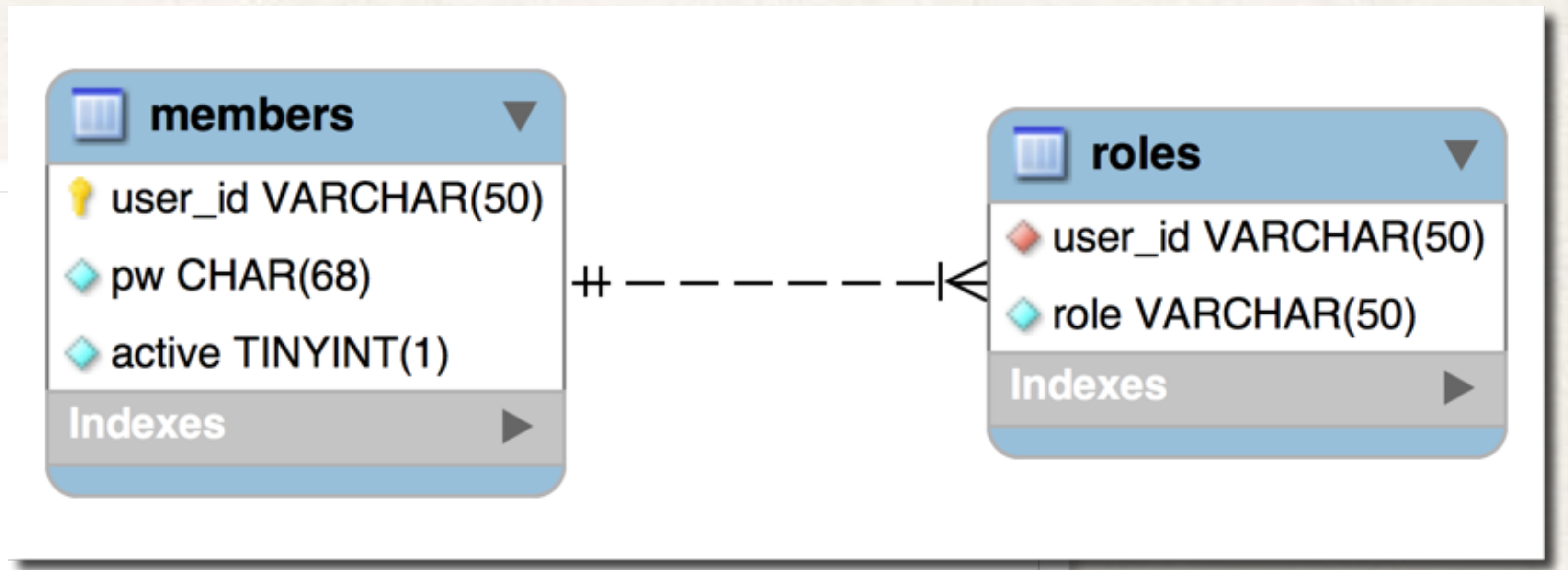
```
#
# JDBC connection properties
#
jdbc.driver=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/spring_security_demo_bcrypt_custom?useSSL=false
jdbc.user=springstudent
jdbc.password=springstudent

#
# Connection pool properties
#
connection.pool.initialPoolSize=5
connection.pool.minPoolSize=5
connection.pool.maxPoolSize=20
connection.pool.maxIdleTime=3000
```

# Step 3: Update Spring Security Configuration

```
@Configuration
@EnableWebSecurity
public class DemoSecurityConfig extends WebSecurityConfigurerAdapter {

   @Autowired
   private DataSource securityDataSource;

   @Override
   protected void configure(AuthenticationManagerBuilder auth) throws Exception {

      auth.jdbcAuthentication().dataSource(securityDataSource)

           .usersByUsernameQuery("select user_id, pw, active from members where user_id=?")

           .authoritiesByUsernameQuery("select user_id, role from roles where user_id=?");

   }

   …

}
```

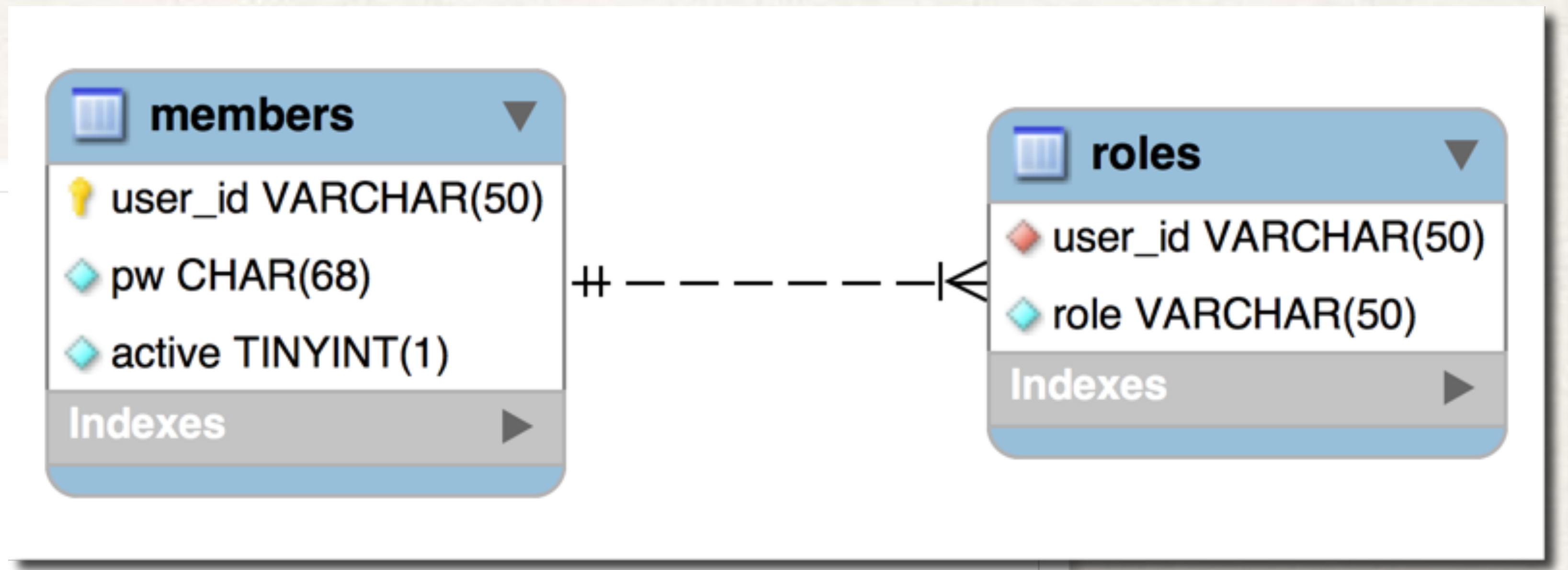
# Step 3: Update Spring Security Configuration

```
@Configuration
@EnableWebSecurity
public class DemoSecurityConfig extends WebSecurityConfigurerAdapter {

  @Autowired
  private DataSource securityDataSource;

  @Override
  protected void configure(AuthenticationManagerBuilder auth) throws Exception {

    auth.jdbcAuthentication().dataSource(securityDataSource)

        .usersByUsernameQuery("select user_id, pw, active from members where user_id=?")

        .authoritiesByUsernameQuery("select user_id, role from roles where user_id=?");

  }

  …

}
```



# Step 3: Update Spring Security Configuration

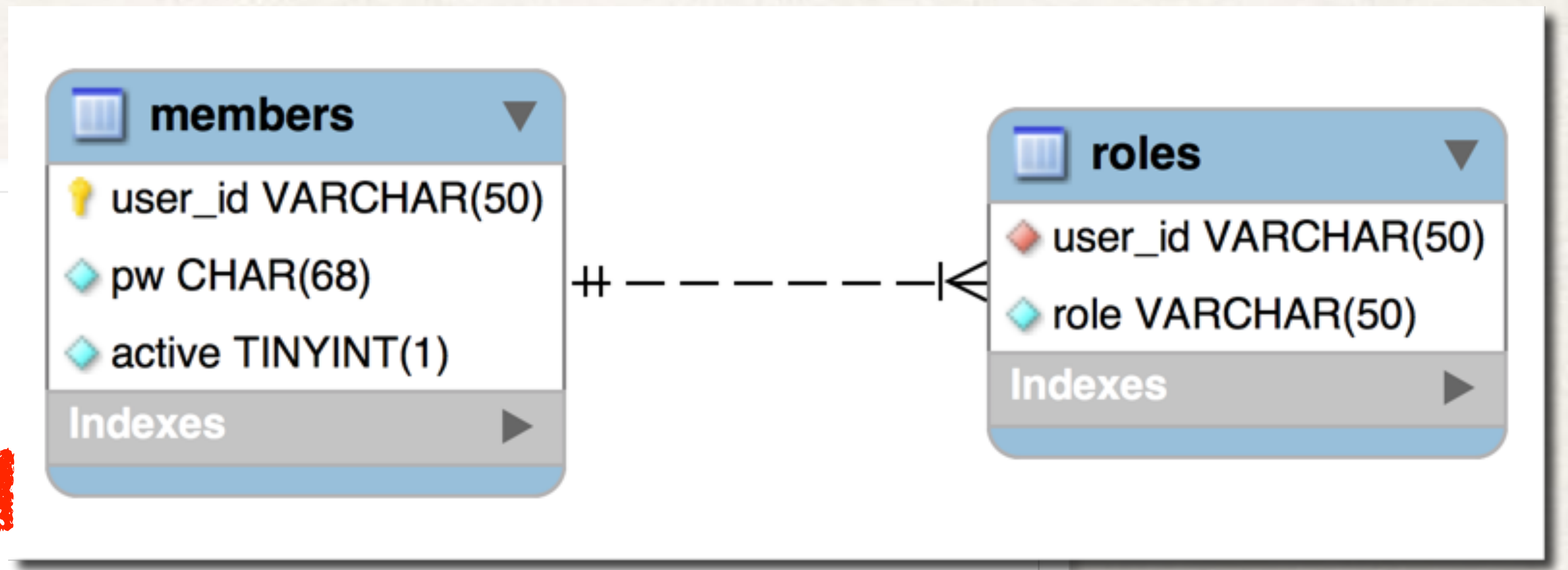
```
@Configuration
@EnableWebSecurity
public class DemoSecurityConfig extends WebSecurityConfigurerAdapter {

   @Autowired
   private DataSource securityDataSource;

   @Override
   protected void configure(AuthenticationManagerBuilder auth) throws Exception {

      auth.jdbcAuthentication().dataSource(securityDataSource)

           .usersByUsernameQuery("select user_id, pw, active from members where user_id=?")

           .authoritiesByUsernameQuery("select user_id, role from roles where user_id=?");

   }

   …

}
```



How to find users

## Step 3: Update Spring Security Configuration

```
@Configuration
@EnableWebSecurity
public class DemoSecurityConfig extends WebSecurityConfigurerAdapter {

   @Autowired
   private DataSource securityDataSource;

   @Override
   protected void configure(AuthenticationManagerBuilder auth) throws Exception {

      auth.jdbcAuthentication().dataSource(securityDataSource)

           .usersByUsernameQuery("select user_id, pw, active from members where user_id=?")

           .authoritiesByUsernameQuery("select user_id, role from roles where user_id=?");

   }

   …

}
```

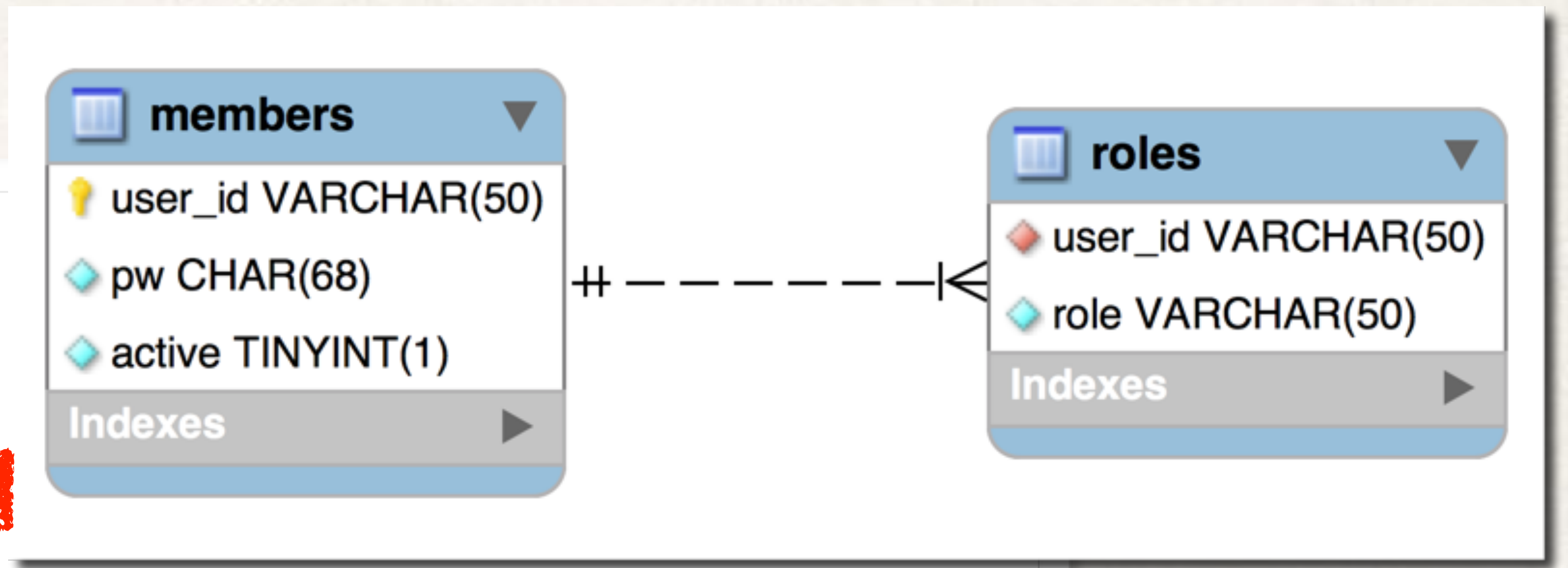


How to find users

luv2code

# Step 3: Update Spring Security Configuration

```
@Configuration
@EnableWebSecurity
public class DemoSecurityConfig extends WebSecurityConfigurerAdapter {

   @Autowired
   private DataSource securityDataSource;

   @Override
   protected void configure(AuthenticationManagerBuilder auth) throws Exception {

      auth.jdbcAuthentication().dataSource(securityDataSource)

           .usersByUsernameQuery("select user_id, pw, active from members where user_id=?")

           .authoritiesByUsernameQuery("select user_id, role from roles where user_id=?");

   }

   …

}
```

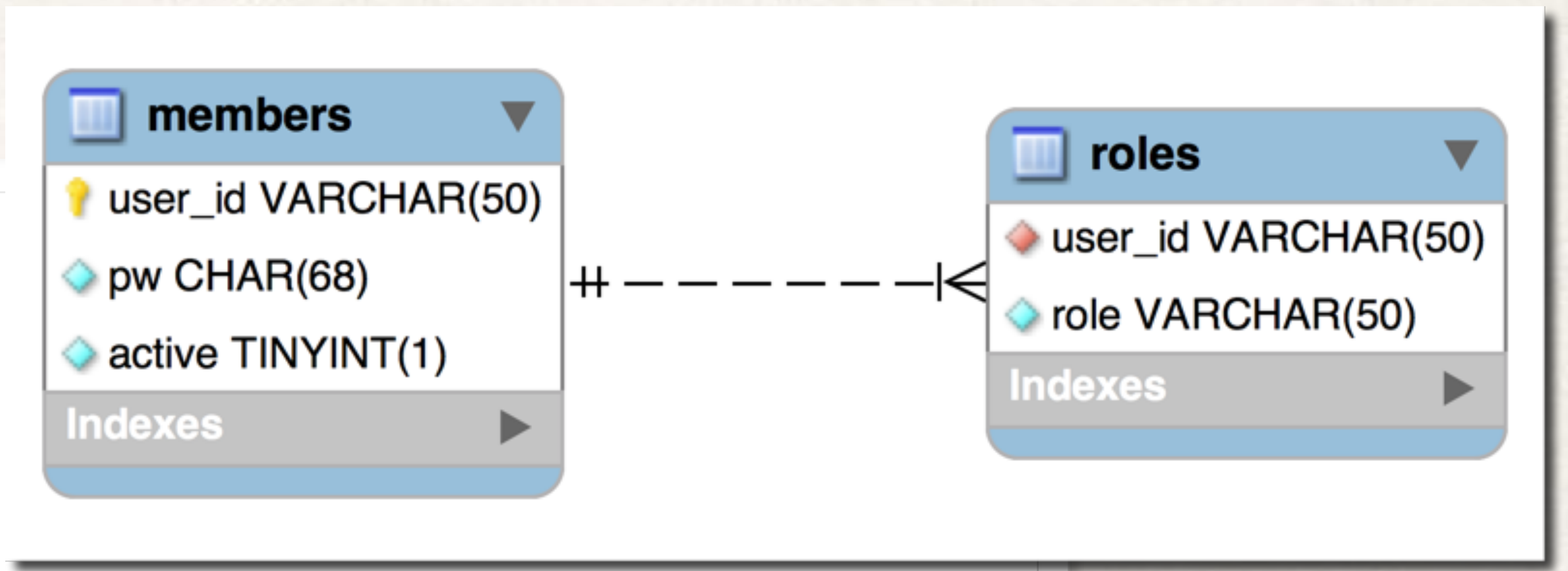


How to find users

Question mark "?"
Parameter value will be the user name from login form

# Step 3: Update Spring Security Configuration

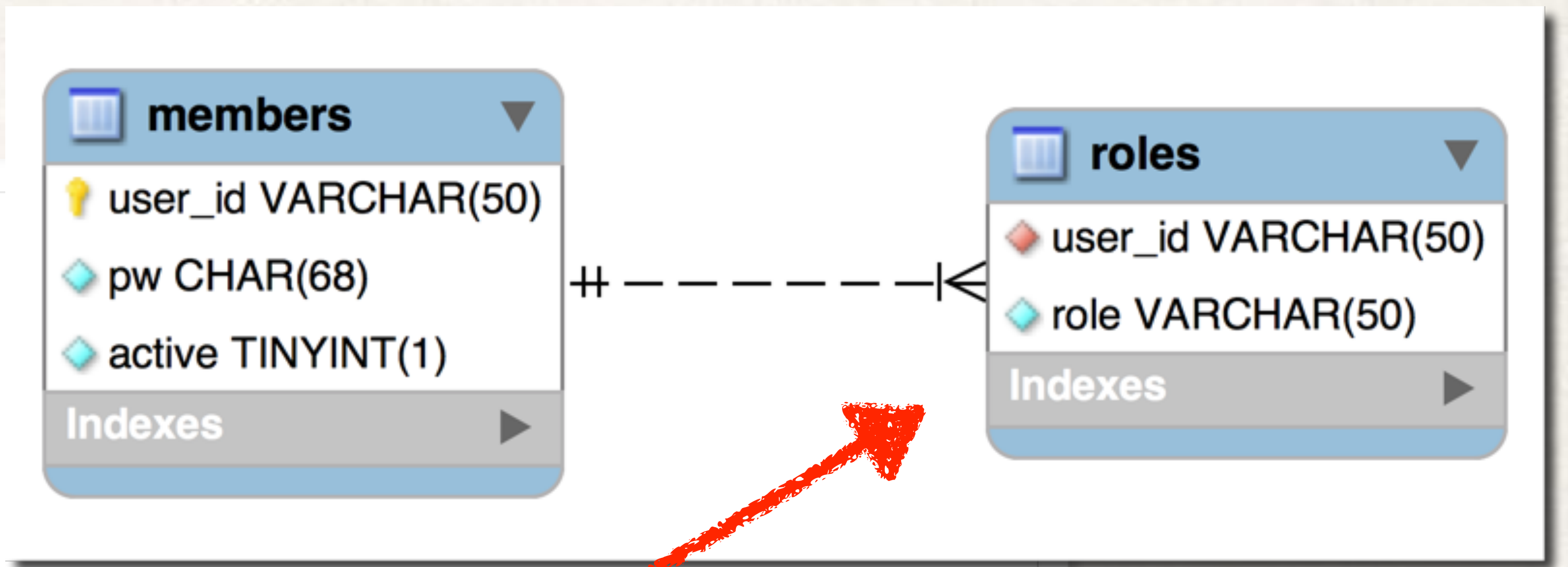
```
@Configuration
@EnableWebSecurity
public class DemoSecurityConfig extends WebSecurityConfigurerAdapter {

   @Autowired
   private DataSource securityDataSource;

   @Override
   protected void configure(AuthenticationManagerBuilder auth) throws Exception {

      auth.jdbcAuthentication().dataSource(securityDataSource)

           .usersByUsernameQuery("select user_id, pw, active from members where user_id=?")

           .authoritiesByUsernameQuery("select user_id, role from roles where user_id=?");

   }

   …

}
```



How to find roles

# Step 3: Update Spring Security Configuration

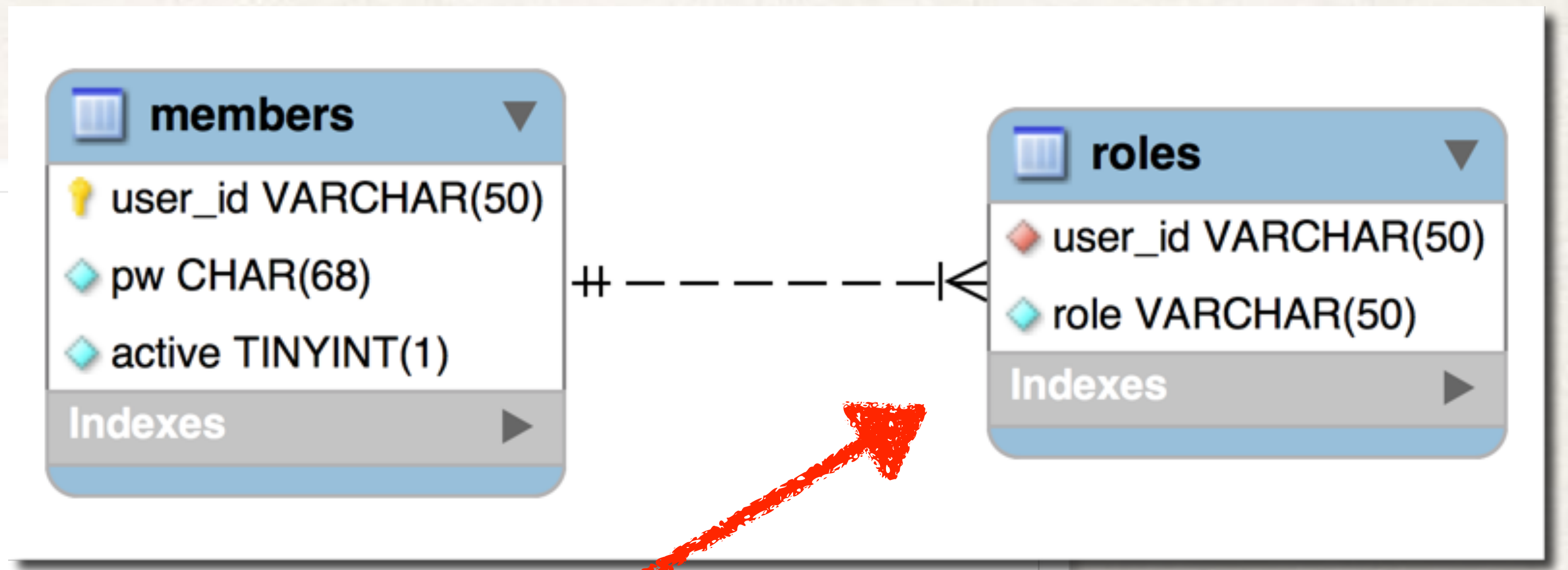
```
@Configuration
@EnableWebSecurity
public class DemoSecurityConfig extends WebSecurityConfigurerAdapter {

   @Autowired
   private DataSource securityDataSource;

   @Override
   protected void configure(AuthenticationManagerBuilder auth) throws Exception {

      auth.jdbcAuthentication().dataSource(securityDataSource)

           .usersByUsernameQuery("select user_id, pw, active from members where user_id=?")

           .authoritiesByUsernameQuery("select user_id, role from roles where user_id=?");

   }

   …

}
```



How to find roles

# Step 3: Update Spring Security Configuration

```
@Configuration
@EnableWebSecurity
public class DemoSecurityConfig extends WebSecurityConfigurerAdapter {

   @Autowired
   private DataSource securityDataSource;

   @Override
   protected void configure(AuthenticationManagerBuilder auth) throws Exception {

      auth.jdbcAuthentication().dataSource(securityDataSource)

           .usersByUsernameQuery("select user_id, pw, active from members where user_id=?")

           .authoritiesByUsernameQuery("select user_id, role from roles where user_id=?");

   }

   …

}
```



How to find roles

Question mark "?"
Parameter value will be the user name from login form