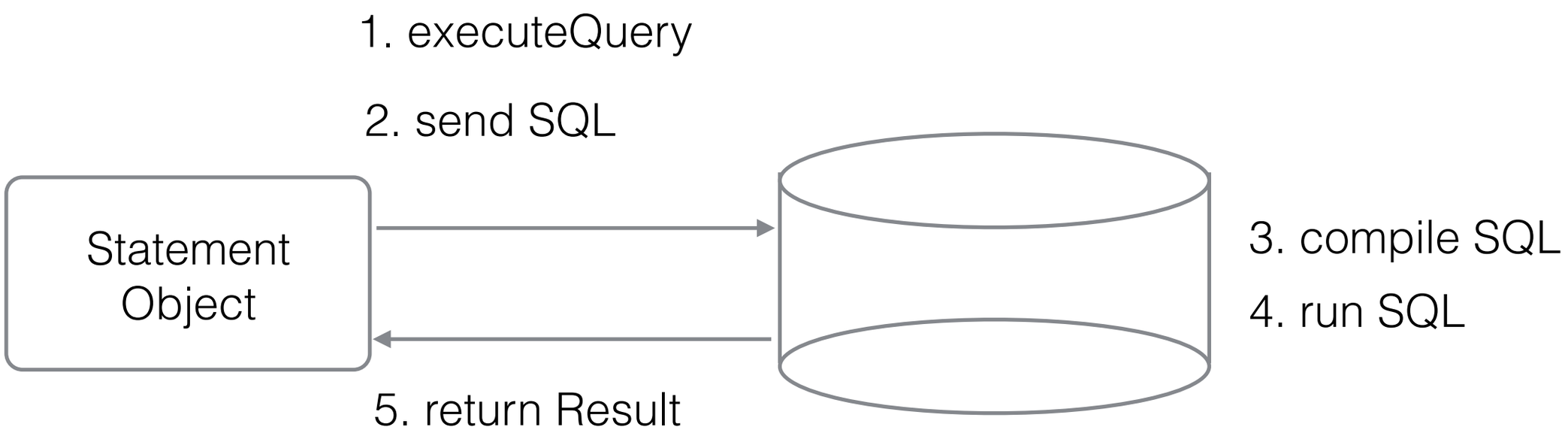
Other Statements

JDBC

Contents

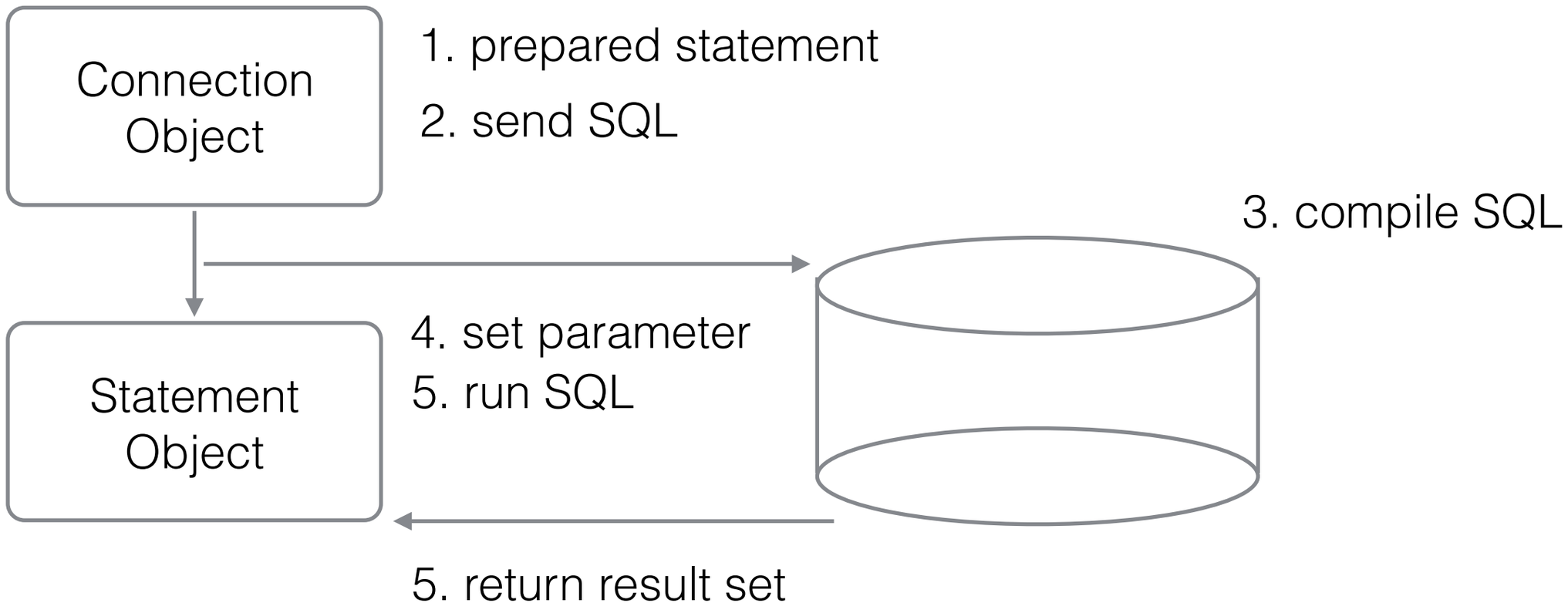
- Why not Statements?
- PreparedStatement
- Parameter Passing
- Using Stored Procedure
- CallableStatement

Why not Statement?



- Statement များသည် executeQuery method ကို ခေါ်ဆိုကာမှ sql အား DB သို့ပို့ကာ Compile လုပ်ပါမည်
- ပြီးခါမှ SQL အား execute လုပ်ကာ ရရှိလာသော Result အား Application ဆီသို့ပြန်ပို့ပေးပါမည်
- ထို့ကြောင့် Performance ပိုင်းတွင် အားနည်းမှုရှိတတ်ပါသည်

PreparedStatement



- PreparedStatement များသည် Object အား Create လုပ်စဉ်ကတည်းက SQL အား ပို့ကာ Compile လုပ်ထားပါသည်
- ထို့နောက် Parameter အား Set လုပ်ကာ execute လုပ်နိုင်သည်
- ထို့ကြောင့် Performance ပိုင်းတွင်ပိုပြီး ကောင်းမွန်ပါသည်

Parameter Passing

```
String sql = “insert into user values (?, ?, ?)”;

try (Connection conn = DriverManager.getConnection(url);
        PreparedStatement stmt = conn.preparedStatement(sql)) {
    // set parameter
    stmt.setString(1, “Aung Aung”);
    stmt.setString(2, “aungaung”);
    stmt.setString(3, “password”);

    stmt.executeUpdate();

} catch (SqlException e) {
    // exception control
}
```

Getting Auto Generated ID

- Auto Increment ID များးအားးအသုံးပြုထားးသော Table များးအား Insert ပြုလုပ်ရာတွင် Application များးအတွင်းမှ Generate လုပ်လိုက်သော ID အား သိရှိရန်လိုအပ်ပါသည်

- Auto Increment ID အား ရယူလိုသော အခါ Connection မှ PreparedStatement အား create လုပ်သည့် method prepareStatement(String sql, int autoGeneratedKeys) တွင် autoGeneratedKeys နေရာတွင် Statement.RETURN_GENERATED_KEYS အား ပေးကာ တည်ဆောက်ရပါမည်

- ထို့နောက်တွင် Statement#getGeneratedKeys method ဖြင့် ResultSet Object အားရယူကာ Generated လုပ်လိုက်သော Key အား ရယူနိုင်ပါသည်

Sample Codes

```
// for auto increase id
try (Connection conn = ConnectionManager.getConnection();
            PreparedStatement stmt = conn.prepareStatement(sql,
                        Statement.RETURN_GENERATED_KEYS)) {

      // set parameter
      stmt.executeUpdate();

      ResultSet rs = stmt.getGeneratedKeys();
      while(rs.next()) {
            Object id = rs.getObject(1);
            return this.setId(t, id);
      }
} catch (Exception e) {
      e.printStackTrace();
}
```

Stored Procedure

- Stored Procedure များသည် DBMS အပေါ်တွင် အသုံးပြုနိုင်သော လုပ်ဆောင်ချက်များဖြစ်ကြသည်

- Programming Language များကဲ့သို့ပင် Variable များအား အသုံးပြုနိုင်ပြီး၊ Decision Controls များ၊ Branching များ နှင့် Iteration Controls များအား ရေးသား အသုံးပြုနိုင်ပါသည်

- မည်သည့် SQL မျိုးမဆို execute လုပ်နိုင်သောကြောင့် Stored Procedure တစ်ခုမှ SQL အများအား အသုံးပြုကာ တစ်ပြိုင်နက်တည်း Execute လုပ်နိုင်ပါသည်

Advantages

- တစ်ကြိမ်ရေးသားပြီးသော Stored Procedure များသည် DBMS အပေါ်တွင် Compile လုပ်ပြီး ပြန်လည် အသုံးပြုနိုင်သောကြောင့် Performance ပိုင်းကို ပိုမိုကောင်းမွန်စေပါသည်

- Stored Procedure များသည် တစ်ကြိမ်တည်းနှင့် SQL များကို အကြိမ်ပေါင်း များစွာ Execute လုပ်နိုင်သောကြောင့် Application နှင့် DB Server အကြားမှ Traffice အား လျော့ချနိုင်ပါသည်

- Stored Procedure များသည် တစ်ခါ ရေးသားပြီးပါက မည်သည့် Application များမှမဆို ပြန်လည်အသုံးပြုနိုင်ပါသည်

- Stored Procedure များအား Permission များအား လိုအပ်သလို Grant လုပ် နိုင်သောကြောင့် Secure ဖြစ်ပါသည်

Disadvantages

- Stored Procedure အများအပြားအား Connection တစ်ခု အတွင်း အသုံးပြုခြင်း သည် Memory အား ဖြုန်းတီးရာရောက်စေပြီး၊ Stored Procedure တစ်ခုအတွင်း Logic အများအပြားရေးသားထားခြင်းသည် CUP Usage အား တိုးပွားစေနိုင် ပါသည်

- ရှုပ်ထွေးသော Business Logic များ အတွင် Stored Procedure များအား ဖွဲ့စည်း ကာ ရေးသားခြင်းသည် Develop လုပ်ငန်းစဉ်အတွင်း ပိုမိုခက်ခဲ စေနိုင်ပါသည်

- အချို့သော DBMS များသာ Stored Procedure အား Debug လုပ်နိုင်သော Feature များအား ပံ့ပိုးထားသောကြောင် Debug လုပ်ရခက်ခဲနိုင်ပါသည်

- Stored Procedure အား Develop လုပ်ရန် အထိုက်အလျောက် အတွေ့အကြုံ လိုအပ်သောကြောင့် Stored Procedure များအား Develop လုပ်ရန်နှင့် Maintain လုပ်ရန်ခက်ခဲနိုင်ပါသည်

CallableStatement

- CallableStatement သည် PreparedStatement ၏ Sub Class ဖြစ်ပြီး Store Procedure များအား Java Program အတွင်းမှ ခေါ်ဆိုရာတွင် အသုံးပြုပါသည်

- PreparedStatement များကဲ့သို့ပင် CallableStatement များသည်လဲ Parameter များအား အသုံးပြုနိုင်ပါသည်

- Store Procedure များတွင် ရလဒ်များအား Return လုပ်သည့် Stored Procedure နှင့် Return မလုပ်သော Stored Procedure များရှိကြသောကြောင့် CallableStatement ကို ခေါ်ဆိုပုံမှာလဲ ကွာခြားပါသည်

Stored Procedure with no return

```
DELIMITER //
CREATE PROCEDURE GetStudentsByCourse(IN courseID INT)
    BEGIN
    SELECT * FROM student s
    WHERE s.id in (
        SELECT sjc.student_id FROM
        student_jdc_class sjc JOIN jdc_class jc
        ON jc.id = sjc.jdc_class_id
        WHERE jc.course_id = courseID
    );
    END //
DELIMITER ;
```

Using CallableStatement

Stored Procedure with return

```
// for auto increase id
try (Connection conn = ConnectionManager.getConnection();
        PreparedStatement stmt = conn.prepareStatement(sql,
                Statement.RETURN_GENERATED_KEYS)) {

    // set parameter
    stmt.executeUpdate();

    ResultSet rs = stmt.getGeneratedKeys();
    while(rs.next()) {
        Object id = rs.getObject(1);
        return this.setId(t, id);
    }
} catch (Exception e) {
    e.printStackTrace();
}
```

Using CallableStatement