

# VanillaCore Walkthrough

## Part 1

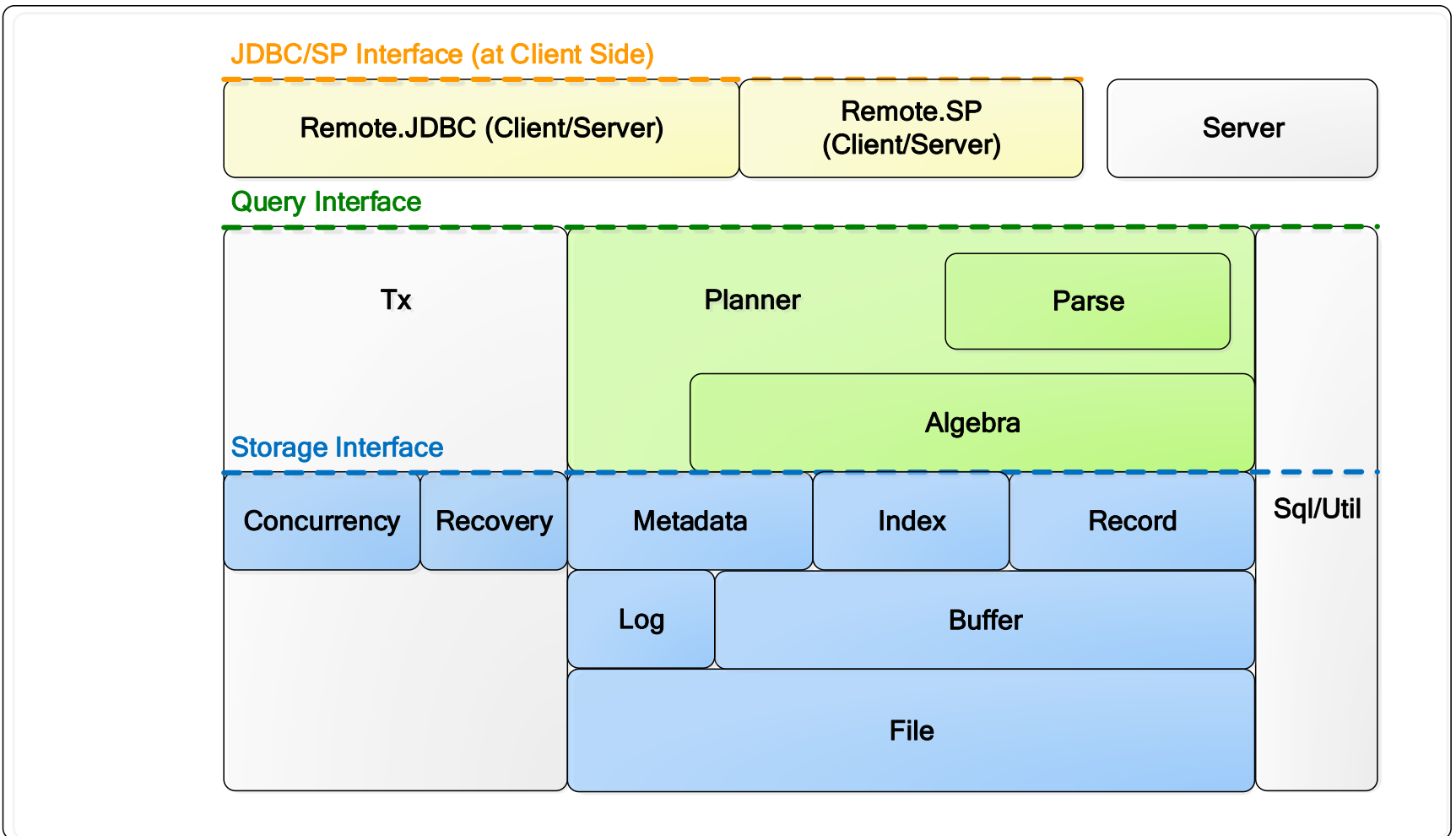
Introduction to Database Systems

DataLab

CS, NTHU

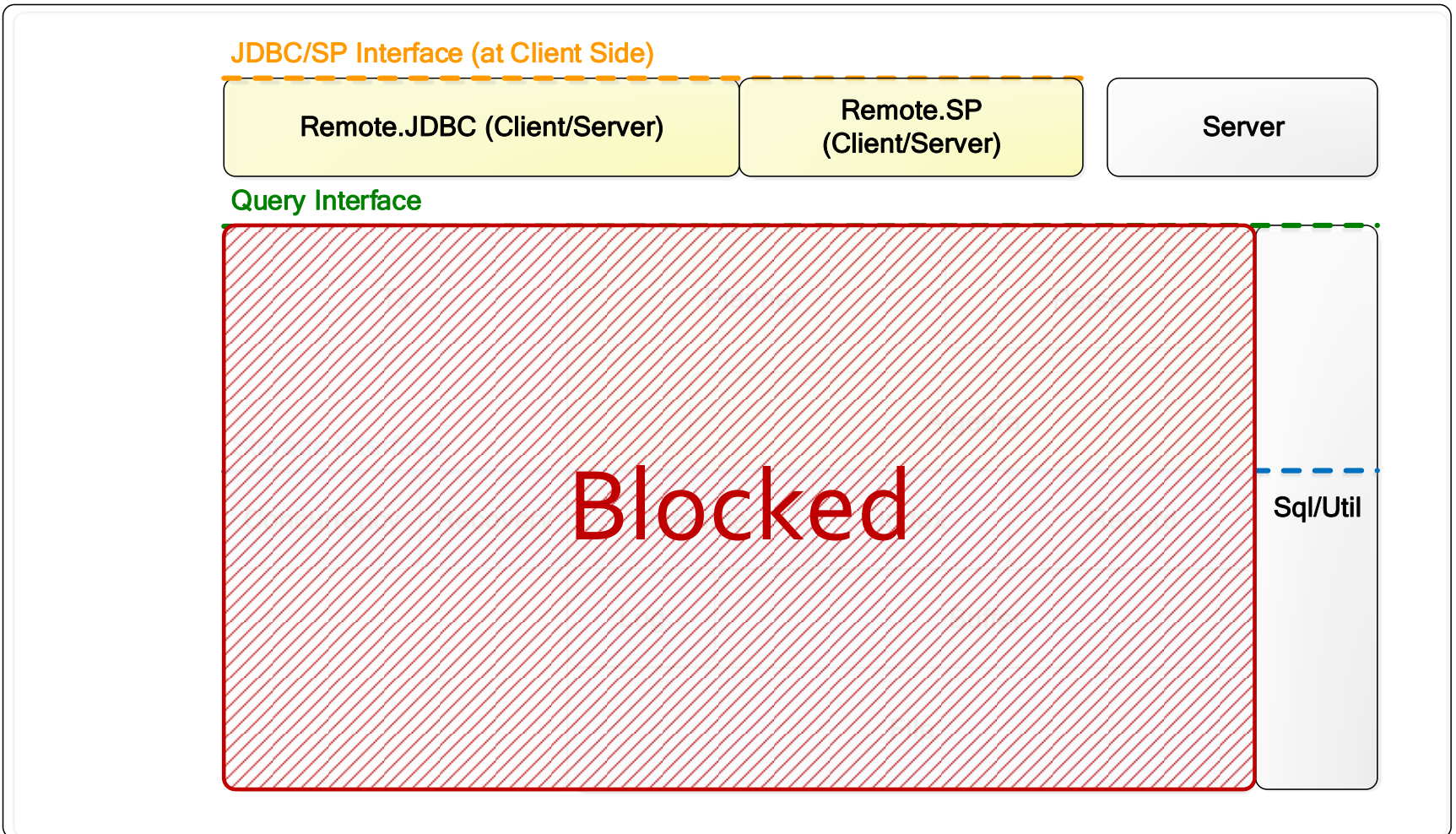
# The Architecture

VanillaDB



# The Architecture

VanillaDB





# Fog

- Now, you can only see a part of VanillaCore.
- As the progress of the course, we will open more packages in the future.

# Outline

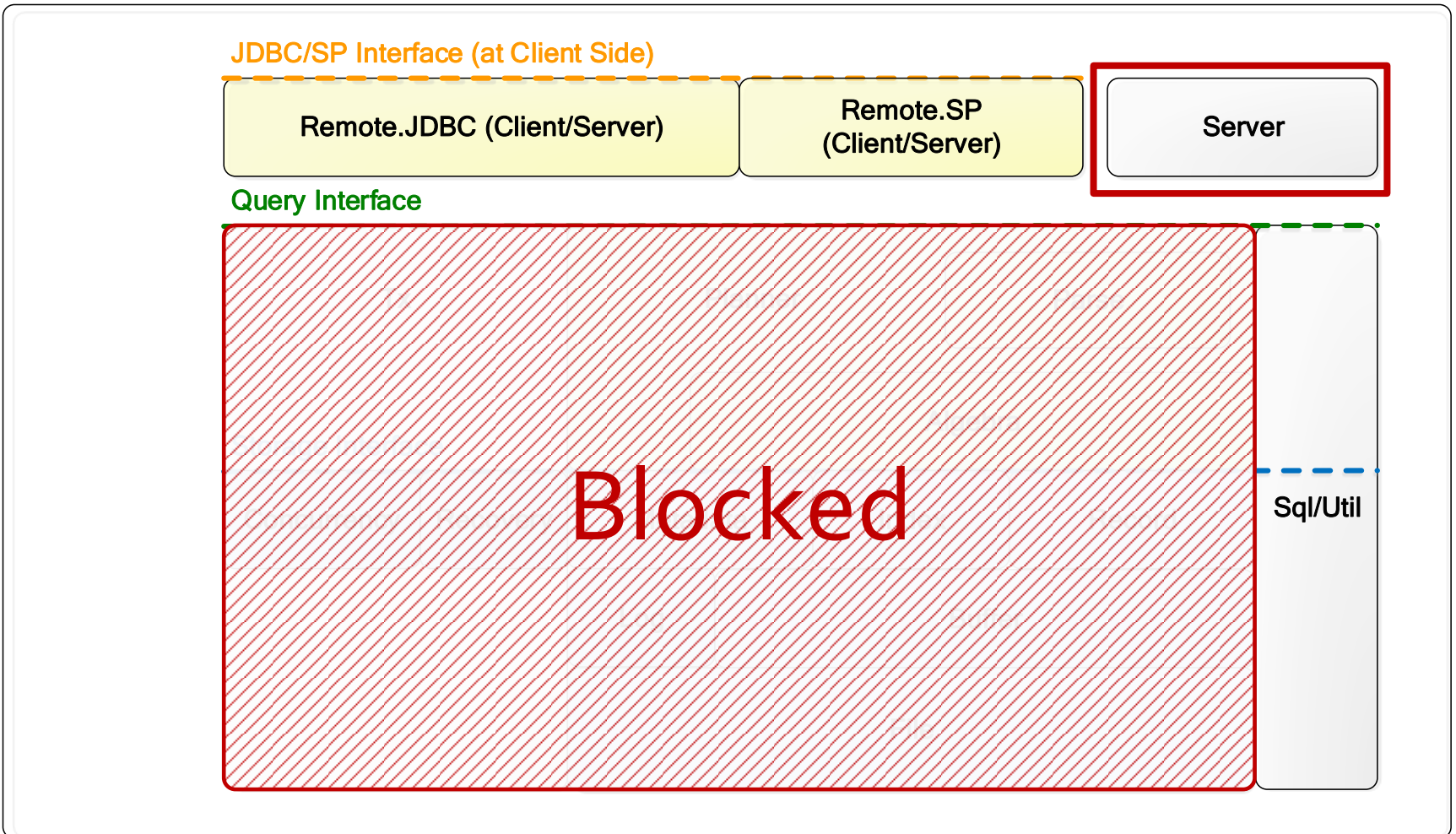
- Server package
- Remote package
- SQL package

# Outline

- Server package
- Remote package
- SQL package

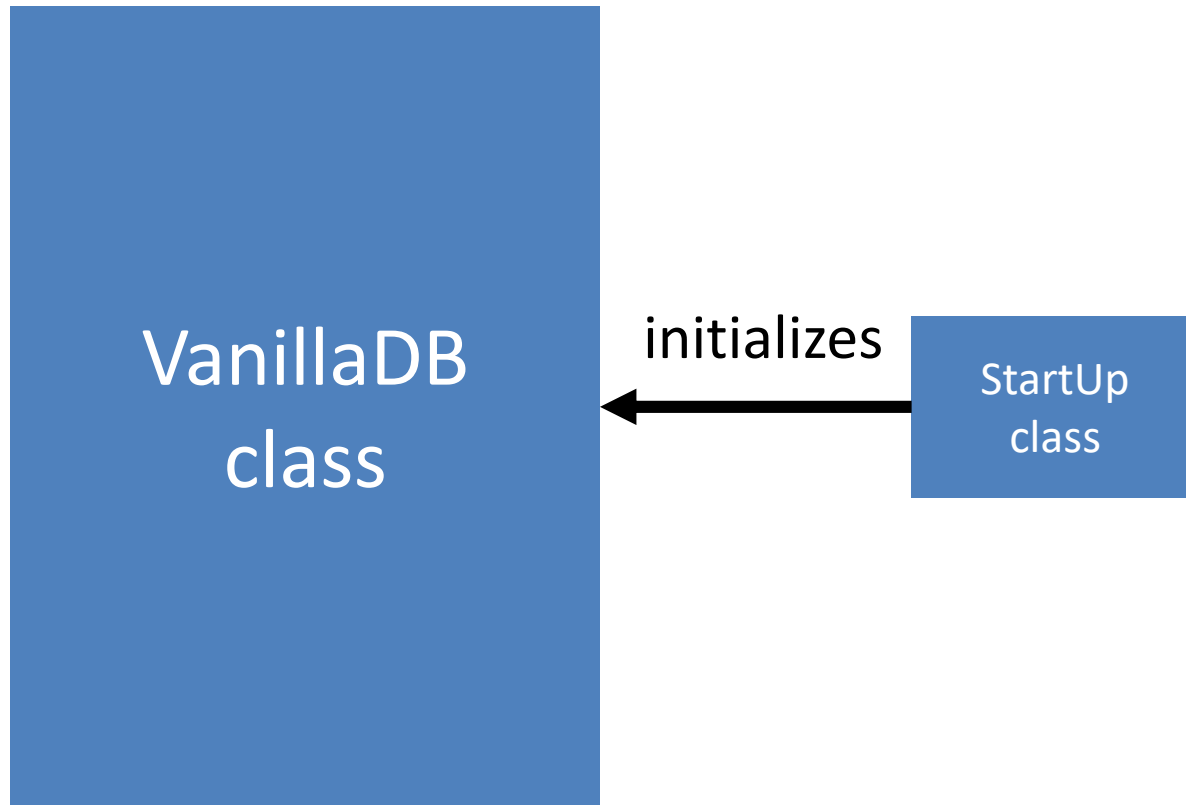
# Where are we?

VanillaDB





# server Package



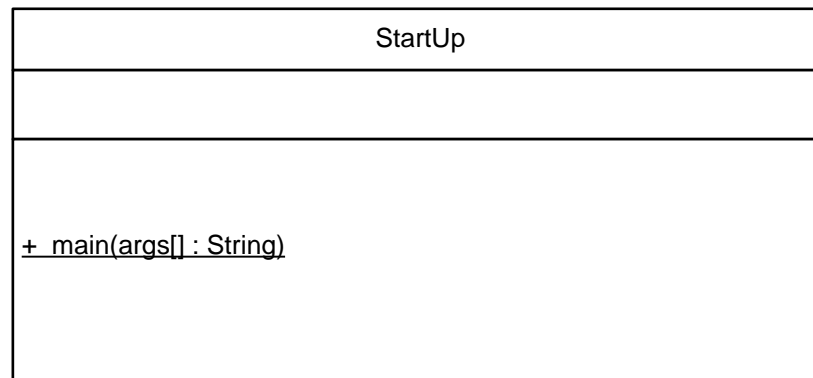
# VanillaDb

- There are four types of methods
  - Initialization
  - Global getters
  - Factory methods
  - Profiler

VanillaDb
<u>+ init(dirName : String)</u> <u>+ isInitd() : boolean</u> <u>+ initFileMgr(dirname : String)</u> <u>+ initFileAndLogMgr(dirname : String)</u> <u>+ initTaskMgr()</u> <u>+ initTxMgr()</u> <u>+ initCatalogMgr(isnew : boolean, tx : Transaction)</u> <u>+ initStatMgr(tx : Transaction)</u> <u>+ initSPFactory()</u> <u>+ initCheckpointingTask()</u>  <u>+ fileMgr() : FileMgr</u> <u>+ bufferMgr() : BufferMgr</u> <u>+ logMgr() : LogMgr</u> <u>+ catalogMgr() : CatalogMgr</u> <u>+ statMgr() : StatMgr</u> <u>+ taskMgr() : TaskMgr</u> <u>+ txMgr() : TransactionMgr</u>  <u>+ spFactory() : StoredProcedureFactory</u> <u>+ newPlanner() : Planner</u>  <u>+ initAndStartProfiler()</u> <u>+ stopProfilerAndReport()</u>

# StartUp

- StartUp provides `main()` that runs VanillaCore as a **JDBC** server
  - Calls `VanillaDB.init()`
    - Sharing global resources through static variables
  - Binds `RemoteDriver` to RMI registry
    - Thread per connection

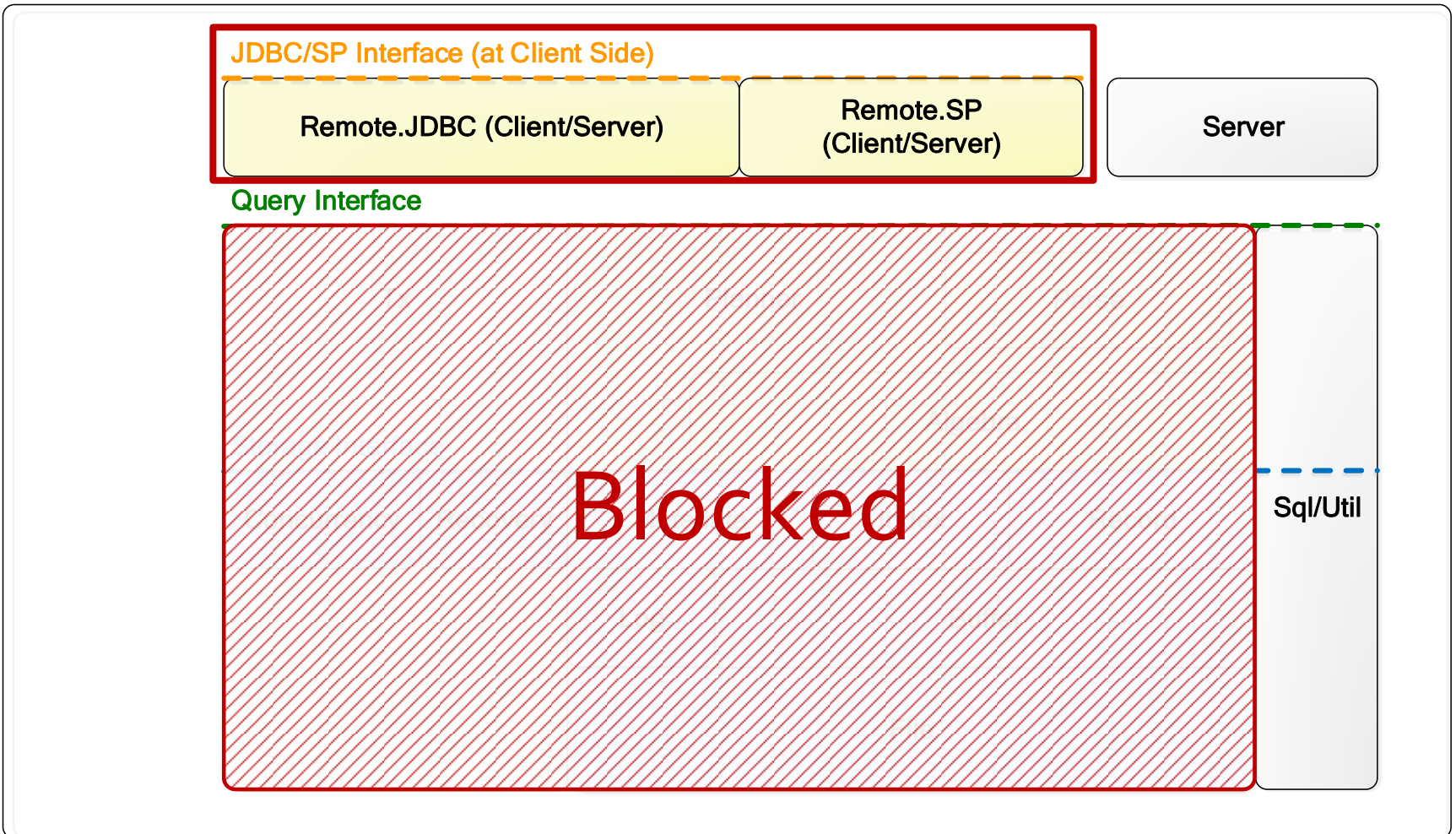


# Outline

- Server package
- **Remote package**
- SQL package

# Where are we?

VanillaDB



# remote Package

A solid purple rectangular box containing the text 'JDBC Package' in white.

JDBC  
Package

A solid purple rectangular box containing the text 'Stored Procedure Package' in white.

Stored Procedure  
Package

# remote Package



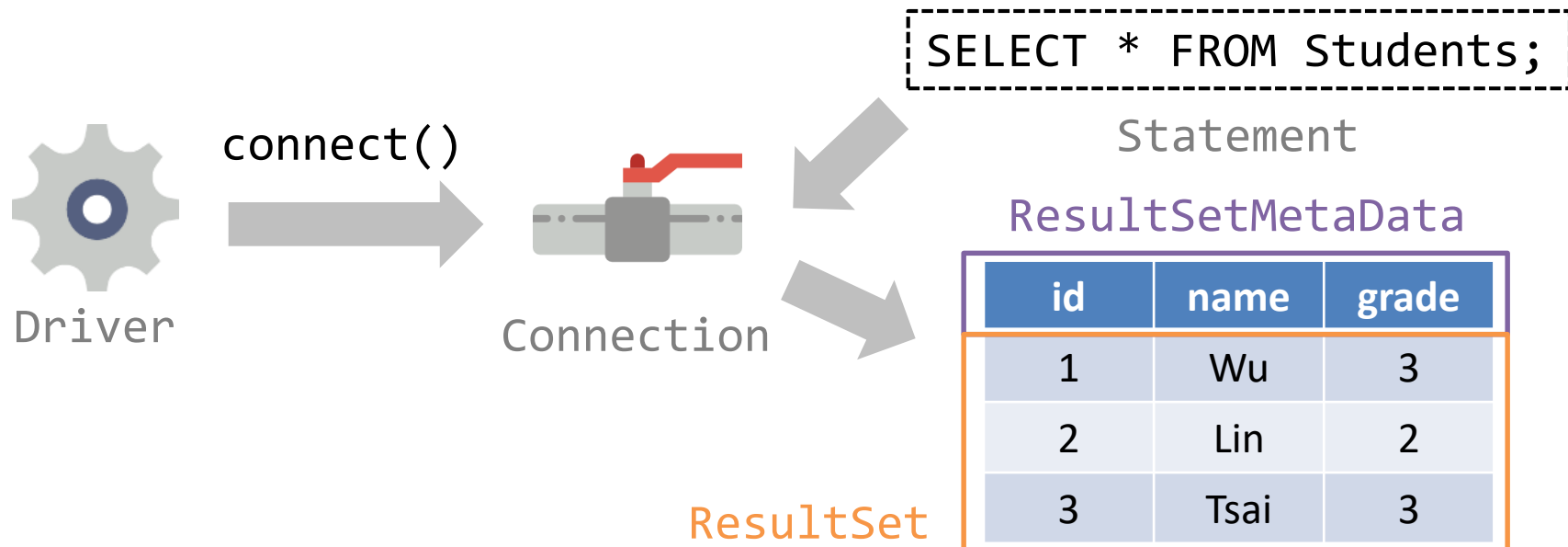
JDBC  
Package



Stored Procedure  
Package

# JDBC

- Java Database Connectivity (JDBC) is an API for Java, that defines how a client may access a database.





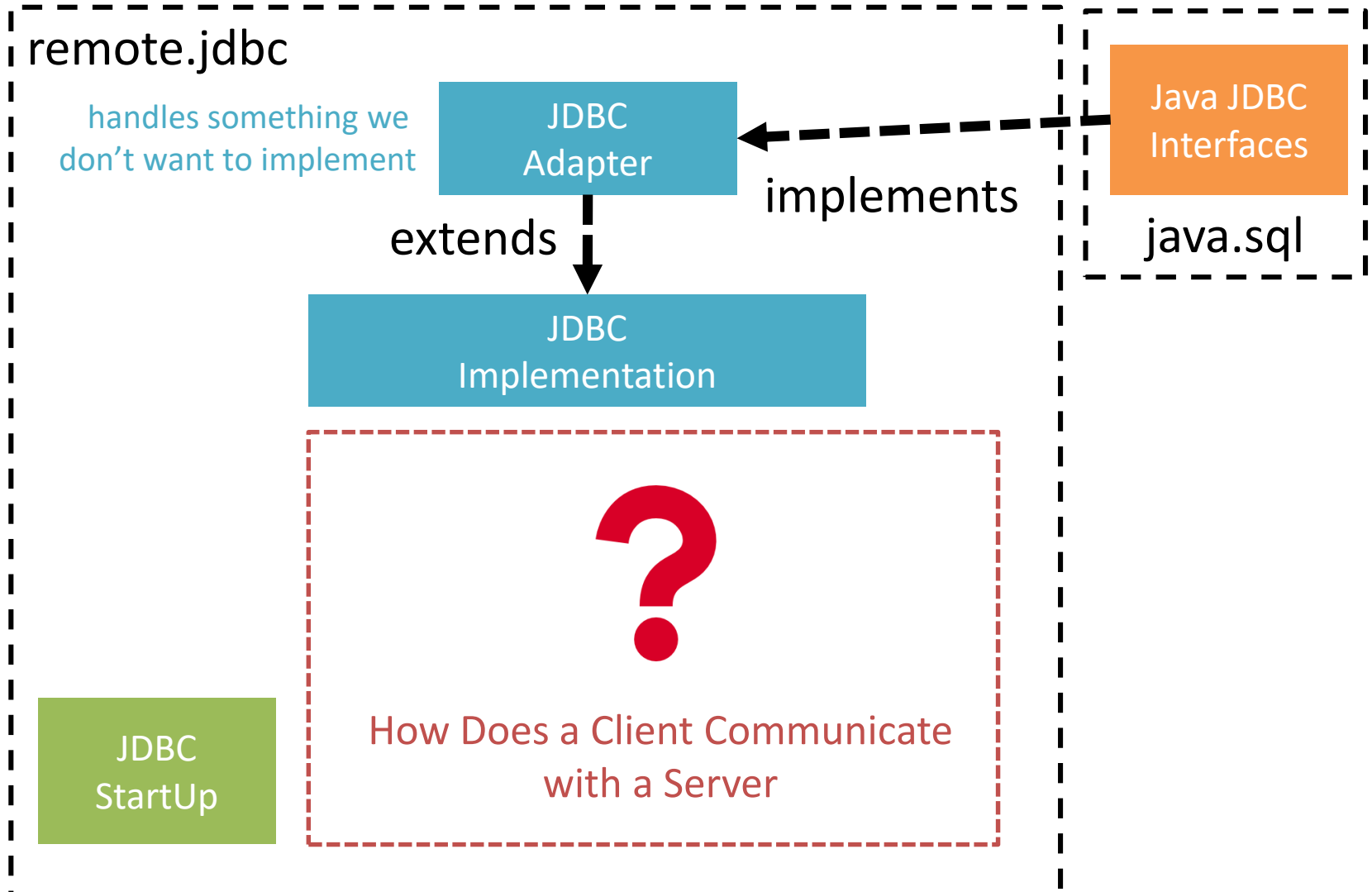
```

Connection conn = null;
try {
    // Step 1: connect to database server
    Driver d = new JdbcDriver();
    conn = d.connect("jdbc:vanilladb://localhost", null);
    conn.setAutoCommit(false);
    conn.setReadOnly(true);
    // Step 2: execute the query
    Statement stmt = conn.createStatement();
    String qry = "SELECT s-name, d-name FROM departments, "
    + "students WHERE major-id = d-id";
    ResultSet rs = stmt.executeQuery(qry);
    // Step 3: loop through the result set
    rs.beforeFirst();
    System.out.println("name\tmajor");
    System.out.println("-----\t-----");
    while (rs.next()) {
        String sName = rs.getString("s-name");
        String dName = rs.getString("d-name");
        System.out.println(sName + "\t" + dName);
    }
    rs.close();
} catch (SQLException e) {
    e.printStackTrace();
} finally {
    try {
        // Step 4: close the connection
        if (conn != null)
            conn.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
}

```

# JDBC Program: Finding Major

# remote.jdbc Package



# RMI

- VanillaCore uses Java Remote Method Invocation (RMI) for communication.
  - It makes a program able to call a method on other program without knowing the implementation of the method.

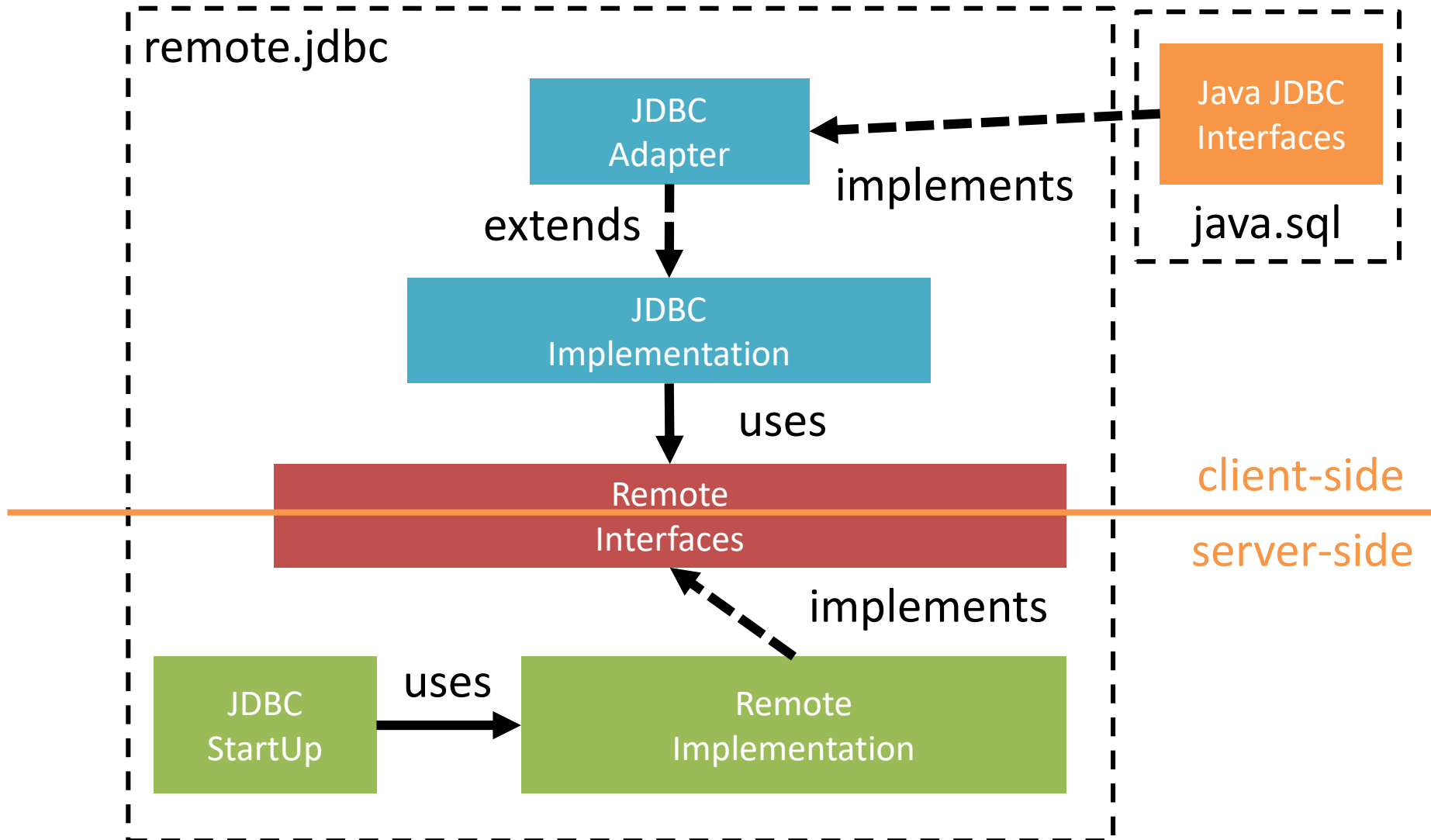
# RMI Example

```
public interface API {  
    int[] sort(int[] numbers);  
}
```

```
public class Server implements API {  
    @Override  
    public int[] sort(int[] numbers) {  
        int[] array = Arrays.copyOf(numbers, numbers.length);  
        Arrays.sort(array);  
        return array;  
    }  
}
```

```
public class Client {  
    public static void main(String[] args) {  
        ...  
        Registry reg = LocateRegistry.getRegistry(host);  
        API api = (API) reg.lookup(regName);  
        array = api.sort(array);  
    }  
}
```

# remote.jdbc Package



# remote Package

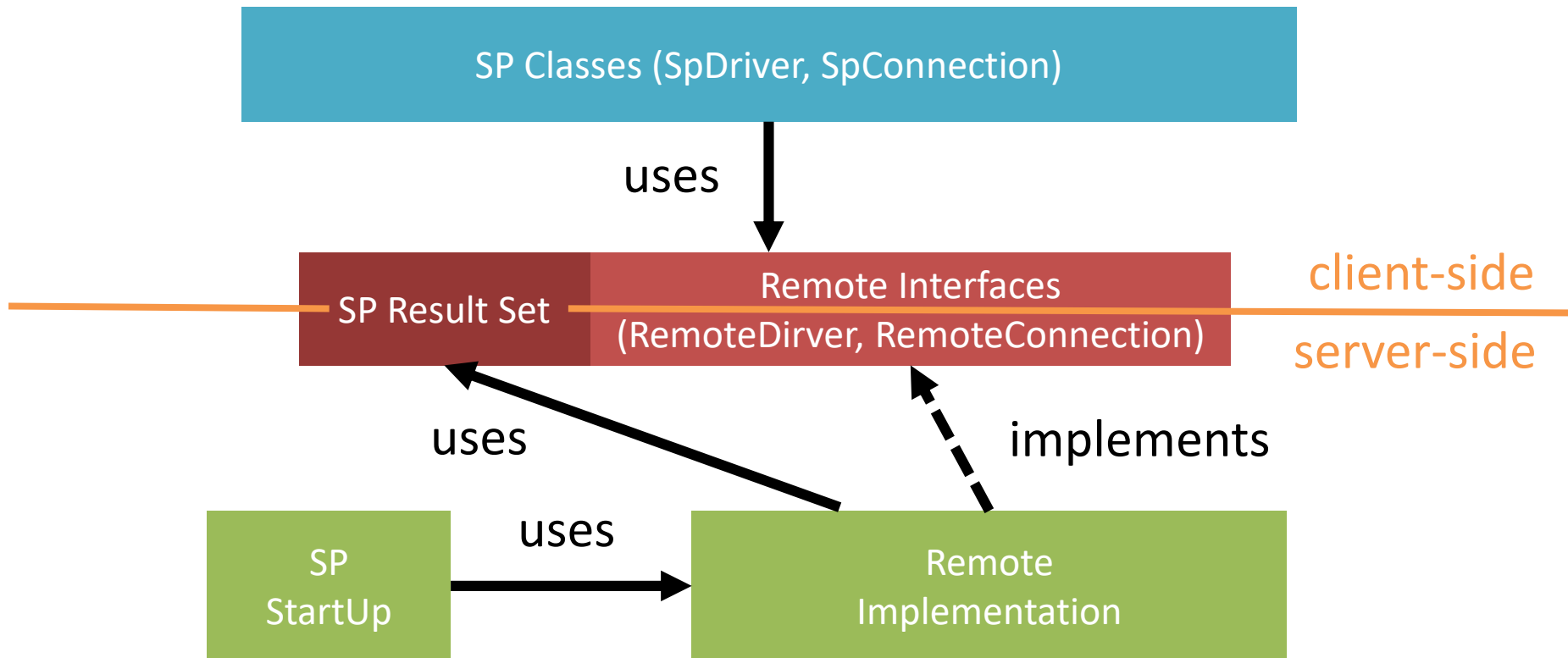


JDBC  
Package



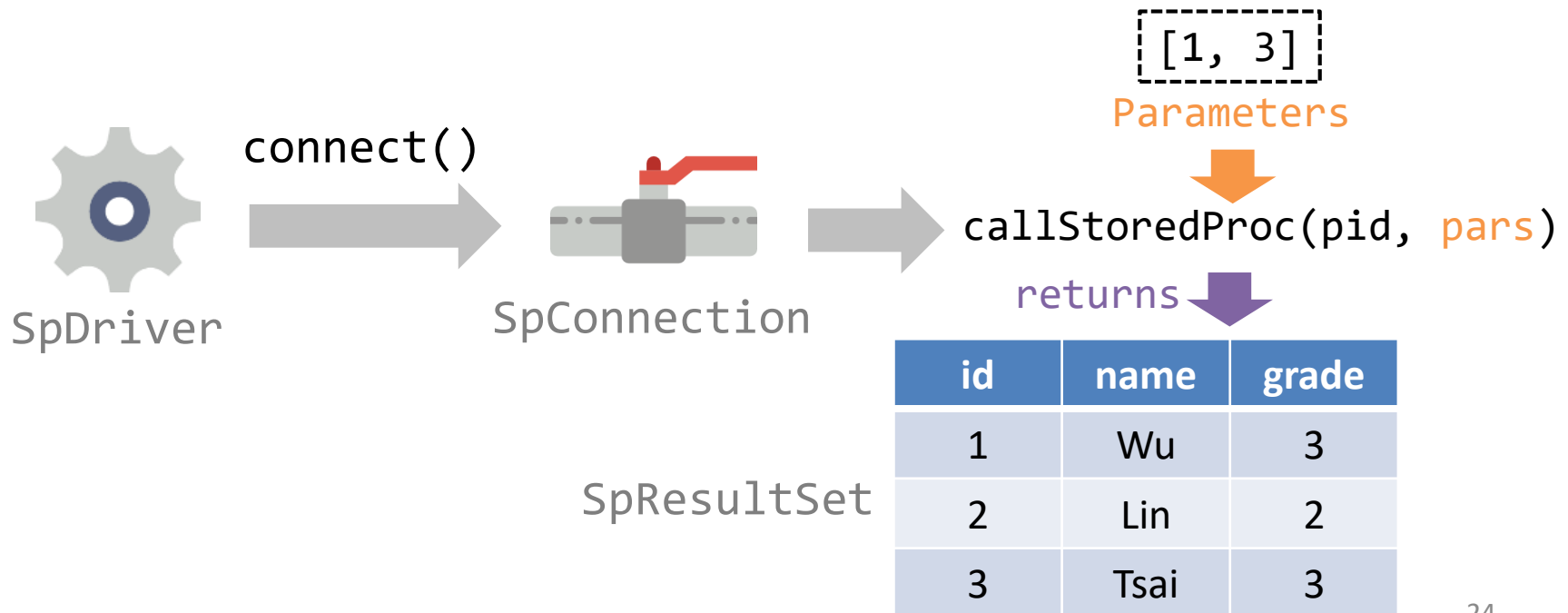
Stored Procedure  
Package

# remote.storedprocedure Package



# Calling Stored Procedure

- To call a stored procedure from clients, it first establishes a connection from the driver.
  - Then send the parameters via the connection



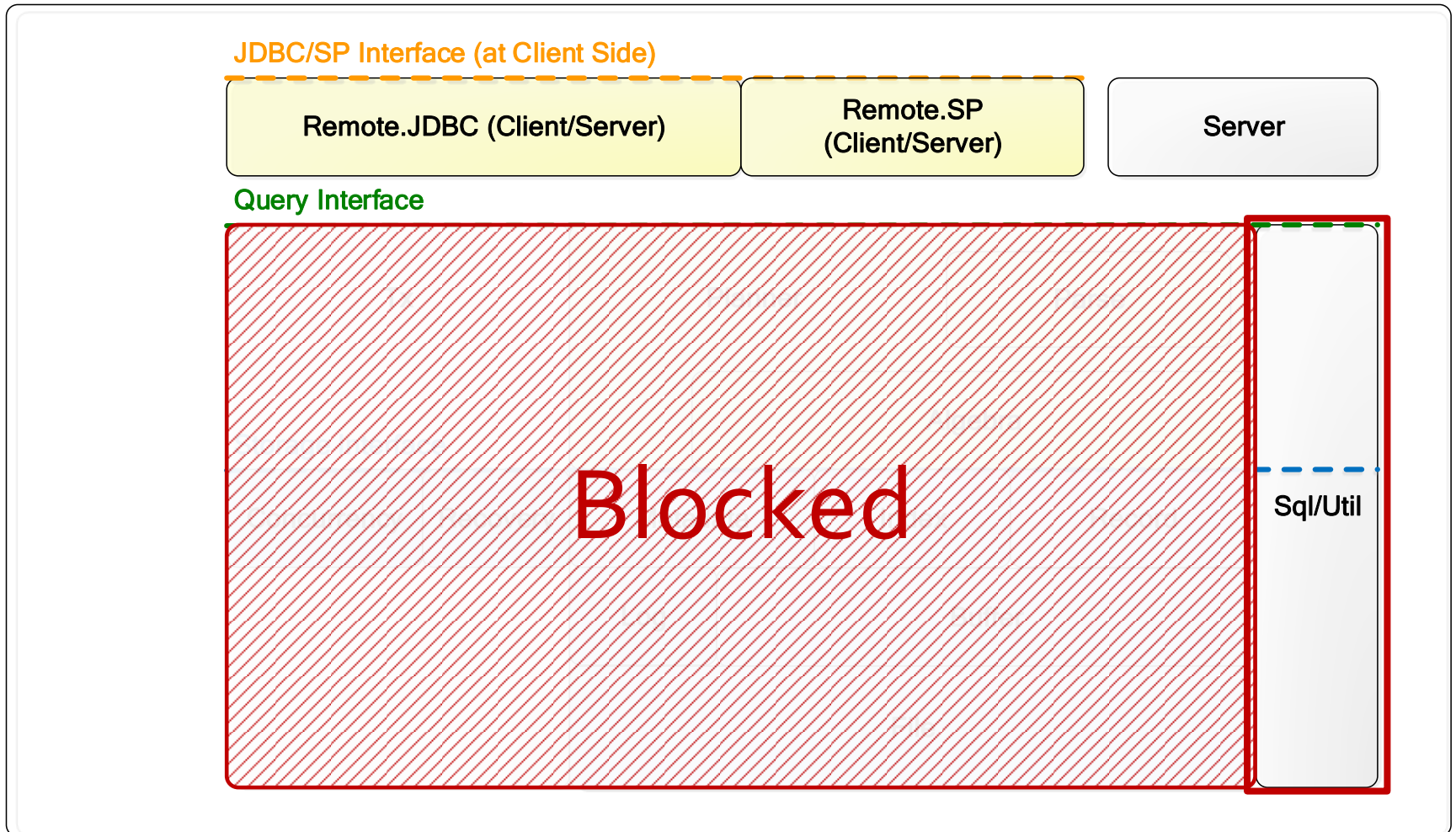


# Outline

- Server package
- Remote package
- SQL package

# Where are we?

VanillaDB



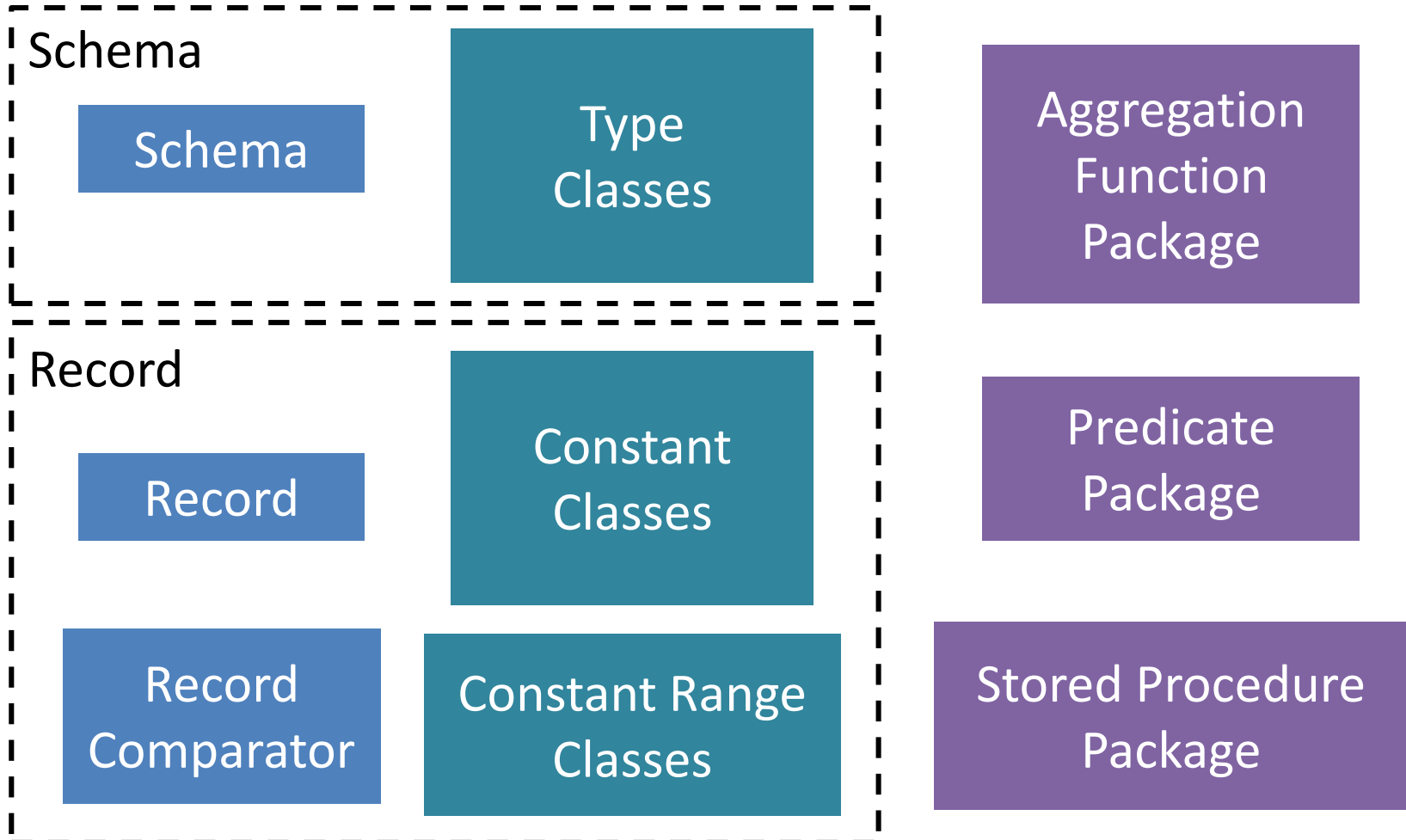
# Schema & Records

blog_id	url	created	author_id
33981	ms.com/...	2012/10/31	729
33982	apache.org/...	2012/11/15	4412

← Schema

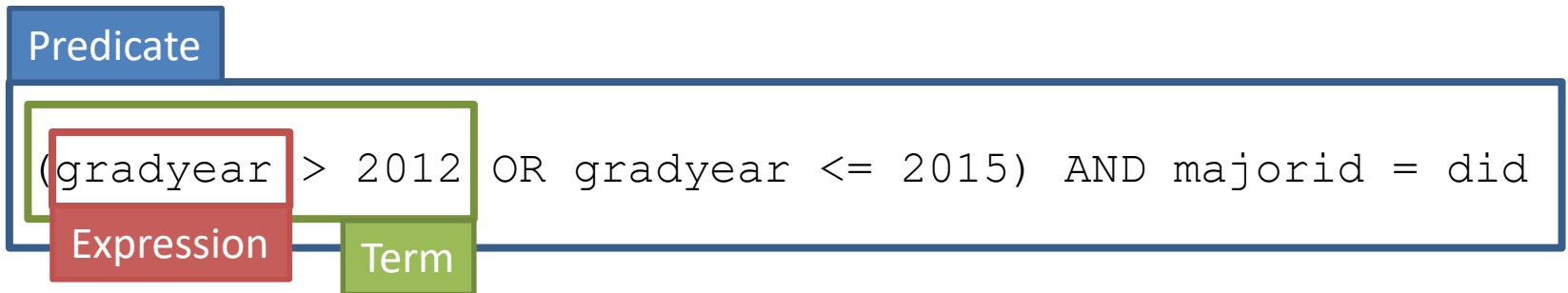
← Record

# sql Package

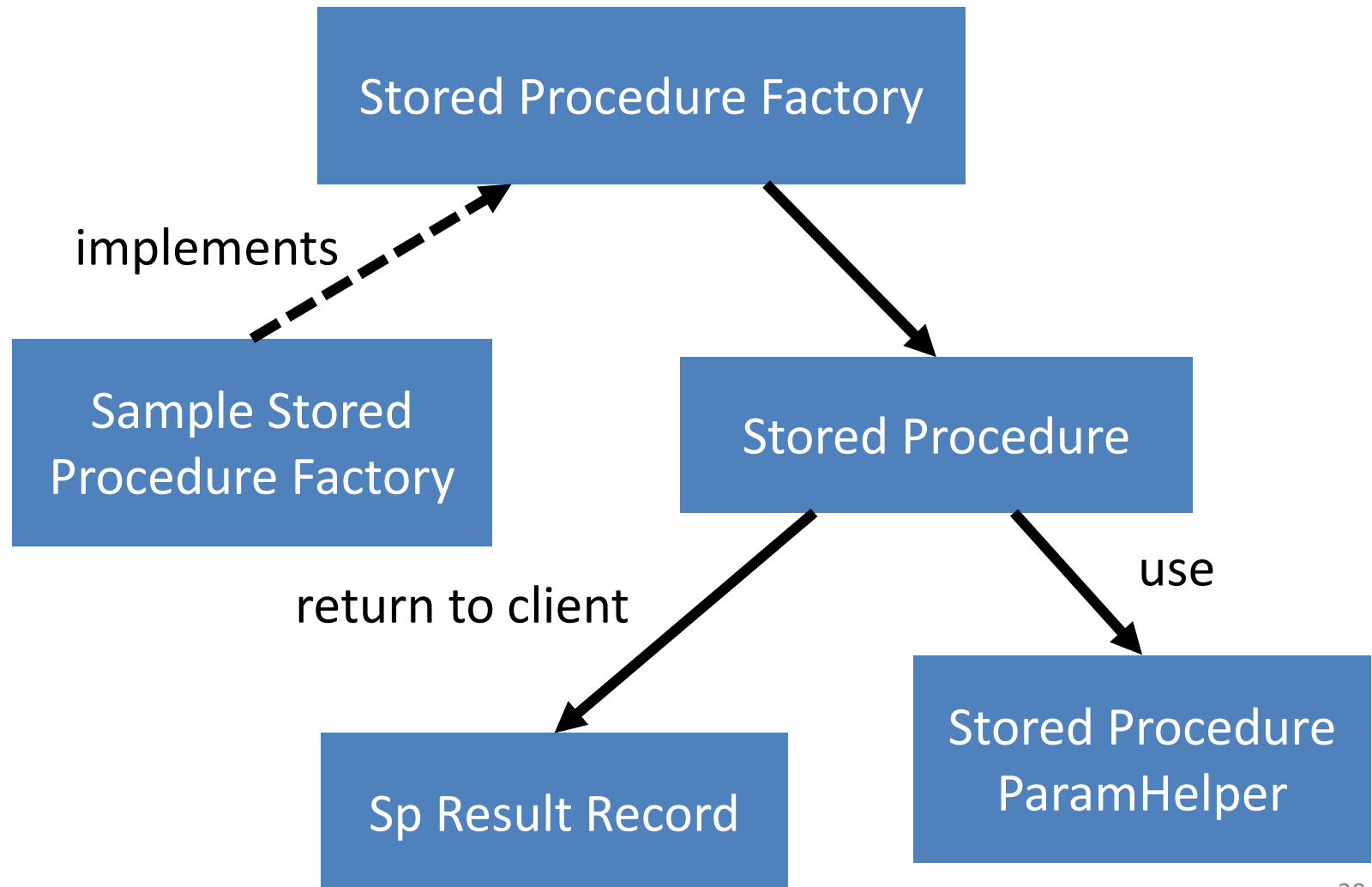


# Predicates

- An **expression** consists of constants, field names, or their operations
- A **term** is a comparison between two expressions
- A **predicate** is a Boolean combination of terms



# sql.storedprocedure Package



# Factory Pattern

- A factory takes care of which implementation should be used.
- The clients only need to pass the parameters to it and wait the results.

