Part 16

Java Programming Language
Mr.Rungrote Phonkam
rungrote@it.kmitl.ac.th

Contents

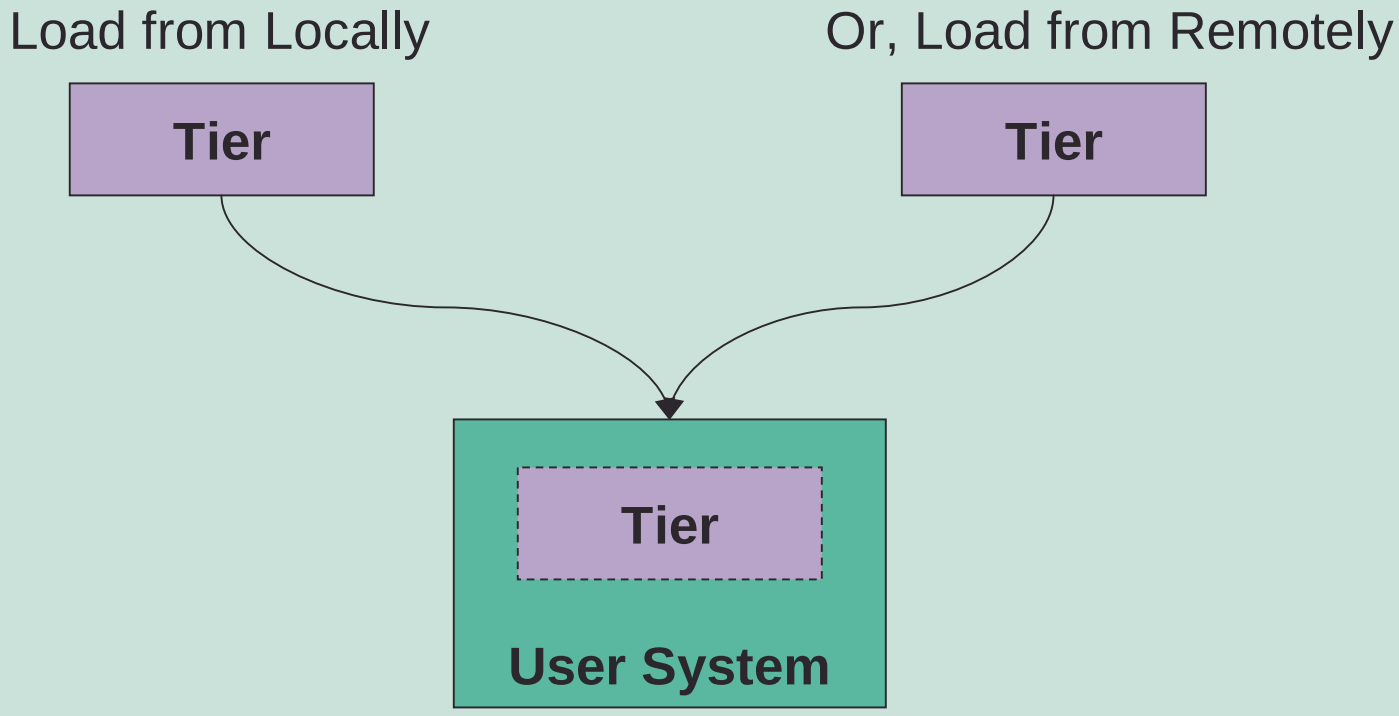
1. Tiers
2. Middleware
3. JDBC Type
4. JDBC Steps
5. JDBC Example

1. Tiers

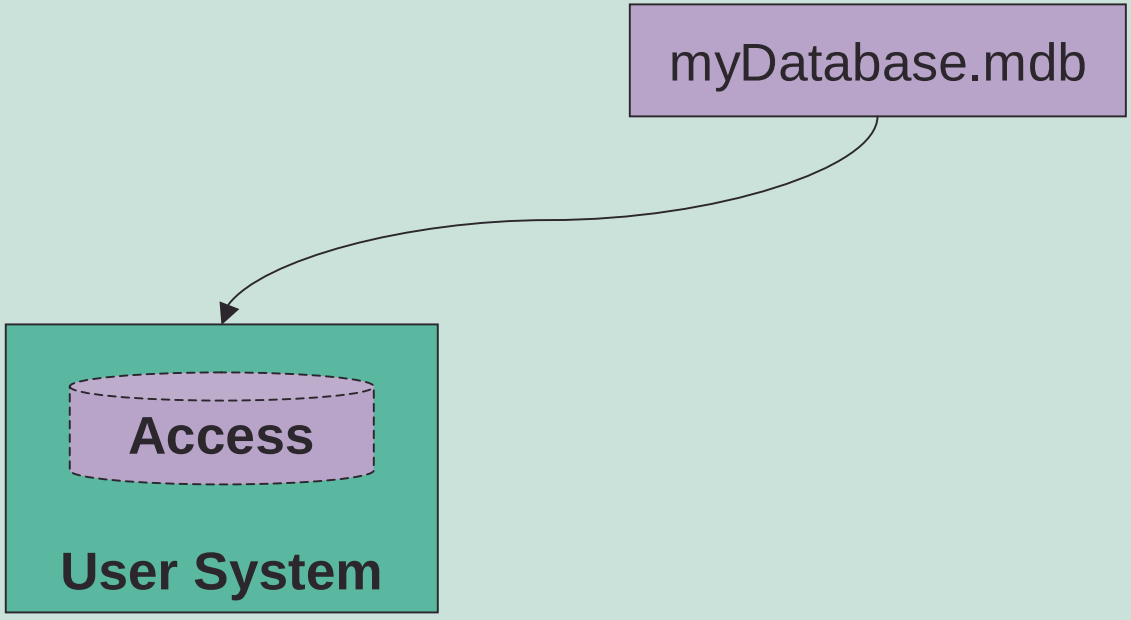
Single Tier

- คือโปรแกรมที่มีโปรเซสในการทำงานบนระบบคอมพิวเตอร์ของผู้ใช้เพียงโปรเซสเดียว
- เป็นรูปแบบการใช้งานโปรแกรมแบบ Standalone
- การทำงานไม่ต้องอาศัยระบบเครือข่าย ยกเว้นไฟล์โปรแกรมถูกเก็บไว้ที่เครื่องคอมพิวเตอร์ประเภท File Server
- ตัวอย่างเช่น PaintBrush, Microsoft Word, FoxPro, Access

Single Tier


Load from Locally
Or, Load from Remotely
Tier
Tier
Tier
User System

Single Tier (Example)

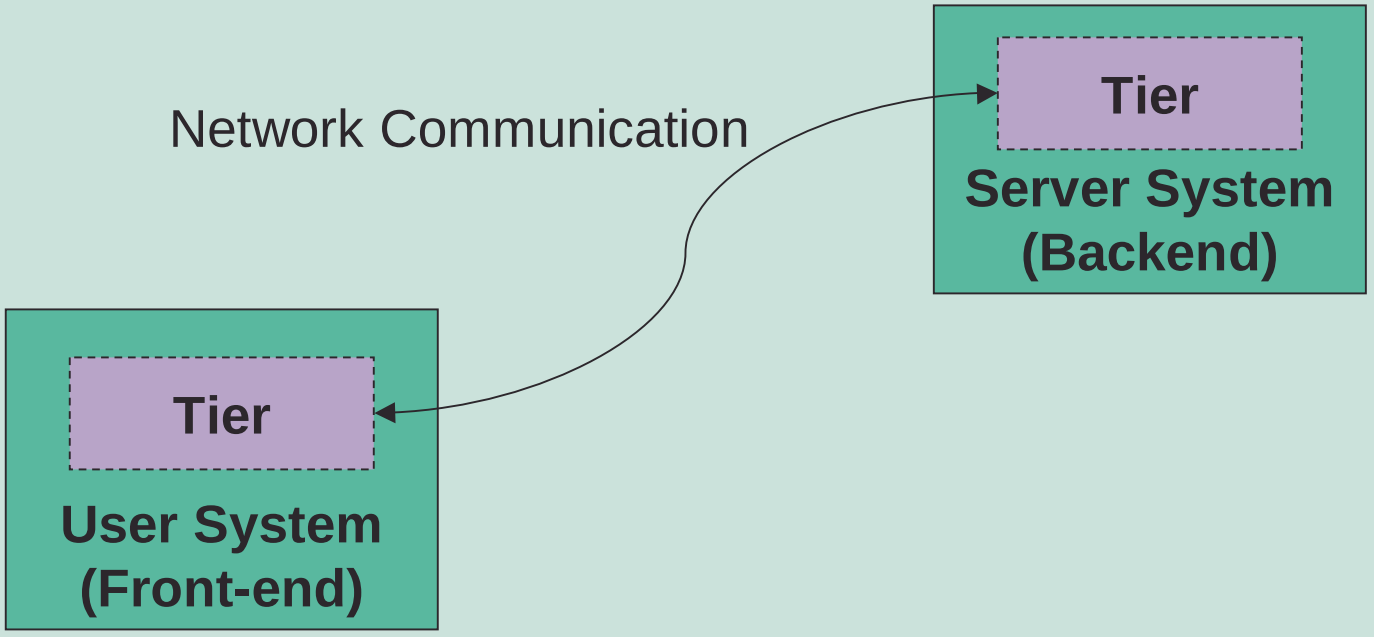


1. Tiers

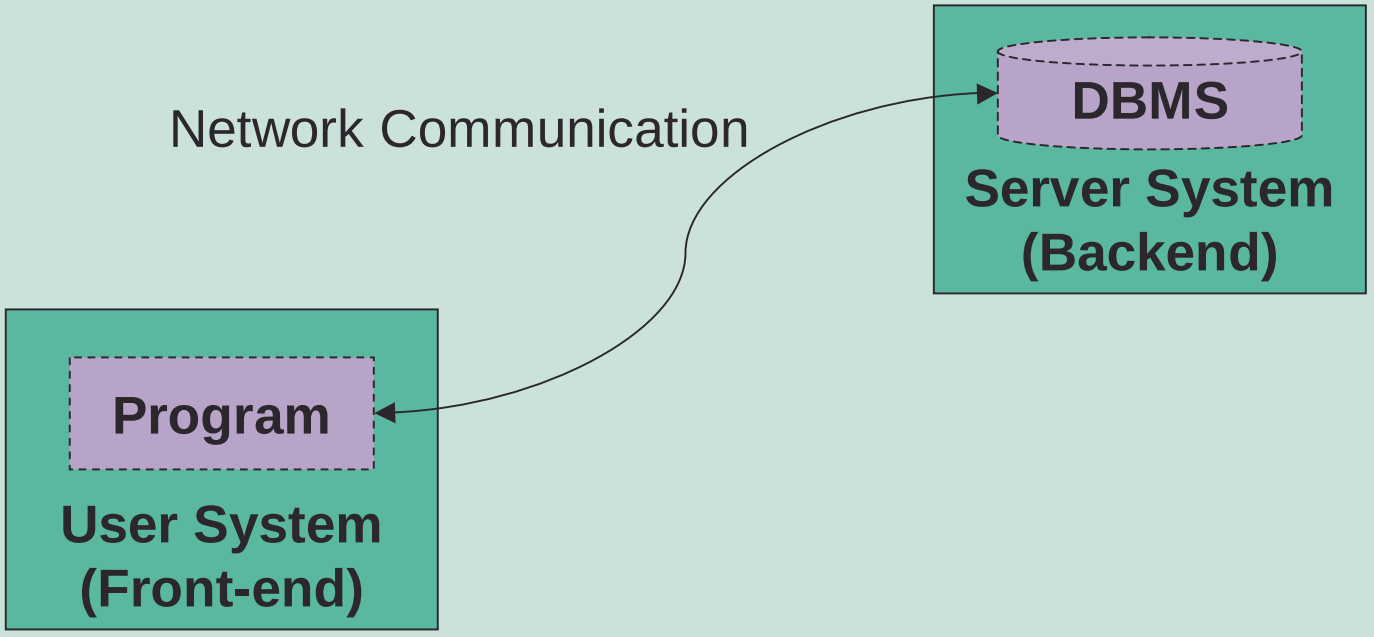
Two Tier

- คือโปรแกรมที่มีโปรเซสในการทำงานบนระบบคอมพิวเตอร์สองโปรเซส
- โปรเซสหนึ่งทำงานในฉากหลังสำหรับการประมวลผล ส่วนอีกโปรเซสอยู่ที่ฉากหน้าสำหรับทำงานในด้านติดต่อกับผู้ใช้
- โปรเซสทั้งสองอาจทำงานบนเครื่องคอมพิวเตอร์เครื่องเดียวกันหรือคนละเครื่องก็ได้
- ถ้าโปรเซสทำงานคนละเครื่องต้องอาศัยระบบเครือข่ายคอมพิวเตอร์ช่วยในการสื่อสารระหว่างโปรเซส
- ตัวอย่างเช่น ระบบเว็บ, ระบบ E-mail, ระบบฐานข้อมูลแบบไคลแอนท์/เซิร์ฟเวอร์

Two Tier


Network Communication
Tier
Server System
(Backend)
Tier
User System
(Front-end)

Two Tier (Example)



1. Tiers

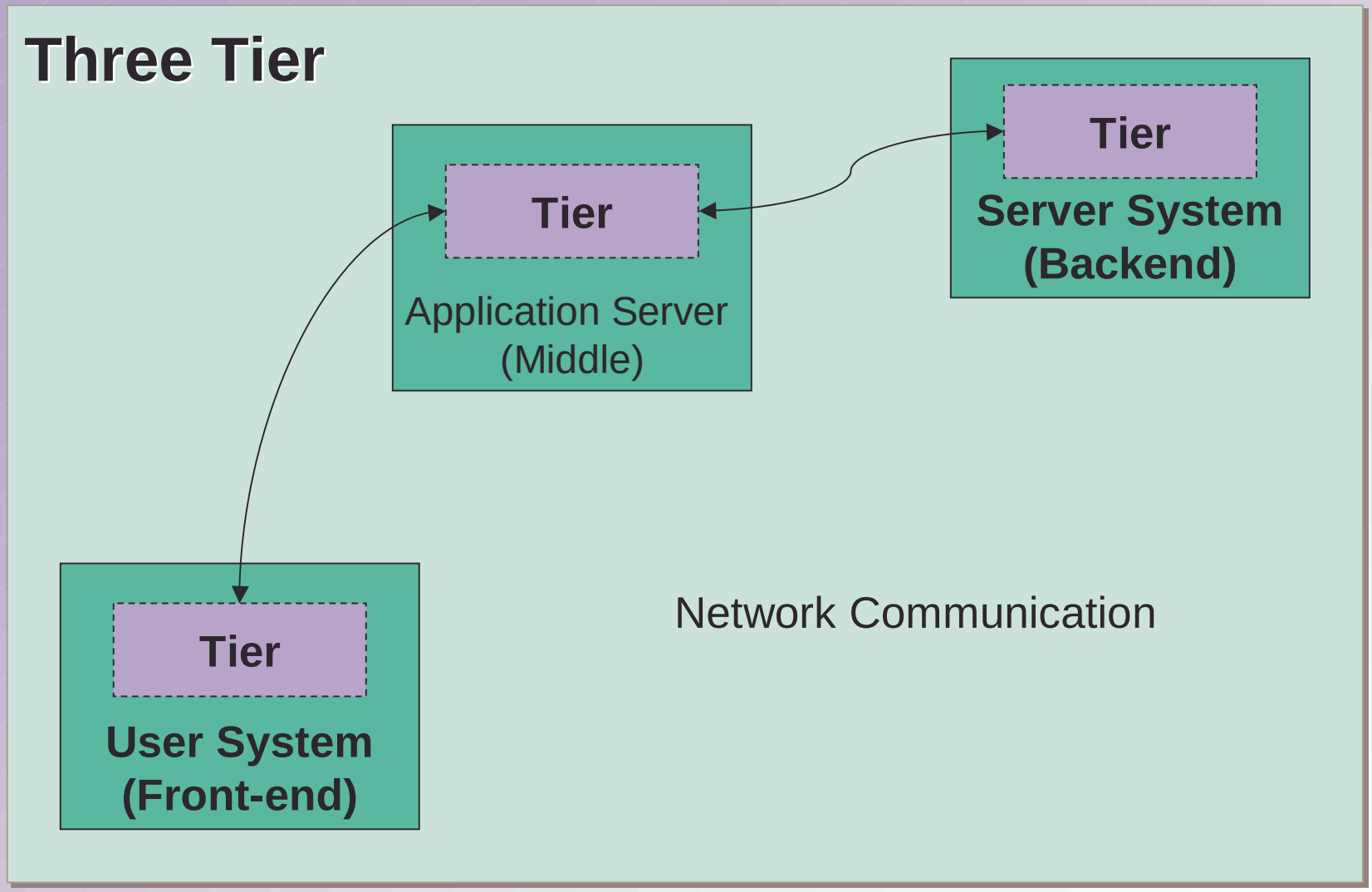
Three Tier

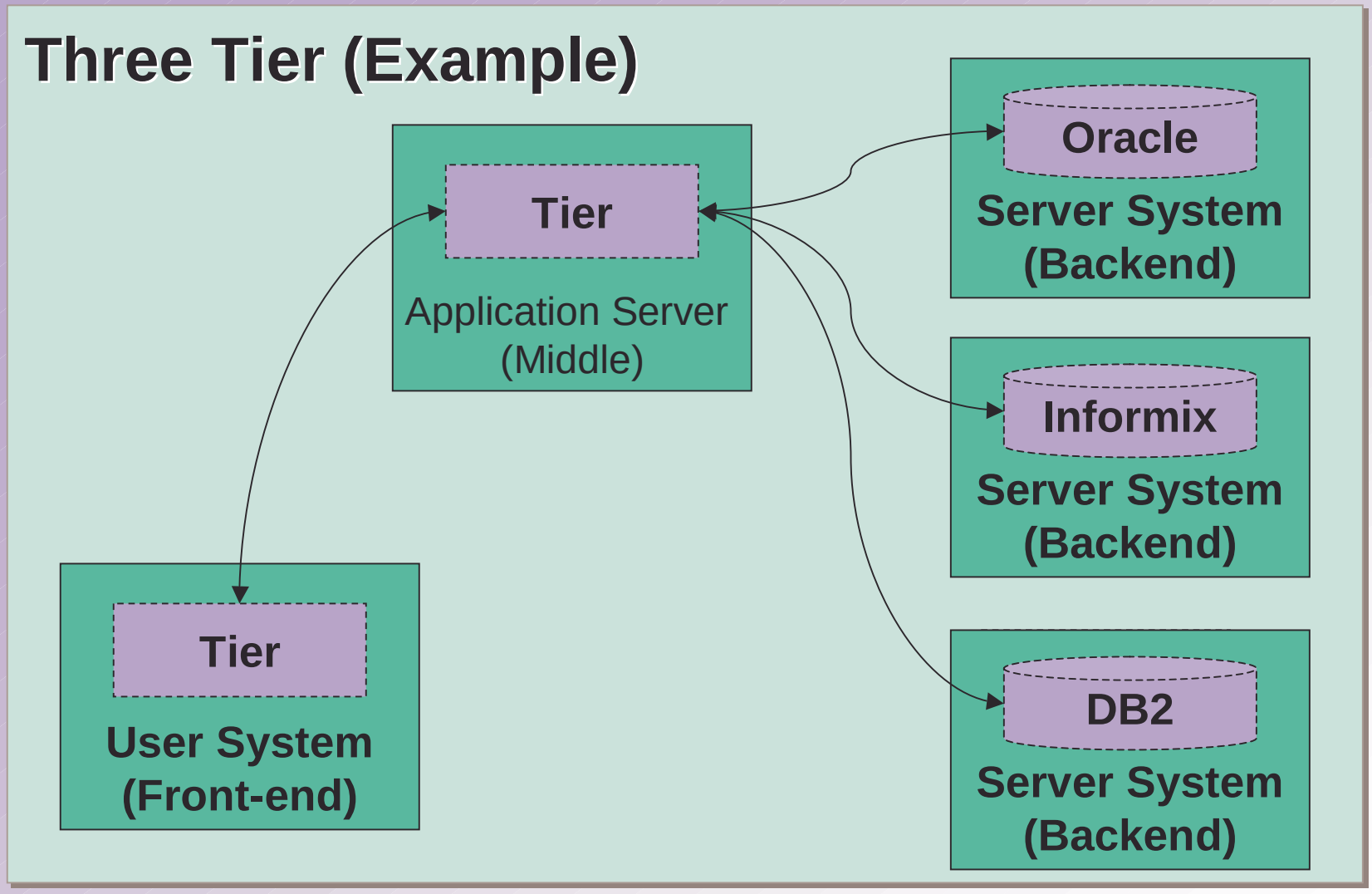
- คือโปรแกรมที่มีโปรเซสในการทำงานบนระบบคอมพิวเตอร์สามโปรเซส
- โปรเซสหนึ่งทำงานในฉากหลังสำหรับการประมวลผล โปรเซสที่สองทำงานอยู่ฉากหน้าสำหรับทำงานในด้านติดต่อกับผู้ใช้ และอีกโปรเซสอยู่ระหว่างกลางของทั้งสองทำหน้าที่แปลการติดต่อสื่อสารทั้งสองให้เข้าใจกัน

1. Tiers

Three Tier

- โปรเซสทั้งสามอาจทำงานบนเครื่องคอมพิวเตอร์เครื่องเดียวกันหรือคนละเครื่องก็ได้ ถ้าโปรเซสทำงานคนละเครื่องต้องอาศัยระบบเครือข่ายคอมพิวเตอร์ช่วยในการสื่อสารระหว่างโปรเซส
- เหมาะสำหรับองค์กรหรือหน่วยงานที่มีโปรเซสฉากหลัง (หรือเซิร์ฟเวอร์หลายระบบ)
- ลักษณะการจัดเทียร์แบบนี้ ช่วยให้การปรับเปลี่ยนระบบเซิร์ฟเวอร์ได้สะดวกและง่าย


Three Tier
Tier
Server System
(Backend)
Tier
Application Server
(Middle)
Tier
User System
(Front-end)
Network Communication

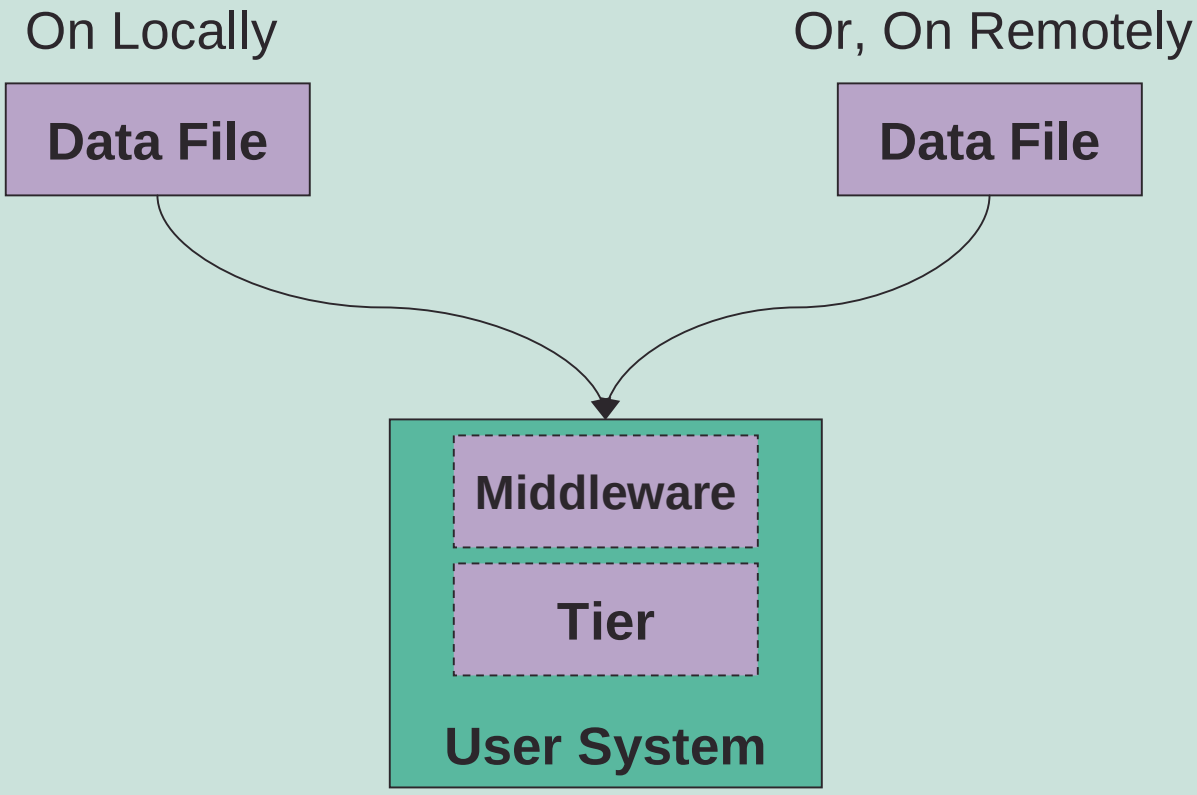

Three Tier (Example)
Tier
Application Server
(Middle)
Oracle
Server System
(Backend)
Informix
Server System
(Backend)
DB2
Server System
(Backend)
Tier
User System
(Front-end)

2. Middleware

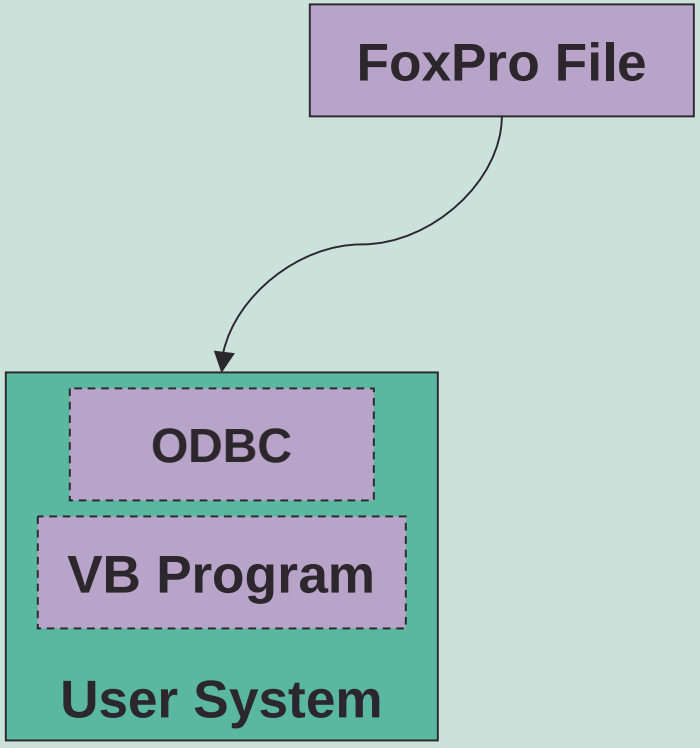
มิดเดิลแวร์

- คือส่วนย่อยที่ใช้เชื่อมการสื่อสารระหว่างสองระบบที่ไม่เข้าใจกัน
- ส่วนใหญ่เห็นได้จากการใช้งานของระบบฐานข้อมูล เช่นการสร้างโปรเซสส่วนติดต่อกับผู้ใช้ด้วย VB แต่ต้องการติดต่อกับฐานข้อมูลระบบ Paradox มิดเดิลแวร์จะสามารถทำให้ VB เข้าใจรูปแบบการจัดเก็บฐานข้อมูลของ Paradox

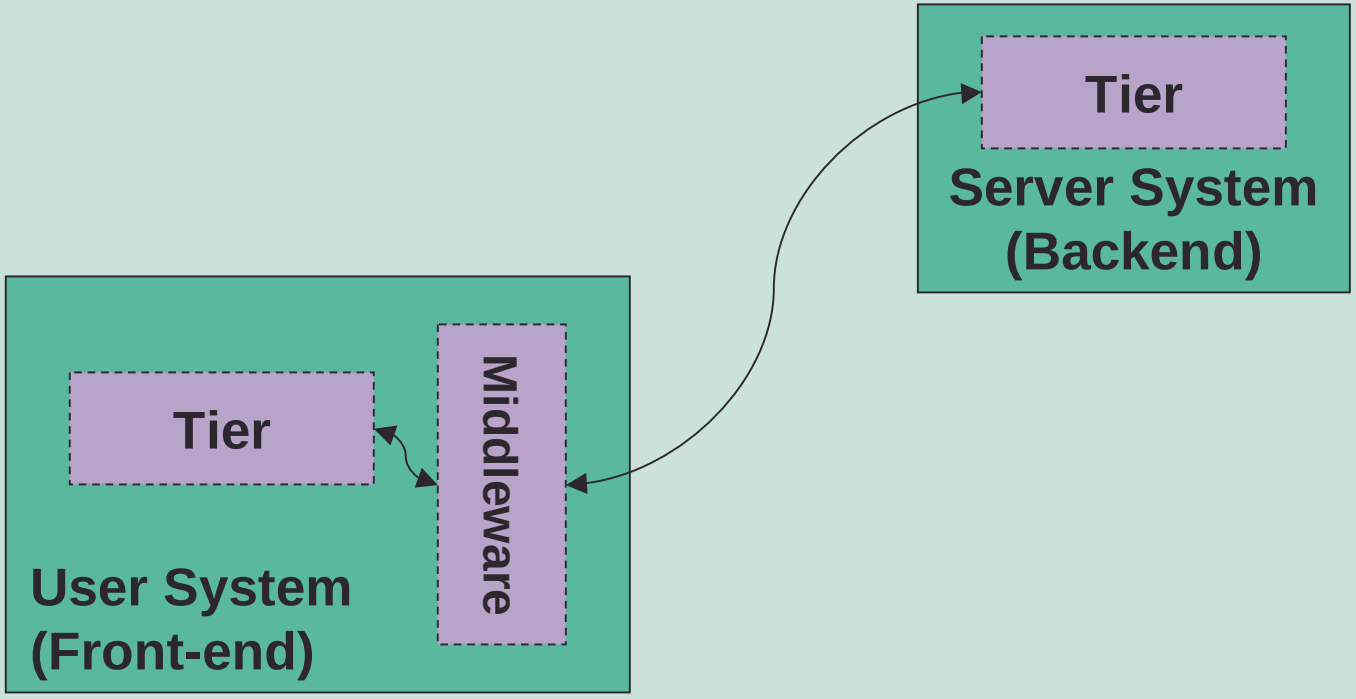
Middleware vs. Single Tier



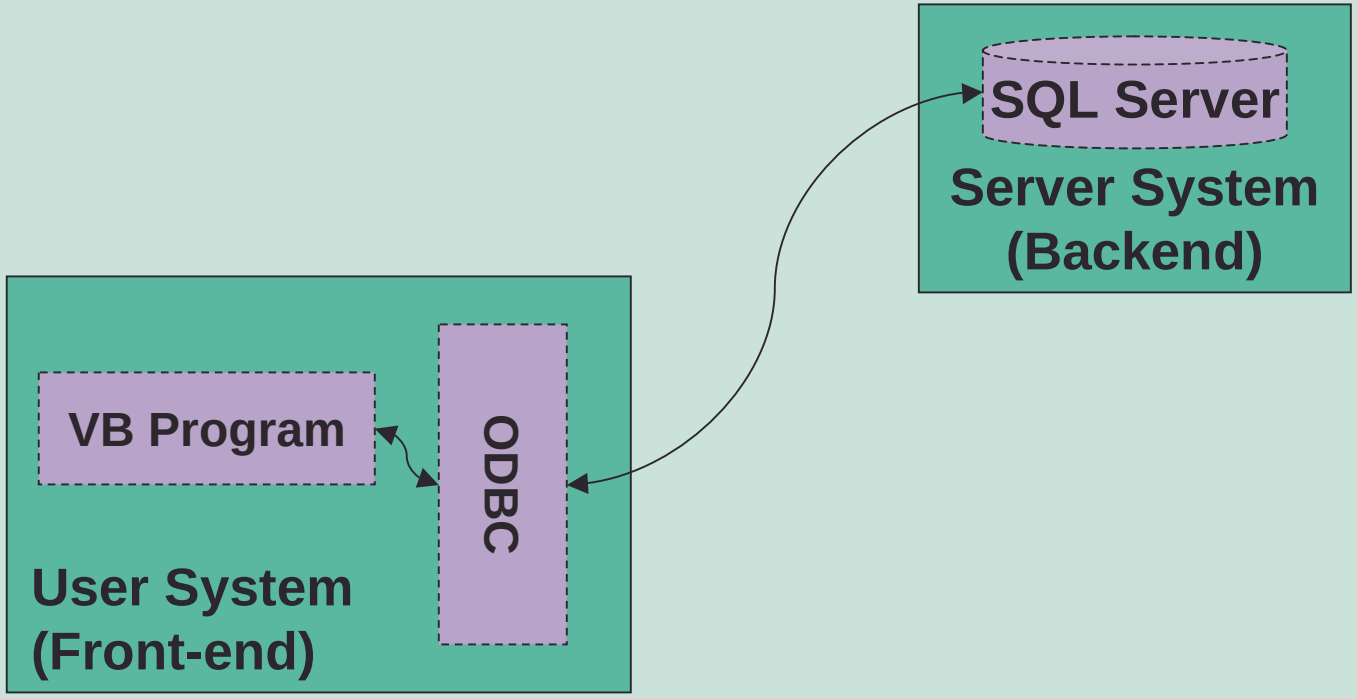
Middleware vs. Single Tier (Example)



Middleware vs. Two Tier

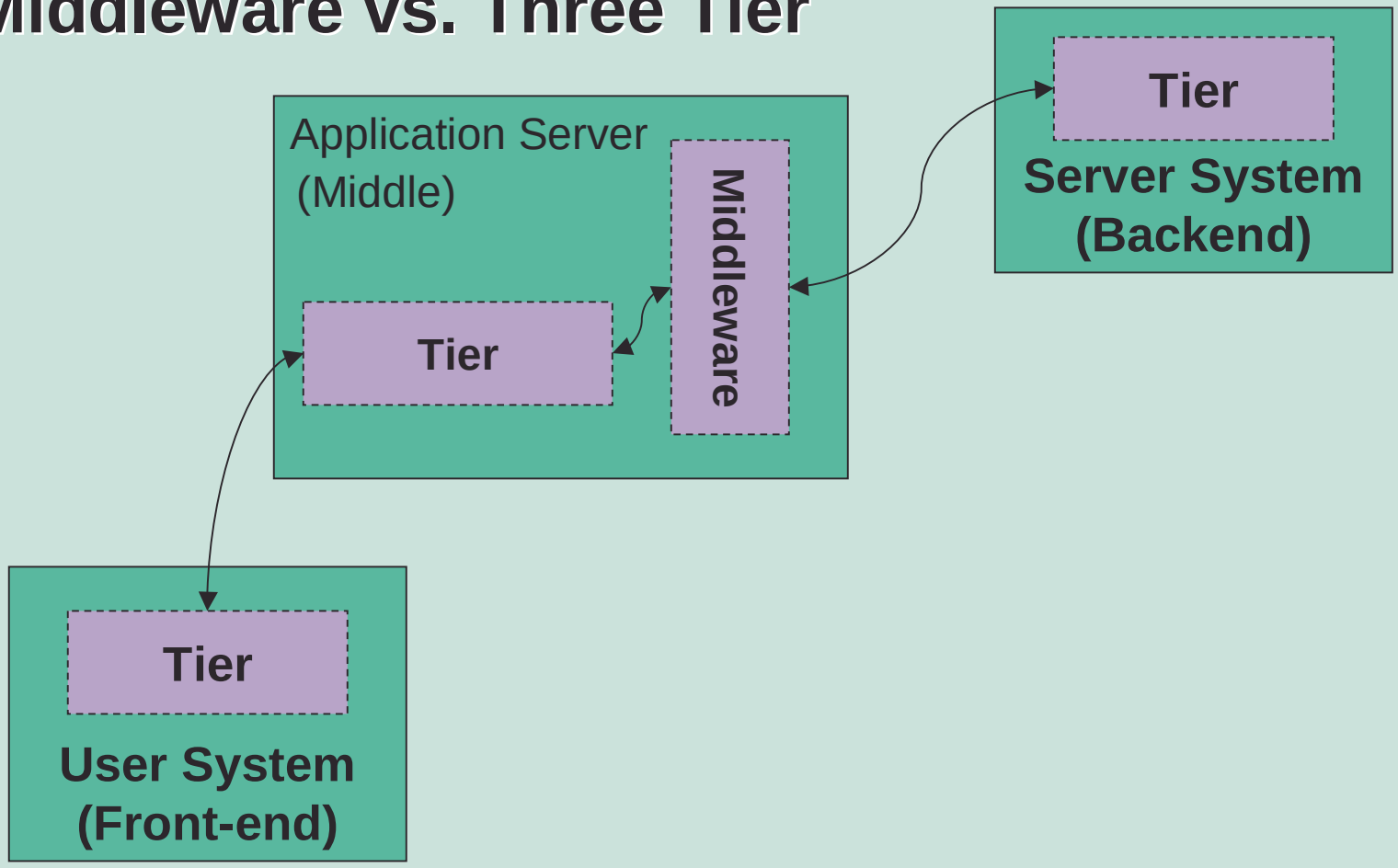


Middleware vs. Two Tier (Example)

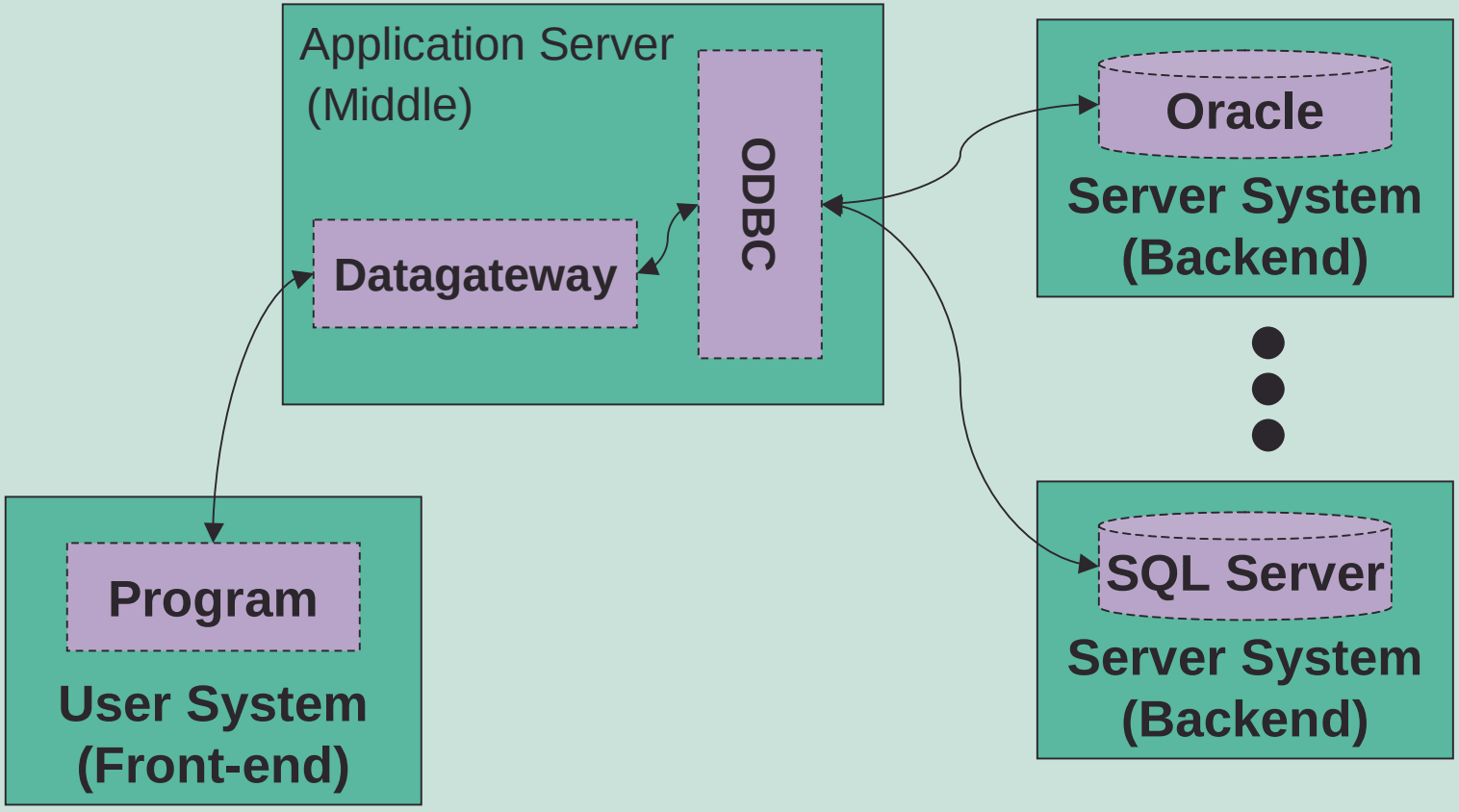


2. Middleware

Middleware vs. Three Tier



Middleware vs. Three Tier (Example)



3. JDBC Type

JDBC(Java DataBase Connectivity)

- คือมิดเดิลแวร์ที่สร้างมาจากภาษาจาวา
- ทำให้ JDBC สามารถใช้งานได้หลายแพลตฟอร์ม เนื่องมาจากข้อกำหนด Write Once, Run Anywhere
- JDBC เป็นรูปแบบในการติดต่อกับฐานข้อมูล มีการแบ่งวิธีการทำงานออกเป็น 4 Type คือ Type 1, Type 2, Type 3 และ Type 4
- JDBC Driver คือไดร์เวอร์ส่วนที่จะนำมาประกอบกับรูปแบบ JDBC ให้เข้าใจฐานข้อมูล เช่น Oracle JDBC Driver คือไดรเวอร์ที่เข้าใจรูปแบบการเก็บข้อมูลของ Oracle

3. JDBC Type

JDBC Type 1(JDBC-ODBC Bridge)

- คือ JDBC ที่ทำงานบน ODBC อีกที
- ช่วยให้สามารถนำเอาฐานข้อมูลเดิมที่ทำงานอยู่บนพื้นฐานของวินโดว์มาใช้งาน Java ได้
- เพื่อหลีกเหลี่ยงปัญหาจากความจำเป็นต้องปรับเปลี่ยนระบบทั้งระบบเป็นการปรับเปลี่ยนเพียงบางส่วน

3. JDBC Type

JDBC Type 1(JDBC-ODBC Bridge)

- ไม่เหมาะกับการนำไปใช้งานบนระบบงานที่มีขนาดใหญ่ๆ เนื่องจากทำให้เกิดความช้าในการทำงาน และประสิทธิภาพในการทำงานที่ไม่ดี
- มีข้อมูลในส่วนโอเวอร์เฮด(Overhead)สูง เนื่องจากต้องมีส่วนในการติดต่อระหว่าง JDBC และ ODBC เพิ่มเติม
- ไม่สนับสนุนความสามารถทั้งหมดของมาตรฐาน JDBC เนื่องจากข้อจำกัดของ ODBC

3. JDBC Type

JDBC Type 2(Partial Java Driver)

- มีประสิทธิภาพในการดีกว่าประเภท 1 เมื่อเปรียบเทียบกัน
- คำสั่งในการติดต่อกับเซิร์ฟเวอร์ จะเป็นคำสั่งที่ดีที่สุดสำหรับเซิร์ฟเวอร์นั้นๆโดยเฉพาะ ทำการทำงานโดยรวมดีกว่า
- ผู้ใช้โปรแกรมในส่วนไคลแอนท์ยังต้องการไดร์เวอร์ สำหรับเซิร์ฟเวอร์โดยเฉพาะ
- เมื่อมีการเปลี่ยนแปลงเซิร์ฟเวอร์เป็นผลิตภัณฑ์ตัวอื่น โปรแกรมในส่วนไคลแอนท์ต้องมีการเปลี่ยนแปลงและคอมไพล์ใหม่เสมอ

3. JDBC Type

JDBC Type 3(Pure Java Driver)

- ระบบที่มีประสิทธิภาพในการทำงานที่ดีกว่าประเภทที่ 1 และ 2 เมื่อเปรียบเทียบกัน
- เหมาะสำหรับองค์กรที่มีเซิร์ฟเวอร์ทางด้านฐานข้อมูลที่หลากหลายรูปแบบ
- การทำงานของไคลแอนท์ไม่จำเป็นต้องติดตั้ง JDBC ไดร์เวอร์ไว้ในทุกตัว
- การติดตั้งและการดูแลระบบไคลแอนท์ทำได้ง่าย และสะดวก
- ยังต้องการไดร์เวอร์สำหรับแต่ละผลิตภัณฑ์ เพื่อติดตั้งไว้ที่แอพปลิเคชั่นเซิร์ฟเวอร์

3. JDBC Type

JDBC Type 4(Direct-to-DB/Native Protocol)

- มีประสิทธิภาพในการทำงานดีที่สุด
- มีความยุ่งยากในการพัฒนา เพราะผู้พัฒนาต้องเรียนรู้การทำงานของเซิร์ฟเวอร์มาก่อน

4. JDBC Steps

ขั้นตอนการสร้างโปรแกรมติดต่อกับระบบฐานข้อมูล

- ขั้นตอนเริ่มติดต่อกับฐานข้อมูล (Establish Database)
- ขั้นตอนการส่งคำสั่ง SQL ให้กับฐานข้อมูล (Send Query)
- ขั้นตอนการรับผลลัพธ์จากฐานข้อมูล (Receive and Display Result)
- ขั้นตอนการปิดการติดต่อกับฐานข้อมูล (Close Connection)

4. JDBC Steps

A. Establish Database

ประกอบด้วยขั้นตอนย่อย 2 ขั้นตอนคือ

ขั้นตอนการโหลดไดรเวอร์

รูปแบบ

```
Class.forName("Driver_Name")
```

เมื่อ

Driver_Name	คือชื่อไดร์เวอร์ที่ต้องการโหลดมาใช้งาน จะต้องระบุชื่อให้ตรงกับระบบฐานข้อมูลที่ต้องการติดต่อ

4. JDBC Steps

A. Establish Database

ตัวอย่าง

1. Class.forName(“sun.jdbc.odbc.dbAccounting”) หรือ
2. Class.forName(“jdbc.DatabaseDriver”) หรือ
3. try { Class.forName(“myDriver.className”) }

```
catch (java.lang.ClassNotFoundException e) {
        System.err.print(e.getMessage());
}
```

4. JDBC Steps

A. Establish Database

ขั้นตอนการเปิดช่องทางการสื่อสาร

ต้องอาศัยข้อมูล 3 ตัวคือ

URL หมายถึงข้อมูลที่ระบบเส้นทางหรือแอดเดรสในระดับเครือข่าย

กรณีที่ใช้แบบ JDBC-ODBC URL เป็นรูปแบบ jdbc:odbc:DSN

กรณีทั่วไปใช้รูปแบบ jdbc:subprotocol

User หมายถึงข้อมูลแสดงตัวผู้ใช้ (ได้จากผู้ดูแลระบบ)

Password หมายถึงข้อมูลรหัสผ่าน (ได้จากผู้ดูแลระบบ)

4. JDBC Steps

A. Establish Database Step

รูปแบบ

```
Connection Instance_Name =
DriverManager.getConnection(URL, Login, Password)
```

ตัวอย่าง

```
String URL = “jdbc:odbc:myData”;
Connection con = DriverManager.getConnection(URL,
“myLogin”, “myPassword”);
```

4. JDBC Steps

B. Send SQL Step

ทำได้โดยการเรียกใช้เมธรอดเหล่านี้

- เมื่อส่งคำสั่งในรูปแบบ Query (SELECT…) เรียก **executeQuery**
- เมื่อส่งคำสั่งในรูปแบบ Update (CREATE, ALTER, DROP, DELETE…) เรียก **executeUpdate**

ทั้ง 2 เมธรอดเรียกใช้งานได้จากคลาส Statement หรือ PreparedStatement

4. JDBC Steps

B. Send SQL Step

ตัวอย่าง

การสร้างตารางชื่อ Students ประกอบด้วยฟิลด์ ID และ Name

```
Statement stmt = con.CreateStatement();

stmt.executeUpdate("CREATE TABLE STUDENTS " +

    "( ID VARCHAR(5), " +" NAME VARCHAR(30) ) " );
```

4. JDBC Steps

B. Send SQL Step

ตัวอย่าง

การแทรกข้อมูลหนึ่งเรคอร์ดในตาราง Student

```
stmt.executeUpdate("INSERT INTO STUDENTS " +
    "('01011', 'Bill Gates')" );
```

4. JDBC Steps

B. Send SQL Step

- การเรียกใช้เมธรอด executeQuery ซึ่งเมื่อส่งคำสั่ง SQL ประเภท SELECT จะได้รับข้อมูลกลับมา
- คลาสที่ใช้รับข้อมูลกลับคือ ResultSet
- เมธรอด executeQuery จะสร้าง Instance ให้หลังจากได้รับข้อมูลกลับ
- ดังนั้นเพียงแค่สร้างตัวอ้างถึงก็สามารถรับ Instance ดังกล่าวได้ดังนี้

```
ResultSet Ref_Name = Statement.executeQuery(…)
```

4. JDBC Steps

B. Send SQL Step

ตัวอย่าง

<u>การส่งคำสั่ง SQL ประเภท SELECT</u>

```
ResultSet rs = stmt.executeQuery("SELECT * " +
    " FROM STUDENTS" );
```

4. JDBC Steps

C. Receive and Display Result

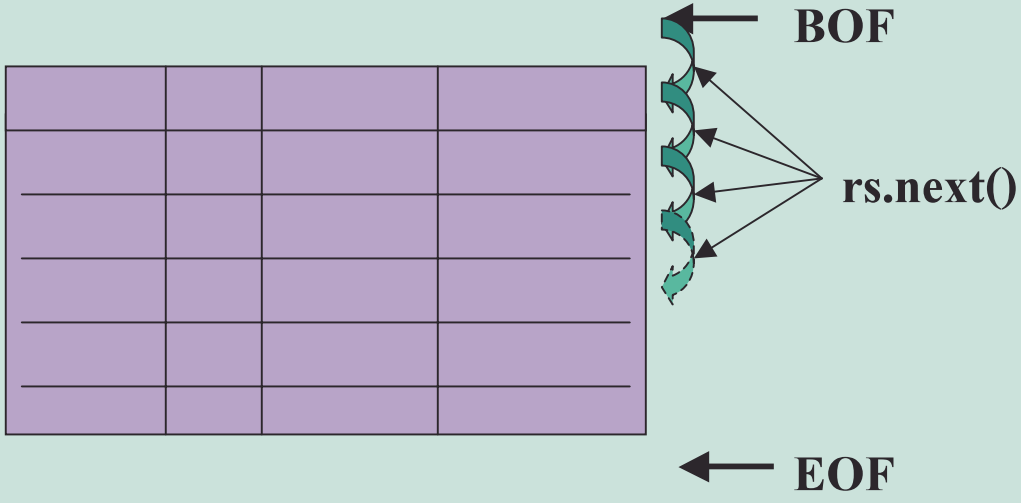
ข้อมูลที่ได้รับกลับจากเมธรอด executeQuery จะอยู่ในรูปของ Record Set โดยมีตัวชี้อ้างไปยังส่วนต้นของชุดข้อมูล โดยใช้ร่วมกับเมธรอดที่เกี่ยวข้องคือ

next()	สำหรับเลื่อนตัวชี้ไปยังเรคอร์ดถัดไป
getXXX	สำหรับการดึงข้อมูลตามชนิดของมูลในฟิลด์
getString	สำหรับ CHAR, VARCHAR
getInt	สำหรับ INTEGER
getFloat	สำหรับ REAL
getDate	สำหรับ DATE

4. JDBC Steps

C. Receive and Display Result

```
while (rs.next()) {        rs.getXXX(Field_Name_1)
                           rs.getXXX(Field_Name_2
}
```



4. JDBC Steps

D. Close Connection

- คือการปิดการใช้งาน Instance ที่ใช้สร้าง Statement และสร้างเส้นทางการสื่อสาร
- ใช้งานผ่านเมธรอด close()
- ตัวอย่าง

```
stmt.close();
conn.close()
```

5. JDBC Example

CreateCoffees.java

```
import java.sql.*;
public class CreateCoffees {
   public static void main(String args[]) {
      String url = "_____"
      Connection con; String createString; Statement stmt;
      createString =       "CREATE TABLE Coffees " +
                           "(Cof_Name      VARCHAR(32), " +
                           "Sup_ID         INTEGER, " +
                           "Price          FLOAT, " +
                           "Sales          INTEGER, " +
                           "Total          INTEGER)";
```

5. JDBC Example

```
    try {      Class.forName("_____________");    }
    catch {
        System.err.print("ClassNotFoundException: ");
        System.err.print(e.getMessage());
    }
    try {    con = DriverManager.getConnection(url,
                    "_________", "_________");
             stmt = con.createStatement();
             stmt.executeUpdate(createString);
             stmt.close(); con.close();
     } catch (SQLException ex) {
         System.err.println("SQLException: " + ex.getMessage()); }
  }
}
```

5. JDBC Example

InsertCoffees.java

```
import java.sql.*;
public class InsertCoffees {
   public static void main(String args[]) {
      String url = "_____"
      Connection con; String query; Statement stmt;
      query = "SELECT Cof_Name, Price FROM Coffees";
      try {      Class.forName("_____");    }
      catch {
           System.err.print("ClassNotFoundException: ");
           System.err.print(e.getMessage());
      }
      try {
          DriverManager.getConnection(url, "____", "_____");
```

5. JDBC Example

```
stmt = con.createStatement();
stmt.executeUpdate("Insert into Coffees " +
   "VALUES('Colombian', 101, 7.99, 0, 0)");
stmt.executeUpdate("Insert into Coffees " +
   "VALUES('French_Roast', 49, 8.99, 0, 0)");
stmt.executeUpdate("Insert into Coffees " +
   "VALUES('Espresso', 150, 9.99, 0, 0)");
stmt.executeUpdate("Insert into Coffees " +
   "VALUES('Colombian_Decaf', 101, 8.99, 0, 0)");
stmt.executeUpdate("Insert into Coffees " +
   "VALUES('French_Roast_Decaf', 49, 9.99, 0, 0)");
```

5. JDBC Example

```
        ResultSet rs = stmt.executeQuery(query);
        System.out.println("Coffee break coffees and prices: ");
        while (rs.next()) {
           String s = rs.getString("Cof_Name");
           float f = rs.getFloat("Price");
           System.out.println(s + "      " + f);
        }
        stmt.close(); con.close();
     } catch (SQLException ex) {
           System.err.println("SQLException: " + ex.getMessage()); }
  }
}
```

5. JDBC Example

java CreateCoffees
java InsertCoffees

Coffee break Coffees and Prices:
Colombian 7.99
French_Roast 8.99
Espresso 9.99
Colombian_Decaf 8.99
French_Roast_Decaf 9.99