

Arsitektur Aplikasi Perangkat Enterprise JDBC



Antonius Rachmat C, S.Kom, M.Cs

JDBC

- ❑ Java Database Connectivity?
- ❑ Java menyediakan JDBC yang berfungsi untuk berhubungan dengan database.
- ❑ Database yang didukung oleh Java cukup banyak, seperti : MySQL, Postgres, Oracle, DB2, Access dan lain-lain.
- ❑ JDBC berisi kumpulan kelas-kelas dan interface yang ditulis dengan bahasa Java.

JDBC (2)

- Yang dilakukan JDBC
 - Membangun koneksi ke data source
 - Mengirim statement ke data source
 - Memproses hasil statement tersebut

- Java menyediakan tiga produk JDBC:
 - JDBC driver manager
 - JDBC driver test suite
 - JDBC ODBC bridge

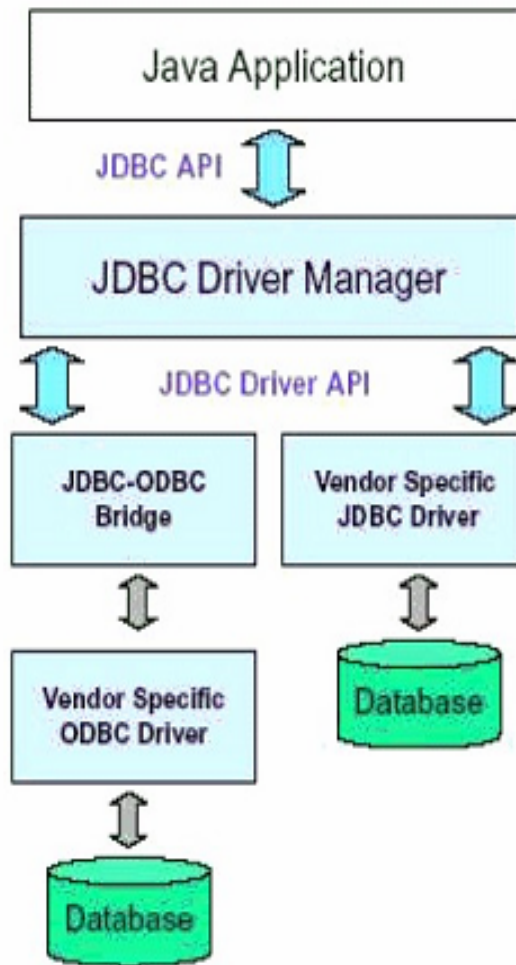
ODBC vs JDBC

- ❑ ODBC tidak cocok dipakai langsung dengan Java karena ditulis dengan bahasa C, pemanggilan dari Java ke C memiliki masalah keamanan, implementasi, robustness, dan portabilitas sistem.
- ❑ Penerjemahan dari C ke Java tidak akan berhasil baik.
 - Contoh: Java tidak memiliki pointer.
- ❑ ODBC **sulit dipelajari** karena optionnya yang sulit walaupun untuk query yang sederhana.
- ❑ Java API diperlukan untuk mempertahankan solusi "murni Java", agar dapat berjalan di berbagai platform. Karena ODBC harus diinstall dahulu di setiap client dan tidak semua platform.

Keunggulan JDBC

- ❑ Mempertahankan data perusahaan yang ada
- ❑ Menyederhanakan development perusahaan
- ❑ Tidak memerlukan konfigurasi pada jaringan komputer
- ❑ Akses penuh ke meta data
- ❑ Koneksi database menggunakan URL dan DataSource (yang menyediakan connection pooling dan distributed transaction)

Arsitektur JDBC



Lapisan Vendor Specific JDBC Driver merupakan driver JDBC yang dikeluarkan oleh para vendor pengembang RDBMS.

Sedangkan JDBC- ODBC Bridge berfungsi sebagai perantara untuk mengakses database melalui ODBC driver.

Baik JDBC driver maupun JDBC-ODBC Bridge diatur dan dapat diakses melalui JDBC Driver Manager.

Aplikasi yang kita kembangkan untuk mengakses database dengan memanfaatkan JDBC akan berinteraksi dengan JDBC Driver Manager.

JDBC API

- ❑ Tersedia dalam paket `java.sql` dan `javax.sql`.
- ❑ `DriverManager` – memanggil driver JDBC ke memori, dan dapat juga digunakan untuk membuka koneksi ke sumber data.
- ❑ `Connection` – mempresentasikan suatu koneksi dengan suatu data source, juga digunakan untuk membuat objek `Statement`, `PreparedStatement` dan `CallableStatement`.
- ❑ `Statement` – mempresentasikan suatu perintah SQL, dan dapat digunakan untuk menerima objek `ResultSet`.

JDBC API (2)

- ❑ PreparedStatement – merupakan alternatif untuk objek Statement SQL yang telah terkompilasi awal.
- ❑ CallableStatement – mempresentasikan suatu stored procedure, dan dapat digunakan untuk menjalankan stored procedures yang terkompilasi dalam suatu RDBMS yang mendukung fasilitas tersebut.
- ❑ ResultSet – mempresentasikan sebuah hasil dari database yang dihasilkan dari statemen SQL SELECT.
- ❑ ResultSetMetaData – mempresentasikan hasil informasi metadata dari kolom-kolom dalam suatu table
- ❑ SQLException – suatu class exception yang membungkus kesalahan (error) pengaksesan database.

JDBC Data Type

JDBC Type	Java Type
BIT	boolean
TINYINT	byte
SMALLINT	short
INTEGER	int
BIGINT	long
REAL	float
FLOAT DOUBLE	double
BINARY VARBINARY LONGVARBINARY	byte[]
CHAR VARCHAR LONGVARCHAR	String

JDBC Type	Java Type
NUMERIC DECIMAL	BigDecimal
DATE	java.sql.Date
TIME TIMESTAMP	java.sql.Timestamp
CLOB	Clob*
BLOB	Blob*
ARRAY	Array*
DISTINCT	mapping of underlying type
STRUCT	Struct*
REF	Ref*
JAVA_OBJECT	underlying Java class

*SQL3 data type supported in JDBC 2.0

Pemrograman JDBC

□ Membangun koneksi

■ Memuat driver ODBC

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

■ Atau

```
DriverManager.registerDriver(new sun.jdbc.odbc.JdbcOdbcDriver ());
```

□ Membangun koneksi URL

Format: jdbc:odbc:<nama_db>

□ Contoh lengkap:

```
String url = "jdbc:odbc:Buku";
```

```
String user = "";
```

```
String pass = "";
```

```
Connection con =
```

```
DriverManager.getConnection(url, user, pass);
```

Akses meta data (optional)

```
DatabaseMetaData dbMetaData = connection.getMetaData();  
String productName =  
    dbMetaData.getDatabaseProductName();  
System.out.println("Database: " + productName);  
String productVersion =  
    dbMetaData.getDatabaseProductVersion();  
System.out.println("Version: " + productVersion);
```

Pemrograman JDBC (2)

- ❑ Membuat Statement
Menggunakan Obyek Connection yang sudah kita buat sebelumnya:
 - `Statement stmt = con.createStatement();`
- ❑ Menjalankan Statement
- ❑ Method **executeUpdate** untuk DDL dan DML insert, update, dan delete.
 - `String query = "delete from tabel where id=1";`
 - `Statement stmt = con.createStatement();`
 - `int hsl = Stmt.executeUpdate(query);`
- ❑ Method **executeQuery** untuk DML select
 - `String query = "select * from tabel";`
 - `Statement stmt = con.createStatement();`
 - `ResultSet rs = Stmt.executeQuery(query);`

Pemrograman JDBC (3)

- ❑ Mengambil hasil Statement dari Query dan Memprosesnya
- ❑ DDL dan DML: update, insert, dan delete
 - `int hsl = Stmt.executeUpdate(query);`
 - `if(hsl == 1)`
 - `System.out.println("Berhasil");`
 - `else`
 - `System.out.println("Gagal");`
- ❑ DML: select
 - `ResultSet rs = Stmt.executeQuery(query);`
 - `while(rs.next()){`
 - ❑ `int a = rs.getInt("fieldA");`
 - ❑ `String b = rs.getString("fieldB");`
 - ❑ `float c = rs.getFloat("fieldC");`
 - `}`

Pemrograman JDBC (4)

- Tutup koneksi yang sudah dibuat.
 - `con.close();`
- Kita dapat membuat class yang berisi semua method yang membantu kita untuk melakukan koneksi dan transaksi ke database!

Penting!

- ❑ Harus mengetahui dan memiliki JDBC driver sesuai dengan database yang digunakan.
- ❑ Harus mengetahui cara koneksi dengan database.
- ❑ Harus mengimport `java.sql.*`;

Contoh: MySQL Create Table

```
import java.sql.*;
public class CreateTableSepatuApp {
    public static void main(String[] args) {
        String createTableSepatu =
            "CREATE TABLE SEPATU "
            + "(NAMA_SEPATU VARCHAR(40),"
            + "SUP_ID INTEGER,"
            + "HARGA REAL,"
            + "PENJUALAN INTEGER,"
            + "TOTAL INTEGER)";

        try {
            Class.forName("org.gjt.mm.mysql.Driver");
            Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/sepatudb?user=root&password=");
            Statement stmt = con.createStatement();
            stmt.executeUpdate(createTableSepatu);
            System.out.println("Tabel SEPATU berhasil dibuat.");
            stmt.close();
            con.close();
        } catch (ClassNotFoundException e) {
            System.out.println("Eksepsi: " + e.getMessage());
        } catch (SQLException e) {
            System.out.println("Eksepsi SQL: " + e.getMessage());
        }
    }
}
```

Membaca Isi Data ODBC

```
public class cobaDB {
    public static void main(String[] args) {
        String selectMhs = "select * from mhs";
        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            String url = "jdbc:odbc:mhs";
            String user = "";
            String pass = "";
            Connection con = DriverManager.getConnection(url,user,pass);

            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery(selectMhs);
            while(rs.next()){
                String nim = rs.getString("nim");
                String nama = rs.getString("nama");
                int ipk = rs.getInt("ipk");
                System.out.println(nim + "\t" + nama + "\t" + ipk);
            }
            rs.close();
            stmt.close();
            con.close();
        } catch (ClassNotFoundException e) {
            System.out.println("Eksepsi: " + e.getMessage());
        } catch (SQLException e) {
            System.out.println("Eksepsi SQL: " + e.getMessage());
        }
    }
}
```

PreparedStatement

```
String insertBuku = "insert into buku(KodeBuku, Judul, Penerbit, ThTerbit) values (?, ?, ?, ?)";
PreparedStatement stmt = con.prepareStatement(insertBuku);
stmt.setString(1, "IPA20");
stmt.setString(2, "aaa");
stmt.setString(3, "bbb");
stmt.setString(4, "1900");
int h = stmt.executeUpdate();

if(h==1)
    System.out.println("Berhasil");
else
    System.out.println("Gagal");
```

Transaction

- ❑ `getAutoCommit / setAutoCommit`
- ❑ `commit()`
- ❑ `rollback()`

Tes Performa MySQL InnoDB

Starting testStatement, turnOffAutoCommit:true

Truncating table

Row:0; Row:200; Row:400; Row:600; Row:800; Row:1000; testStatement finish: 531 ms

Starting testStatement, turnOffAutoCommit:false

Truncating table

Row:0; Row:200; Row:400; Row:600; Row:800; Row:1000; testStatement finish: 41125 ms

Starting testPreparedStatement, turnOffAutoCommit:true

Truncating table

Row:0; Row:200; Row:400; Row:600; Row:800; Row:1000; testPreparedStatement finish: 469 ms

Starting testPreparedStatement, turnOffAutoCommit:false

Truncating table

Row:0; Row:200; Row:400; Row:600; Row:800; Row:1000; testPreparedStatement finish: 39078 ms

Starting testBatch, turnOffAutoCommit:true

Truncating table

Row:0; Row:200; Row:400; Row:600; Row:800; Row:1000; testBatch finish: 453 ms

Starting testBatch, turnOffAutoCommit:false

Truncating table

Row:0; Row:200; Row:400; Row:600; Row:800; Row:1000; testBatch finish: 38734 ms

Process completed.

Tes Performa Oracle 10g, (10.2.0)

Starting testStatement, turnOffAutoCommit:true

Truncating table

Row:0; Row:200; Row:400; Row:600; Row:800; Row:1000; testStatement finish: 2078 ms

Starting testStatement, turnOffAutoCommit:false

Truncating table

Row:0; Row:200; Row:400; Row:600; Row:800; Row:1000; testStatement finish: 2594 ms

Starting testPreparedStatement, turnOffAutoCommit:true

Truncating table

Row:0; Row:200; Row:400; Row:600; Row:800; Row:1000; testPreparedStatement finish: 656 ms

Starting testPreparedStatement, turnOffAutoCommit:false

Truncating table

Row:0; Row:200; Row:400; Row:600; Row:800; Row:1000; testPreparedStatement finish: 922 ms

Starting testBatch, turnOffAutoCommit:true

Truncating table

Row:0; Row:200; Row:400; Row:600; Row:800; Row:1000; testBatch finish: 16 ms

Starting testBatch, turnOffAutoCommit:false

Truncating table

Row:0; Row:200; Row:400; Row:600; Row:800; Row:1000; testBatch finish: 16 ms

Callable Statement – stored procedure

Stored Procedure Syntax

- Procedure with no parameters

```
{ call procedure_name }
```

- Procedure with input parameters

```
{ call procedure_name(?, ?, ...) }
```

- Procedure with output parameters

```
{ ? = call procedure_name(?, ?, ...) }
```

CallableStatement statement =

```
connection.prepareCall("{ call procedure(?, ?) }");
```

Contoh callable statement

```
String procedure = "{ ? = call isValidUser(?, ?) }";
CallableStatement statement =
    connection.prepareCall(procedure);
statement.setString(2, username);
statement.setString(3, password);
statement.registerOutParameter(1, Types.BIT);
statement.execute();

if (statement.getBoolean(1)) {
    // Valid Username, password.
    ...
} else {
    // Invalid username, password.
    ...
}
```

Cursor ResultSet

- Method pergerakan kursor yang didukung oleh ResultSet:
 - `previous()` ke record sebelumnya
 - `next()` ke record selanjutnya
 - `first()` ke record pertama
 - `last()` ke record terakhir
 - `absolute()` ke nomor baris tertentu
 - `relative()` ke nomor baris dari baris sekarang
 - `beforeFirst()` ke nomor baris sebelum pertama
 - `afterLast()` ke nomor baris setelah terakhir

Cursor ResultSet

- ❑ Jika suatu ResultSet dibuat, selalu ResultSet tersebut berada pada posisi record sebelum record pertama (`rs.beforeFirst()`).
- ❑ Sehingga untuk mengambil data yang hanya terdiri dari satu baris, harus terlebih dahulu digunakan method `rs.next()` sekali.

Informasi kolom ResultSetMetaData

- getColumnCount
 - Jumlah kolom
- getColumnDisplaySize
 - Jumlah karakter kolom tersebut
- getColumnLabel/getColumnName
 - Nama kolom tersebut
- getColumnType
 - Tipe data kolom tersebut

Cursor ResultSet

- ❑ `getMetaData()`, untuk yg mengembalikan `ResultSetMetaData` mengambil informasi metadata
 - `System.out.println(dbkolom.getColumnNames(1) + "\t" + dbkolom.getColumnNames(2) + "\t" + dbkolom.getColumnNames(3));`
- ❑ Method untuk mengambil jumlah baris:
 - `getRow()` yang mengembalikan nilai integer
- ❑ Method `findColumn(<namastringkolom>)`
 - Kembaliannya int posisi
- ❑ Method untuk membatasi jumlah baris hasil query select:
 - `Statement.setFetchSize(number)`

Contoh ambil nama kolom dinamis

```
// Look up information about a particular table.  
ResultSetMetaData resultsMetaData =  
    resultSet.getMetaData();  
int columnCount = resultsMetaData.getColumnCount();  
// Column index starts at 1 (a la SQL) not 0 (a la Java)  
for(int i=1; i<columnCount+1; i++) {  
    System.out.print(resultsMetaData.getColumnName(i) +  
        " ");  
}
```

setFetchSize()

- setFetchSize() memiliki arah, yaitu:
 - ResultSet.FETCH_FORWARD untuk proses maju
 - ResultSet.FETCH_REVERSE untuk proses berbalik
 - ResultSet.FETCH_UNKNOWN untuk proses yang tidak diketahui

- Contoh:

```
Statement stmt = con.createStatement();  
stmt.setFetchDirection(ResultSet.FETCH_FORWARD);  
stmt.setFetchSize(30);  
ResultSet rs = stmt.executeQuery(...);
```

Kembalian ResultSet

□ null

- Untuk metode getXXX yang mengembalikan obyek

□ 0

- Untuk metode getXXX yang mengembalikan tipe data primitif biasa

□ false

- Untuk metode getXXX yang mengembalikan tipe data boolean.

Exception dalam JDBC

- ❑ SQLException: ketika ada masalah pengaksesan data
- ❑ SQLWarning: ketika ada peringatan
- ❑ DataTruncation: ketika data mungkin terpotong
- ❑ BatchUpdateException: ketika tidak semua perintah update berhasil dilakukan.

Access Database

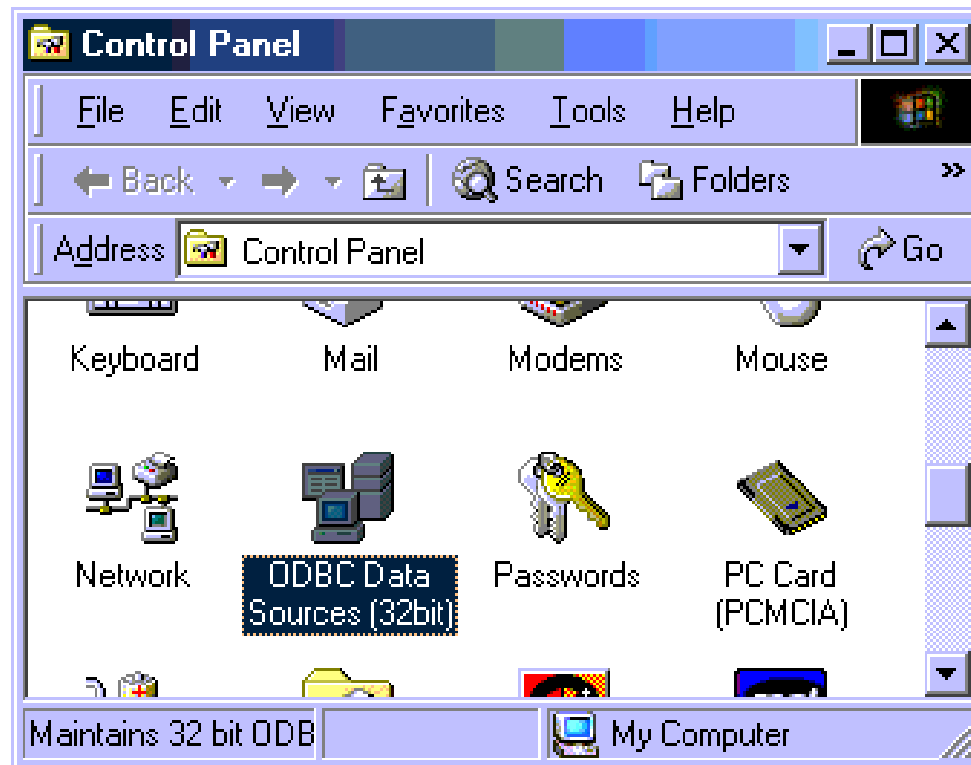
□ Microsoft Access Database : db1

tabel01 : Table					
	Nomer	Judul	Pengarang	Penerbit	harga
▶	1	Java 2 Complete Reference	Patrick Naughton	McGraw Hill	600000
	2	Distributed Systems	Tanenbaum	Prentice Hall	500000
	3	Home Networking Bible	Plumley	IDG Books	375000

tabel01 : Table		
	Field Name	Data Type
▶	Nomer	AutoNumber
	Judul	Text
	Pengarang	Text
	Penerbit	Text
	harga	Number

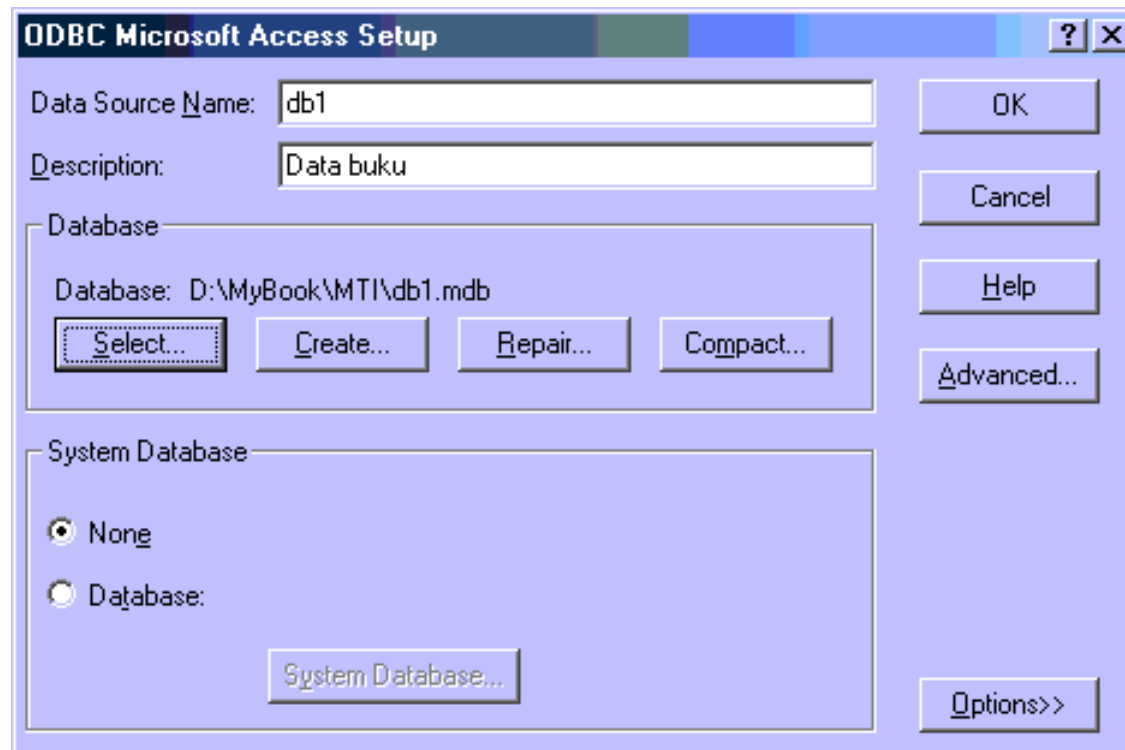
Setting Access Database

- Menset Database yang dipakai sebagai acuan dalam Program
- Start -> Control Panel -> ODBC Data Source

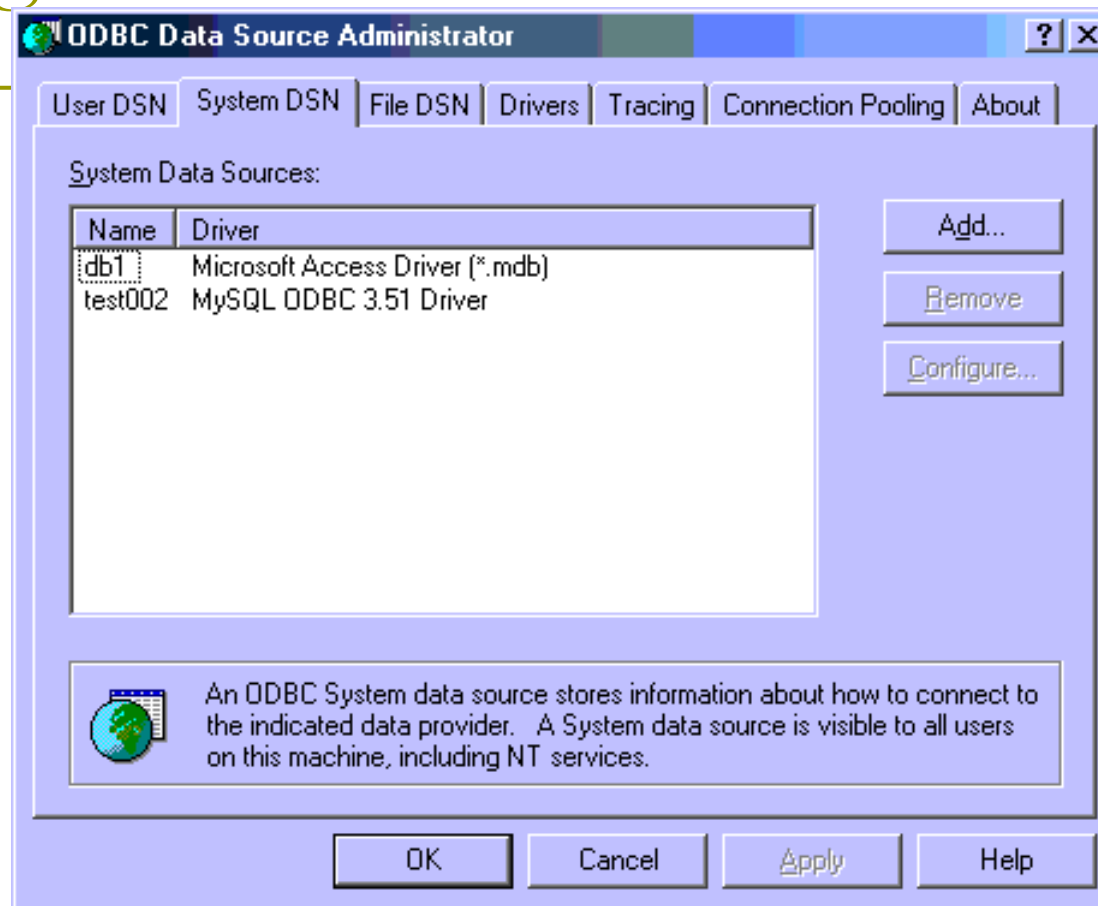


Setting Access Database

- ❑ Tampil jendela ODBC Data Source Administrator
- ❑ Click System DSN-> Add-> pilih Driver (Microsoft Access Driver) -> Finish. Dalam hal ini kita memakai Access Database.
- ❑ Tentukan Letak File Database yang digunakan



Setting Access Database



- ❑ Click OK
- ❑ Data base db1 siap diakses oleh Program

PRAKT JDBC

- Next : Enterprise Application Framework