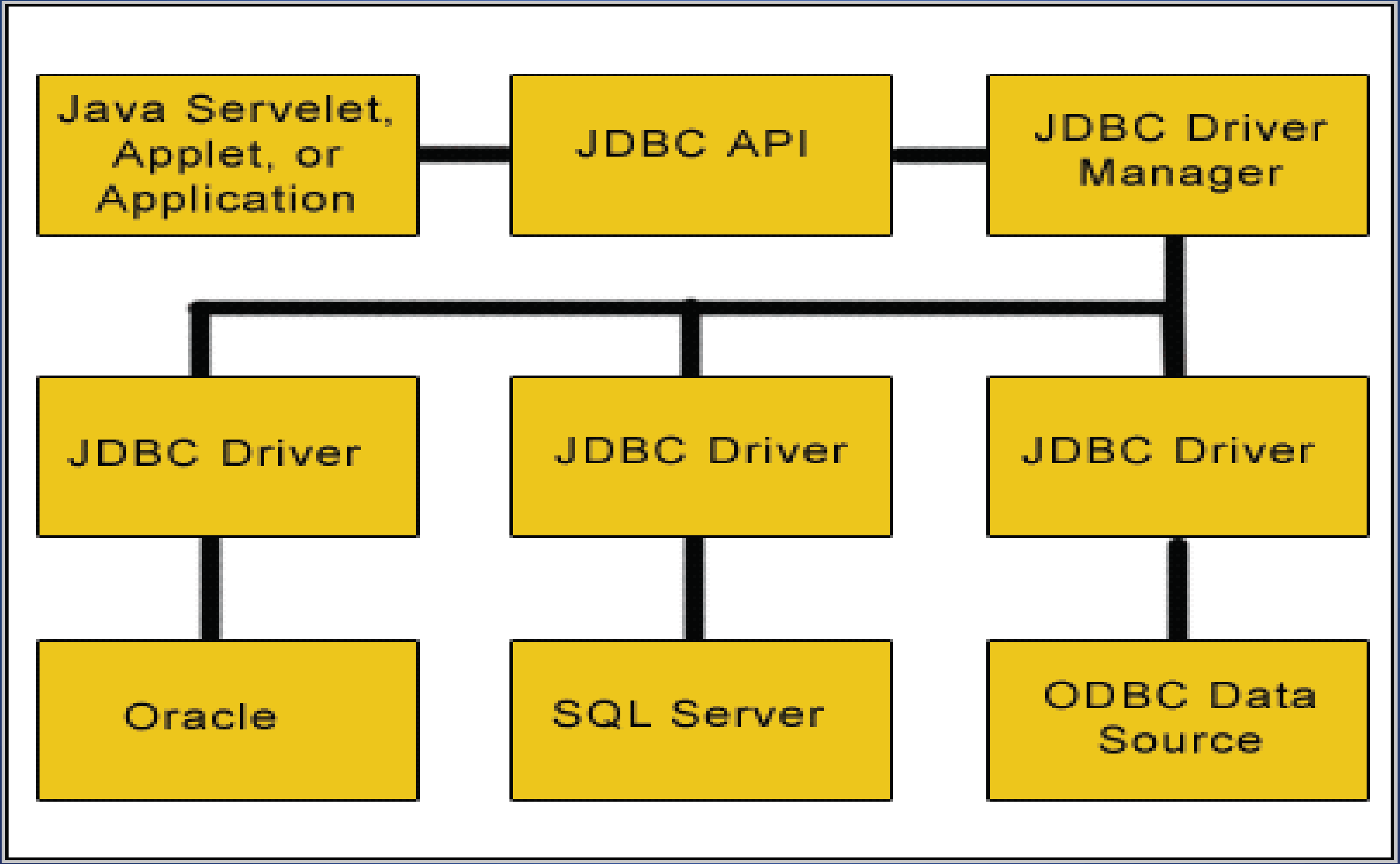
What is JDBC

- **JDBC** is Java application programming interface that allows the Java programmers to access database management system from Java code.
- It is a java API which enables the java programs to execute SQL statements.
- It was developed by **JavaSoft**, a subsidiary of **Sun Microsystems**.

JDBC APIs

- **java.sql**
- **javax.sql**

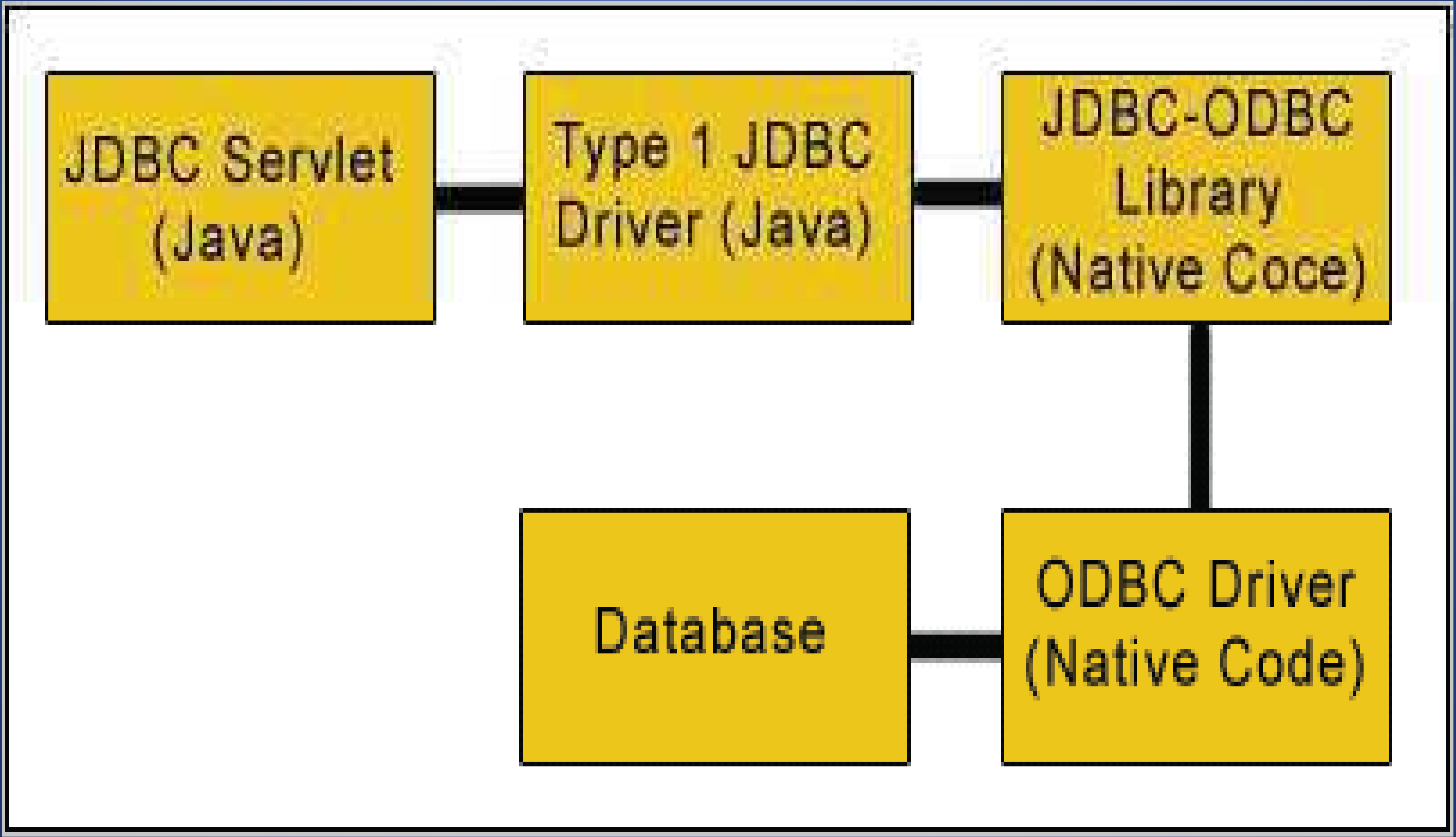
JDBC Architecture



JDBC Drivers

- **JDBC-ODBC Driver (JDBC-ODBC Bridge)**
- **Java/Native Code Driver**
- **Type – 3 JDBC Driver**
- **Type – 4 JDBC Driver**

JDBC-ODBC Driver


JDBC Servlet (Java)
Type 1 JDBC Driver (Java)
JDBC-ODBC Library (Native Coce)
Database
ODBC Driver (Native Code)

DriverManager Class

- Defines objects which connect Java Applications to a JDBC driver.
- Used to provide a means of managing the different types of JDBC database driver running on an application.
- The main responsibility of JDBC database driver is to load all the drivers found in the system properly as well as to select the most appropriate driver from opening a connection to a database.

Connection Interface

- A **Connection** object represents a connection with a database.
- When we connect to a database by using connection method, we create a Connection Object, which represents the connection to the database.
- It is used to create statements, to commit or rollback the jdbc transactions, to provice metadata information and to close the data source.

Statement Interface

- Used to execute SQL queries.
- Three types of statements:-
  - Statement
  - PreparedStatement
  - CallableStatement

ResultSet Interface

- It stores the results derived from the database in the form of number of tuples after the statement object has fired the query.

- We can access our data achieved from database through the ResultSet object only.

JDBC Programming Steps

- Loading Driver
- Establishing Connection
- Executing Statements
- Getting Results
- Closing Database Connection

Loading Drivers

- To initialize the corresponding driver object for our program related to a specific database.
- Class.forName(String *DriverClassPath*);
- E.g.
  - Class.forName(“sun.jdbc.odbc.JdbcOdbcDriver”);
  - Is the driver path to be followed for Jdbc-Odbc Drivers.
  - If drivers are to be changed, the corresponding path will also be changed.

Establishing Connection

- A bridge between the database and the java program is to be established in the form of connection.
- We use *Connection* Inteface to hold the connection object returned by the DriverManager Class.
- E.g.-
  - Connection con = DriverManager.getConnection(String dbUrl, Sting username, String password);

Creating Statements

- Statements are to be built up using createStatement() method of Connection.
  - Statement s = con.createStatement();

- Query is fired with the use of the statement object based on whether the query is of type DDL (select) or DML (insert, update, delete).

Executing Statements

- For DDL Queries:
  - ResultSet rs = s.executeQuery(String *query*);
  - The returned tuples are to be stored in the ResultSet Object.
- For DML Queries:
  - s.executeUpdate();
  - It returns an integer value stating the number of columns have been modified due to the execution of query.

Getting Results

- To get the results, we need to check continuously for availability of data/tuples in ResultSet object by next() method of ResultSet. Then we can get the specific data by getxxx() method of the same.
  – while(rs.next()){

```
while(rs.next()){
            String str = rs.getString(1);
            int I = rs.getInt(2);
}
```

Close Database Connection

- To close the database connection, we need to close the connection object by its close() method. But before that, the ResultSet object and the Statement objects are getting closed by their close() methods to avoid their future use after the connection closing event.
  - rs.close()
    s.close()
    con.close()

Handling Exceptions

- While dealing with the various methods here to load the drivers, to establish the connection, to create connections or to execute the queries, we need to take care about various exceptions thrown by these methods.
- E.g. ClassNotFoundException, SQLException

Types of Connections

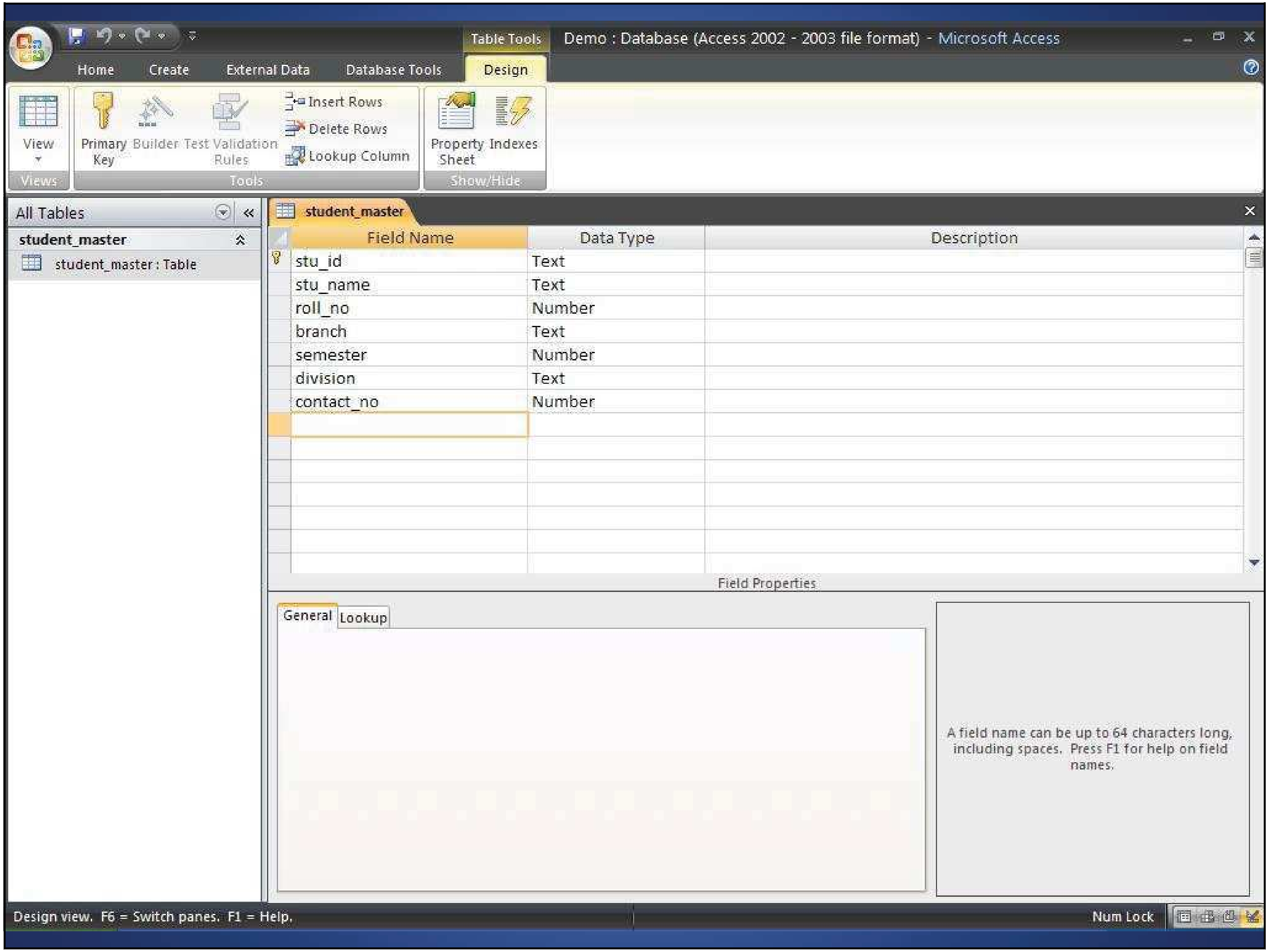
- DSN Oriented Connection
  - A layer of Data Source Name (DSN) is kept between the Java Program and the actual Database.
- DSN Less Connection
  - A direct connection to the Database is managed here by directly specifying the driver path while preparing the connection.

Demo JDBC Program

- Let us have a look to a sample program stating the use of JDBC.
- Here we will connect our java program with Microsoft Access Database and will perform various SQL queries upon the same.
- First we will see the DSN Oriented and later, DSN Less Connectivity with the same database.
- Here we go…

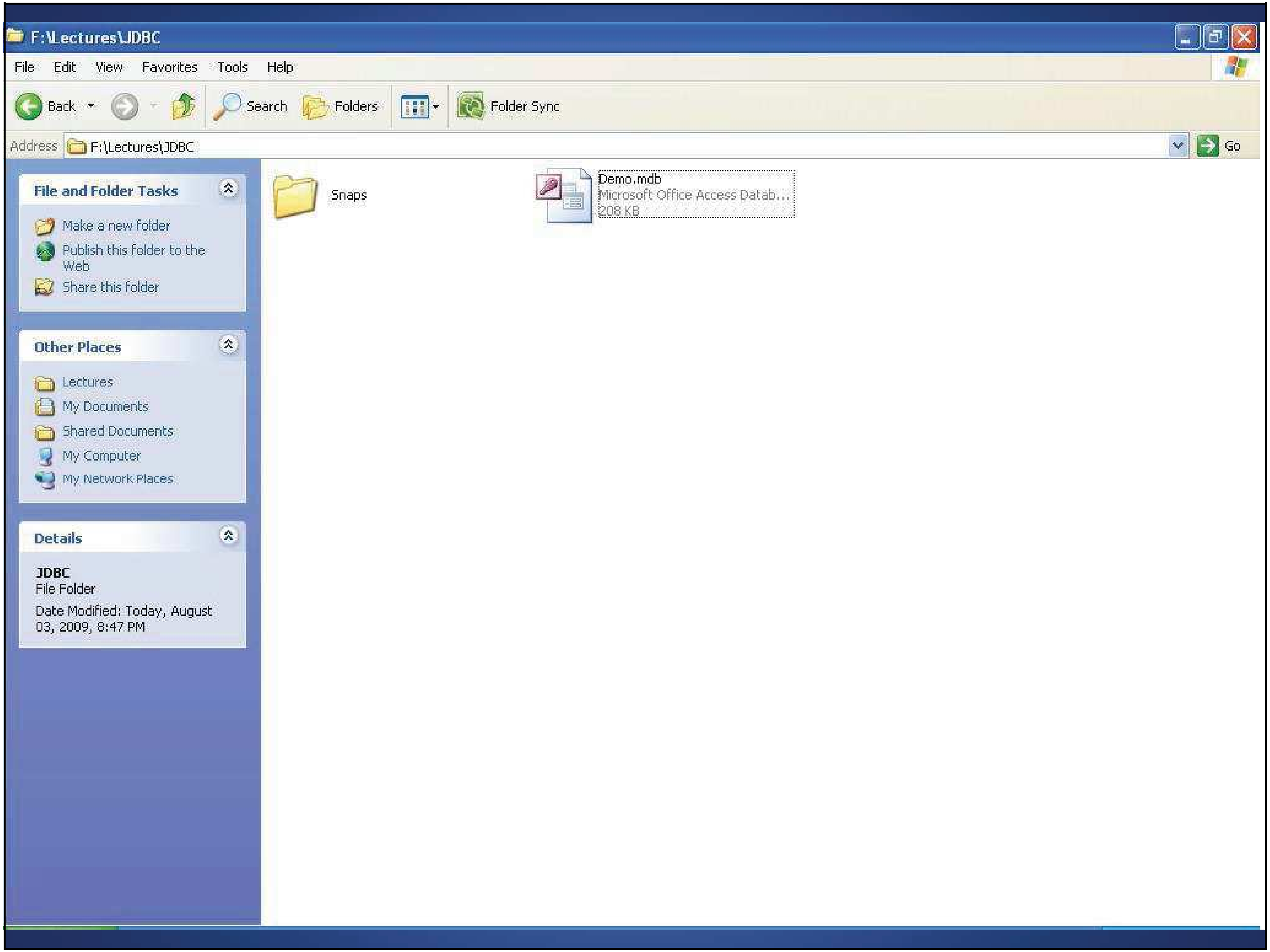
DSN Oriented Connection

- Here, first we need to create a DSN for our database by selecting the appropriate driver for it.
- Then we need to specify the DSN path as url while preparing the connection.
- The steps to be followed to create the DSN are as furnished in the subsequent slides in the form of snapshots.



Field Name	Data Type	Description
stu_id	Text	
stu_name	Text	
roll_no	Number	
branch	Text	
semester	Number	
division	Text	
contact_no	Number	

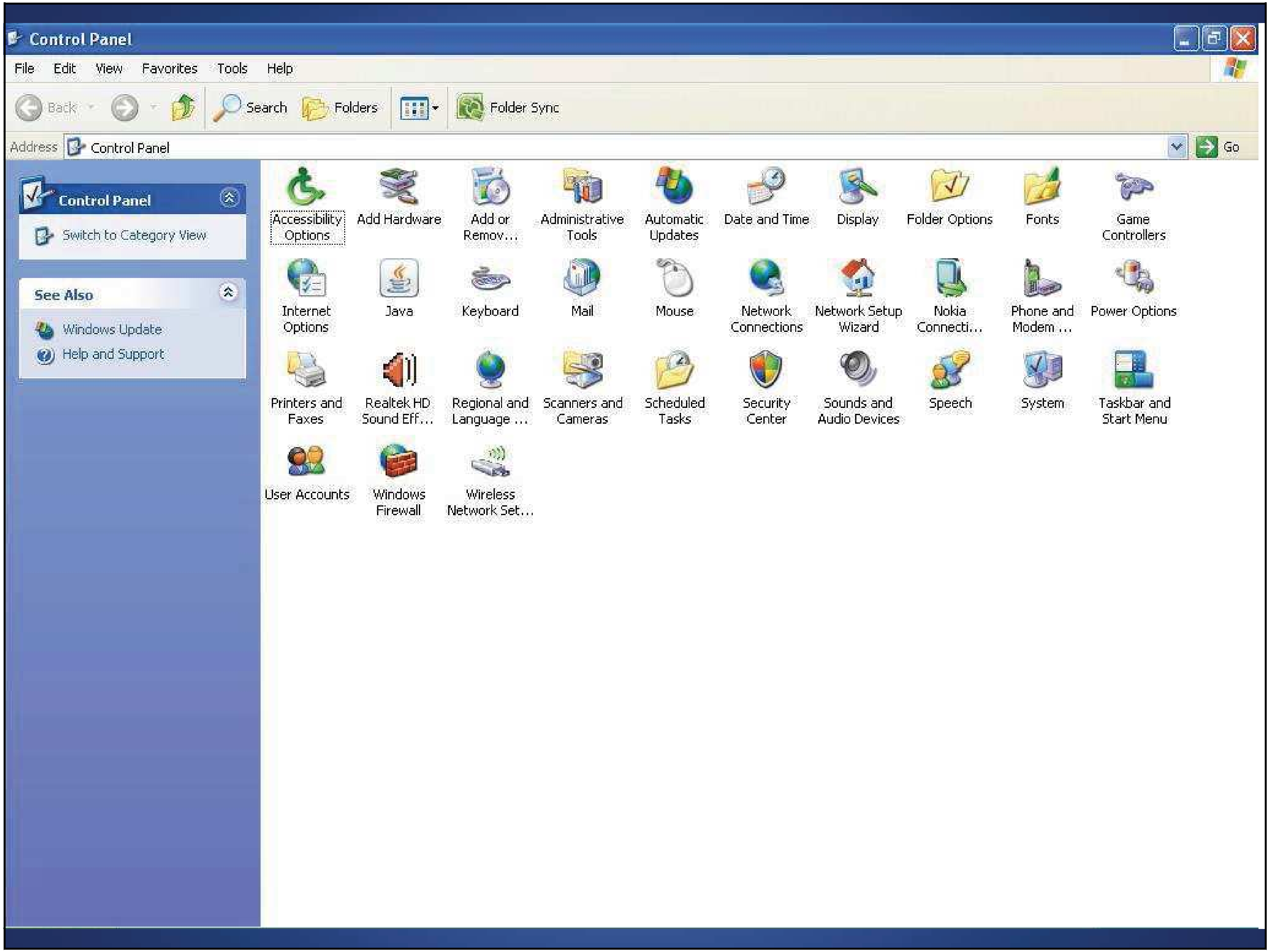
A field name can be up to 64 characters long, including spaces. Press F1 for help on field names.

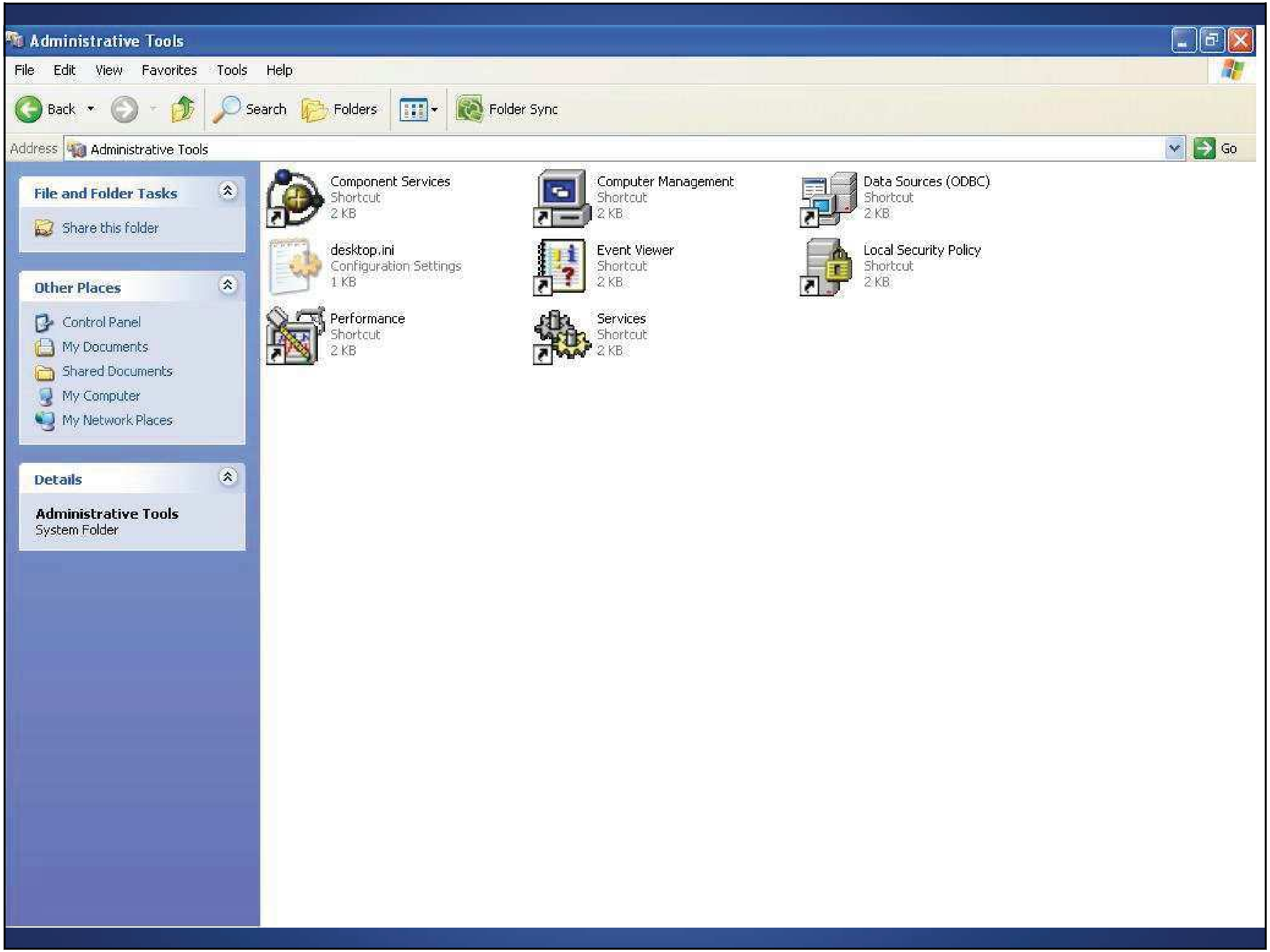


JDBC
File Folder
Date Modified: Today, August 03, 2009, 8:47 PM

Snaps

Demo.mdb
Microsoft Office Access Datab...
208 KB


Control Panel
File
Edit
View
Favorites
Tools
Help
Back
Search
Folders
Folder Sync
Address
Control Panel
Go
Control Panel
Switch to Category View
See Also
Windows Update
Help and Support
Accessibility Options
Add Hardware
Add or Remov…
Administrative Tools
Automatic Updates
Date and Time
Display
Folder Options
Fonts
Game Controllers
Internet Options
Java
Keyboard
Mail
Mouse
Network Connections
Network Setup Wizard
Nokia Connecti…
Phone and Modem …
Power Options
Printers and Faxes
Realtek HD Sound Eff…
Regional and Language …
Scanners and Cameras
Scheduled Tasks
Security Center
Sounds and Audio Devices
Speech
System
Taskbar and Start Menu
User Accounts
Windows Firewall
Wireless Network Set…



Administrative Tools
System Folder

Component Services
Shortcut
2 KB

Computer Management
Shortcut
2 KB

Data Sources (ODBC)
Shortcut
2 KB

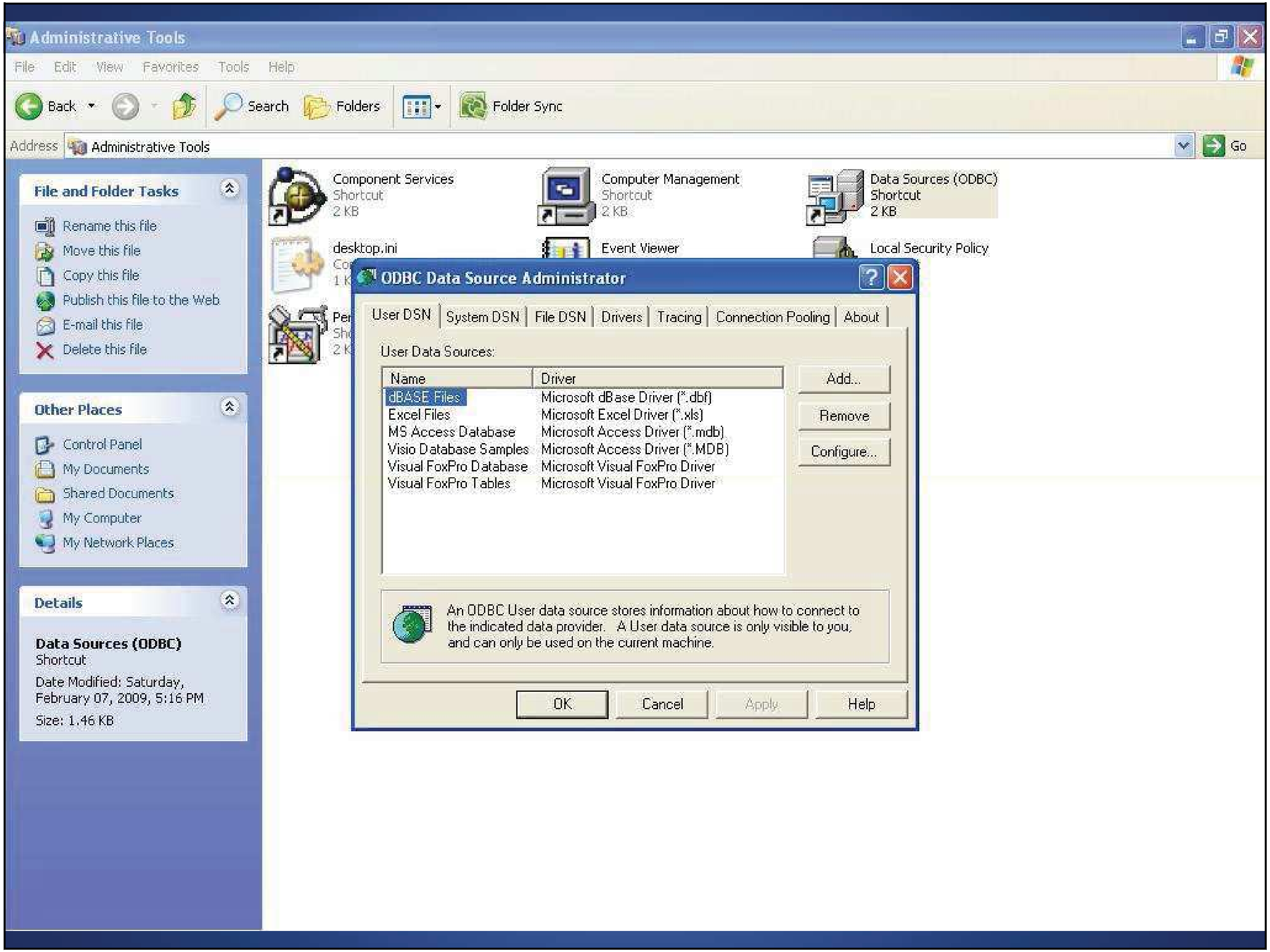
desktop.ini
Configuration Settings
1 KB

Event Viewer
Shortcut
2 KB

Local Security Policy
Shortcut
2 KB

Performance
Shortcut
2 KB

Services
Shortcut
2 KB

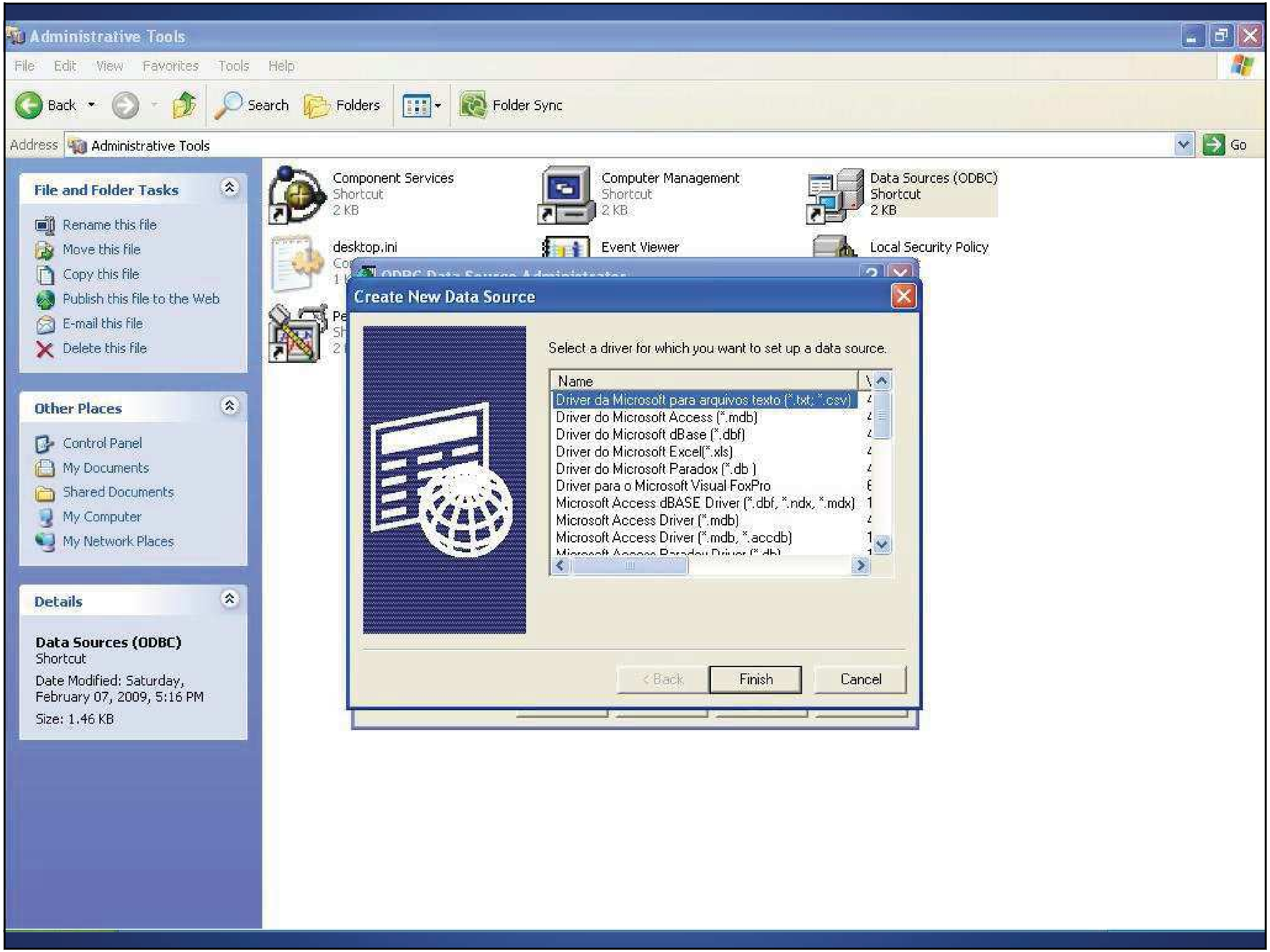


Data Sources (ODBC)
Shortcut
Date Modified: Saturday, February 07, 2009, 5:16 PM
Size: 1.46 KB

User Data Sources:

Name	Driver
dBASE Files	Microsoft dBase Driver (*.dbf)
Excel Files	Microsoft Excel Driver (*.xls)
MS Access Database	Microsoft Access Driver (*.mdb)
Visio Database Samples	Microsoft Access Driver (*.MDB)
Visual FoxPro Database	Microsoft Visual FoxPro Driver
Visual FoxPro Tables	Microsoft Visual FoxPro Driver

An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only visible to you, and can only be used on the current machine.



Data Sources (ODBC)
Shortcut
Date Modified: Saturday, February 07, 2009, 5:16 PM
Size: 1.46 KB

Component Services
Shortcut
2 KB

Computer Management
Shortcut
2 KB

Data Sources (ODBC)
Shortcut
2 KB

desktop.ini

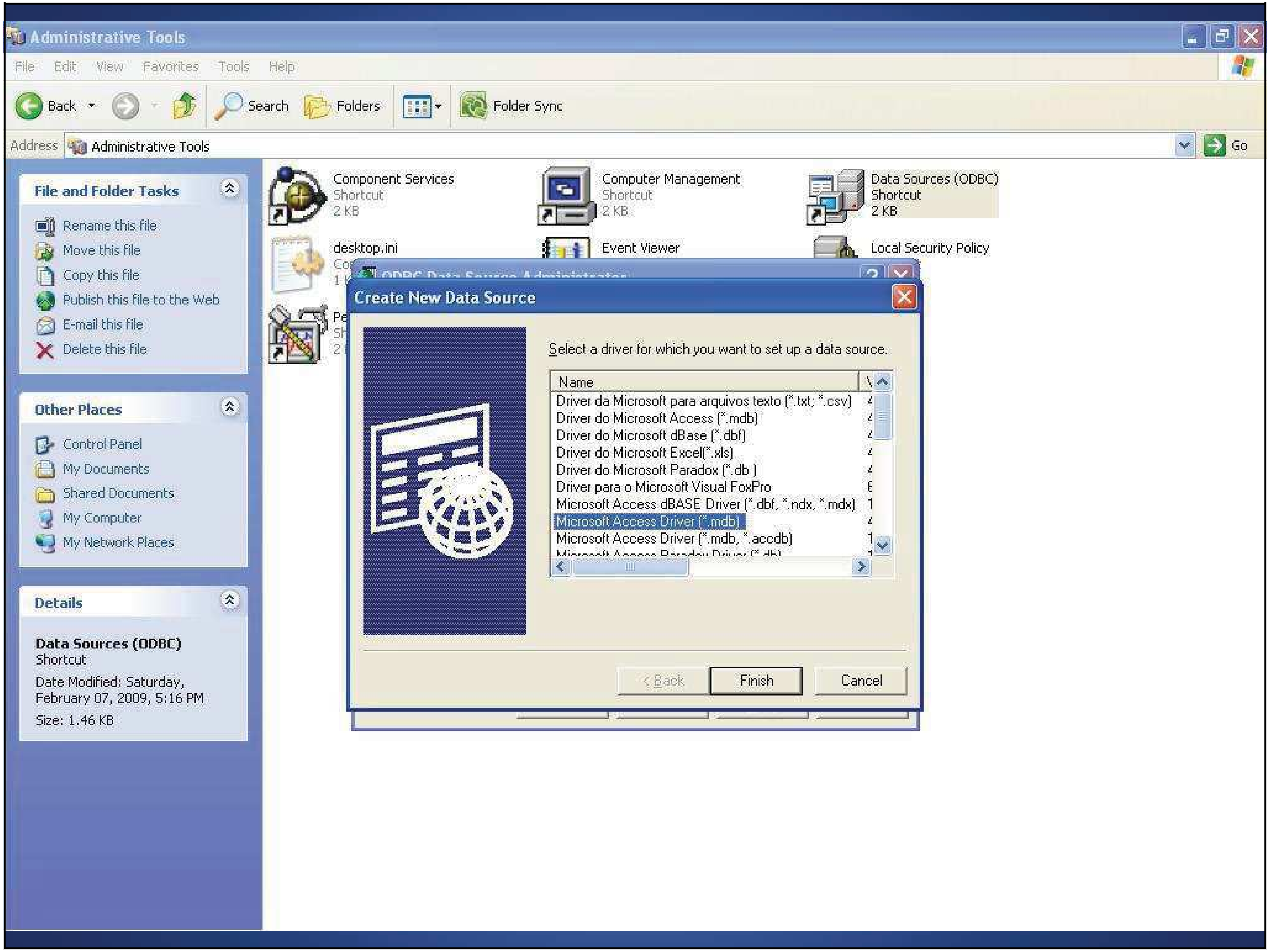
Event Viewer

Local Security Policy

Create New Data Source

Select a driver for which you want to set up a data source.

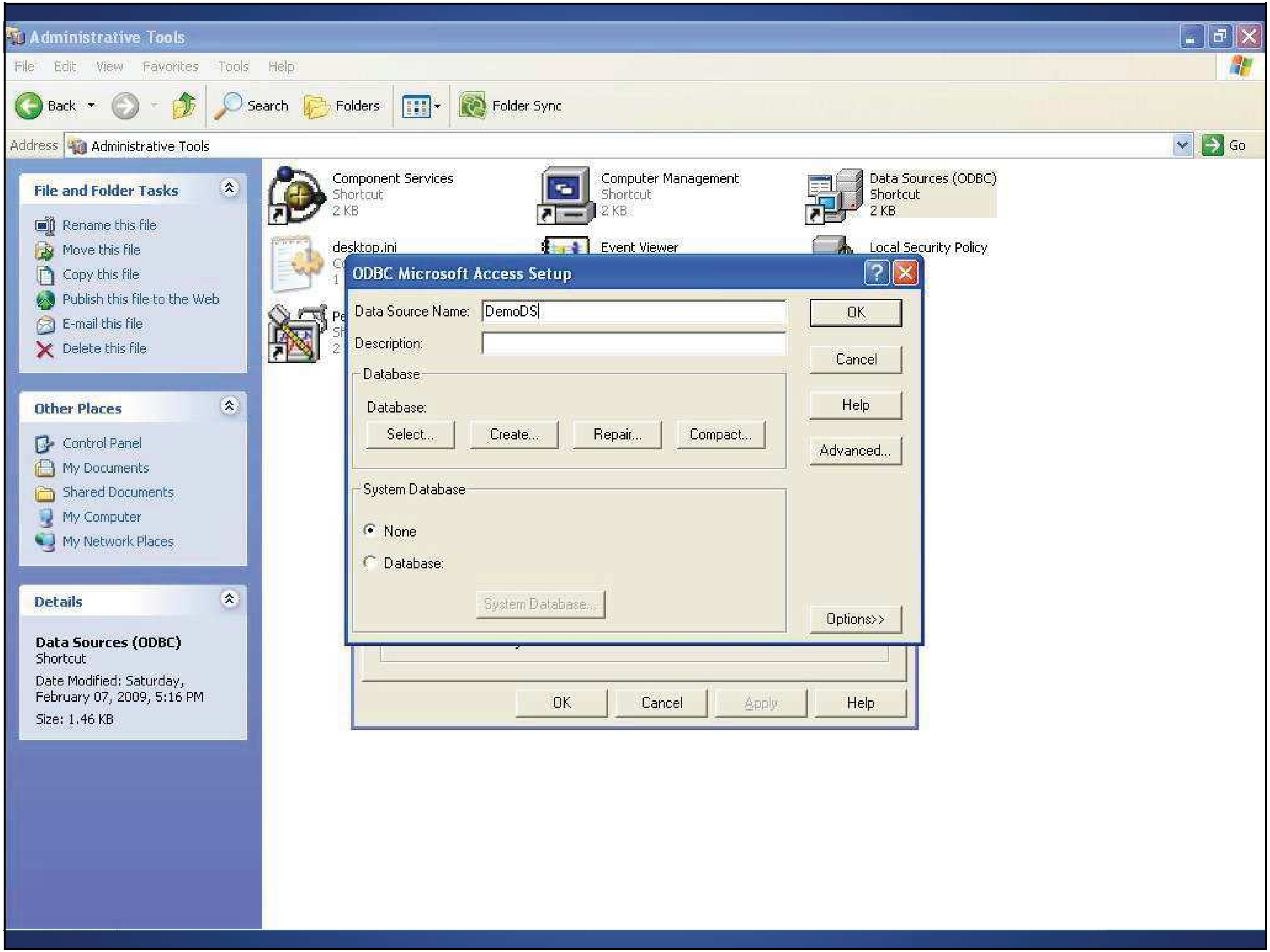
Name
Driver da Microsoft para arquivos texto (*.txt; *.csv)
Driver do Microsoft Access (*.mdb)
Driver do Microsoft dBase (*.dbf)
Driver do Microsoft Excel(*.xls)
Driver do Microsoft Paradox (*.db)
Driver para o Microsoft Visual FoxPro
Microsoft Access dBASE Driver (*.dbf, *.ndx, *.mdx)
Microsoft Access Driver (*.mdb)
Microsoft Access Driver (*.mdb, *.accdb)



Data Sources (ODBC)
Shortcut
Date Modified: Saturday, February 07, 2009, 5:16 PM
Size: 1.46 KB

Select a driver for which you want to set up a data source.

Name
Driver da Microsoft para arquivos texto (*.txt; *.csv)
Driver do Microsoft Access (*.mdb)
Driver do Microsoft dBase (*.dbf)
Driver do Microsoft Excel(*.xls)
Driver do Microsoft Paradox (*.db)
Driver para o Microsoft Visual FoxPro
Microsoft Access dBASE Driver (*.dbf, *.ndx, *.mdx)
[✓ Microsoft Access Driver (*.mdb)]
Microsoft Access Driver (*.mdb, *.accdb)


Administrative Tools

File Edit View Favorites Tools Help

Back Search Folders Folder Sync

Address: Administrative Tools

File and Folder Tasks
- Rename this file
- Move this file
- Copy this file
- Publish this file to the Web
- E-mail this file
- Delete this file

Other Places
- Control Panel
- My Documents
- Shared Documents
- My Computer
- My Network Places

Details

Data Sources (ODBC)
Shortcut
Date Modified: Saturday, February 07, 2009, 5:16 PM
Size: 1.46 KB

Component Services
Shortcut
2 KB

Computer Management
Shortcut
2 KB

Data Sources (ODBC)
Shortcut
2 KB

desktop.ini

Event Viewer

Local Security Policy

ODBC Microsoft Access Setup

Data Source Name: DemoDS
Description:

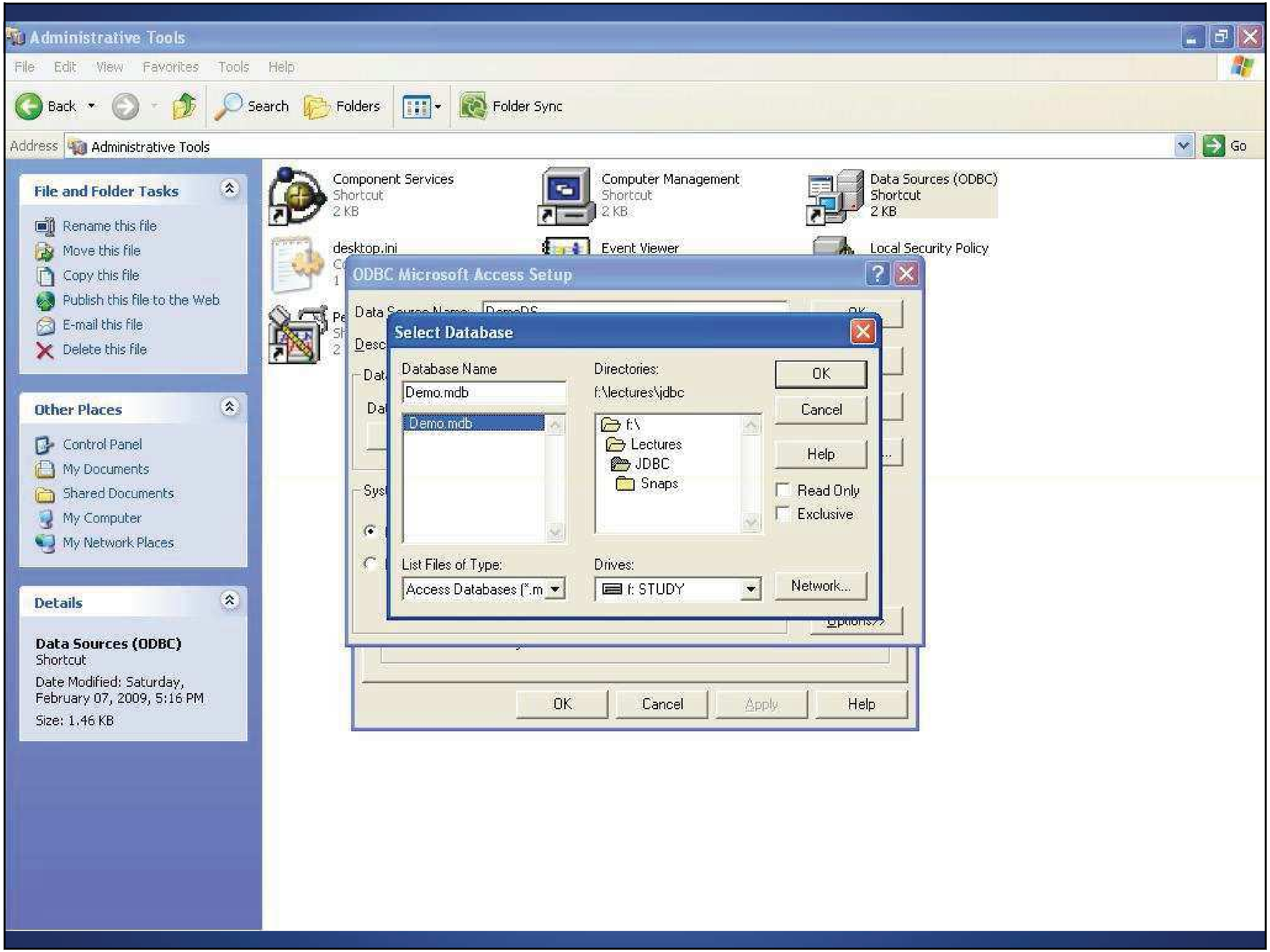
Database
Database:
Select... Create... Repair... Compact...

System Database
- [✓ None]
- Database:

System Database...

OK Cancel Help Advanced... Options>>

OK Cancel Apply Help



Administrative Tools

Address: Administrative Tools

Component Services
Shortcut
2 KB

Computer Management
Shortcut
2 KB

Data Sources (ODBC)
Shortcut
2 KB

desktop.ini

Event Viewer

Local Security Policy

ODBC Microsoft Access Setup

Select Database

Database Name: Demo.mdb

Directories: f:\lectures\jdbc

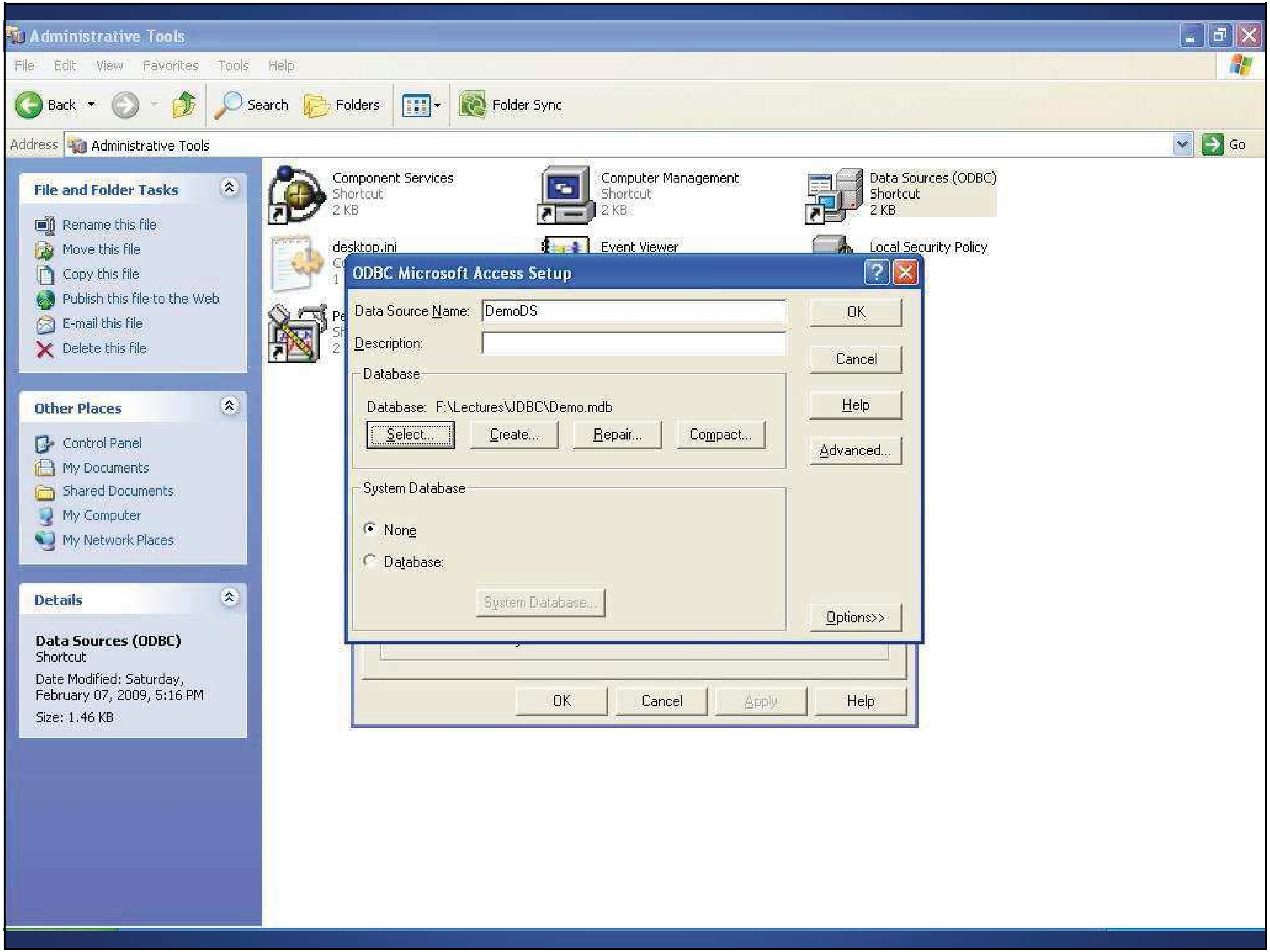
List Files of Type: Access Databases (*.m

Drives: f: STUDY

Read Only [ ]
Exclusive [ ]

Details

Data Sources (ODBC)
Shortcut
Date Modified: Saturday, February 07, 2009, 5:16 PM
Size: 1.46 KB



Data Sources (ODBC)
Shortcut
Date Modified: Saturday, February 07, 2009, 5:16 PM
Size: 1.46 KB

Data Source Name: DemoDS

Description:

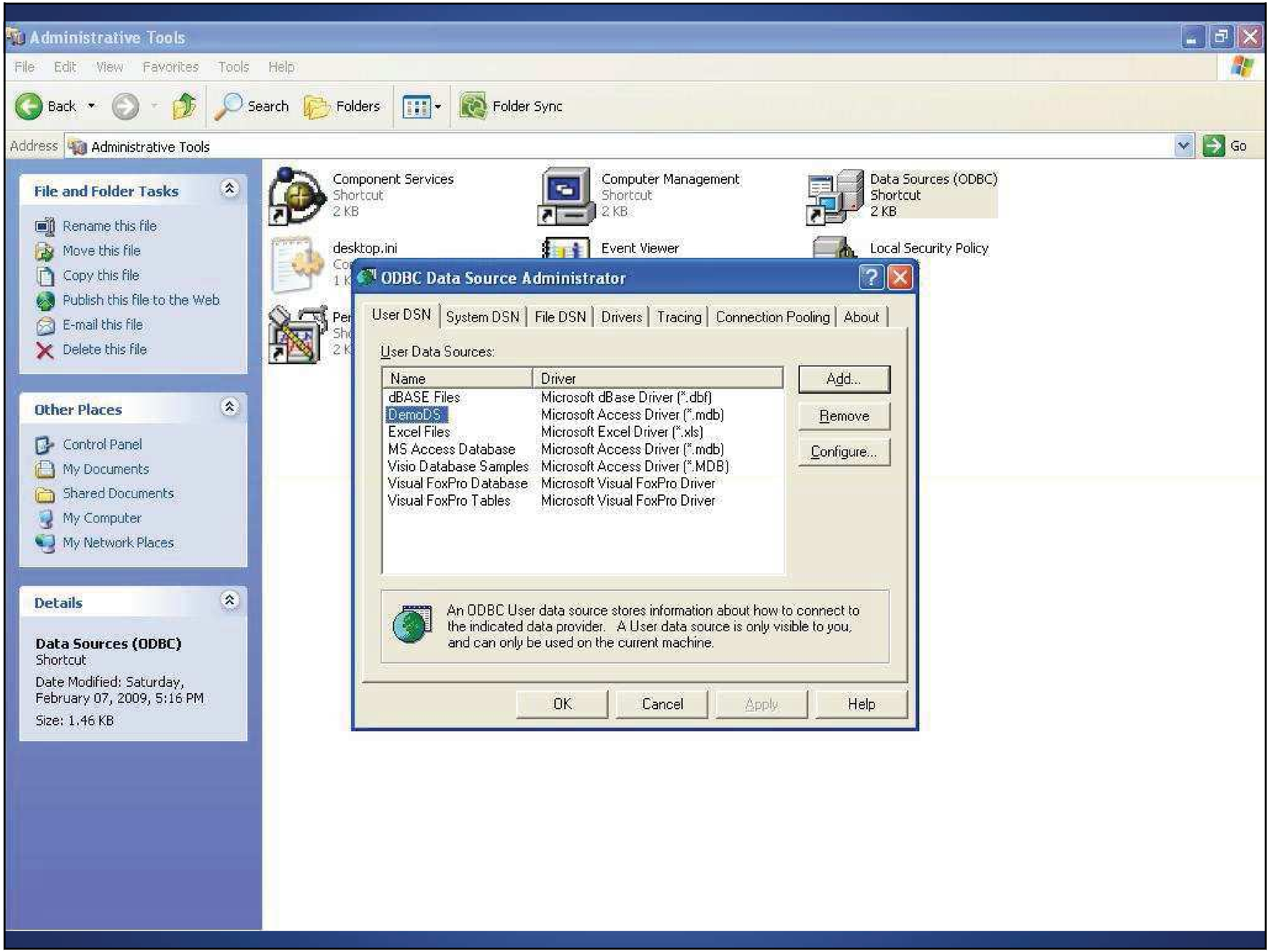
Database

Database: F:\Lectures\JDBC\Demo.mdb

System Database

[✓ None]

Database:



Data Sources (ODBC)
Shortcut
Date Modified: Saturday, February 07, 2009, 5:16 PM
Size: 1.46 KB

User Data Sources:

Name	Driver
dBASE Files	Microsoft dBase Driver (*.dbf)
DemoDS	Microsoft Access Driver (*.mdb)
Excel Files	Microsoft Excel Driver (*.xls)
MS Access Database	Microsoft Access Driver (*.mdb)
Visio Database Samples	Microsoft Access Driver (*.MDB)
Visual FoxPro Database	Microsoft Visual FoxPro Driver
Visual FoxPro Tables	Microsoft Visual FoxPro Driver

An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only visible to you, and can only be used on the current machine.

DSN Less Connection

- Here, we need not to create and connect to any DSN to have the JDBC connectivity with our database.
- Instead of that, we directly specify the drivers we want to use and the fully qualified path of our database while preparing the connection.
- E.g.
  - Connection con = DriverManager.getConnection("jdbc:odbc:Driver={ Microsoft Access Driver (*.mdb)};DBQ=F:\\Lectures\\JDBC\\Demo.mdb");

DSN Less Connection

- Here, DBQ is TNS Service Name.
- TNS (Transparent Network Substrate) is a communication layer used by Oracle databases.
- For Oracle, we can have:
  - DRIVER={Oracle ODBC Driver};DBQ=<<TNS alias>>;UID=system;PWD=manager

Prepared Statement

- Whatever we had discussed previously, that was related to the simple Statement Interface used to fire any SQL query.
- We use simple Statement when we want the query to be compiled and executed at the DBMS side which may increase load at the database side if the query is executed several times by the same instance of J2EE component during the same session.

Prepared Statement

- To resolve this problem, we have facility to fire a precompiled query using PreparedStatement interface available with us.
- A SQL query can be precompiled and executed by using the PreparedStatement object.
- We keep placeholders (? sign) in our query for a value that is inserted into the query after the query is compiled. It is the value that changes each time the query is executed.
- This pre-compilation is done by DBMS and termed as “Late Binding”.

Prepared Statement

- To set the values at the placeholders, we have various setxxx() methods available where xxx here is replaced by a specific datatype.
- E.g.
  - PreparedStatement s = con.prepareStatement("select * from student_master where stu_id = ?");

    s.setString(1,"s002");
    ResultSet rs = s.executeQuery();

Callable Statement

- Callable Statement is used to call a stored procedure from within a J2EE object.
- Stored procedure is a block of code and is identified by a unique name.
- CallableStatement object uses three types of parameters when calling a stored procedure.
  - IN
  - OUT
  - INOUT

Callable Statement

- IN:
  - The IN Parameter contains any data that needs to be passed to the stored procedure and whose value is assigned using the setxxx() method.
- OUT:
  - The OUT parameter contains the value returned by the stored procedures, if any.
  - OUT parameter must be registered using the registerOutParameter() method and then is retrivde by the J2EE component using the getxxx() method.

Callable Statement

- INOUT:
  - It is a single parameter used for both passing and retrieving the information from to/from a stored procedure.

How to call a stored procedure?

- To call a stored procedure, the CALL keyword is used.
  - String query = “{CALL procedure_name(?,?)}”;
  - “?” sign is a place holder for the argument value.
- To execute a query related to calling a procedure needs the help of CallableStatement interface.
- The query related to calling a procedure is passed to the method prepareCall(String query) of connection object.
  - CallableStatement cs = con.prepareCall(query);

How to call a stored procedure?

- After this you can register the IN parameter which you need to pass as the procedure parameters.
  - cs. setString(1,”Atmiya”);
- Before executing the query actually, we need to set the OUT parameter (if any) i.e. the value which has been returned from the procedure and to be captured in the program.

How to call a stored procedure?

- For this, first we need to register OUT parameter.
  - `cs.registerOutParameter(1,Types.VARCHAR);`
  - Here Types.VARCHAR is the data type of the argument to be returned.
- After everything is set, you can execute the query by the execute() method of CallableStatement object.
- After the execution of this query, you can retrieve the out parameters.
  - `cs.getString(1);`

Scrollable Resultset

- With the use of scrollable resultset, the virtual curser on the resultset rows can be moved forward as well as backward and even can be positioned at a specific row.
- This facility was not available in the previous case of ResultSet.
- It was introduced after the release of JDBC 2.1 API.

Scrollable Resultset

- The statement object must be set up to handle a scrollable ResultSet by passing one of the below mentioned arguments in the createStatement() method.
  - TYPE_FORWARD_ONLY (Default)
    - Restricts the virtual cursor to downward movement.
  - TYPE_SCROLL_INSENSITIVE
  - TYPE_SCROLL_SENSITIVE
    - Both of the above permit the virtual cursor to move in both the directions.

Scrollable Resultset

- TYPE_SCROLL_INSENSITIVE
  - Makes the ResultSet insensitive to changes made by another J2EE component to data in the table whose rows are reflected in the resultset.
- TYPE_SCROLL_SENSITIVE
  - Makes the ResultSet sensitive to those changes.
- E.g.
  - Statement s = con.createStatement(TYPE_SCROLL_INSENSITIVE);

Scrollable ResultSet

- After enabling the scrollable property of the resultset, you can use any of the below mentioned six methods to achieve various results.

Method	Purpose
first()	Moves virtual cursor to the first row in the ResultSet.
last()	Moves virtual cursor to the last row in the ResultSet.
previous()	Moves virtual cursor to the previous row in the ResultSet.
absolute()	Moves virtual cursor at the row number specified by the integer passed as a parameter to this method.
relative()	Moves virtual cursor to the specified number of rows relative to its current position. Here the number specified in the relative method can be positive or negative which will show the direction of the relative movement.
getRow()	Returns an integer that represents the number of the current row in the ResultSet.

Updatable ResultSet

- This updating will not take place in the database.
- Update, Delete and Insert operations are possible here.
- This is made possible by passing CONCUR_UPDATABLE constant as an argument in the createStatement() of Connection object.
- To prevent the resultset by getting updated, we can again pass CONCUR_READ_ONLY constant.
  - Statement s = con.createStatement(ResultSet.CONCUR_UPDATABLE);

Updatable ResultSet

- Update Operation:
  - updatexxx() method is used to update a specific column and then after updateRow() method is called.
    - Resultset.updateString(“LastName”,”Mahera”);
      ResultSet.updateRow();
- Delete Operation:
    - ResultSet.deleteRow(*int rowNo*);
- Insert Operation:
    - ResultSet.updateString(1,”Atmiya”);
      ResultSet.updateString(2,”Institute”);
      ResultSet.insertRow();

Database Metadata

- The DatabaseMetaData interface is used to retrieve information about databases, tables, columns, and indexes.
- It is retrieved by calling the getMetadata() method of Connection object.
- Once the DatabaseMetaData object is obtained, an assortment of methods contained in that object are called to retrieve specific metadata.

Database Metadata

Method	Purpose
getDatabaseProductName()	Returns the product name of the database.
getUserName()	Returns the usename.
getURL()	Returns the URL of the database.
getSchemas()	Returns all the schema names available in this database.
getPrimaryKeys()	Returns primary keys.
getProcedures()	Returns stored procedure names.
getTables()	Returns names of the tables in the database.

ResultSet Metadata

- It is metadata related to a resultset.
- It is retrieved by the getMetadata() method of resultset object.
  - ResultSetMetaData rm = rs.getMetaData();
- Once ResultSetMetaData is achieved, we can call its methods to retrieve specific kinds of metadata.

Database Metadata

Method	Purpose
getColumnCount()	Returns the number of columns contained in the ResultSet.
getColumnName(int number)	Returns the name of the column specified by the column number.
getColumnType(int number)	Returns the data type of the column specified by the column number.