



Java Avanzado

Acceso a Bases de Datos (JDBC)

Copyright

- Copyright (c) 2004
José M. Ordax
- Este documento puede ser distribuido solo bajo los términos y condiciones de la Licencia de Documentación de javaHispano v1.0 o posterior.
- La última versión se encuentra en
<http://www.javahispano.org/licencias/>

JDBC

- Son las siglas de Java Database Connectivity.
- Consta de un conjunto de clases e interfaces Java que nos permiten acceder de una forma genérica a las Bases de Datos independientemente del proveedor del SGBD.
- Cada proveedor dispondrá de una implementación de dichos interfaces. Dicha implementación se sabe comunicar con el SGBD de ese proveedor.
- Se encuentra en el paquete `java.sql.*`;

JDBC (cont.)

- Básicamente, una aplicación que usa JDBC realiza los siguientes pasos:
 - Establece una conexión con una Base de Datos.
 - Crea y envía una sentencia SQL a la Base de Datos.
 - Procesa el resultado.
- SQL son las siglas de Structured Query Language.
- SQL es el lenguaje estándar de interrogación de Bases de Datos.

ODBC

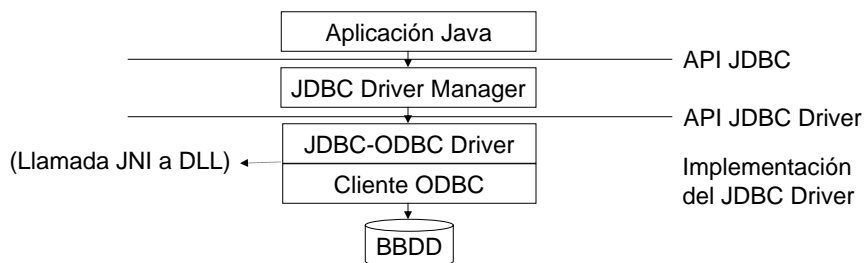
- Son las siglas de Open Database Connectivity.
- Consiste en otra implementación de la misma idea que persigue el JDBC pero:
 - Está escrito en C (luego habría que usar JNI).
 - Copiar directamente el API ODBC en Java era difícil puesto que usa punteros, etc... inexistentes en Java.
 - Es mas complejo de utilizar.
 - Requiere instalación de código adicional en las máquinas.
- No obstante existe un 'bridge' JDBC-ODBC.

Drivers JDBC

- Los drivers JDBC son la implementación que cada proveedor ha realizado del API JDBC.
- Existen cuatro tipos:
 - Tipo 1: JDBC-ODBC Bridge.
 - Tipo 2: Native-API partly-Java.
 - Tipo 3: JDBC-Net pure Java.
 - Tipo 4: Native protocol pure Java.
- Los SGBD tendrán un fichero JAR o ZIP con las clases del Driver JDBC que habrá que añadir a la variable CLASSPATH del sistema.

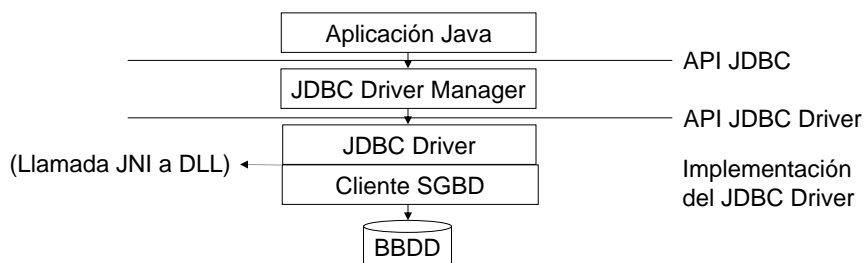
Tipo 1: JDBC-ODBC Bridge.

- Viene incluido con la JDK.
`sun.jdbc.odbc.JdbcOdbcDriver`
- Traduce llamadas JDBC en llamadas ODBC.
- Requiere de la instalación y configuración del cliente ODBC.



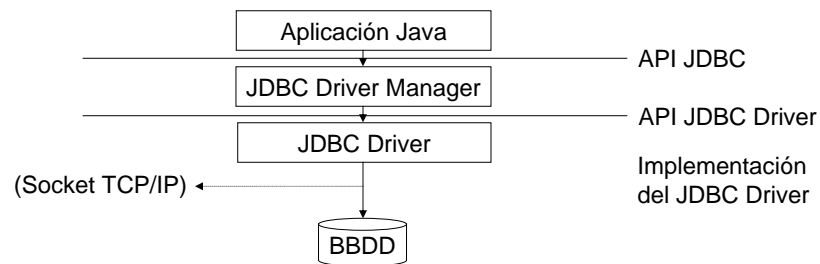
Tipo 2: Native-API partly-Java

- No viene incluido con la JDK.
- Traduce llamadas JDBC a llamadas propietarias del SGBD.
- Requiere instalación y configuración del cliente del SGBD.



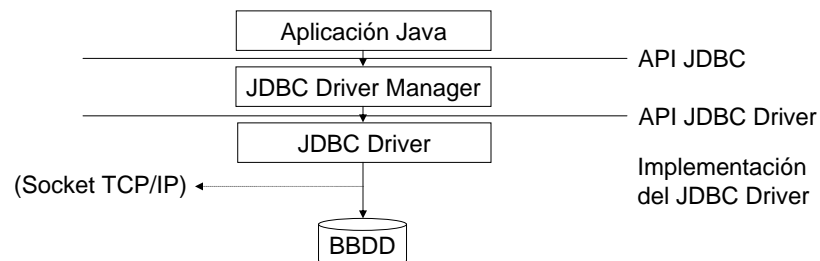
Tipo 3: JDBC-Net pure Java

- No viene incluido con la JDK.
- Conecta de manera remota (vía TCP/IP) con un daemon (listener) del SGBD (local o remoto).
- El daemon traduce las llamadas al SGBD.
- No requiere ninguna instalación previa.



Tipo 4: Native-protocol pure Java

- No viene incluido con la JDK.
- Conecta de manera remota (vía TCP/IP) con el SGBD (local o remoto).
- No requiere ninguna instalación previa.



Ejemplos de drivers



Genérico ODBC:



Tipo 1: `sun.jdbc.odbc.JdbcOdbcDriver`



IBM DB2 v7 o anteriores (o v8 en modo legacy):



Tipo 2: `COM.ibm.db2.jdbc.app.DB2Driver`



Tipo 3: `COM.ibm.db2.jdbc.net.DB2Driver`



IBM DB2 v8 (universal):



Tipo 2 y 4: `com.ibm.db2.jcc.DB2Driver`



Oracle:



Tipo 3: `oracle.jdbc.OracleDriver`

Ejemplos de drivers (cont.)



Sybase:



Tipo 3: `com.sybase.jdbc2.jdbc.SybDriver`



MySQL:



Tipo 3: `com.mysql.jdbc.Driver`



Informix:



Tipo 3: `com.informix.jdbc.IfxDriver`



Apache Derby:



Tipo 2: `org.apache.derby.jdbc.EmbeddedDriver`



Tipo 4: `com.ibm.db2.jcc.DB2Driver` (el de IBM DB2 v8)

Componentes del JDBC



Los componentes del API JDBC son:



El gestor de los drivers (`java.sql.DriverManager`).



La conexión con la Base de Datos (`java.sql.Connection`).



La sentencia a ejecutar (`java.sql.Statement`).



El resultado (`java.sql.ResultSet`).



Otros:



Otro tipo de sentencias (`java.sql.PreparedStatement`)



Procedimiento almacenado (`java.sql.CallableStatement`)

`java.sql.DriverManager`



El Driver Manager lleva el control de los drivers JDBC cargados en memoria.



Además es el encargado de realizar la conexiones con la Base de Datos.



Hay que cargar en memoria los drivers JDBC para registrarlos al Driver Manager:



Se cargan mediante el método estático `forName()` de la clase `java.lang.Class`









Por ejemplo: `Class.forName("com.mysql.jdbc.Driver");`

Ejemplo

```
import java.sql.*;

public class TestJDBC
{
    public static void main(String[] args)
    {
        try
        {
            Class.forName("com.ibm.db2.jcc.DB2Driver");
            .....
        }
        catch(ClassNotFoundException ex)
        {
            ex.printStackTrace();
        }
    }
}
```

java.sql.Connection


-  Representa una conexión con la Base de Datos.
-  El encargado de abrir una conexión es el Driver Manager mediante el método estático:
public static Connection getConnection(url, user, pwd) **throws** SQLException
-  Donde:
 -  url: es el identificador de la Base de Datos.
 -  user: usuario con el que se abre la conexión (opcional).
 -  pwd: contraseña del usuario (opcional)

Formato de la URL


 Las URLs JDBC constan de las siguientes partes:

 jdbc:%subprotocolo%:%subnombre%

 Donde:

 jdbc: el protocolo (es fijo, siempre se pone jdbc).

 %subprotocolo%: especifica el tipo o nombre del Driver.

 %subnombre%: especifica la BBDD. Depende del tipo de driver tendrá una información u otra.


Ejemplos de URLs

 Genérico ODBC:


 Tipo 1: "jdbc:odbc:sample"

 IBM DB2:

 Tipo 2: "jdbc:db2:sample"

 Tipo 3/4: "jdbc:db2://127.0.0.1:6789/sample"

 Informix:

 Tipo 3: "jdbc:informix-sqli://127.0.0.1:1526/sample"

 Oracle:

 Tipo 3: "jdbc:oracle:thin:@127.0.0.1:1521:sample"

Ejemplos de URLs (cont.)



Sybase:



Tipo 3: "jdbc:sybase:Tds:127.0.0.1/sample"



MySQL:



Tipo 3: "jdbc:mysql://127.0.0.1/sample"



Apache Derby:



Tipo 2: "jdbc:derby:sample"



Tipo 4: "jdbc:derby:net://127.0.0.1:1527/sample"

java.sql.Connection (cont.)



A través de la conexión nos comunicaremos con la Base de Datos, enviándole sentencias SQL.



Las sentencias SQL se envían a través de 'statements'.



Existen tres tipos de 'statements' y un método para generar cada tipo.



java.sql.Statement: createStatement();



java.sql.PreparedStatement: prepareStatement();



java.sql.CallableStatement: prepareCall();

java.sql.Connection (cont.)

Una vez hayamos terminado de trabajar con una conexión debemos liberarla. Una conexión abierta significa recursos consumiéndose en el SGBD.

Las conexiones se cierran mediante el método:

```
public void close() throws java.sql.SQLException
```

Ejemplo

```
import java.sql.*;

public class TestJDBC
{
    public static void main(String[] args)
    {
        try
        {
            Class.forName("com.ibm.db2.jcc.DB2Driver");
            Connection con = DriverManager.getConnection("jdbc:db2:sample");
            .....
            .....
            con.close();
        }
        catch(ClassNotFoundException ex)
        {
            ex.printStackTrace();
        }
        catch(SQLException ex)
        {
            ex.printStackTrace();
        }
    }
}
```

java.sql.Statement

- Se utiliza para ejecutar una sentencia SQL sin información dinámica contra una Base de Datos.
- Siempre lleva asociada una conexión que sirvió como origen para su creación.
- Se crean con el siguiente método de la clase `java.sql.Connection`:
`public Statement createStatement() throws SQLException;`
- Las sentencias se cierran mediante el método:
`public void close() throws java.sql.SQLException;`

java.sql.Statement (cont.)






- El método para ejecutarla, depende del tipo de sentencia SQL que contenga.
- Sentencias **SELECT**:
 - Se usa el método: `executeQuery(String sql);`
 - Devuelve una instancia de `java.sql.ResultSet`
- Sentencias **INSERT, UPDATE y DELETE**:
 - Se usa el método: `executeUpdate(String sql);`
 - Devuelve un `int` con el número de filas afectadas.
- Sentencias **CREATE TABLE y DROP TABLE**:
 - Se usa el método: `executeUpdate(String sql);`
 - Devuelve un `int` que siempre vale 0.

Ejemplo

```
import java.sql.*;

public class TestJDBC
{
    public static void main(String[] args)
    {
        try
        {
            Class.forName("com.ibm.db2.jcc.DB2Driver");
            Connection con = DriverManager.getConnection("jdbc:db2:sample");
            Statement stmt = con.createStatement();
            stmt.executeQuery("SELECT * FROM employee");
            .....
            .....
            stmt.close();
            con.close();
        }
        catch(ClassNotFoundException ex)
        {
            ex.printStackTrace();
        }
        catch(SQLException ex)
        {
            ex.printStackTrace();
        }
    }
}
```

java.sql.ResultSet

-  Representa el resultado de la ejecución de una sentencia SQL.
-  Es decir, lleva asociadas todas las filas que cumplían las condiciones de la sentencia SQL.
-  Implementa métodos para:
 -  Acceder a las filas que componen el resultado.
 -  Acceder al valor de cada columna de la fila seleccionada.

java.sql.ResultSet (cont.)

- El método para acceder a la fila siguiente del resultado es:
public boolean next() **throws** SQLException;
- El método para acceder a la fila anterior del resultado es:
public boolean previous() **throws** SQLException;
- El método a usar para acceder al valor de una columna, dependerá del tipo de dato, siendo 1 el valor del índice para la primera columna.
public xxxx getXXXX(**int** column) **throws** SQLException;

Relación método/tipo

	getByte	getShort	getInt	getLong	getFloat	getDouble	getString	getBigDecimal	getBoolean	getTimestamp	getBinaryStream	getCharacterStream	getBlob	getClob	getArray	getRef	getURL	getObject
TINYINT	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
SMALLINT	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
INTEGER	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
BIGINT	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
REAL	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
FLOAT	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
DOUBLE	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
DECIMAL	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
NUMERIC	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
BIT	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
BOOLEAN	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
CHAR	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
VARCHAR	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
LONGVARCHAR	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
BINARY	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
VARBINARY	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
LONGVARIABLE	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
DATE	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
TIME	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
TIMESTAMP	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
CLOB	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
BLOB	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
ARRAY	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
REF	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
DATALINK	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
STRUCT	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
JAVA OBJECT	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Nota: una x indica una solución legal mientras que una X indica la solución recomendada.

java.sql.ResultSet (cont.)



Se suele implementar un bucle de tipo while para recorrer y tratar todo el ResultSet.



Los ResultSets se cierran mediante el método:

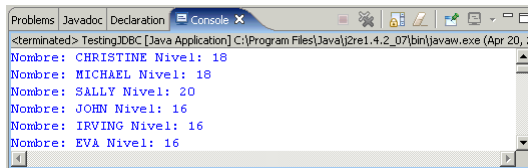
```
public boolean close() throws SQLException;
```

Nota: el ResultSet se cierra automáticamente al cerrar el Statement que le creó. No obstante, no está de más cerrarlo.

Ejemplo

```
import java.sql.*;

public class TestJDBC
{
    public static void main(String[] args)
    {
        try
        {
            Class.forName("com.ibm.db2.jcc.DB2Driver");
            Connection con = DriverManager.getConnection("jdbc:db2:sample");
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT * FROM employee");
            while(rs.next())
            {
                System.out.println("Nombre: " + rs.getString(2) + " Nivel: " + rs.getInt(9));
            }
            rs.close();
            stmt.close();
            con.close();
        }
        catch(ClassNotFoundException ex)
        {
            ex.printStackTrace();
        }
        catch(SQLException ex)
        {
            ex.printStackTrace();
        }
    }
}
```



java.sql.PreparedStatement

- Hereda de java.sql.Statement
- Se diferencia de java.sql.Statement en dos cosas:
 - La sentencia SQL es compilada por el SGBD previamente.
 - La sentencia puede contener variables marcadas con ?.
- Siempre lleva asociada una conexión que sirvió como origen para su creación.
- Se crean con el siguiente método de la clase java.sql.Connection:
 - **public** PreparedStatement prepareStatement(String sql) **throws** SQLException;

java.sql.PreparedStatement (cont.)

- Para ejecutar la sentencia se llama al método:
 - **public** ResultSet executeQuery() **throws** SQLException;
- En el caso de que la sentencia contenga alguna variable se la deberá dar un valor antes de ejecutar la sentencias con el método;
 - **public void** setXXXX(int col, xxxx value) **throws** SQLException;
- Las sentencias se cierran mediante el método:
 - **public void** close() **throws** java.sql.SQLException;

Relación método/tipo

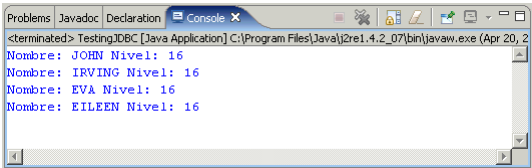
[illegible]

Ejemplo

Ejemplo

```
import java.sql.*;

public class TestJDBC
{
    public static void main(String[] args)
    {
        try
        {
            Class.forName("com.ibm.db2.jcc.DB2Driver");
            Connection con = DriverManager.getConnection("jdbc:db2://127.0.0.1:50000/sample", "chemi", "password");
            PreparedStatement pstmt =
                con.prepareStatement("SELECT * FROM employee WHERE JOB = ? AND EDLEVEL = ?");
            pstmt.setString(1, "MANAGER");
            pstmt.setInt(2, 16);
            ResultSet rs = pstmt.executeQuery();
            while(rs.next())
            {
                System.out.println("Nombre: " + rs.getString(2) + " Nivel: " + rs.getInt(9));
            }
            rs.close();
            pstmt.close();
            con.close();
        }
        catch(ClassNotFoundException ex)
        {
            ex.printStackTrace();
        }
        catch(SQLException ex)
        {
            ex.printStackTrace();
        }
    }
}
```



```
Problems Javadoc Declaration Console x
<terminated> TestingJDBC [Java Application] C:\Program Files\Java\j2re1.4.2_07\bin\javaw.exe (Apr 20, 2
Nombre: JOHN Nivel: 16
Nombre: IRVING Nivel: 16
Nombre: EVA Nivel: 16
Nombre: EILEEN Nivel: 16
```

java.sql.CallableStatement

- Hereda de java.sql.PreparedStatement
- Sirve para ejecutar procedimientos almacenados en la Base de Datos.
- Siempre lleva asociada una conexión que sirvió como origen para su creación.
- Se crean con el siguiente método de la clase java.sql.Connection:
`public CallableStatement prepareCall(String call) throws SQLException;`

java.sql.CallableStatement (cont.)

- Las llamadas pueden tener tres formas:
 - Con parámetros de entrada:
{call nombre_procedimiento(?, ?, ...)}
 - Con parámetros de entrada y de salida:
{? = call nombre_procedimiento(?, ?, ...)}
 - Sin parámetros:
{call nombre_procedimiento}

java.sql.CallableStatement (cont.)

- Para ejecutar la sentencia se llama al método:
`public ResultSet executeQuery() throws SQLException;`
- En el caso de que la sentencia contenga alguna variable se la deberá dar un valor antes de ejecutar la sentencias con el método;
`public void setXXXX(int col, xxxx value) throws SQLException;`
- Las sentencias se cierran mediante el método:
`public void close() throws java.sql.SQLException;`

Conversión de JDBC a Java

Tipo JDBC	Tipo Java
CHAR	java.lang.String
VARCHAR	java.lang.String
LONGVARCHAR	java.lang.String
NUMERIC	java.math.BigDecimal
DECIMAL	java.math.BigDecimal
BIT	boolean
TINYINT	byte
SMALLINT	short
INTEGER	int
BIGINT	long

Conversión de JDBC a Java

Tipo JDBC	Tipo Java
REAL	float
FLOAT	double
DOUBLE	double
BINARY	byte[]
VARBINARY	byte[]
LONGVARBINARY	byte[]
DATE	java.sql.Date
TIME	java.sql.Time
TIMESTAMP	java.sql.TimeStamp
CLOB	java.sql.Clob

Conversión de JDBC a Java

Tipo JDBC	Tipo Java
BLOB	java.sql.Blob
ARRAY	java.sql.Array
DISTINCT	Depende del valor
STRUCT	java.sql.Struct
REF	java.sql.Ref
JAVA_OBJECT	Depende del valor






Conversión de Java a JDBC

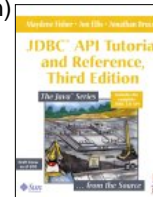
Tipo Java	Tipo JDBC
java.lang.String	CHAR, VARCHAR o LONGVARCHAR
java.math.BigDecimal	NUMERIC
boolean	BIT
byte	TINYINT
short	SMALLINT
int	INTEGER
long	BIGINT
float	REAL
double	DOUBLE

Conversión de Java a JDBC

Tipo Java	Tipo JDBC
byte[]	BINARY, VARBINARY o LONGVARBINARY
java.sql.Date	DATE
java.sql.Time	TIME
java.sql.Timestamp	TIMESTAMP
java.sql.Clob	CLOB
java.sql.Blob	BLOB
java.sql.Array	ARRAY
java.sql.Struct	STRUCT
java.sql.Ref	REF
Una clase Java	JAVA_OBJECT

Bibliografía

-  **Database Programming with JDBC** (2nd edition)
George Reese.
O'Reilly.
-  **JDBC Pocket Reference**
Donald Bales.
O'Reilly.
-  **Java Database Best Practices**
George Reese.
O'Reilly.
-  **JDBC API Tutorial and Reference** (3rd edition)
Maydene Fisher, John Ellis y Jonathan Bruce.
Addison-Wesley.
-  **The Java tutorial** (on-line)
<http://java.sun.com/docs/books/tutorial/jdbc/index.html>



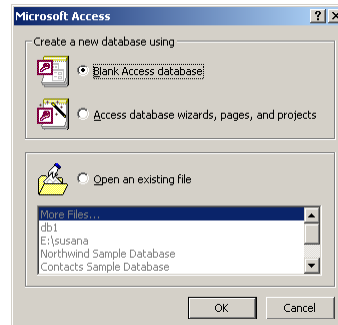
Apéndice A: MS-Access

-  Arrancar MS-Access:



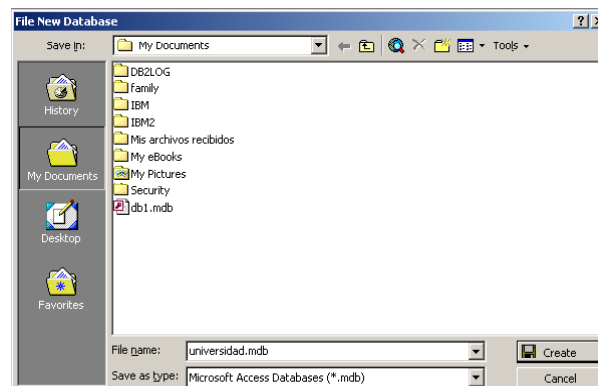
Apéndice A: MS-Access

○ Crear una Base de Datos nueva:



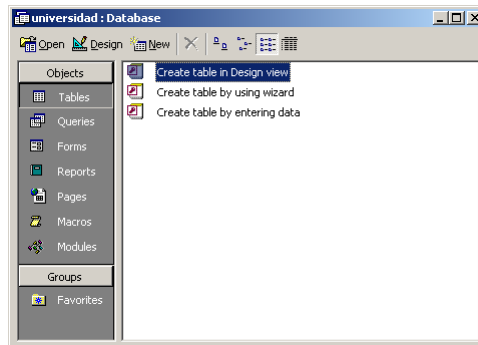
Apéndice A: MS-Access

○ Dar un nombre a la Base de Datos:



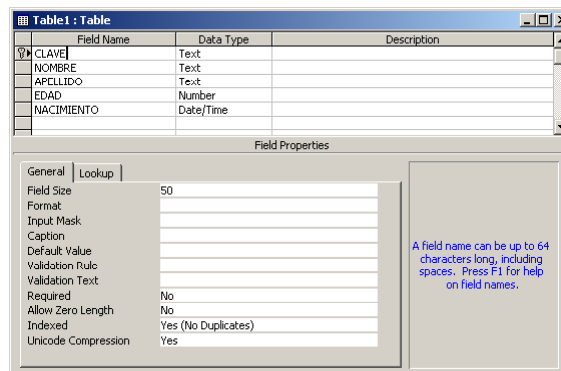
Apéndice A: MS-Access

○ Crear una tabla nueva en la Base de Datos:



Apéndice A: MS-Access

○ Añadir campos (columnas) a la tabla:

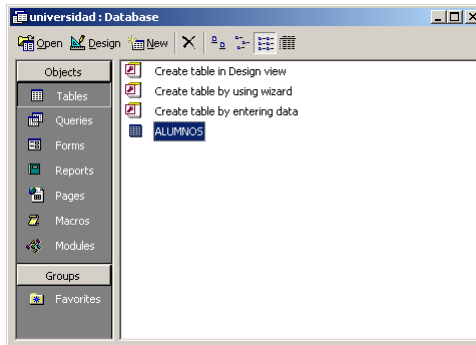
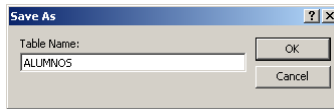


○ 5 campos.

○ El campo CLAVE es la clave primaria.

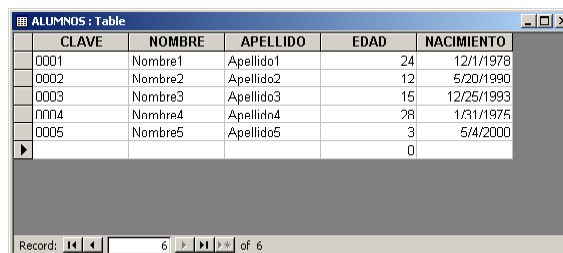
Apéndice A: MS-Access

Salvar la definición de la tabla:



Apéndice A: MS-Access

Introducir tuplas (filas) en la tabla:

A screenshot of the 'ALUMNOS : Table' window in Microsoft Access, showing the data entered into the table. The table has five columns: CLAVE, NOMBRE, APELLIDO, EDAD, and NACIMIENTO. There are five rows of data, plus a sixth row with empty fields. The status bar at the bottom indicates 'Record: 6 of 6'.

Apéndice A: MS-Access



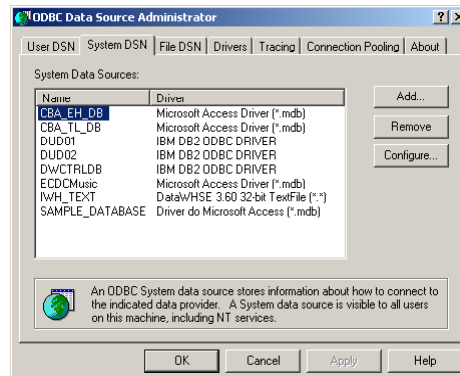
Definir una fuente de datos ODBC en el sistema:



Inicio -> Configuración -> Panel de Control



Herramientas Administrativas -> Orígenes de Datos (ODBC)



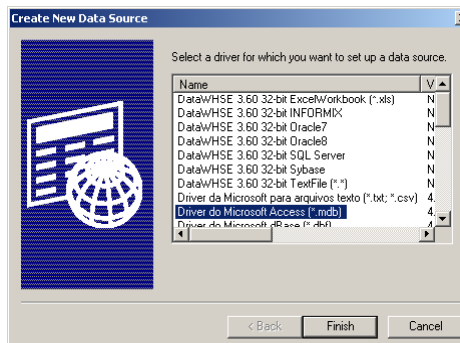
Apéndice A: MS-Access



Seleccionar la pestaña DSN de sistema y pulsar el botón Agregar...

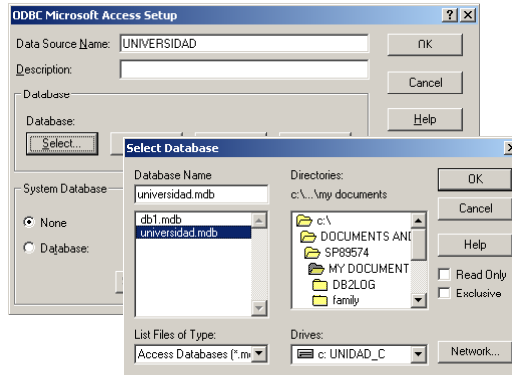


Seleccionar el controlador: Microsoft Access Driver (*.mdb)



Apéndice A: MS-Access

- Establecer como nombre del Origen de Datos: UNIVERSIDAD.
- Seleccionar la Base de Datos: universidad.mdb



Apéndice A: MS-Access

- Desarrollar la aplicación Java:
- MS-Access no incluye una implementación JDBC.
- Hay que acceder utilizando el 'bridge' JDBC-ODBC.
- Dicho 'bridge' o Driver JDBC está incluido en la JDK.
- El Driver JDBC es: `sun.jdbc.odbc.JdbcOdbcDriver`
- La URL JDBC en nuestro caso es: `jdbc:odbc:UNIVERSIDAD` si hemos definido la Fuente de Datos ODBC.
- La URL JDBC también podría ser esta otra, en caso de que no hubiesemos definido la Fuente de Datos ODBC:
`jdbc:odbc:DRIVER=Microsoft Access Driver (*.mdb);DBQ=c:\\universidad.mdb`

```
import java.sql.*;
```

```
public class AccessTest
```

```
{  
    public static void main(String[] args)
```

```
{  
    try
```

```
{
```

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

```
        Connection con = DriverManager.getConnection("jdbc:odbc:UNIVERSIDAD");
```

```
        PreparedStatement pstmt = con.prepareStatement("SELECT * FROM ALUMNOS WHERE EDAD > ?");
```

```
        pstmt.setInt(1,21);
```

```
        ResultSet rs = pstmt.executeQuery();
```

```
        while(rs.next())
```

```
            System.out.println("Nombre: " + rs.getString(2) + " Apellido: " + rs.getString(3) + " Edad: " +
```

```
                + rs.getInt(4) + " Fecha: " + rs.getDate(5));
```

```
        rs.close();
```

```
        pstmt.close();
```

```
        con.close();
```

```
    }
```

```
    catch(ClassNotFoundException ex)
```

```
{
```

```
        ex.printStackTrace();
```

```
}
```

```
    catch(SQLException ex)
```

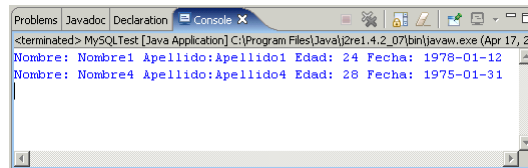
```
{
```

```
        ex.printStackTrace();
```

```
}
```

```
}
```

Apéndice A



Apendice A: Bibliografía



Access Database Design & Programming

Steven Roman, Ph.D

(3rd edition)

O'Reilly.



Access Cookbook (2nd edition)

Ken Getz, Paul Litwin y Andy Baron.

O'Reilly.



Microsoft Office Access 2003 Step by Step

Online Training Solutions Inc.

Microsoft Press.



MSDN (on-line)

<http://msdn.microsoft.com/>

Apéndice B: IBM DB2

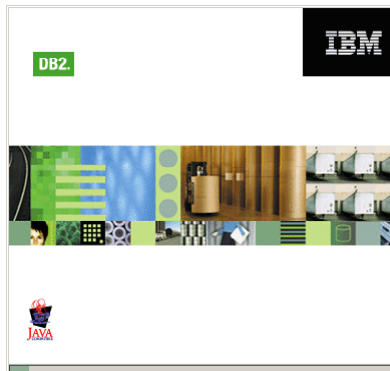
Descargar IBM DB2 8.2 de la web:



<http://www.ibm.com/software/data/db2/udb/>

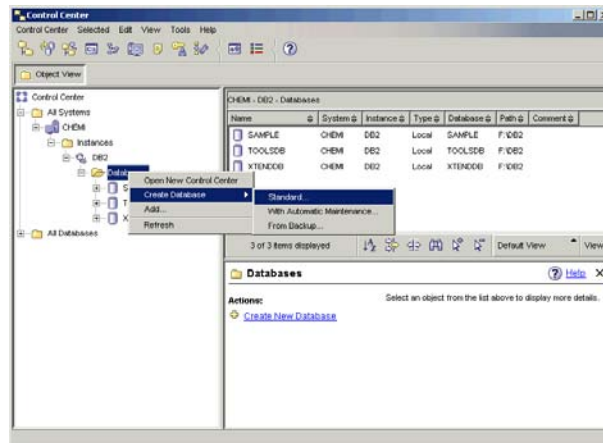
Apéndice B: IBM DB2

Arrancar IBM DB2 Control Center:



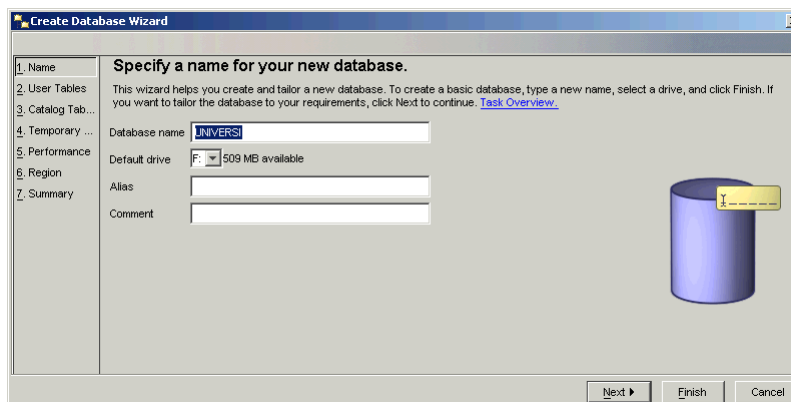
Apéndice B: IBM DB2

○ Crear una Base de Datos nueva:



Apéndice B: IBM DB2

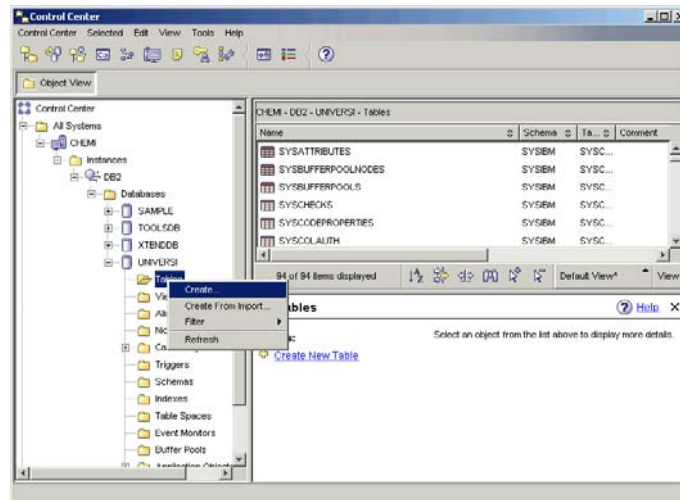
○ Dar un nombre a la Base de Datos:



○ Pulsar el botón Final.

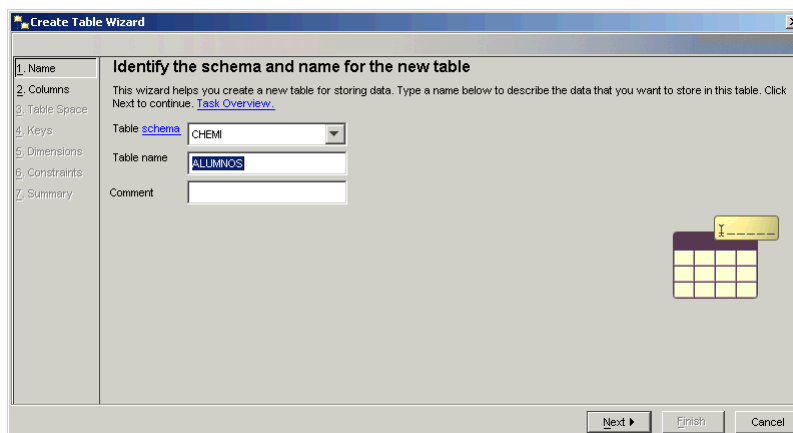
Apéndice B: IBM DB2

● Crear una tabla nueva en la Base de Datos:



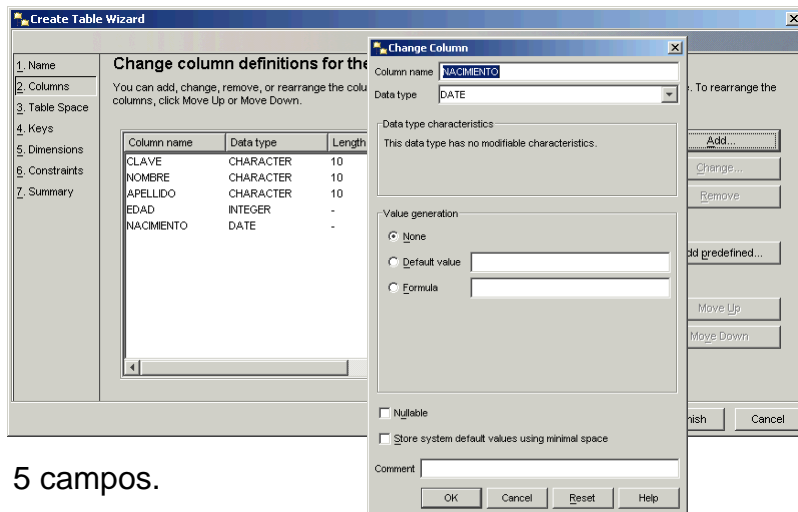
Apéndice B: IBM DB2

● Dar un nombre a la nueva tabla:



Apéndice B: IBM DB2

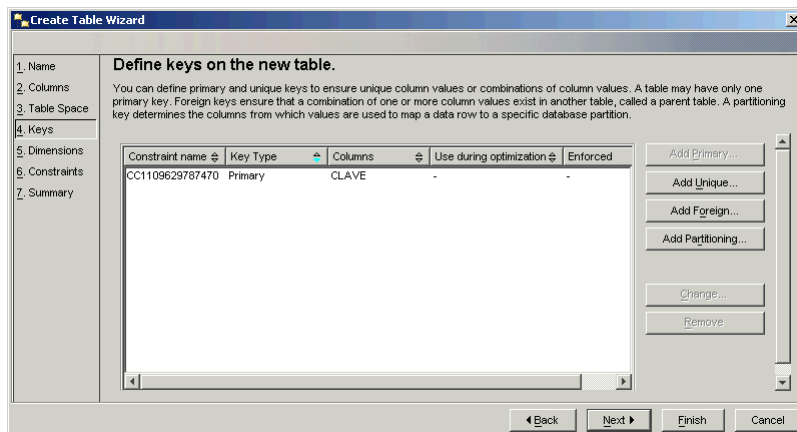
➊ Añadir campos (columnas) a la tabla:



5 campos.

Apéndice B: IBM DB2

➋ Establecer la clave primaria:



Apéndice B: IBM DB2



Desarrollar la aplicación Java:



IBM DB2 incluye varias implementaciones JDBC: tipo 2,3 y 4.



Estos Drivers JDBC no están incluidos en la JDK.



Por tanto, hay que tener en cuenta añadir el fichero que contenga los drivers en la variable de sistema CLASSPATH.

set CLASSPATH=%CLASSPATH%;c:\sql\lib\java\db2java.zip
Nota: db2jcc.jar y db2jcc_license_cu.jar para el universal.



Vamos a usar el driver de tipo 2:

com.ibm.db2.jcc.DB2Driver



La URL JDBC en nuestro caso es: jdbc:db2:UNIVERSI

```
import java.sql.*;
```

```
public class InsertDB2
```

```
{  
    public static void main(String[] args)
```

```
{
```

```
    try
```

```
    {
```

```
        Class.forName("com.ibm.db2.jcc.DB2Driver");
```

```
        Connection con = DriverManager.getConnection("jdbc:db2:UNIVERSI");
```

```
        PreparedStatement pstmt = con.prepareStatement("INSERT INTO ALUMNOS VALUES (?, ?, ?, ?, ?)");
```

```
        pstmt.setString(1, "0005");
```

```
        pstmt.setString(2, "Nombre5");
```

```
        pstmt.setString(3, "Apellido5");
```

```
        pstmt.setInt(4, 3);
```

```
        pstmt.setDate(5, new Date(100, 4, 4));
```

```
        pstmt.executeUpdate();
```

```
        pstmt.close();
```

```
        con.close();
```

```
    }
```

```
    catch(ClassNotFoundException ex)
```

```
    {
```

```
        ex.printStackTrace();
```

```
    }
```

```
    catch(SQLException ex)
```

```
    {
```

```
        ex.printStackTrace();
```

```
    }
```

```
}
```

```
}
```

Apéndice B



Introducir tuplas (filas) en la tabla:



Esta vez lo hacemos mediante Java.

```
import java.sql.*;
```

```
public class DB2Test
```

```
{
```

```
    public static void main(String[] args)
```

```
    {
```

```
        try
```

```
        {
```

```
            Class.forName("com.ibm.db2.jcc.DB2Driver");
```

```
            Connection con = DriverManager.getConnection("jdbc:db2:UNIVERSI");
```

```
            PreparedStatement pstmt = con.prepareStatement("SELECT * FROM ALUMNOS WHERE EDAD > ?");
```

```
            pstmt.setInt(1,21);
```

```
            ResultSet rs = pstmt.executeQuery();
```

```
            while(rs.next())
```

```
            {
                System.out.println("Nombre: " + rs.getString(2).trim() + " Apellido: " + rs.getString(3).trim() + " Edad: " +
                    + rs.getInt(4) + " Fecha: " + rs.getDate(5));
            }
```

```
            rs.close();
```

```
            pstmt.close();
```

```
            con.close();
```

```
        }
```

```
    } catch(ClassNotFoundException ex)
```

```
    {
```

```
        ex.printStackTrace();
```

```
    }
```

```
    } catch(SQLException ex)
```

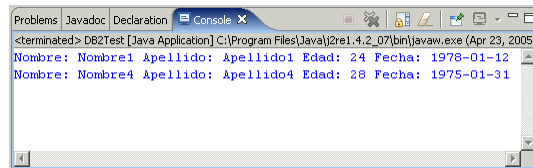
```
    {
```

```
        ex.printStackTrace();
```

```
    }
```

```
}
```

Apéndice B



Apendice B: Bibliografía



DB2 UDB v8 Administration Cert. Guide (5th edition)

George Baklarz y Bill Wong.

IBM Press.



DB2 UDB v8 Development Cert. Guide (2nd edition)

Steva Sanval, David Martineau, Kevin Gashyna y Michael Kyprianou.

IBM Press.



IBM RedBooks (on-line)

<http://www.redbooks.ibm.com/>



DB2 Library (on-line)

<http://www-306.ibm.com/software/data/technical/>



Apéndice C: MySQL

Descargar MySQL 4.1.11 de la web:



<http://www.mysql.com/downloads/index.html>

Apéndice C: MySQL

Arrancar MySQL:

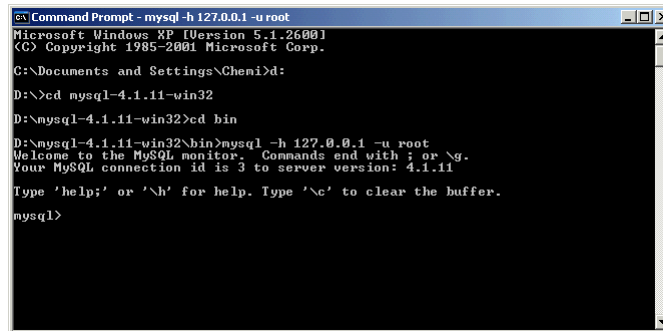
```
Command Prompt - mysqld --standalone
Microsoft Windows [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Chemi>cd
D:\>cd mysql-4.1.11-win32
D:\mysql-4.1.11-win32>cd bin
D:\mysql-4.1.11-win32\bin>mysqld --standalone
=
```

`mysqld.exe --standalone`

Apéndice C: MySQL

Conectar con MySQL:



```
Command Prompt - mysql -h 127.0.0.1 -u root
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

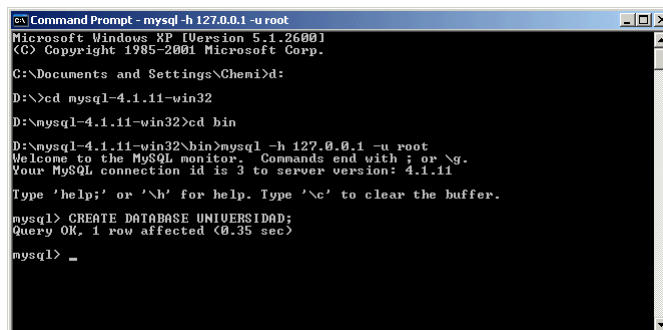
C:\Documents and Settings\Chemi>d:
D:\>cd mysql-4.1.11-win32
D:\mysql-4.1.11-win32>cd bin
D:\mysql-4.1.11-win32\bin>mysql -h 127.0.0.1 -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3 to server version: 4.1.11
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

mysql.exe -h 127.0.0.1 -u root

Apéndice C: MySQL

Crear una Base de Datos nueva:



```
Command Prompt - mysql -h 127.0.0.1 -u root
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Chemi>d:
D:\>cd mysql-4.1.11-win32
D:\mysql-4.1.11-win32>cd bin
D:\mysql-4.1.11-win32\bin>mysql -h 127.0.0.1 -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3 to server version: 4.1.11
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

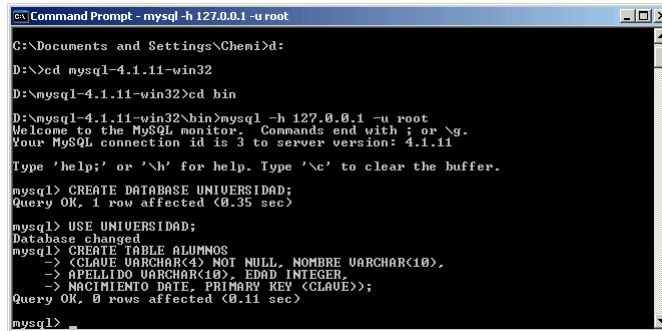
mysql> CREATE DATABASE UNIVERSIDAD;
Query OK, 1 row affected (0.35 sec)

mysql>
```

CREATE DATABASE UNIVERSIDAD;

Apéndice C: MySQL

Crear una tabla nueva en la Base de Datos:



```
Command Prompt - mysql -h 127.0.0.1 -u root
G:\Documents and Settings\Cheni>d:
D:\>cd mysql-4.1.11-win32
D:\mysql-4.1.11-win32>cd bin
D:\mysql-4.1.11-win32\bin>mysql -h 127.0.0.1 -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3 to server version: 4.1.11
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> CREATE DATABASE UNIVERSIDAD;
Query OK, 1 row affected (0.35 sec)

mysql> USE UNIVERSIDAD;
Database changed
mysql> CREATE TABLE ALUMNOS
-> <CLAVE VARCHAR(4) NOT NULL, NOMBRE VARCHAR(10),
-> APELLIDO VARCHAR(10), EDAD INTEGER,
-> NACIMIENTO DATE, PRIMARY KEY (CLAVE));
Query OK, 0 rows affected (0.11 sec)

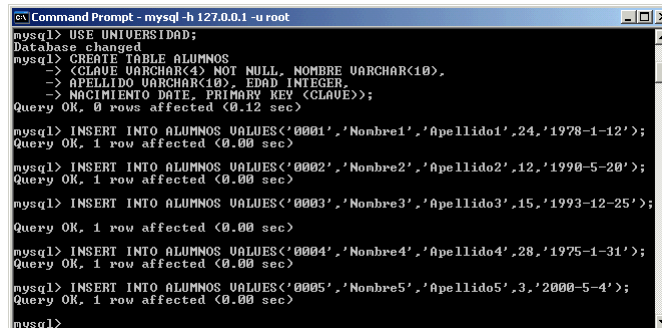
mysql>
```

USE UNIVERSIDAD;

CREATE TABLE ALUMNOS (CLAVE VARCHAR(10),.....);

Apéndice C: MySQL

Introducir tuplas (filas) en la tabla:



```
Command Prompt - mysql -h 127.0.0.1 -u root
mysql> USE UNIVERSIDAD;
Database changed
mysql> CREATE TABLE ALUMNOS
-> <CLAVE VARCHAR(4) NOT NULL, NOMBRE VARCHAR(10),
-> APELLIDO VARCHAR(10), EDAD INTEGER,
-> NACIMIENTO DATE, PRIMARY KEY (CLAVE));
Query OK, 0 rows affected (0.12 sec)

mysql> INSERT INTO ALUMNOS VALUES('0001','Nombre1','Apellido1',24,'1978-1-12');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO ALUMNOS VALUES('0002','Nombre2','Apellido2',12,'1990-5-20');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO ALUMNOS VALUES('0003','Nombre3','Apellido3',15,'1993-12-25');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO ALUMNOS VALUES('0004','Nombre4','Apellido4',28,'1975-1-31');
Query OK, 1 row affected (0.00 sec)

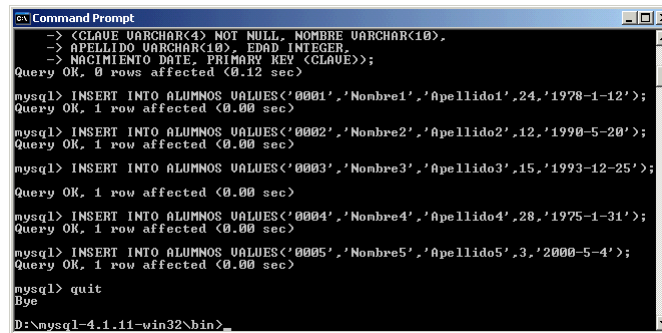
mysql> INSERT INTO ALUMNOS VALUES('0005','Nombre5','Apellido5',3,'2000-5-4');
Query OK, 1 row affected (0.00 sec)

mysql>
```

INSERT INTO ALUMNOS VALUES('0001','Nombre1',.....);

Apéndice C: MySQL

Desconectar de MySQL:



```
C:\mysql-4.1.11-win32\bin>
-> <CLAVE VARCHAR(4) NOT NULL, NOMBRE VARCHAR(10),
-> APELLIDO VARCHAR(10), EDAD INTEGER,
-> NACIMIENTO DATE, PRIMARY KEY (CLAVE));
Query OK, 0 rows affected (0.12 sec)

mysql> INSERT INTO ALUMNOS VALUES('0001','Nombre1','Apellido1',24,'1978-1-12');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO ALUMNOS VALUES('0002','Nombre2','Apellido2',12,'1990-5-20');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO ALUMNOS VALUES('0003','Nombre3','Apellido3',15,'1993-12-25');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO ALUMNOS VALUES('0004','Nombre4','Apellido4',28,'1975-1-31');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO ALUMNOS VALUES('0005','Nombre5','Apellido5',3,'2000-5-4');
Query OK, 1 row affected (0.00 sec)

mysql> quit
Bye
D:\mysql-4.1.11-win32\bin>
```

quit;

Apéndice C: MySQL

Desarrollar la aplicación Java:

MySQL no incluye Drivers JDBC. Se descargan aparte.

Estos Drivers JDBC no están incluidos en la JDK.

Por tanto, hay que tener en cuenta añadir el fichero que contenga los drivers en la variable de sistema CLASSPATH.

set CLASSPATH=%CLASSPATH%;c:\mysql-connector-java-3.1.7-bin.jar

Vamos a usar el driver de tipo 3:

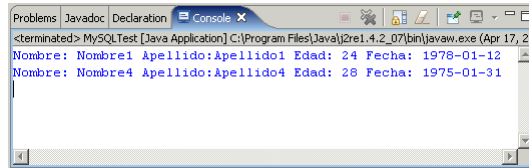
com.mysql.jdbc.Driver

La URL JDBC es: jdbc:mysql://127.0.0.1/UNIVERSIDAD

Apéndice C

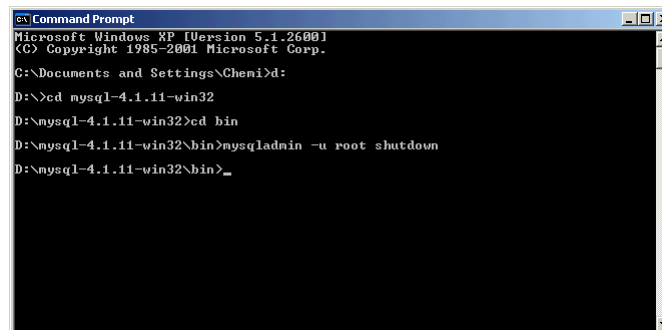
```
import java.sql.*;

public class MySQLTest
{
    public static void main(String[] args)
    {
        try
        {
            Class.forName("com.mysql.jdbc.Driver");
            Connection con = DriverManager.getConnection("jdbc:mysql://127.0.0.1/UNIVERSIDAD");
            PreparedStatement pstmt = con.prepareStatement("SELECT * FROM ALUMNOS WHERE EDAD > ?");
            pstmt.setInt(1,21);
            ResultSet rs = pstmt.executeQuery();
            while(rs.next())
            {
                System.out.println("Nombre: " + rs.getString(2) + " Apellido: " + rs.getString(3) + " Edad: " +
                    rs.getInt(4) + " Fecha: " + rs.getDate(5));
            }
            rs.close();
            pstmt.close();
            con.close();
        }
        catch(ClassNotFoundException ex)
        {
            ex.printStackTrace();
        }
        catch(SQLException ex)
        {
            ex.printStackTrace();
        }
    }
}
```



Apéndice C: MySQL

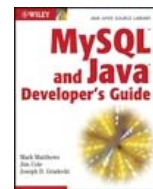
 Detener MySQL:



 mysqladmin.exe -u root shutdown

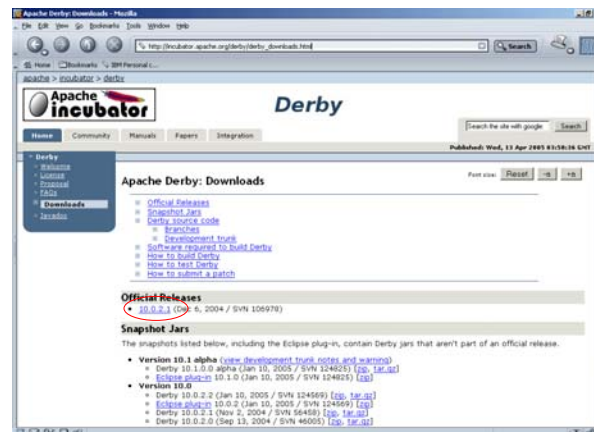
Apendice C: Bibliografía

- Managing & Using MySQL (2nd edition)
George Reese, Randy Jay Yarger y Tim King.
O'Reilly.
- MySQL Cookbook
Paul DuBois.
O'Reilly.
- MySQL and Java Developer's Guide
Mark Matthews, Jim Cole y Joseph D. Gradecki.
Wiley.
- MySQL Documentation (on-line)
<http://www.mysql.com/documentation/index.html>



Apéndice D: Apache Derby

- Descargar Apache Derby 10.0.2.1 de la web:



http://incubator.apache.org/derby/derby_downloads.html

Apéndice D: Apache Derby

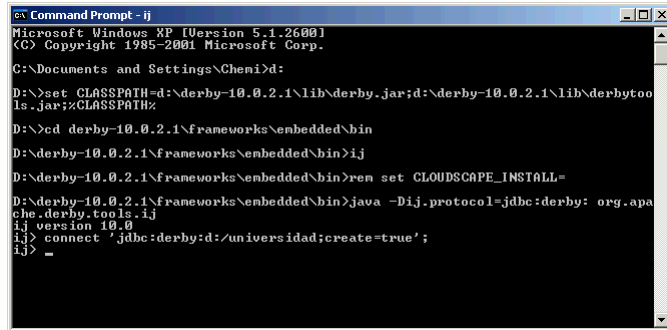
- Apache Derby es un SGDB 100% Java.
- Es la edición Open Source de IBM Cloudscape.
- Permite dos modos de trabajo:
 - Modo embebido (Derby Embedded).
 - Modo servidor (Derby Network Server).
- Modo embebido:
 - El gestor se ejecuta en la misma JVM que la aplicación.
 - No es multiusuario.
 - El gestor se arranca al cargar el Driver JDBC.
 - Utiliza Driver JDBC Tipo 2.

Apéndice D: Apache Derby

- Modo servidor:
 - El gestor se ejecuta en su propia JVM.
 - Es multiusuario.
 - La comunicación con el gestor es vía TCP/IP.
 - Utiliza Driver JDBC Tipo 4.
 - El gestor se arranca con el script startNetworkServer.bat/ksh
- Incluye un intérprete de línea de comandos: ij
 - ij.bat/ksh ó java.exe org.apache.derby.tools.ij

Apéndice D: Apache Derby

Crear una Base de Datos nueva:



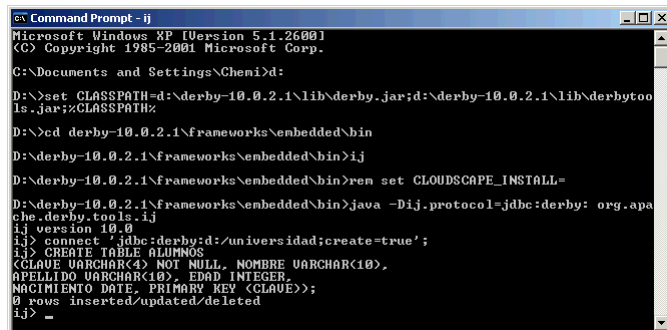
```
Command Prompt - ij
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Cheni>d:
D:\>set CLASSPATH=d:\derby-10.0.2.1\lib\derby.jar;d:\derby-10.0.2.1\lib\derbytools.jar;%CLASSPATH%
D:\>cd derby-10.0.2.1\frameworks\embedded\bin
D:\derby-10.0.2.1\frameworks\embedded\bin>ij
D:\derby-10.0.2.1\frameworks\embedded\bin>ren set CLOUDSCAPE_INSTALL=
D:\derby-10.0.2.1\frameworks\embedded\bin>java -Dij.protocol=jdbc:derby: org.apache.derby.tools.ij
ij version 10.0
ij> connect 'jdbc:derby:d:/universidad;create=true';
ij> _
```

Añadir al CLASSPATH: derby.jar, derbytools.jar
ij.bat
connect 'jdbc:derby:d:/universidad;create=true';

Apéndice D: Apache Derby

Crear una tabla nueva en la Base de Datos:



```
Command Prompt - ij
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Cheni>d:
D:\>set CLASSPATH=d:\derby-10.0.2.1\lib\derby.jar;d:\derby-10.0.2.1\lib\derbytools.jar;%CLASSPATH%
D:\>cd derby-10.0.2.1\frameworks\embedded\bin
D:\derby-10.0.2.1\frameworks\embedded\bin>ij
D:\derby-10.0.2.1\frameworks\embedded\bin>ren set CLOUDSCAPE_INSTALL=
D:\derby-10.0.2.1\frameworks\embedded\bin>java -Dij.protocol=jdbc:derby: org.apache.derby.tools.ij
ij version 10.0
ij> connect 'jdbc:derby:d:/universidad;create=true';
ij> CREATE TABLE ALUMNOS
(CLAVE VARCHAR(4) NOT NULL, NOMBRE VARCHAR(10),
 APELLIDO VARCHAR(10), EDAD INTEGER,
 NACIMIENTO DATE, PRIMARY KEY (CLAVE));
0 rows inserted/updated/deleted
ij> _
```

CREATE TABLE ALUMNOS (CLAVE VARCHAR(10),.....);

Apéndice D: Apache Derby

Introducir tuplas (filas) en la tabla:

```
Command Prompt - ij
D:\derby-10.0.2.1\frameworks\embedded\bin>ij
D:\derby-10.0.2.1\frameworks\embedded\bin>run set CLOUDSCAPE_INSTALL=
D:\derby-10.0.2.1\frameworks\embedded\bin>java -Dij.protocol=jdbc:derby: org.apa
che.derby.tools.ij
ij version 10.0
ij> connect 'jdbc:derby:d:/universidad;create=true';
ij> CREATE TABLE ALUMNOS
(CLAUE VARCHAR(4) NOT NULL, NOMBRE VARCHAR(10),
APELLIDO VARCHAR(10), EDAD INTEGER,
NACIMIENTO DATE, PRIMARY KEY (CLAUE));
0 rows inserted/updated/deleted
ij> INSERT INTO ALUMNOS VALUES('0001','Nombre1','Apellido1',24,'1978-1-12');
1 row inserted/updated/deleted
ij> INSERT INTO ALUMNOS VALUES('0002','Nombre2','Apellido2',12,'1990-5-20');
1 row inserted/updated/deleted
ij> INSERT INTO ALUMNOS VALUES('0003','Nombre3','Apellido3',15,'1993-12-25');
1 row inserted/updated/deleted
ij> INSERT INTO ALUMNOS VALUES('0004','Nombre4','Apellido4',28,'1975-1-31');
1 row inserted/updated/deleted
ij> INSERT INTO ALUMNOS VALUES('0005','Nombre5','Apellido5',3,'2000-5-4');
1 row inserted/updated/deleted
ij>
```

INSERT INTO ALUMNOS VALUES('0001','Nombre1',.....);

Apéndice D: Apache Derby

Desconectar de Apache Derby:

```
Command Prompt
D:\derby-10.0.2.1\frameworks\embedded\bin>run set CLOUDSCAPE_INSTALL=
D:\derby-10.0.2.1\frameworks\embedded\bin>java -Dij.protocol=jdbc:derby: org.apa
che.derby.tools.ij
ij version 10.0
ij> connect 'jdbc:derby:d:/universidad;create=true';
ij> CREATE TABLE ALUMNOS
(CLAUE VARCHAR(4) NOT NULL, NOMBRE VARCHAR(10),
APELLIDO VARCHAR(10), EDAD INTEGER,
NACIMIENTO DATE, PRIMARY KEY (CLAUE));
0 rows inserted/updated/deleted
ij> INSERT INTO ALUMNOS VALUES('0001','Nombre1','Apellido1',24,'1978-1-12');
1 row inserted/updated/deleted
ij> INSERT INTO ALUMNOS VALUES('0002','Nombre2','Apellido2',12,'1990-5-20');
1 row inserted/updated/deleted
ij> INSERT INTO ALUMNOS VALUES('0003','Nombre3','Apellido3',15,'1993-12-25');
1 row inserted/updated/deleted
ij> INSERT INTO ALUMNOS VALUES('0004','Nombre4','Apellido4',28,'1975-1-31');
1 row inserted/updated/deleted
ij> INSERT INTO ALUMNOS VALUES('0005','Nombre5','Apellido5',3,'2000-5-4');
1 row inserted/updated/deleted
ij> disconnect;
ij> exit;
D:\derby-10.0.2.1\frameworks\embedded\bin>
```

disconnect;

exit;

Apéndice D: Apache Derby

- Desarrollar la aplicación Java (modo embebido):
 - Apache Derby incluye los Drivers JDBC tipo 2.
 - Estos Drivers JDBC no están incluidos en la JDK.
 - Por tanto, hay que tener en cuenta añadir el fichero que contenga los drivers en la variable de sistema CLASSPATH.
set CLASSPATH=%CLASSPATH%;c:\derby-10.0.2.1\lib\derby.jar
 - Vamos a usar el driver de tipo 2:
org.apache.derby.jdbc.EmbeddedDriver
 - La URL JDBC es: jdbc:derby:d:/universidad

```
import java.sql.*;
```

```
public class ApacheDerbyTest
```

```
{  
    public static void main(String[] args)
```

```
{  
    try
```

```
{  
        Class.forName("org.apache.derby.jdbc.EmbeddedDriver");  
        Connection con = DriverManager.getConnection("jdbc:derby:d:/universidad");  
        PreparedStatement pstmt = con.prepareStatement("SELECT * FROM ALUMNOS WHERE EDAD > ?");  
        pstmt.setInt(1,21);  
        ResultSet rs = pstmt.executeQuery();  
        while(rs.next())  
            System.out.println("Nombre: " + rs.getString(2) + " Apellido: " + rs.getString(3) + " Edad: " +  
                + rs.getInt(4) + " Fecha: " + rs.getDate(5));  
        rs.close();  
        pstmt.close();  
        con.close();  
        DriverManager.getConnection("jdbc:derby:d:/universidad;shutdown=true");  
    }  
}
```

```
catch(ClassNotFoundException ex)
```

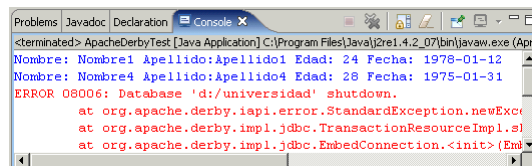
```
{  
    ex.printStackTrace();
```

```
}  
catch(SQLException ex)
```

```
{  
    ex.printStackTrace();
```

```
}  
}
```

Apéndice D



Apéndice D: Apache Derby



Desarrollar la aplicación Java (modo servidor):



Apache Derby no incluye los Drivers JDBC tipo 4.



Estos Drivers JDBC no están incluidos en la JDK.



Por tanto, hay que tener en cuenta añadir el fichero que contenga los drivers en la variable de sistema CLASSPATH.

```
set CLASSPATH=c:\sql\lib\java\db2jcc.jar;c:\sql\lib\java\db2jcc_license_cu.jar
```



Vamos a usar el driver de tipo 4:

```
com.ibm.db2.jcc.DB2Driver
```



La URL JDBC es:

```
jdbc:derby:net://127.0.0.1:1527/"d:/universidad":user=APP;password=APP;
```

Nota: es obligatorio usar user/password. Por defecto es APP/APP.

Apéndice D: Apache Derby



Arrancar Apache Derby:

```
Microsoft Windows [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Chemi>d:
D:\>set CLASSPATH=d:\derby-10.0.2.1\lib\derby.jar;d:\derby-10.0.2.1\lib\derbytools.jar;d:\derby-10.0.2.1\lib\derbynet.jar
D:\>cd derby-10.0.2.1\frameworks\NetworkServer\bin
D:\derby-10.0.2.1\frameworks\NetworkServer\bin>startNetworkServer.bat
D:\derby-10.0.2.1\frameworks\NetworkServer\bin>ren set CLOUDSCAPE_INSTALL=
D:\derby-10.0.2.1\frameworks\NetworkServer\bin>java org.apache.derby.drda.NetworkServerControl start
Server is ready to accept connections on port 1527.
```



Añadir al CLASSPATH: derby.jar, derbytools.jar, derbynet.jar



startNetworkServer.bat

```
import java.sql.*;
```

```
public class ApacheDerbyTest
```

```
{
    public static void main(String[] args)
```

```
{
    try
```

```
{
```

```
        Class.forName("com.ibm.db2.jcc.DB2Driver");
```

```
        Connection con = DriverManager.getConnection
```

```
            ("jdbc:derby:net://127.0.0.1:1527/^d:/universidad":user=APP;password=APP;");
```

```
        PreparedStatement pstmt = con.prepareStatement("SELECT * FROM ALUMNOS WHERE EDAD > ?");
```

```
        pstmt.setInt(1,21);
```

```
        ResultSet rs = pstmt.executeQuery();
```

```
        while(rs.next())
```

```
            System.out.println("Nombre: " + rs.getString(2) + " Apellido: " + rs.getString(3) + " Edad: " +
```

```
                + rs.getInt(4) + " Fecha: " + rs.getDate(5));
```

```
        rs.close();
```

```
        pstmt.close();
```

```
        con.close();
```

```
    }
```

```
    catch(ClassNotFoundException ex)
```

```
    {
        ex.printStackTrace();
```

```
    }
    catch(SQLException ex)
```

```
    {
        ex.printStackTrace();
```

```
    }
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

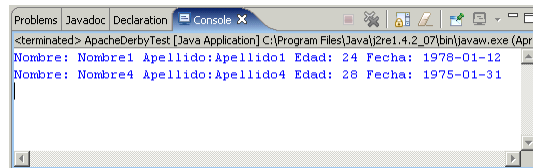
```
}
```

```
}
```

```
}
```

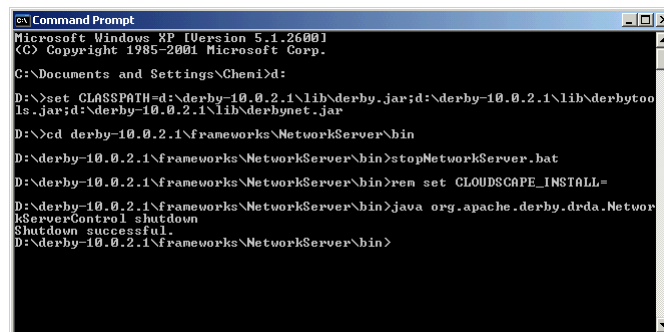
```
}
```


Apéndice D




Apéndice D: Apache Derby

 Detener Apache Derby:



 Añadir al CLASSPATH: derby.jar, derbytools.jar, derbynet.jar

 stopNetworkServer.bat

Apéndice D: Bibliografía



Apache Derby -- Off to the Races

Paul C. Zikopolus, George Baklarz y Dan Scott.
Prentice Hall.



Apache Derby Documentation (on-line)

<http://incubator.apache.org/derby/manuals/index.html>