Select

¿Por qué no usar ODBC desde Java? Se puede usar ODBC desde Java Puente JDBC-ODBC ¿Por qué se necesita JDBC? ODBC no es apropiado para su uso directo desde Java porque usa una interface en C Una traducción de la ODBC API en C a una API en Java no sería deseable ODBC es duro de aprender Una API en Java como JDBC es necesaria para conseguir una solución âpuramente Javaâ JDBC API es una interface natural de Java getCatalogName() Nombre de la columna en el catálogo de la base de datos getColumnName() Nombre de la columna getColumnLabel() Nombre a utilizar a la hora de imprimir el nombre de la columna getColumnDisplaySize() Ancho máximo en caracteres necesario para mostrar el contenido de la columna getColumnCount() Número de columnas en el ResultSet getTableName()Nombre de la tabla a que pertenece la columna getPrecision() Número de dígitos de la columna getScale() Número de decimales para la columna getColumnType() Tipo de la columna (uno de los tipos SQL en java.sql.Types) getColumnTypeName() Nombre del tipo de la columna isSigned() Para números, indica si la columna corresponde a un número con sign isAutoIncrement() Indica si la columna es de tipo autoincremento isCurrency() Indica si la columna contiene un valor monetario isCaseSensitive() Indica si la columna contiene un texto sensible a mayúsculas isNullable() Indica si la columna puede contener un NULL SQL. Puede devolver los valores columnNoNulls, columnNullable o columnNullableUnknown, miembros finales estáticos de ResultSetMetaData (constantes) isReadOnly() Indica si la columna es de solo lectura isWritable() Indica si la columna puede modificarse, aunque no lo garantiza isDefinitivelyWritable() Indica si es absolutamente seguro que la columna se puede modificar isSearchable() Indica si es posible utilizar la columna para determinar los criterios de búsqueda de un SELECT getSchemaName() Devuelve el texto correspondiente al esquema de la base de datos para esa columna Connection Representa la conexión con la base de datos. Es el objeto que permite realizar las consultas SQL y obtener los resultados de dichas consultas. Es el objeto base para la creación de los objetos de acceso a la base de datos. DriverManager Encargado de mantener los drivers que están disponibles en una aplicación concreta. Es el objeto que mantiene las funciones de administración de las operaciones que se realizan con la base de datos. Statement Se utiliza para enviar las sentencias SQL simples, aquellas que no necesitan parámetros, a la base de datos. PreparedStatement Tiene una relación de herencia con el objeto Statement, añadiéndole la funcionalidad de poder utilizar parámetros de entrada. Además, tiene la particularidad de que la pregunta ya ha sido compilada antes de ser realizada, por lo que se denomina preparada. La principal ventaja, aparte de la utilización de parámetros, es la rapidez de ejecución de la pregunta. CallableStatement Tiene una relación de herencia cn el objeto PreparedStatement. Permite utilizar funciones implementadas directamente sobre el sistema de gestión de la base de datos. Teniendo en cuenta que éste posee información adicional sobre el uso de las estructuras internas, índices, etc.; las funciones se realizarán de forma más eficiente. Este tipo de operaciones es muy utilizada en el caso de ser funciones muy complicadas o bien que vayan a ser ejecutadas varias veces a lo largo del tiempo de vida de la aplicación. ResultSet Contiene la tabla resultado de la pregunta SQL que se haya realizado. En párrafos anteriores se han comentado los métodos que proporciona este objeto para recorrer dicha tabla. Clase que hace la conexion java con Access, solo hace la conexion no hay consultas import java.sql.*; public class Conexion{ Connection con=null; //ResultSet rs=null; //registros de la consulta Statement st=null; //prepara la base de datos String driver= "sun.jdbc.odbc.JdbcOdbcDriver"; // url del driver String url= "jdbc:odbc:Hotel";//nombre de la base de datos String usuario=""; String clave=""; Conexion(){ try{ Class.forName(driver); System.out.println("Driver Ok"); } catch(ClassNotFoundException e){ System.out.println("Error al cargar Driver"+e); } try{ con=DriverManager.getConnection(url,usuario,clave) ; System.out.println("Conexion OK"); } catch(SQLException e){ System.out.println("No hay Conexion"+e); } try{ st=con.createStatement(); System.out.println("Preparada"); } catch(SQLException e){ System.out.println("no Preparada " +e); } } }