Accés a base de dades

Mysql

Connexió a base de dades

- Crearem instàncies de les classes que estan en els paquets java.sql i javax.sql i, si ens interessa, les manipularem amb els mecanismes convencionals d'herència.
- Els mecanismes de connexió dels drivers JDBC són fàcils i, com gairebé tot en Java, independents de plataforma. Per tal d'aconseguir una connexió amb una base de dades, senzillament hem de crear un objecte connexió seguint aquesta fórmula:

```
String usuari = "";
String clau = "";
String  urlDades="jdbc:mysql://localhost/NOM_BASE_DADES";
Class.forName("com.mysql.jdbc.Driver ");
Connection connexio = DriverManager.getConnection(urlDades, usuari,clau);
Statement pregunta = connexio.createStatement();
ResultSet resposta = pregunta.executeQuery("select * from usuaris");
```

Connexió a base de dades

- Les cadenes urlDades, usuari i clau descriuen la Url de la base de dades, el nom d'usuari i la contrasenya d'accés a la base de dades. Si una base de dades no està protegida per contrasenya, les podem crear en blanc.
- Carreguem el controlador amb **Class.forname("com.mysql.jdbc.Driver")**

  Cada fabricant de bases de dades té el seu.
- Creem un objecte **java.sql.Connection**, que és el que conté la informació de la connexió amb la base de dades.

  Aquest objecte es crea des del gestor de drivers en passar-li la ubicació de la base de dades, el nom d'usuari i la contrassenya:

  Connection connexio = DriverManager.getConnection(urlDades,usuari,clau)

  a on **urlDades="jdbc:mysql://localhost/NOM_BASE_DADES"**
- Quan ja tenim una connexió de dades, ja podem demanar-li la informació. Ho fem creant consultes o preguntes:

  Statement pregunta = connexio.createStatement();

  i cedint el seu resultat als objectes resposta:

  ResultSet resposta =pregunta.executeQuery("select * from usuaris");

  Aquesta darrera cadena conté el codi SQL mitjançant el qual li demanem al servidor Access això: "selecciona i retorna tots els camps i registres de la taula usuaris".
- Els objectes resposta **ResultSet** representen una matriu de dades ordenades en forma de registres i camps, als quals podem recórrer per extreure'n la informació.

Connexió a base de dades

- Quan ja tenim la connexió, llancem la pregunta **Statement** i, llavors, el motor de dades ens retorna la resposta **ResultSet**.
- L'objecte **ResultSet** conté una matriu de dades, amb un registre per a cada registre de la taula que Access ens ha retornat. Podem navegar a través d'un **ResultSet** amb els mètodes **first()**, **last()**, o **next()**. El que fem en aquest cas és fer una lectura seqüencial de ResultSet des del primer registre fins al darrer amb un cicle **while (resposta.next())**.

  Per a cada pas del cicle, fem una sortida a consola amb els camps id, nom i cognom1 de l'usuari de la biblioteca.
- Finalment, tanquem **Statement** -que ens tanca automàticament **ResultSet**, i tanquem la connexió amb la base de dades: **pregunta.close()** i **connexio.close()**.
- Tot està estrictament protegit per possibles excepcions: **ClassNotFoundException** per si no està disponible el driver JDBC del motor de dades i **SQLException** per les excepcions que es produeixin en el nostre diàleg amb la base de dades.

Objecte ResultSet

- El cursor és la posició activa o actual del **ResulSet**. Un **ResultSet** no pot"veure" totes les files simultàniament, necessita una eina per a saber quina és la fila que està tractant en un moment donat, això és el cursor, una forma d'apuntar o senyalar a la posició activa o actual.
- D'aquesta manera una crida un mètode **getXXX()** sap en tot moment a quina fila es refereix la petició de dades. Hi ha una particularitat, al executar la sentencia **executeQuery** el cursor es situa en una pseudofila: l'anterior a la primera, llavors diem que el conjunt "apunta" a la posició anterior ala primera fila. Per aquesta raó per a obtenir la primera fila el primer que hem de fer es cridar a **next()**, es a dir, demanar-li al conjunt que avanci el cursor una posició, a la primera fila "real":

```
/*** Recorrer fila a fila todo el resultado ****/
while ( rs.next() )
    System.out.println( rs.getString( col1 ) + ", " + rs.getString( col2 ) );

/*** Nos ponemos en el primero y lo imprimimos ***/
rs.first();
System.out.println( "PRIMERO: " + rs.getString( col1 ) + ", " + rs.getString( col2 ) );

/*** Nos ponemos en el último y lo imprimimos ***/
rs.last();
System.out.println( "ULTIMO: " + rs.getString( col1 ) + ", " + rs.getString( col2 ) );

/*** Nos ponemos en el antepenúltimo y lo imprimimos ***/
rs.relative(-2);
System.out.println( "PENULTIMO: " + rs.getString( col1 ) + ", " + rs.getString( col2 ) );

/*** Volvemos a recorrer fila a fila todo el resultado ****/
rs.beforeFirst();  // Ojo: si no lo pongo, no comienzo en el primero
while ( rs.next() )
    System.out.println( rs.getString( col1 ) + ", " + rs.getString( col2 ) );
```

Objecte ResultSet

- **first()**, **last()** o **relative()** són auto evidents. Són formes de desplaçar-nos per un conjunt de resultats d'una sentència.

  👁: **NAVEGUEM PER UN CONJUNT DE RESULTADOS D'UNA SENTENCIA, NO PER LA TOTALITAT DE LA TAULA.**

  El que succeeix es que en el nostre cas no tenim cap clàusula WHERE i, per tant, les files de la sentencia coincideixen amb les files de la taula (però no sempre va a ser així).

- Per a tornar a recórrer tot el conjunt des del principi, abans d'usar **next()** hem de posar el cursor en la pseudofila, la posició anterior a la primera:

```
/*** Volvemos a recorrer fila a fila todo el resultado ****/
rs.beforeFirst();  // Ojo: si no lo pongo, no comienzo en el primero
while ( rs.next() )
    System.out.println( rs.getString( col1 ) + ", " + rs.getString( col2 ) );
```

Objecte ResultSet

- Tots els mètodes (tant els anteriors com els que veurem ara, a excepció de **getRow**) retornen **false** en cas de que el desplaçament es faci fora del **ResultSet** (a una pseudofila). No hi ha desplaçaments cap a enrere si el cursor apunta a la posició anterior a la primera fila. De la mateixa manera, no hi ha desplaçament cap a endavant si el cursor apunta a la posició posterior a la última.
- Altres mètodes:
  - **previous()**: desplaça el cursor a la posició anterior.
  - **afterLast()**: desplaça el cursor a la posició següent a la última (una pseudofila).
  - **absolute(int posició)**: desplaçament absolut a la posició senyalada per l'argument.
  - **getRow()**: retorna la posició d'una fila, **zero** si el cursor apunta a una paseudofila.
  - **isFirst()**: **true** si el cursor apunta a la primera, **false** en cas contrari.
  - **isLast()**: idem per a l'última.

ResultSet amb desplaçament

- ResulSet pot tenir o no desplaçament i pot ser sensible o no als canvis en la base de dades que poden produir altres usuaris o processos.
- Tipus de ResultSet:

TYPE_FORWARD_ONLY	Només es pot recórrer cap endavant i no és sensible a canvis en base de dades.
TYPE_SCROLL_INSENSITIVE	Té desplaçament (endavant/enrere), però és insensible a canvis en base de dades
TYPE_SCROLL_SENSITIVE	Té desplaçament (endavant/enrere) i és sensible a canvis en base de dades

- Amb el següent codi comprovem que el ResultSet tingui desplaçament:

```
DatabaseMetaData dbmd = con.getMetaData();
if ( dbmd.supportsResultSetType( ResultSet.TYPE_FORWARD_ONLY ) )
    System.out.println("Esta base de datos no admite scroll");
else
    System.out.println("Esta base de datos admite scroll");
```

- A més a més hi ha una versió de createStatement que dispararà una excepció del tipus SQLException, en el cas de que la base de dades no permeti el tipus assenyalat en (resultSetType):

  public Statement createStatement(int resultSetType, int resultSetConcurrency) throws SQLException

- També podem saber el tipus d'un conjunt mitjançant un mètode de ResulSet:
  - **public int getType()**: Retorna el tipus en la forma de les constants abans descrites: TYPE_FORWARD_ONLY, etc.

ResultSet amb desplaçament

ResultSet.CONCUR_READ_ONLY	**Si volem que les dades es puguin llegir però no actualitzar**
ResultSet.CONCUR_UPDATABLE	Si volem permetre que la base de dades sigui actualitzable mitjançant l'objecte ResultSet

- Si no s'utilitza el mètode sense paràmetres l'objecte serà TYPE_FORWARD_ONLY i CONCUR_READ_ONLY.

Obtenir dades

- Obtenir dades sense paràmetres:

```
Statement pregunta = connexio.createStatement();
ResultSet resposta = pregunta.executeQuery(
                     "select * from usuaris order by
cognom1");
    while (resposta.next()) {}
```

Quan es tracta d'una consulta d'obtenció de dades fem **pregunta.executeQuery()**.

Afegir Registres

- Insertar dades passant paràmetres:

```
PreparedStatement pregunta =
  connexio.prepareStatement(
  "insert into usuaris (nom,cognom1,cognom2) values
(?,?,?)");
pregunta.setString(1,nom);
pregunta.setString(2,cognom1);
pregunta.setString(3,cognom2);
pregunta.executeUpdate();
```

Quan la consulta és d'actualització resolem **Statement** amb **pregunta.executeUpdate()**.

Esborrar-ne registres

- Esborrar dades passant paràmetres:

```
PreparedStatement pregunta =
  connexio.prepareStatement("delete from usuaris where
id=?");
pregunta.setString(1,usuari);
pregunta.executeUpdate();
```

Quan la consulta és d'actualització resolem **Statement** amb **pregunta.executeUpdate()**.

Actualitzar informació i llistar-la

- Actualitzar dades passant paràmetres:

```
PreparedStatement pregunta =
   connexio.prepareStatement(
   "update usuaris set nom=?,cognom1=?,cognom2=? where
id=?");
pregunta.setString(1,nom);
pregunta.setString(2,cognom1);
pregunta.setString(3,cognom2);
pregunta.setString(4,usuari);
pregunta.executeUpdate();
```

Per a les consultes amb paràmetres, no utilitzem **Statement** sinó **PreparedStatement**. Els llocs de la consulta que han de ser substituïts pel paràmetre els indiquem amb el símbol '?'. Seguidament, per assignar el valor al paràmetre fem **pregunta.setString(1,nom)**.Això substitueix el primer símbol '?' del la consulta SQL pel contingut de la variable nom. Si hem de passar un nombre enter hem d'escriure **pregunta.setInt(1,numero)**.