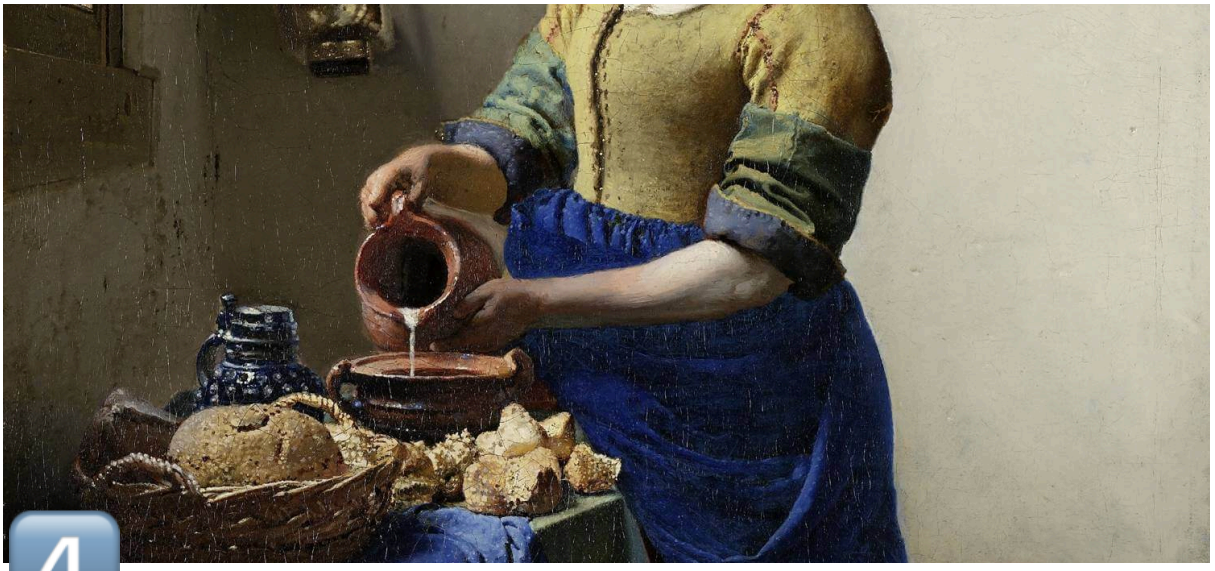

RESULTSET

1. Interfaz ResultSet

💡 La interfaz ResultSet dispone de métodos para recuperar y manipular los resultados de consultas ejecutadas.

1.1 Características principales:

▼ Tipos de cursor:

El primer argumento de los métodos `createStatement`, `prepareStatement` y `prepareCall` define el tipo de ResultSet

- `TYPE_FORWARD_ONLY`: Cursor avanza solo hacia adelante (desde antes de la primera fila hasta después de la última).
- `TYPE_SCROLL_INSENSITIVE`: Cursor puede moverse en ambas direcciones y hacía una posición absoluta, pero no refleja cambios en la base de datos.
- `TYPE_SCROLL_SENSITIVE`: posee las mismas características que el anterior, pero en este caso si que refleja cambios en la base de datos.

```
con.getMetaData().supportsResultSetType(ResultSet.TYPE_SCROLL_INSENS
System.out.println("Soporta TYPE_SCROLL_INSENSITIVE");
```

▼ Concurrencia (Actualizable o no):

El segundo argumento de los métodos `createStatement`, `prepareStatement` y `prepareCall` es el tipo de concurrencia, que determina que **nivel de funcionalidad de actualización se admite**

- `CONCUR_READ_ONLY`: Solo lectura (predeterminado).
- `CONCUR_UPDATABLE`: Permite actualizar datos.

> *No todos los controladores JDBC admiten concurrenncia*

```
con.getMetaData().supportsResultSetConcurrency(ResultSet.TYPE_SCROLL_
                    ResultSet.CONCUR_UPDATABLE)
```

▼ Retención del cursor:

Sirve para configurar el cierre (o no cierre del objeto ResultSet) cuando se llama al método Commit

- `HOLD_CURSORS_OVER_COMMIT`: Cursor permanece abierto tras `commit`.
- `CLOSE_CURSORS_AT_COMMIT`: Cursor se cierra tras `commit`.

---

2. Recuperación de Valores

- **Métodos principales**:
  - `getInt`, `getString`, `getDate`, etc., para recuperar valores desde la fila actual.
- **Acceso a columnas**:
  - Por índice (más eficiente, columnas numeradas desde 1).
  - Por nombre o alias (menos eficiente, pero útil para columnas nombradas explícitamente en la consulta).
- **Ejemplo**:

```
while (rs.next()) {
    String nombreCafe = rs.getString(1);
    int idProveedor = rs.getInt(2);
    // Otros valores...
}
```

---

3. Moviendo el Cursor

- **Posicionamiento del cursor**:
  - `next`: Avanza una fila.
  - `previous`: Retrocede una fila.
  - `first`: Primera fila.
  - `last`: Última fila.
  - `beforeFirst`: Antes de la primera fila.
  - `afterLast`: Después de la última fila.
  - `absolute(n)`: Fila n específica.
  - `relative(n)`: Desplazamiento relativo a la posición actual.
- **Limitaciones**:

- Tipo predeterminado (`TYPE_FORWARD_ONLY`) no admite movimientos, salvo `next`.

---

4. Actualización de Filas

- **Requisitos**:
  - El `ResultSet` debe ser `CONCUR_UPDATABLE` y permitir desplazamiento (`TYPE_SCROLL_SENSITIVE` o `TYPE_SCROLL_INSENSITIVE`).
- **Métodos de actualización**:
  - Ejemplo: `updateInt`, `updateFloat`, `updateString`, etc.
  - Debe llamarse a `updateRow` para hacer efectivos los cambios en la base de datos.
- **Ejemplo**:

```
ResultSet uprs = stmt.executeQuery("SELECT * FROM Cafe");
while (uprs.next()) {
   float precio = uprs.getFloat("precio");
   uprs.updateFloat("precio", precio * porcentaje);
   uprs.updateRow();
}
```

---

5. Uso de Objetos Statement

- Un `ResultSet` se puede crear usando:
  - `Statement`
  - `PreparedStatement`
  - `CallableStatement`
  - `RowSet`

6. Notas Adicionales

- **Compatibilidad**:

- No todos los tipos de `ResultSet` o niveles de concurrencia son soportados por todos los controladores JDBC.
- Métodos como `DatabaseMetaData.supportsResultSetType` o `supportsResultSetConcurrency` verifican compatibilidad.

- **Manejo de Excepciones**: Siempre envolver las operaciones con bloques `try-catch`.

Código de Ejemplo: Mostrar una Tabla

```
public static void showCafes(Connection con) throws SQLException {
    String query = "SELECT nome, idProveedor, precio, ventas, total FROM Ca
fe";
    try (Statement stmt = con.createStatement()) {
        ResultSet rs = stmt.executeQuery(query);
        while (rs.next()) {
            System.out.println(rs.getString("nome") + ", " + rs.getInt("idProveedo
r") +
                ", " + rs.getFloat("precio") + ", " + rs.getInt("ventas") +
                ", " + rs.getInt("total"));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

Código de Ejemplo: Actualizar Precios

```
public void actualizarPrecios(float porcentaje) throws SQLException {
    try (Statement stmt = con.createStatement(
        ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE))
{
        ResultSet rs = stmt.executeQuery("SELECT * FROM Cafe");
        while (rs.next()) {
            float precioActual = rs.getFloat("precio");
            rs.updateFloat("precio", precioActual * porcentaje);
            rs.updateRow();
```

```
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```