


Open · Opened 2 days ago by  **Rubén Montero**

Más de una columna

Resumen

- Introduciremos consultas que recuperan más de una columna
- Hablaremos del carácter *wildcard* (`*`)
- Realizaremos aclaraciones sobre *qué índice* de columna se pasa a `.getXXX`
- Añadiremos un nuevo método a `MoviesDataProvider` que:
 - Abra una conexión
 - Lance una consulta que recupere más de una columna
 - Acumule los resultados en un `ArrayList`
 - Cierre la conexión
 - Devuelva (`return`) los resultados

Descripción

Nuestra primera consulta era:

```
SELECT title
FROM TMovies;
```

...que recupera los **títulos** de cada fila:

id	title	year	duration	countryIso3166	genre	synopsis
1	The Shawshank Redemption	1994	142	US	drama	Acusado del asesinato de su mujer, Andrew Dufresne (Tim Robbins), tras ser condenado a cadena perpetua, es enviado a la cárcel de Shawshank. Con el paso de los años...
2	Secondhand Lions	2003	109	US	comedy	Cuando al introvertido Walter (Haley Joel Osment) lo obliga su irresponsable madre (Kyra Sedgwick) a pasar el verano en un rancho de Texas con sus viejos y excéntricos tíos (Michael Caine y Robert Duvall), a los que apenas conoce, la idea no le hace ninguna gracia. Pero tampoco a ellos les agrada mucho la idea de cuidar del chaval. Sin embargo...
...

Esta consulta sólo selecciona una columna

Correcto. La columna `title`.

¿Se puede seleccionar **más** de un campo en una consulta?

Sí. Basta con especificarlos separados por comas (`,`).

Por ejemplo:

```
SELECT title, year, duration
FROM TMovies;
```

ó

```
SELECT id, countryIso3166, synopsis
FROM TMovies;
```

El *wildcard* asterisco (`*`)

Si queremos consultar **todos** los campos, podemos hacerlo con:

```
SELECT *
FROM TMovies;
```

¡Ojo! Índices

Recuerda que a la invocación `.getXXX` (por ejemplo: `.getString`, `.getInt`, `.getDouble` ...) debemos pasarle el **nombre** de la columna, o bien, su **índice**.

Si pasamos el **nombre** no hay mucho margen de error.

Pero si pasamos el **índice** tengamos en cuenta:

- Se pasa el **índice relativo** de la columna **en** la consulta.

Por ejemplo

Como puedes ver arriba, `year` es la **tercera** columna de la tabla.

Pero si ejecutamos la consulta `SELECT title, year`, corresponderá a la **segunda** columna *dentro* del resultado:

```
Statement statement = conn.createStatement();
ResultSet result = statement.executeQuery("SELECT title, year FROM TMovies");
while(result.next()) {
    System.out.println("La película es del año:");
    System.out.println(result.getInt(2)); // Funciona
    System.out.println(result.getInt(3)); // Falla. Hay que especificar el índice de la columna dentro del resultado
}
```

Un nuevo método

Añadamos un nuevo método en nuestra clase `MoviesDataProvider`:

- Público
- Devolverá una lista de *Strings* (`ArrayList<String>`)
- Se llamará `getTwoColumns`
- **No** recibirá parámetros:

Así:

```
public ArrayList<String> getTwoColumns() {
}
```

Algo muy típico: Acumular elementos a devolver

En primer lugar, declarará e inicializará una variable `ArrayList` vacía.

Al final, la devolverá:

```
public ArrayList<String> getTwoColumns() {
    ArrayList<String> finalResult = new ArrayList<>();
    // ...
    return finalResult;
}
```

¡Ahora sólo falta rellenarla con los resultados deseados!

Abrimos y cerramos una conexión...

...como ya hemos visto:

```
public ArrayList<String> getTwoColumns() {
    ArrayList<String> finalResult = new ArrayList<>();
    String connectionStr = "jdbc:sqlite:db/sqlite3/movies.db";
    try {
        Connection conn = DriverManager.getConnection(connectionStr);

        // ...
    }
}
```

```
        conn.close();
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
    return finalResult;
}
```

Lanzamos la consulta y la procesamos

Y lanzaremos una consulta que recupera **dos** columnas de todas las filas de la tabla, como se muestra a continuación.

Para *cada* fila, *fabricamos* un nuevo **String** que concatena los **dos** valores y lo añadimos al resultado final.

Este código debería resultarte bastante comprensible:

```
public ArrayList<String> getTwoColumns() {
    ArrayList<String> finalResult = new ArrayList<>();
    String connectionStr = "jdbc:sqlite:db/sqlite3/movies.db";
    try {
        Connection conn = DriverManager.getConnection(connectionStr);
        // Crear y ejecutar la consulta
        Statement statement = conn.createStatement();
        ResultSet result = statement.executeQuery("SELECT genre, title FROM TMovies");
        while(result.next()) {
            String firstColumnValue = result.getString(1);
            String secondColumnValue = result.getString(2);
            String concatenated = firstColumnValue + ", " + secondColumnValue;
            finalResult.add(concatenated);
        }
        conn.close();
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
    return finalResult;
}
```

La tarea

Se pide que añadas el método arriba expuesto a tu clase **MoviesDataProvider**.

Si quieres probarlo desde **Main.java**, algo como esto será suficiente:

```
MoviesDataProvider provider = new MoviesDataProvider();
for (String lineOfMyArray : provider.getTwoColumns()) {
    System.out.println(lineOfMyArray);
}
```

¿Tienen sentido los resultados que se ven por pantalla?

Por último

Verifica que el **test** funciona correctamente.

Haz **commit** y **push** para subir los cambios al repositorio.



Rubén Montero @ruben.montero changed milestone to [%Sprint 2](#) 2 days ago