Anotación @Query

Ejemplo: Método para buscar una película por título:

```
public interface PeliculasRepository extends JpaRepository<Pelicula, Integer> {

   @Query("select p from Pelicula p where p.titulo = ?1")
   Pelicula anyNameMethod(String titulo);

}
```

Sentencia SQL que se generaría internamente…

```
select p.* from Peliculas p where p.titulo = ?
```

Notas importantes:

- ✓ El método solo regresa un objeto de tipo Pelicula. Si la consulta devuelve más de una entidad, obtendremos la excepción:

  javax.persistence.NonUniqueResultException: query did not return a unique result: 2

- ✓ Cuando usamos la anotación @Query el nombre del método en nuestra interfaz (Repository) puede ser cualquiera.
- ✓ La consulta es construida utilizando nombres de entidades (clase Java y sus atributos). No se utiliza los nombres reales de las tablas y campos como están en la base de datos. En este ejemplo se puede ver como es `select p from Pelicula` y no `select p from Peliculas`
- ✓ Los parámetros se especifican con placeholders (?1). El orden es importante.

Anotación @Query

Ejemplo: Método para buscar películas por estatus y genero:

```
public interface PeliculasRepository extends JpaRepository<Pelicula, Integer> {

   @Query("select p from Pelicula p where p.estatus = ?1 and p.genero = ?2")
   List<Pelicula> obtenerPorEstatusGenero(String estatus, String genero);

}
```

Sentencia SQL que se generaría internamente…

```
select p.* from Peliculas p where p.estatus = ? and p.genero = ?
```

Notas importantes:

- ✓ La consulta puede encontrar varias películas, por lo tanto el resultado es de tipo List<Pelicula>.
- ✓ Ejemplo con 2 parámetros (?1 y ?2). El orden es importante.

Anotación @Query

Ejemplo: Método para buscar **SOLO LOS TITULOS** de películas por genero:

```
public interface PeliculasRepository extends JpaRepository<Pelicula, Integer> {

   @Query("select p.titulo from Pelicula p where p.genero = ?1")
   List<String> obtenerTitulosPorGenero(String genero);

}
```

Sentencia SQL que se generaría internamente…

```
select p.titulo from Peliculas p where p.genero = ?
```

Notas importantes:

- ✓ En este ejemplo solo nos interesa el título de las películas (no el objeto de tipo Pelicula completo). Por eso retornamos List<String>.
- ✓ La consulta puede encontrar varias películas, por lo tanto el resultado es de tipo List<String>.

Anotación @Query

Ejemplo: Buscar películas que contengan la cadena de texto pasada como parámetro en el título (LIKE):

```
public interface PeliculasRepository extends JpaRepository<Pelicula, Integer> {

   @Query("select p from Pelicula p where p.titulo like %?1%")
   List<Pelicula> buscarPorTitulo(String titulo);

}
```

Sentencia SQL que se generaría internamente…

```
select p.* from Peliculas p where p.titulo like %?%
```

Notas importantes:

- ✓ Ejemplo de uso de la expresión **LIKE** de SQL.
- ✓ La consulta puede encontrar varias películas, por lo tanto el resultado es de tipo List<Pelicula>.

Anotación @Query

Ejemplo: Buscar películas por rango de duración (BETWEEN):

```
public interface PeliculasRepository extends JpaRepository<Pelicula, Integer> {

   @Query("select p from Pelicula p where p.duracion between ?1 and ?2")
   List<Pelicula> buscarPorRangoDuracion(int duration1, int duration2);

}
```

Sentencia SQL que se generaría internamente…

```
select p.* from Peliculas p where p.duracion between ? and ?
```

Notas importantes:

- ✓ Ejemplo de uso de la expresión **BETWEEN** de SQL.
- ✓ La consulta puede encontrar varias películas, por lo tanto el resultado es de tipo List<Pelicula>.

Anotación @Query

Ejemplo: Obtener la duración total en minutos de todas las películas por género:

```
public interface PeliculasRepository extends JpaRepository<Pelicula, Integer> {

   @Query("select sum(p.duracion) from Pelicula p where p.genero = :genero")
   int duracionTotalPorGenero(@Param("genero") String genero);

}
```

Sentencia SQL que se generaría internamente…

```
select sum(p.duracion) from Peliculas p where p.genero = ?
```

Notas importantes:

- ✓ Ejemplo de uso de la expresión **SUM** de SQL.
- ✓ El resultado es númerico (sumatoria). Por lo tanto el método regresa un tipo int.
- ✓ Anotación **@Param**, otra forma de indicar placeholders para los parámetros de la consulta (Named Parameter).

Anotación @Query

Ejemplo: Obtener la película con la mínima duración en minutos:

```
public interface PeliculasRepository extends JpaRepository<Pelicula, Integer> {

   @Query("select min(p.duracion) from Pelicula p where p.estatus = 'Activa'")
   int peliculaMinDuracion();
}
```

Sentencia SQL que se generaría internamente…

```
select min(p.duracion) from Peliculas p where p.estatus='Activa'
```

Notas importantes:

- ✓ Ejemplo de uso de la expresión **MIN** de SQL.
- ✓ El resultado es numérico (valor mínimo). Por lo tanto el método regresa un tipo int.

Anotación @Query

Ejemplo: Obtener la película con la máxima duración en minutos:

```
public interface PeliculasRepository extends JpaRepository<Pelicula, Integer> {

   @Query("select max(p.duracion) from Pelicula p where p.estatus = 'Activa'")
   int peliculaMaxDuracion();
}
```

Sentencia SQL que se generaría internamente…

```
select max(p.duracion) from Peliculas p where p.estatus='Activa'
```

Notas importantes:

- ✓ Ejemplo de uso de la expresión **MAX** de SQL.
- ✓ El resultado es numérico (valor máximo). Por lo tanto el método regresa un tipo int.

Anotación @Query

Ejemplo: Obtener la duración promedio en minutos:

```
public interface PeliculasRepository extends JpaRepository<Pelicula, Integer> {

   @Query("select avg(p.duracion) from Pelicula p where p.estatus = 'Activa'")
   double obtenerDuracionPromedio();
}
```

Sentencia SQL que se generaría internamente…

```
select avg(p.duracion) from Peliculas p where p.estatus='Activa'
```

Notas importantes:

- ✓ Ejemplo de uso de la expresión **AVG** de SQL.
- ✓ El resultado es numérico con decimales (valor promedio). Por lo tanto el método regresa un tipo double.

Anotación @Query – Consultas avanzadas

Ejemplo: Consultar en la tabla de Horarios (HorariosRepository), agrupando por idPelicula para buscar películas **ACTIVAS** que estarán disponibles (películas con horarios registrados) para una fecha en particular:

```
public interface HorariosRepository extends JpaRepository<Horario, Integer> {
   @Query("select h from Horario h where h.fecha = :fecha and pelicula.estatus='Activa' group by h.pelicula
           order by pelicula.id asc")
   public List<Horario> getByFecha(@Param("fecha") Date fecha)
}
```

Sentencia SQL que se generaría internamente…

```
select h.*
from Horarios h
inner join Peliculas p
where h.idPelicula = p.id
and h.fecha = ?
and p.estatus='Activa'
group by h.idPelicula
order by h.idPelicula asc;
```

Notas importantes:

- ✓ Ejemplo de uso de la expresión **GROUP BY** de SQL.
- ✓ Esté método es usado cuando filtramos POR FECHA en la página principal del proyecto final. En este caso buscamos en la tabla de horarios AGRUPANDO por idPelicula, pero solo películas ACTIVAS.
- ✓ Anotación **@Param**, otra forma de indicar placeholders para los parámetros de la consulta (Named Parameter).