

CURSO HIBERNATE Y JPA

CONSULTAS CON HQL Y JPQL



Por el experto: Ing. Ubaldo Acosta



Experiencia y Conocimiento para tu vida

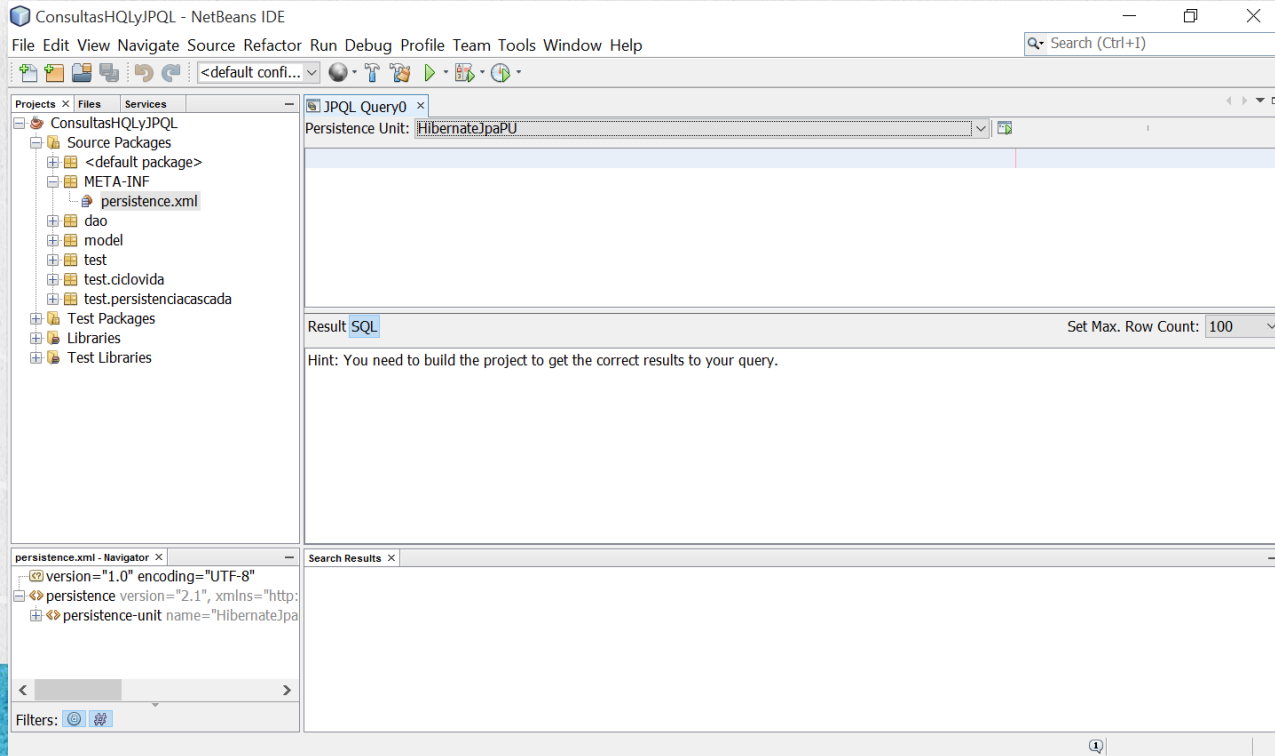


CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx

OBJETIVO DEL EJERCICIO

Crear consultas con HQL/JPQL. Al finalizar deberemos ver:



AGREGAR DATOS A LA BASE DE DATOS SGA_DB

Agregamos algunos registros a la base de datos utilizando la aplicación Web que creamos anteriormente o directamente en la base de datos de mysql de sga_db, los valores pueden variar, pero como ejemplo son:

Ubaldo

Listar Alumnos

← → ↻ 🏠

localhost:8080/PersistenciaAlumnosWeb/ServletControlador

☆

2

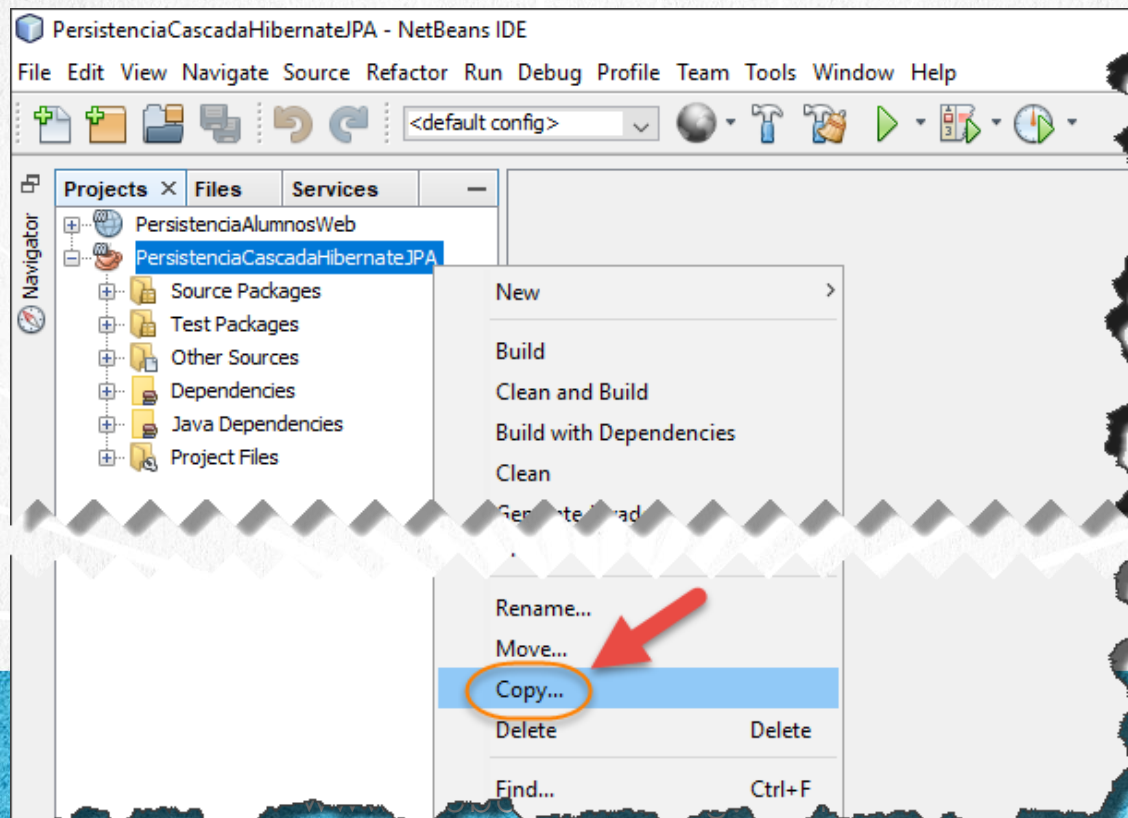
Listar Alumnos

[Agregar](#)

| Alumno ID | Nombre | Apellido Paterno | Apellido Materno | Domicilio | Teléfono | Email |
|-------------------|---------|------------------|------------------|-----------|----------|-------------------|
| 1 | Alfonso | Ugalde | Martinez | Londres | 55551111 | alfonso@mail.com |
| 2 | Martha | Martinez | Garcia | Allende | 55717189 | contacto@mail.com |
| 7 | Carlos | Mendez | Velez | Jupiter | 44441111 | carlos@mail.com |
| 8 | Maria | Perez | Juarez | Darwin | 33332222 | maria@mail.com |

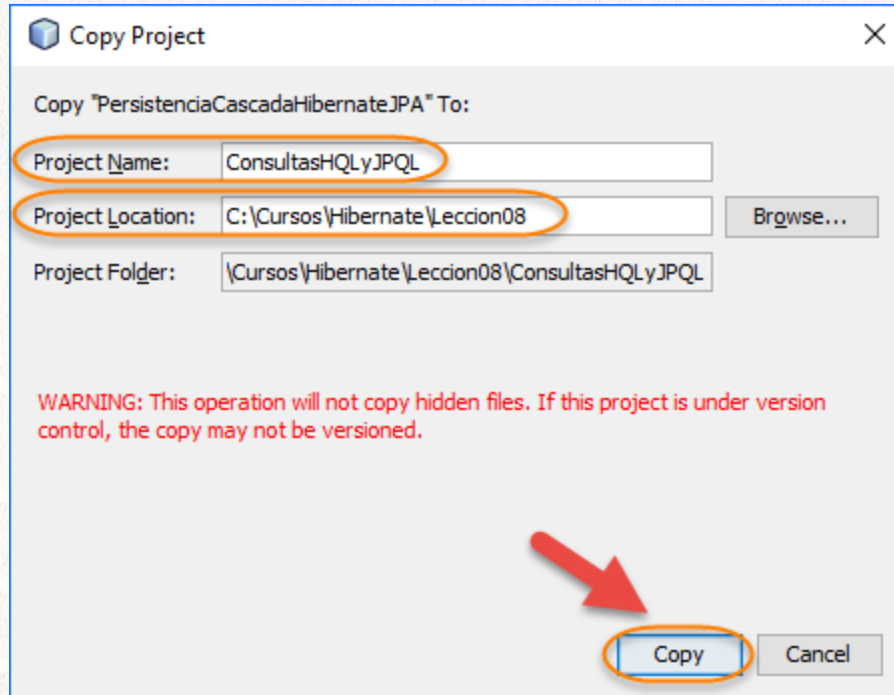
PASO 1. COPIAMOS EL PROYECTO

Copiamos el proyecto PersistenciaCascadaHibernateJPA:



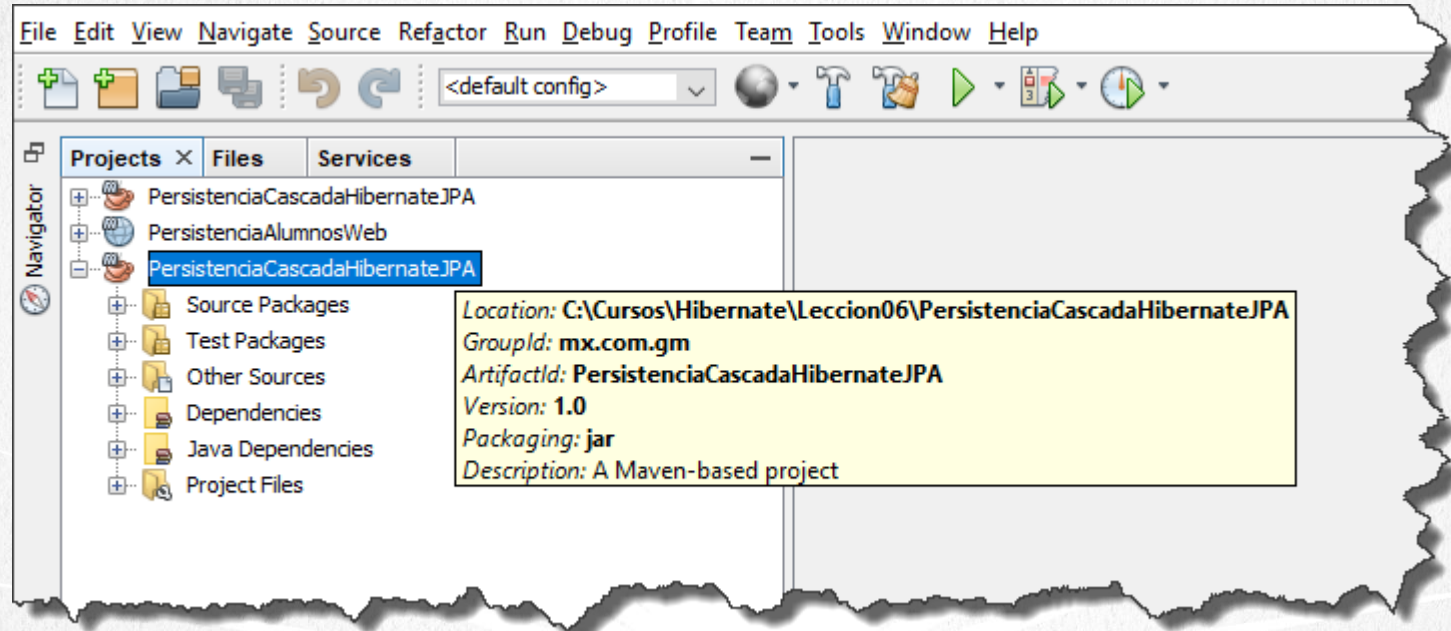
PASO 1. COPIAMOS EL PROYECTO

Creamos el nuevo proyecto ConsultasHQLyJPQL:



PASO 2. CERRAMOS EL PROYECTO YA NO USADO

Cerramos el proyecto que ya no utilizamos:

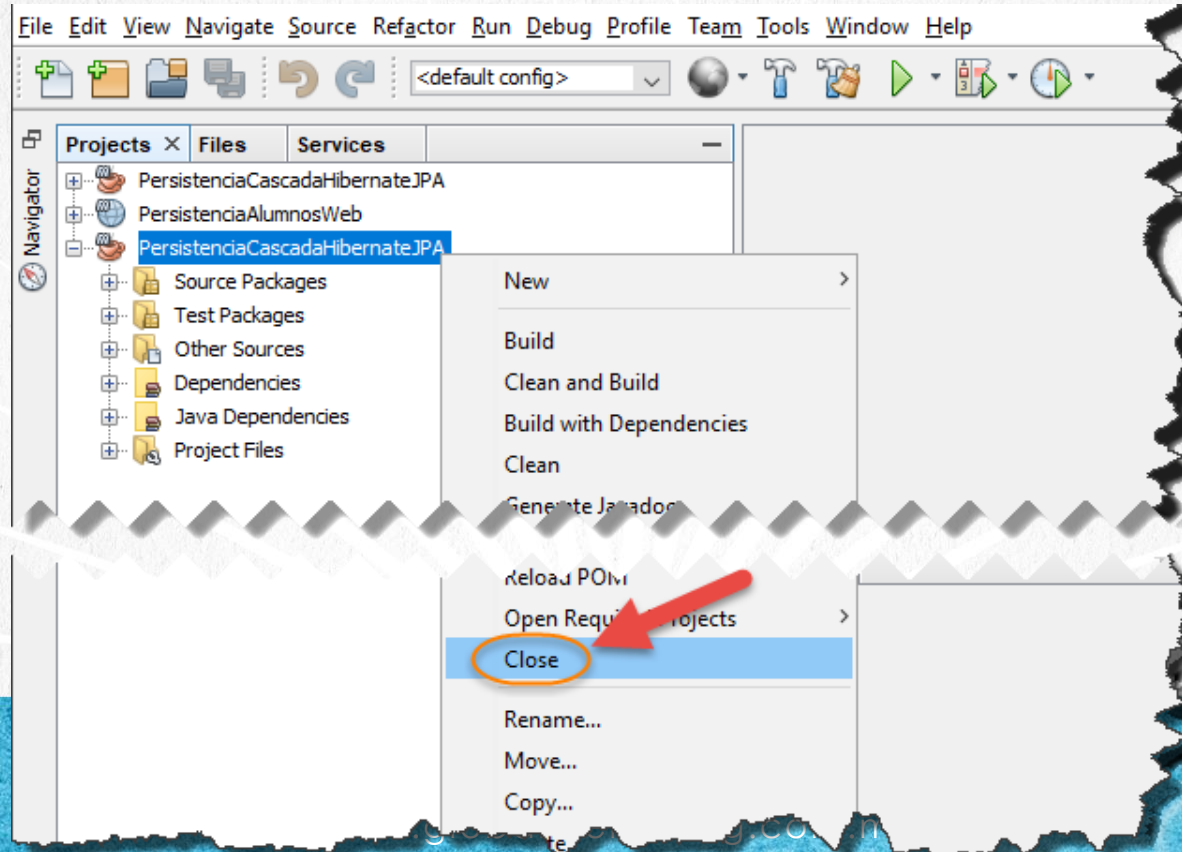


CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx

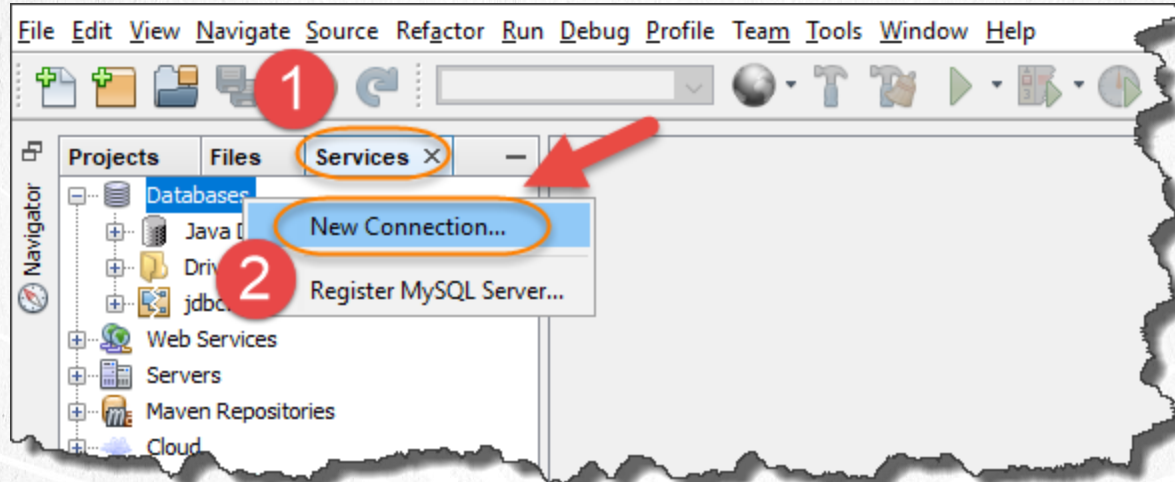
PASO 2. CERRAMOS EL PROYECTO YA NO USADO

Cerramos el proyecto que ya no utilizamos:



PASO 3. CREAMOS UNA CONEXIÓN DE MYSQL

Creamos una conexión a Mysql en la sección de Servicios:



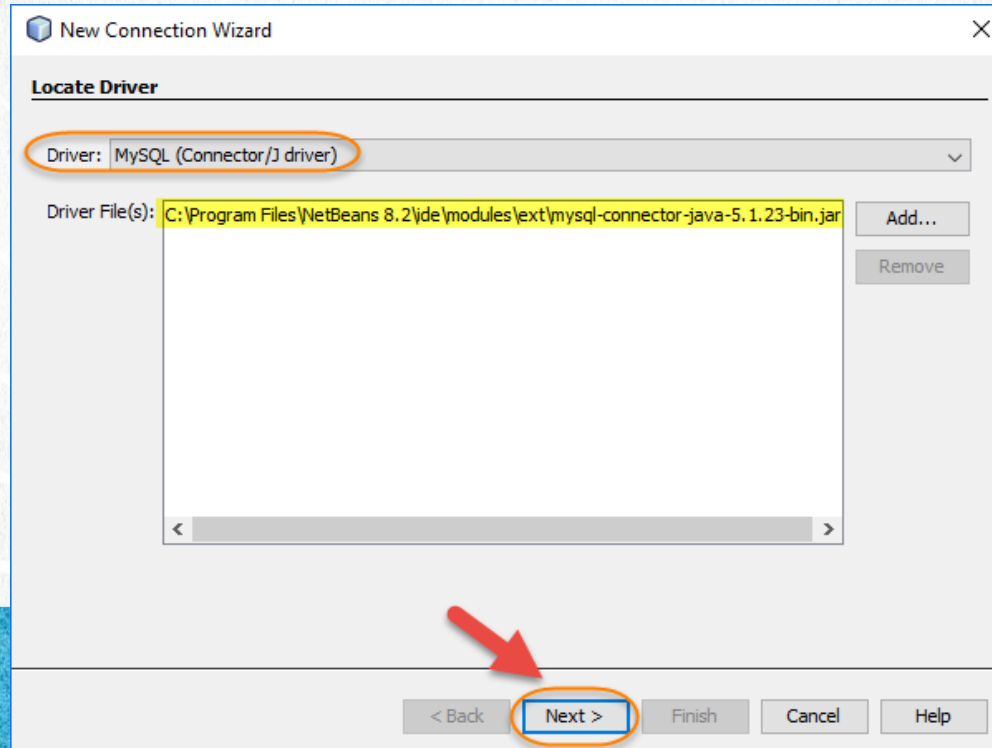
CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx

PASO 3. CREAMOS UNA CONEXIÓN DE MYSQL

Si no reconoce el .jar de mysql automáticamente, lo agregamos de las librerías del curso:

<http://icursos.net/cursos/Hibernate/libs/hibernate-libs-1.zip>



PASO 3. CREAMOS UNA CONEXIÓN DE MYSQL

Agregamos los valores solicitados por la conexión a la base de datos sga_db de mysql:

New Connection Wizard

Customize Connection

Driver Name: MySQL (Connector/J driver)

Host: localhost Port: 3306

Database: sga_db

User Name: root

Password: **password: admin**

☒ Remember password

Connection Properties Test Connection

JDBC URL: jdbc:mysql://localhost:3306/sga_db?useSSL=true **Agregamos el parámetro ?useSSL=true**

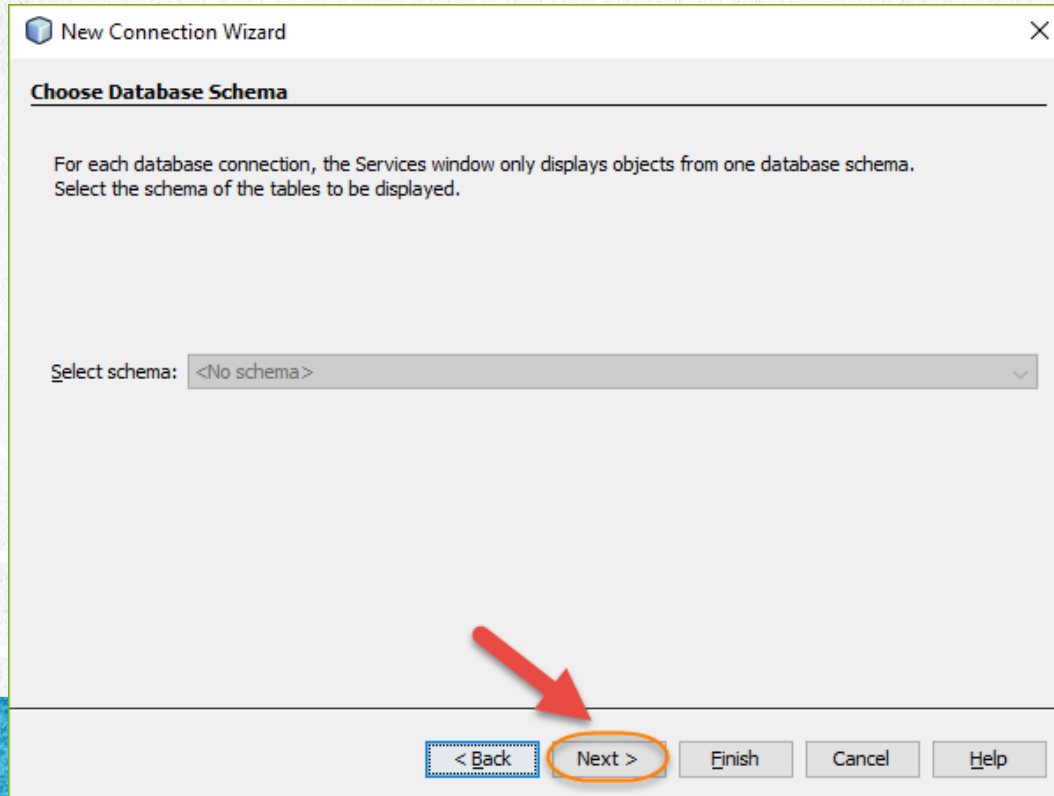
Connection Succeeded.

< Back **Next >** Finish Cancel Help

Todos los valores deben coincidir con los valores contenidos en el archivo persistence.xml

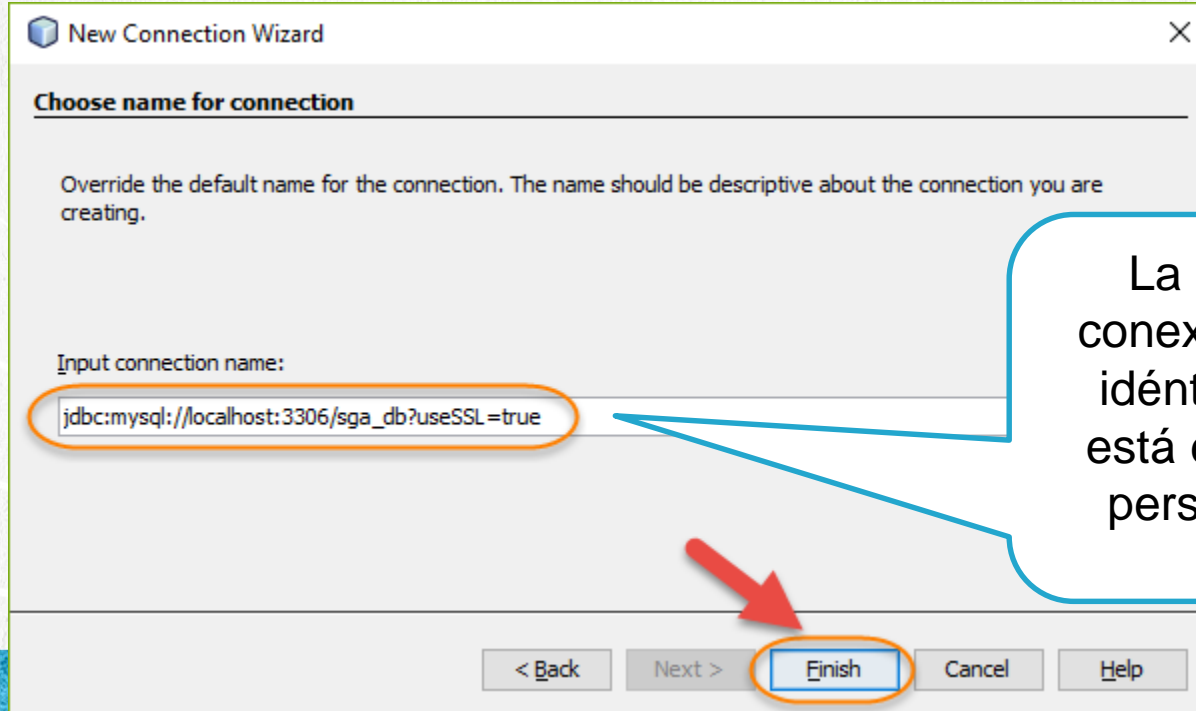
PASO 3. CREAMOS UNA CONEXIÓN DE MYSQL

No seleccionamos nada, solo damos click en Next:



PASO 3. CREAMOS UNA CONEXIÓN DE MYSQL

No seleccionamos nada, solo damos click en Next:



Choose name for connection

Override the default name for the connection. The name should be descriptive about the connection you are creating.

Input connection name:

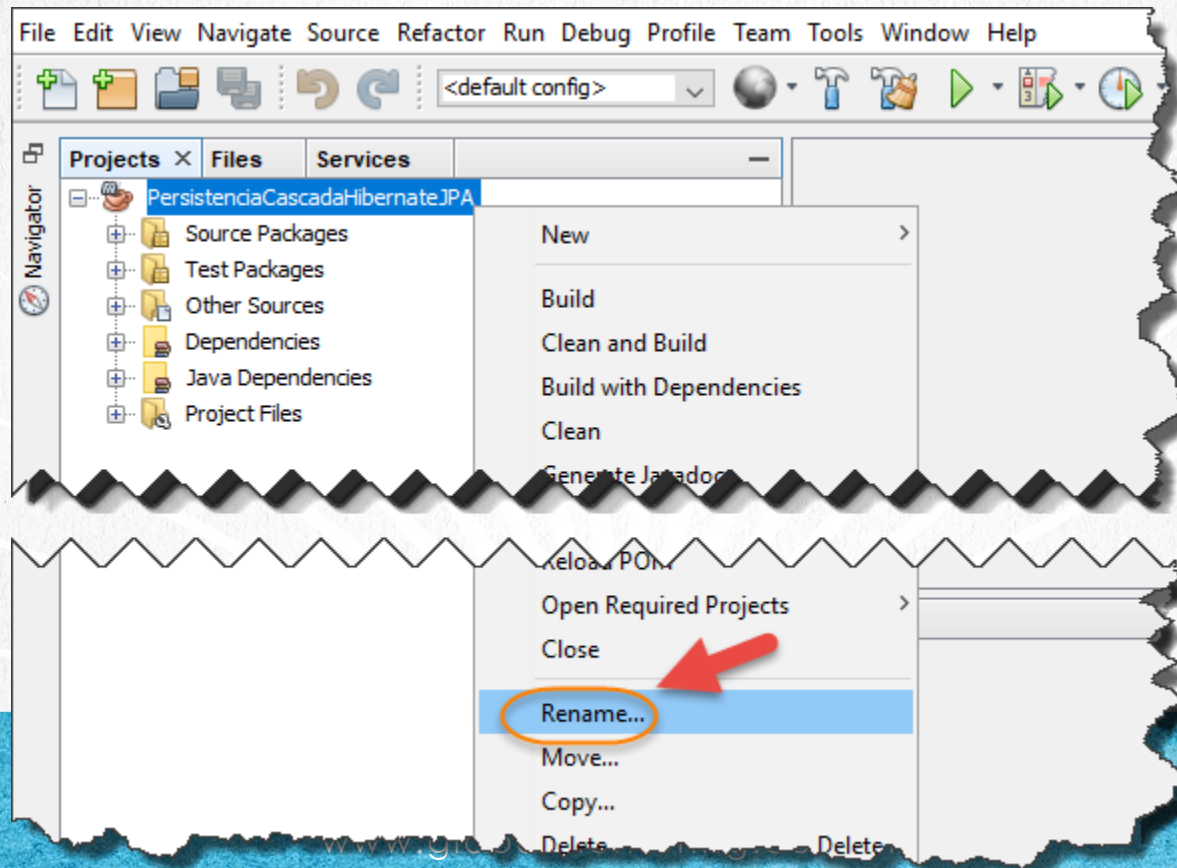
`jdbc:mysql://localhost:3306/sga_db?useSSL=true`

< Back Next > **Finish** Cancel Help

La cadena de conexión debe ser idéntica a la que está en el archivo persistence.xml

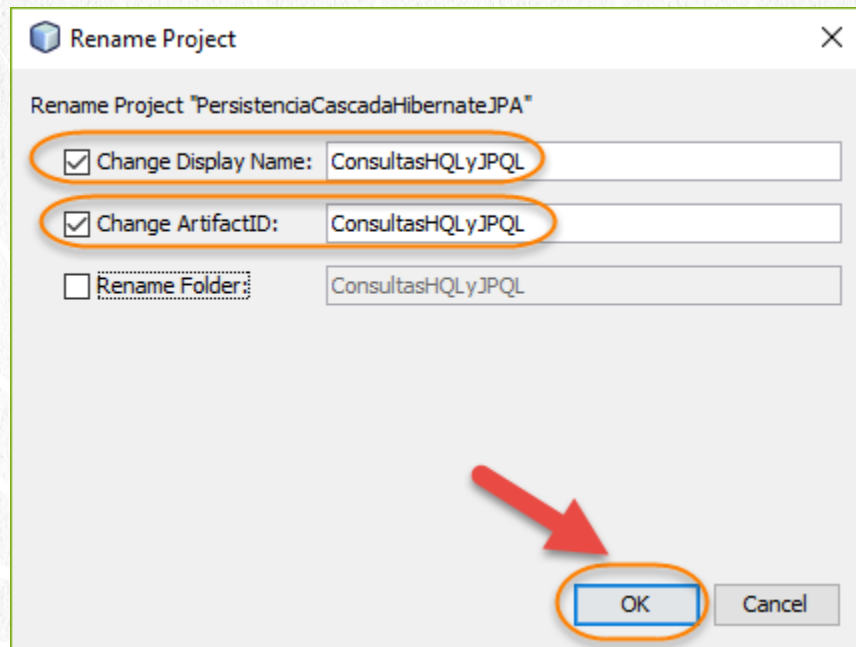
PASO 4. RENOMBRAMOS EL PROYECTO

Renombramos el proyecto:



PASO 4. RENOMBRAMOS EL PROYECTO

Renombramos el proyecto:

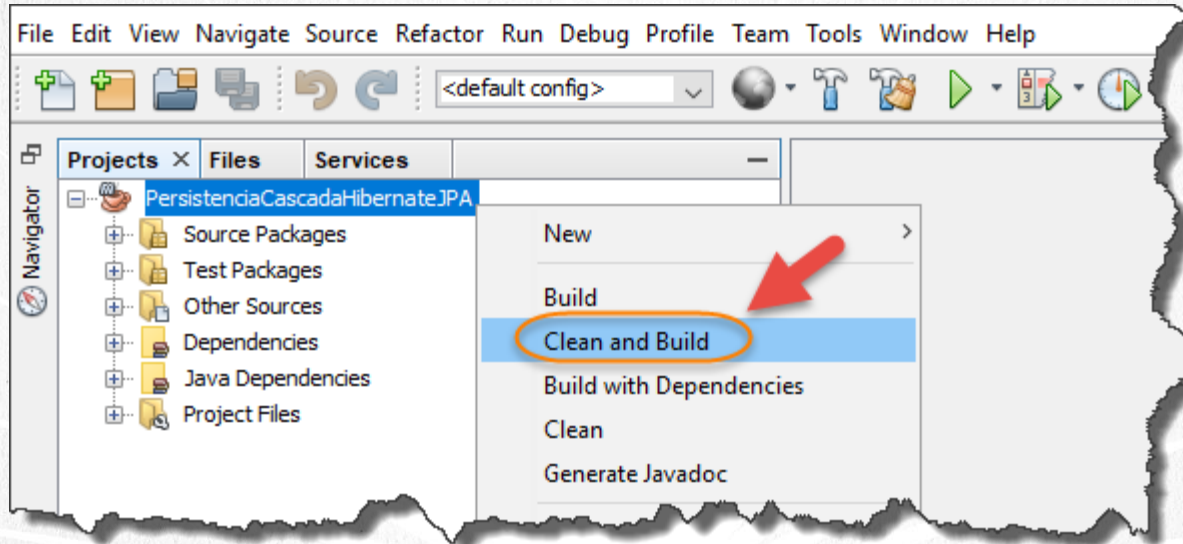


CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx

PASO 5. HACEMOS UN BUILD DEL PROYECTO

Hacemos un clean & build del proyecto, y lo tenemos que hacer cada que abramos una consola de JPQL para asegurar que tenemos la última versión en el código Java y JPA:

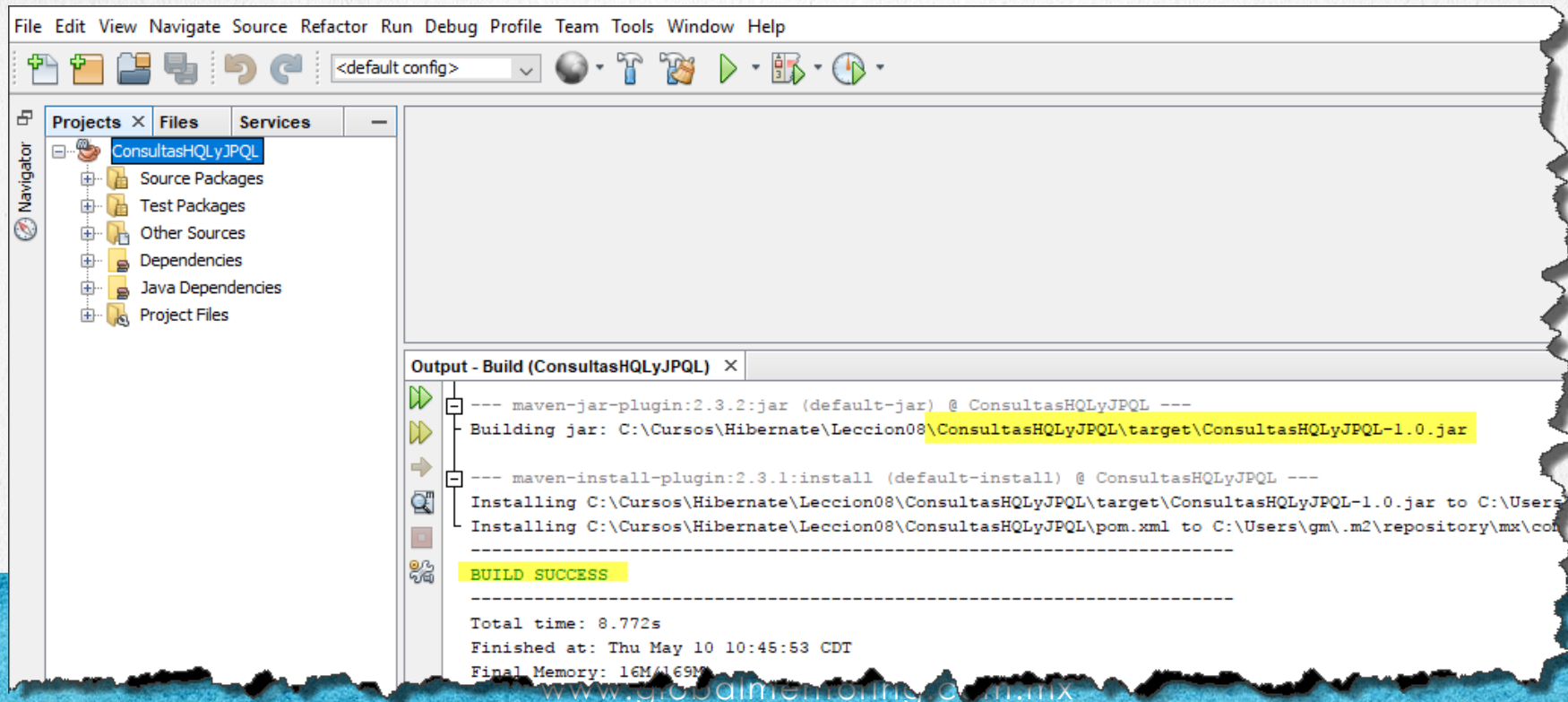


CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx

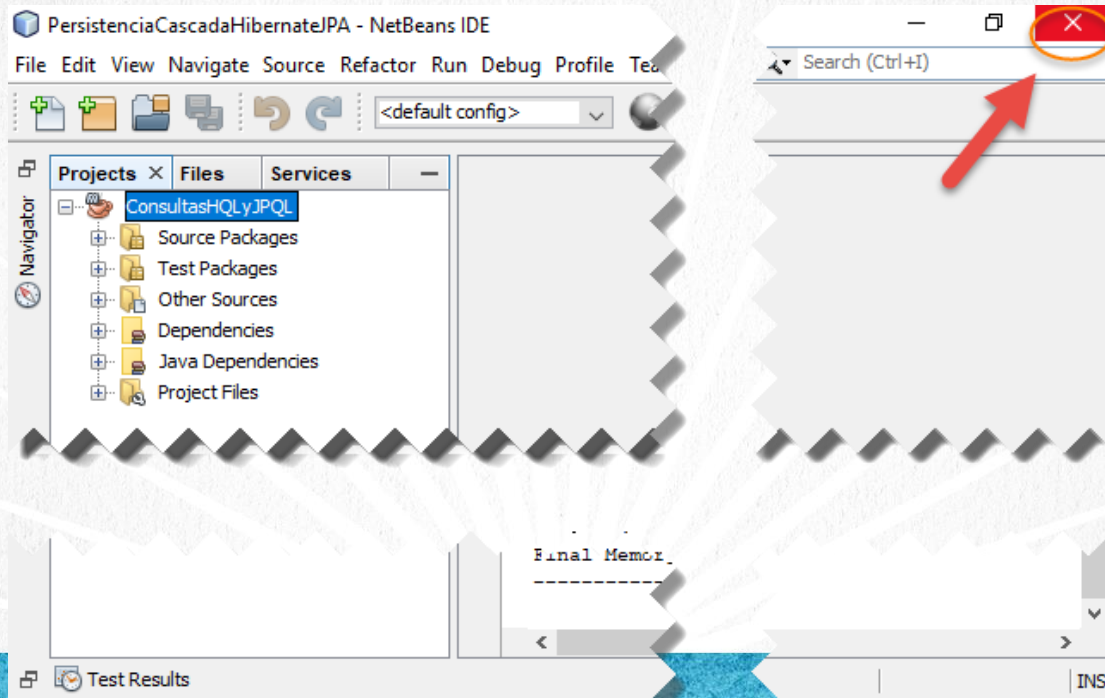
PASO 5. HACEMOS UN BUILD DEL PROYECTO

Hacemos un clean & build del proyecto, y lo tenemos que hacer cada que abramos una consola de JPQL para asegurar que tenemos la última versión en el código Java y JPA:



PASO 6. REINICIAMOS EL IDE

Cerramos y volvemos a abrir Netbeans para que reconozca el archivo persistence.xml como un archivo válido para ejecutar consultas JPQL con los nuevos cambios:

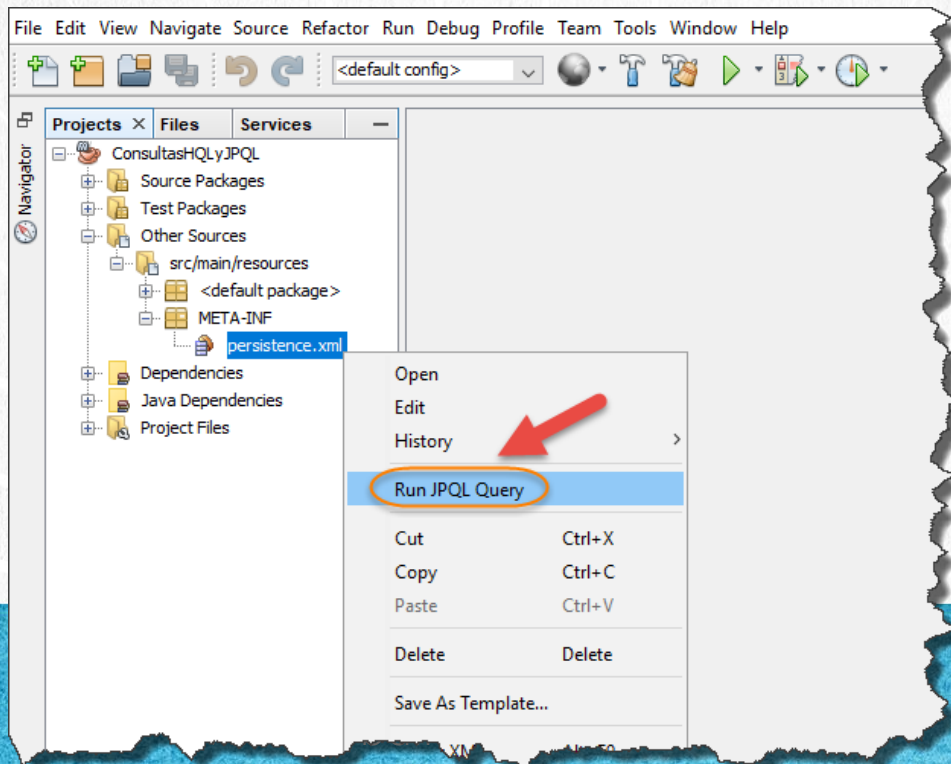


CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx

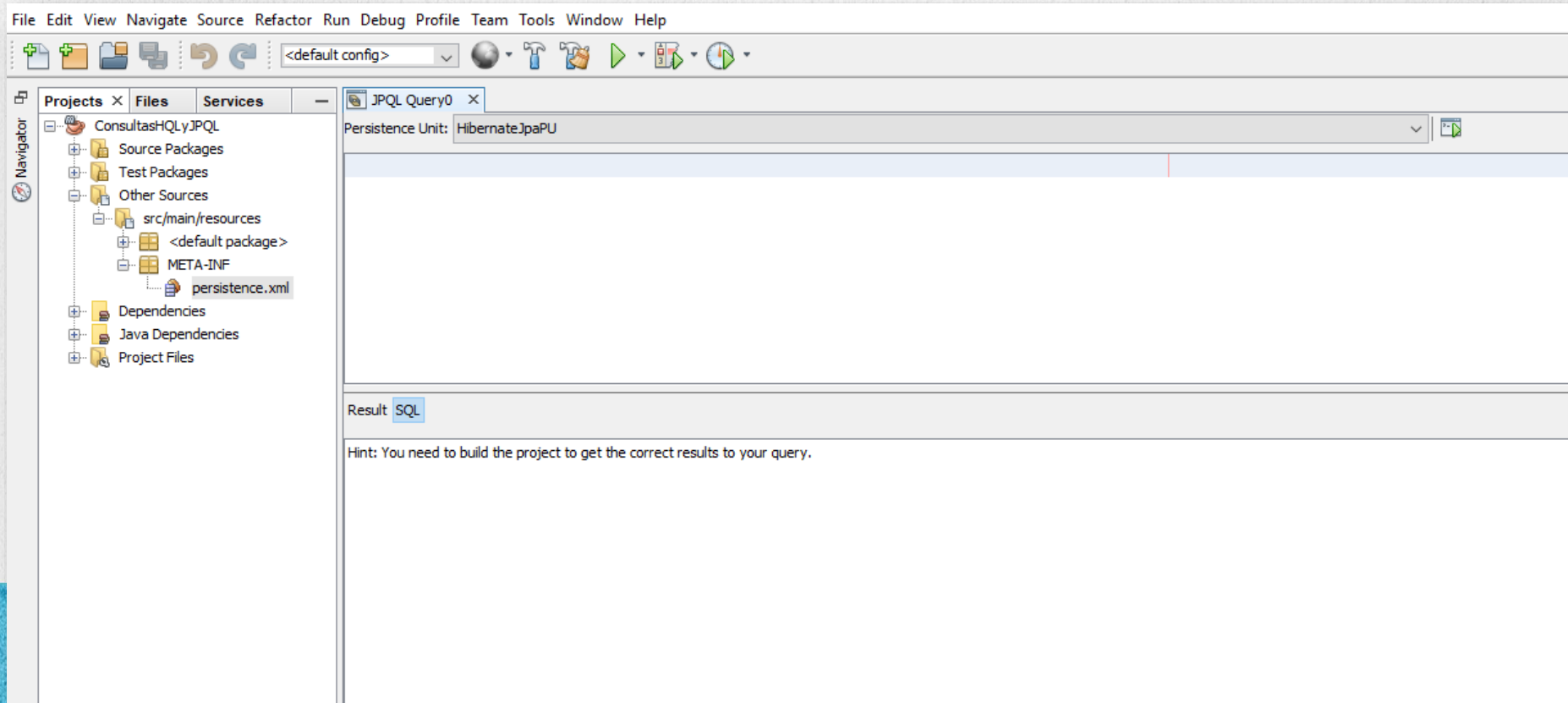
PASO 7. ABRIMOS LA CONSOLA DE JPQL

Una vez abierto nuevamente el IDE, abrimos la consola para ejecutar consultas JPQL. Si por alguna razón falla, hay que revisar que la cadena de conexión del archivo persistence.xml sea la misma que la creada en Netbeans, según los pasos indicados anteriormente:



PASO 7. ABRIMOS LA CONSOLA DE JPQL

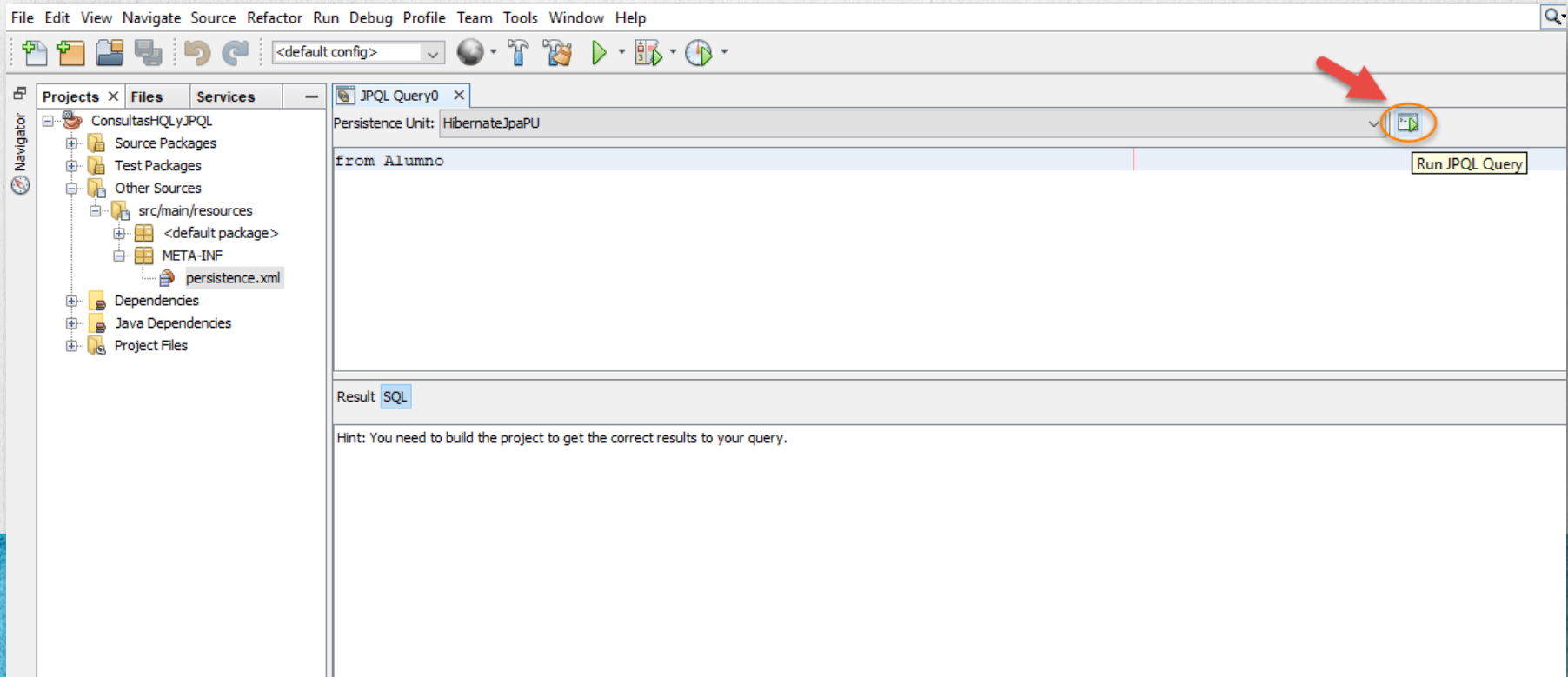
Abrimos la consola para ejecutar consultas JPQL:



PASO 7. ESCRIBIMOS LA CONSULTA JPQL

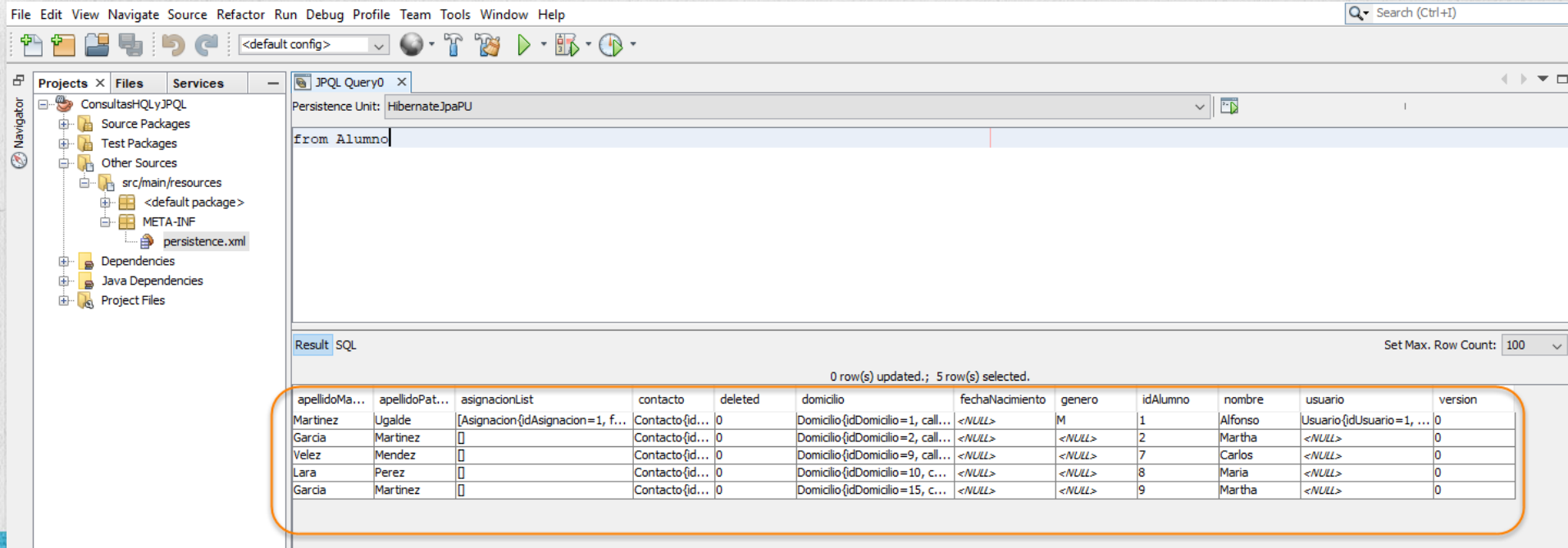
Escribimos la consulta: `from Alumno`

Y ejecutamos la consulta presionando el botón de Run JPQL Query según se muestra



PASO 8. EJECUTAMOS LA CONSOLA DE JPQL

Observamos el resultado de ejecutar la consulta. Los datos pueden variar según la información que se tenga en la base de datos.



The screenshot shows the Eclipse IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, and Help. Below the menu is a toolbar with icons for file operations and a search bar labeled 'Search (Ctrl+I)'. The left sidebar shows the 'Projects' view with a tree structure of the project 'ConsultashQLyJPQL', including source packages, test packages, other sources, and dependencies. The main editor area displays the 'JPQL Query0' editor with the text 'from Alumno'. Below the editor, the 'Result SQL' tab is active, showing the execution results. The status bar indicates '0 row(s) updated.; 5 row(s) selected.' The results are displayed in a table with 12 columns: apellidoMa..., apellidoPat..., asignacionList, contacto, deleted, domicilio, fechaNacimiento, genero, idAlumno, nombre, usuario, and version. The table contains 5 rows of data.

| apellidoMa... | apellidoPat... | asignacionList | contacto | deleted | domicilio | fechaNacimiento | genero | idAlumno | nombre | usuario | version |
|---------------|----------------|----------------------------------|----------------|---------|----------------------------------|-----------------|--------|----------|---------|--------------------------|---------|
| Martinez | Ugalde | [Asignacion{idAsignacion=1, f... | Contacto{id... | 0 | Domicilio{idDomicilio=1, call... | <NULL> | M | 1 | Alfonso | Usuario{idUsuario=1, ... | 0 |
| Garcia | Martinez | [] | Contacto{id... | 0 | Domicilio{idDomicilio=2, call... | <NULL> | <NULL> | 2 | Martha | <NULL> | 0 |
| Velez | Mendez | [] | Contacto{id... | 0 | Domicilio{idDomicilio=9, call... | <NULL> | <NULL> | 7 | Carlos | <NULL> | 0 |
| Lara | Perez | [] | Contacto{id... | 0 | Domicilio{idDomicilio=10, c... | <NULL> | <NULL> | 8 | Maria | <NULL> | 0 |
| Garcia | Martinez | [] | Contacto{id... | 0 | Domicilio{idDomicilio=15, c... | <NULL> | <NULL> | 9 | Martha | <NULL> | 0 |

PASO 9. EJECUTAMOS LA CONSULTA JPQL

Basado en los pasos anteriores, ejecutamos las siguientes consultas. Es posible que deban modificar los valores según existan en su base de datos, así que los valores pueden variar:

```
from Alumno a where a.idAlumno = 1
```

JPQL Query0 x

Persistence Unit: Hibernate.JpaPU

from Alumno a where a.idAlumno = 1

Result SQL

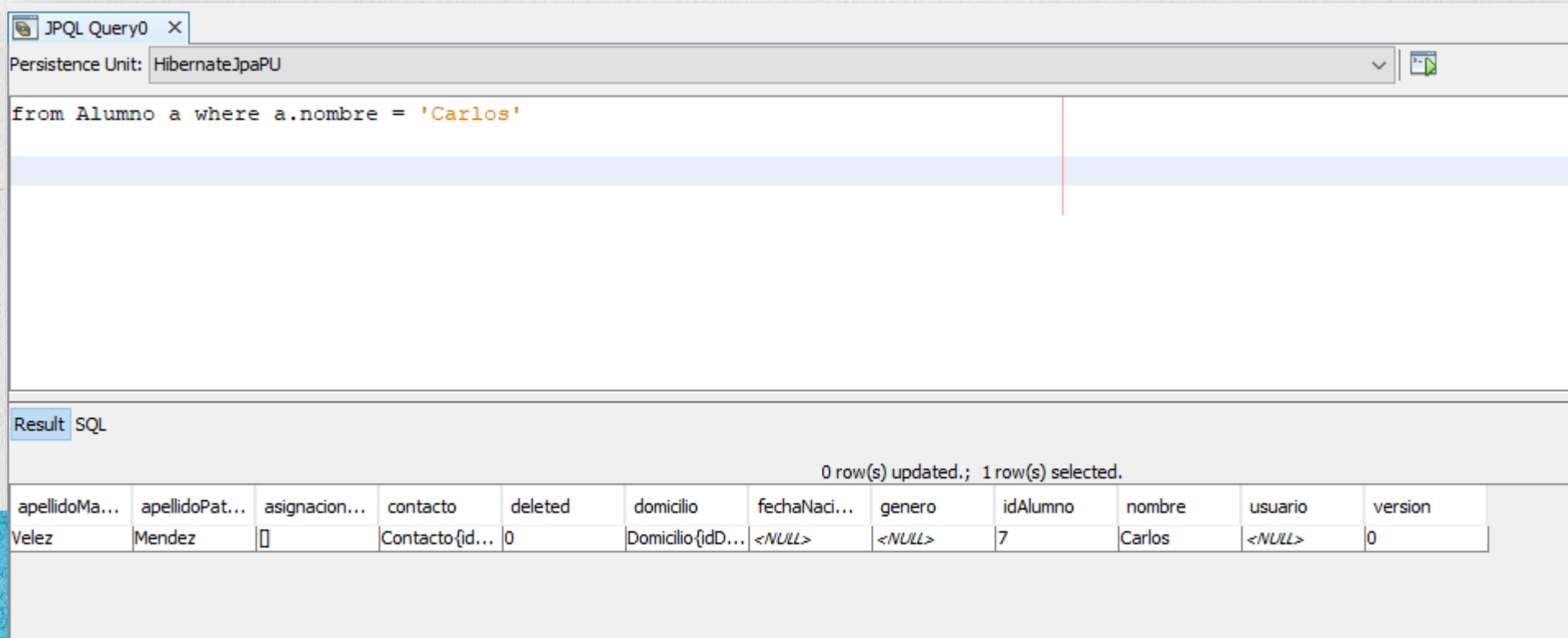
0 row(s) updated.; 1 row(s) selected.

| apellidoMa... | apellidoPat... | asignacion... | contacto | deleted | domicilio | fechaNaci... | genero | idAlumno | nombre | usuario | version |
|---------------|----------------|-----------------|----------------|---------|------------------|--------------|--------|----------|---------|----------------|---------|
| Martinez | Ugalde | [Asignacion{... | Contacto{id... | 0 | Domicilio{idD... | <NULL> | M | 1 | Alfonso | Usuario{idU... | 0 |

PASO 10. EJECUTAMOS LA CONSULTA JPQL

Ejecutamos la siguiente consulta:

```
from Alumno a where a.nombre = 'Carlos'
```



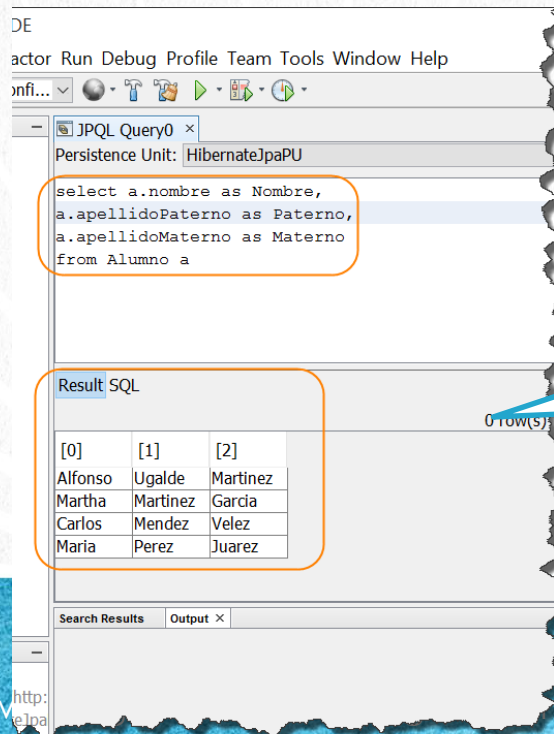
The screenshot shows a software interface for executing JPQL queries. At the top, there's a title bar 'JPQL Query0'. Below it, a 'Persistence Unit' dropdown is set to 'HibernateJpaPU'. The main text area contains the query: `from Alumno a where a.nombre = 'Carlos'`. Below the query area, there's a 'Result' tab and an 'SQL' tab. The 'Result' tab is active, displaying the message '0 row(s) updated.; 1 row(s) selected.' above a table of results. The table has 12 columns: 'apellidoMa...', 'apellidoPat...', 'asignacion...', 'contacto', 'deleted', 'domicilio', 'fechaNaci...', 'genero', 'idAlumno', 'nombre', 'usuario', and 'version'. The first row of data shows 'Velez' for 'apellidoMa...', 'Mendez' for 'apellidoPat...', an empty array for 'asignacion...', 'Contacto{id...' for 'contacto', '0' for 'deleted', 'Domicilio{idD...' for 'domicilio', '<NULL>' for 'fechaNaci...', '<NULL>' for 'genero', '7' for 'idAlumno', 'Carlos' for 'nombre', '<NULL>' for 'usuario', and '0' for 'version'.

| apellidoMa... | apellidoPat... | asignacion... | contacto | deleted | domicilio | fechaNaci... | genero | idAlumno | nombre | usuario | version |
|---------------|----------------|---------------|----------------|---------|------------------|--------------|--------|----------|--------|---------|---------|
| Velez | Mendez | [] | Contacto{id... | 0 | Domicilio{idD... | <NULL> | <NULL> | 7 | Carlos | <NULL> | 0 |

PASO 11. EJECUTAMOS LA CONSULTA JPQL

Ejecutamos la siguiente consulta:

```
select a.nombre as Nombre, a.apellidoPaterno as Paterno, a.apellidoMaterno as Materno from Alumno a
```



The screenshot shows an IDE window titled "JPQL Query0" with a persistence unit of "HibernateJpaPU". The query is: `select a.nombre as Nombre, a.apellidoPaterno as Paterno, a.apellidoMaterno as Materno from Alumno a`. Below the query, the "Result SQL" section shows the results in a table format. The table has three columns: [0], [1], and [2]. The rows are: Alfonso Ugalde Martinez, Martha Martinez Garcia, Carlos Mendez Velez, and Maria Perez Juarez. The "Search Results" and "Output" tabs are visible at the bottom.

| [0] | [1] | [2] |
|---------|----------|----------|
| Alfonso | Ugalde | Martinez |
| Martha | Martinez | Garcia |
| Carlos | Mendez | Velez |
| Maria | Perez | Juarez |

Se crea un arreglo de tipo Object de 3 columnas

PASO 12. EJECUTAMOS LA CONSULTA JPQL

Ejecutamos la siguiente consulta:

```
select a, a.idAlumno from Alumno a
```

The screenshot shows an IDE window with a JPQL query editor and a results pane. The query is `select a, a.idAlumno from Alumno a`. The results pane shows 4 rows selected. A speech bubble points to the query, stating: "Se crea un arreglo de tipo Object con 2 columnas". A bracket below the results table spans the first two columns, labeled "1er Col" and "2da Col".

JPQL Query0 x
Persistence Unit: HibernateJpaPU
`select a, a.idAlumno from Alumno a`

Se crea un arreglo de tipo Object con 2 columnas

1er Col 2da Col

Result SQL Set Max. Row Count: 10

0 row(s) updated.; 4 row(s) selected.

| [0].ap... | [0].ap... | [0].asi... | [0].co... | [0].del... | [0].do... | [0].fec... | [0].ge... | [0].idA... | [0].no... | [0].us... | [0].ver... | [1] |
|-----------|-----------|------------|------------|------------|-------------|------------|-----------|------------|-----------|------------|------------|-----|
| Martinez | Ugalde | [Asigna... | Contact... | 0 | Domicili... | <NULL> | M | 1 | Alfonso | Usuario... | 0 | 1 |
| Garcia | Martinez | [] | Contact... | 0 | Domicili... | <NULL> | <NULL> | 2 | Martha | <NULL> | 0 | 2 |
| Velez | Mendez | [] | Contact... | 0 | Domicili... | <NULL> | <NULL> | 7 | Carlos | <NULL> | 0 | 7 |
| Juarez | Perez | [] | Contact... | 0 | Domicili... | <NULL> | <NULL> | 8 | Maria | <NULL> | 0 | 8 |

Search Results Output x

PASO 13. EJECUTAMOS LA CONSULTA JPQL

Ejecutamos la siguiente consulta:

```
select new Alumno(a.idAlumno) from Alumno a
```

The screenshot shows an IDE window titled "JPQL Query0" with the persistence unit "HibernateJpaPU". The query `select new Alumno(a.idAlumno) from Alumno a` is entered in the editor and highlighted with an orange box. A blue callout bubble points to this query with the text "Utilizamos el constructor del idAlumno". Below the editor, the "Result" tab is active, displaying the SQL statement and a table of results. The table has 12 columns: apellidoMaterno, apellidoPaterno, asigna..., contacto, deleted, domicilio, fechaN..., genero, idAlumno, nombre, usuario, and version. The "idAlumno" column is highlighted with an orange box. The results show 4 rows selected, with idAlumno values 2, 7, 8, and 1.

Factor Run Debug Profile Team Tools Window Help

onfi... [Icons]

JPQL Query0 x

Persistence Unit: HibernateJpaPU

`select new Alumno(a.idAlumno) from Alumno a`

Utilizamos el constructor del idAlumno

Result SQL Set Max. Row Count: 100

0 row(s) updated.; 4 row(s) selected.

| apellidoMaterno | apellidoPaterno | asigna... | contacto | deleted | domicilio | fechaN... | genero | idAlumno | nombre | usuario | version |
|-----------------|-----------------|-----------|----------|---------|-----------|-----------|--------|----------|--------|---------|---------|
| <NULL> | <NULL> | <NULL> | <NULL> | 0 | <NULL> | <NULL> | <NULL> | 2 | <NULL> | <NULL> | 0 |
| <NULL> | <NULL> | <NULL> | <NULL> | 0 | <NULL> | <NULL> | <NULL> | 7 | <NULL> | <NULL> | 0 |
| <NULL> | <NULL> | <NULL> | <NULL> | 0 | <NULL> | <NULL> | <NULL> | 8 | <NULL> | <NULL> | 0 |
| <NULL> | <NULL> | <NULL> | <NULL> | 0 | <NULL> | <NULL> | <NULL> | 1 | <NULL> | <NULL> | 0 |

PASO 14. EJECUTAMOS LA CONSULTA JPQL

Ejecutamos la siguiente consulta:

```
select min(a.idAlumno) as MinId, max(a.idAlumno) as MaxId,  
count(a.idAlumno) as Count from Alumno a
```

The screenshot shows an IDE window titled "JPQL Query0" with a persistence unit of "HibernateJpaPU". The query being executed is:

```
select min(a.idAlumno) as MinId,  
max(a.idAlumno) as MaxId,  
count(a.idAlumno) as Count  
from Alumno a
```

A callout bubble points to the query, stating: "Regresa el valor mínimo y máximo del idAlumno (Scalar results)".

Below the query, the "Result SQL" section shows the following table:

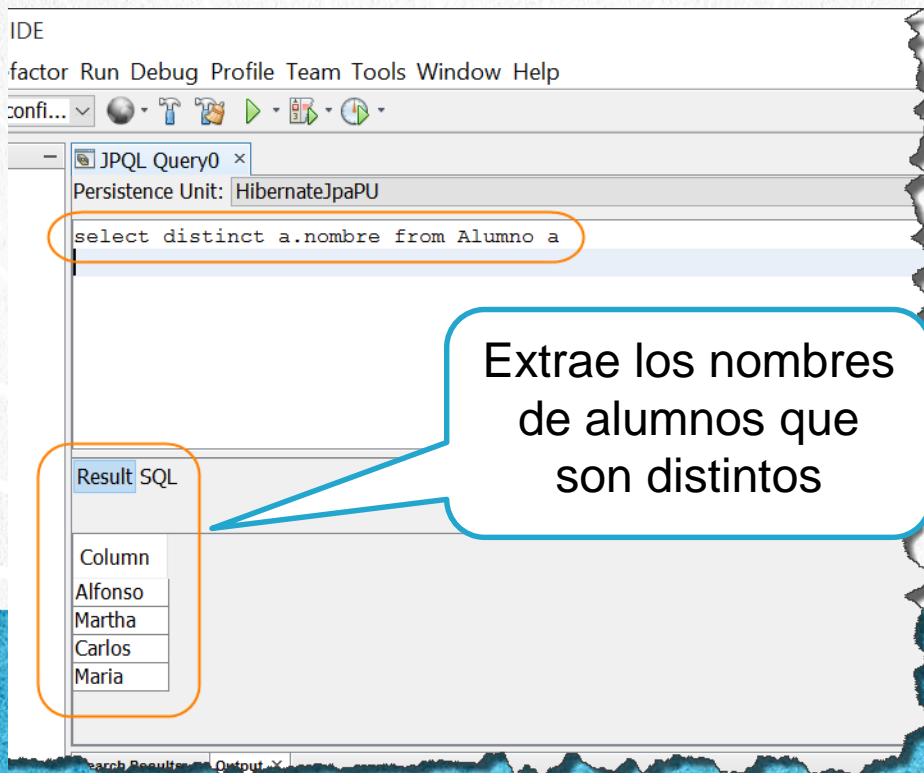
| [0] | [1] | [2] |
|-----|-----|-----|
| 1 | 8 | 4 |

At the bottom of the IDE, a status bar indicates "0 row(s) updated.; 1 row(s) selected." and a "Set Max. Row Count: 100" dropdown is visible.

PASO 15. EJECUTAMOS LA CONSULTA JPQL

Ejecutamos la siguiente consulta:

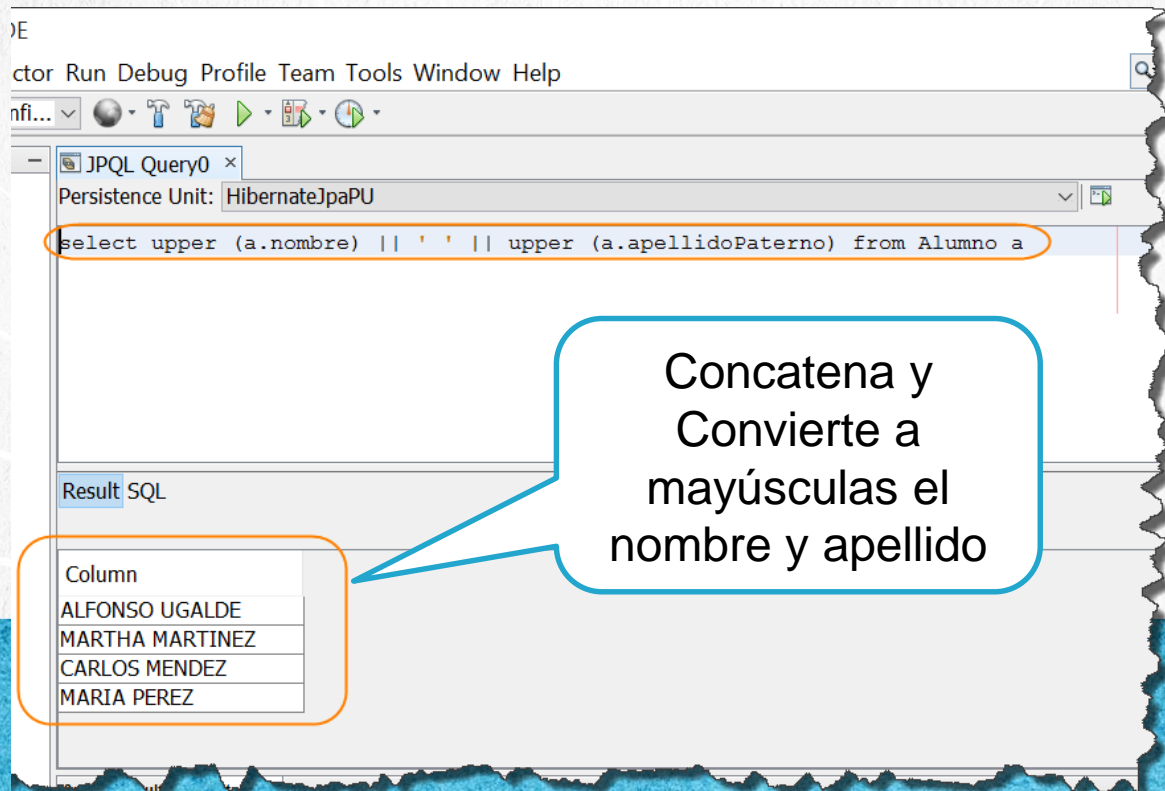
```
select distinct a.nombre from Alumno a
```



PASO 16. EJECUTAMOS LA CONSULTA JPQL

Ejecutamos la siguiente consulta:

```
select upper (a.nombre) || ' ' || upper (a.apellidoPaterno) from Alumno a
```



PASO 17. EJECUTAMOS LA CONSULTA JPQL

La consulta con parámetros no se puede ejecutar en la consola de Netbeans, pero la ejecutaremos con código Java.

from Alumno a where a.idAlumno = :id

Resultado: Obtiene el objeto alumno con id igual al parámetro proporcionado

The screenshot shows the NetBeans IDE interface. At the top, the menu bar includes 'Factor', 'Run', 'Debug', 'Profile', 'Team', 'Tools', 'Window', and 'Help'. Below the menu bar is a toolbar with various icons. The main editor window displays a JPQL query in a file named 'JPQL Query0'. The query is 'from Alumno a where a.idAlumno = 1'. A blue callout bubble points to the value '1' in the query, containing the text 'Suponiendo que el parámetro proporcionado es id = 1'. Below the query editor, the 'Result SQL' tab is active, showing the execution status '0 row(s) updated.; 1 row(s) selected.' and a table of results. The table has 12 columns: 'apellid...', 'apellid...', 'asigna...', 'contacto', 'deleted', 'domicilio', 'fechaN...', 'genero', 'idAlu...', 'nombre', 'usuario', and 'version'. The first row of data contains the values: 'Martinez', 'Ugalde', '[Asigna...', 'Contact...', '0', 'Domicili...', '<NULL>', 'M', '1', 'Alfonso', 'Usuario...', and '0'.

DE

Factor Run Debug Profile Team Tools Window Help

onfi... [Icons]

JPQL Query0 x

Persistence Unit: HibernateJpaPU

from Alumno a where a.idAlumno = 1

Suponiendo que el parámetro proporcionado es id = 1

Result SQL

Set Max. Rows

0 row(s) updated.; 1 row(s) selected.

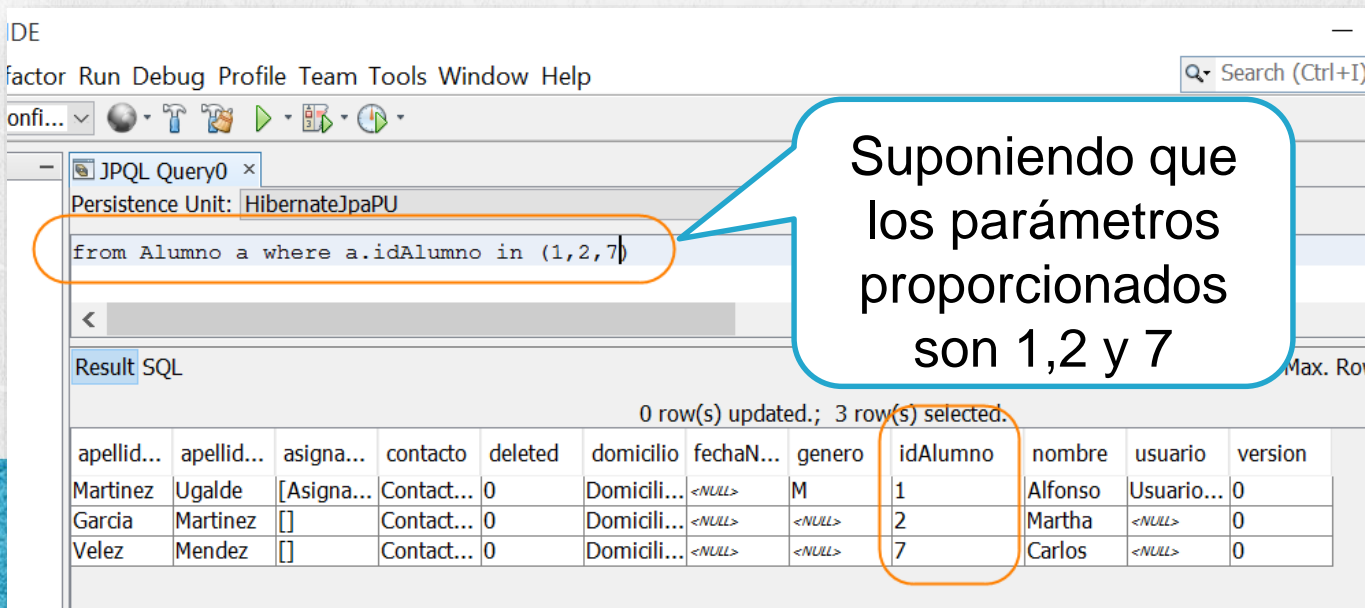
| apellid... | apellid... | asigna... | contacto | deleted | domicilio | fechaN... | genero | idAlu... | nombre | usuario | version |
|------------|------------|------------|------------|---------|-------------|-----------|--------|----------|---------|------------|---------|
| Martinez | Ugalde | [Asigna... | Contact... | 0 | Domicili... | <NULL> | M | 1 | Alfonso | Usuario... | 0 |

PASO 18. EJECUTAMOS LA CONSULTA JPQL

La consulta con parámetros no se puede ejecutar en la consola de Netbeans, ya que se debe proporcionar una lista de datos. Esto lo haremos desde código Java.

```
from Alumno a where a.idAlumno in (:lista)
```

Resultado: Obtiene los alumnos en un rango de ids (no se puede ejecutar en la consola)



The screenshot shows the NetBeans IDE interface. The 'JPQL Query0' editor is active, displaying the query: `from Alumno a where a.idAlumno in (1,2,7)`. The 'Persistence Unit' is set to 'HibernateJpaPU'. Below the query editor, the 'Result SQL' tab shows the results of the query. A status bar indicates '0 row(s) updated.; 3 row(s) selected.' The results are displayed in a table with columns: 'apellid...', 'apellid...', 'asigna...', 'contacto', 'deleted', 'domicilio', 'fechaN...', 'genero', 'idAlumno', 'nombre', 'usuario', and 'version'. The 'idAlumno' column is highlighted with an orange box. A blue callout bubble points to the query, stating: 'Suponiendo que los parámetros proporcionados son 1,2 y 7'.

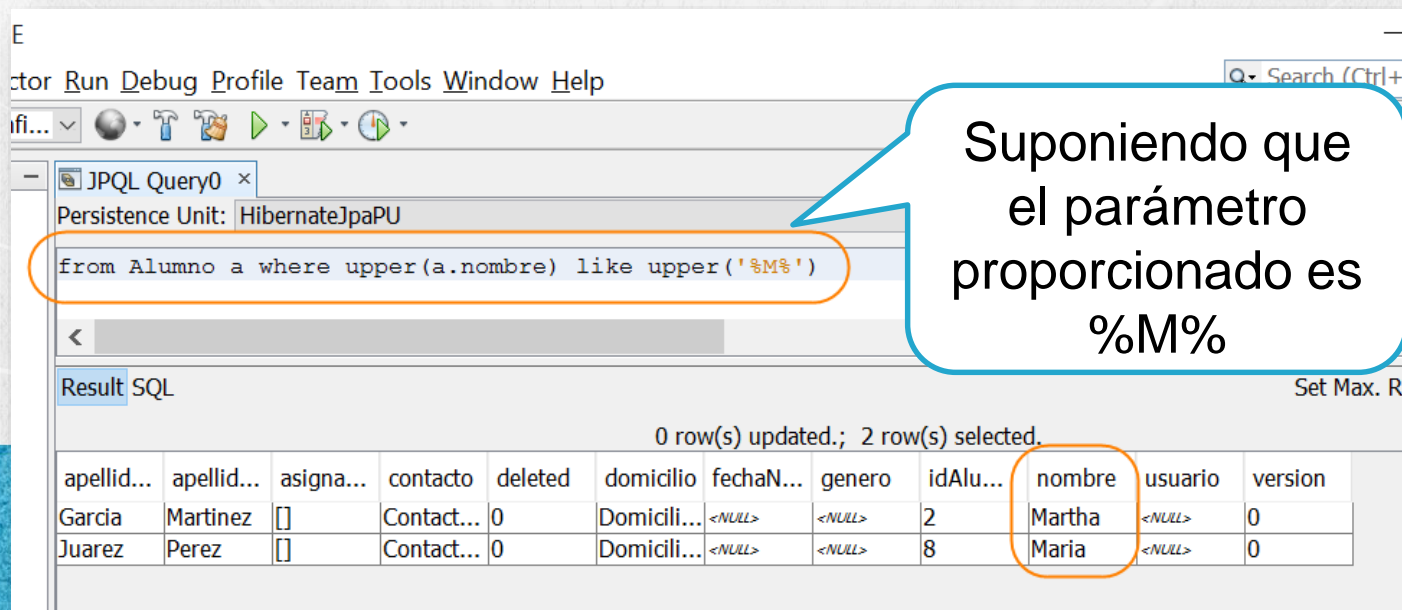
| apellid... | apellid... | asigna... | contacto | deleted | domicilio | fechaN... | genero | idAlumno | nombre | usuario | version |
|------------|------------|------------|------------|---------|-------------|-----------|--------|----------|---------|------------|---------|
| Martinez | Ugalde | [Asigna... | Contact... | 0 | Domicili... | <NULL> | M | 1 | Alfonso | Usuario... | 0 |
| Garcia | Martinez | [] | Contact... | 0 | Domicili... | <NULL> | <NULL> | 2 | Martha | <NULL> | 0 |
| Velez | Mendez | [] | Contact... | 0 | Domicili... | <NULL> | <NULL> | 7 | Carlos | <NULL> | 0 |

PASO 19. EJECUTAMOS LA CONSULTA JPQL

La consulta con parámetros no se puede ejecutar en la consola de Netbeans, pero la ejecutaremos con código Java.

```
from Alumno a where upper(a.nombre) like upper(:param1)
```

Resultado: Obtiene los alumnos que contenga una letra M, sin importar mayúscula/minúsculas.



The screenshot shows the NetBeans IDE interface. The 'JPQL Query0' window displays the query: `from Alumno a where upper(a.nombre) like upper('%M%')`. The 'Persistence Unit' is set to 'HibernateJpaPU'. The 'Result SQL' window shows the results of the query, indicating '0 row(s) updated.; 2 row(s) selected.' The results table is as follows:

| apellid... | apellid... | asigna... | contacto | deleted | domicilio | fechaN... | genero | idAlu... | nombre | usuario | version |
|------------|------------|-----------|------------|---------|-------------|-----------|--------|----------|--------|---------|---------|
| Garcia | Martinez | [] | Contact... | 0 | Domicili... | <NULL> | <NULL> | 2 | Martha | <NULL> | 0 |
| Juarez | Perez | [] | Contact... | 0 | Domicili... | <NULL> | <NULL> | 8 | Maria | <NULL> | 0 |

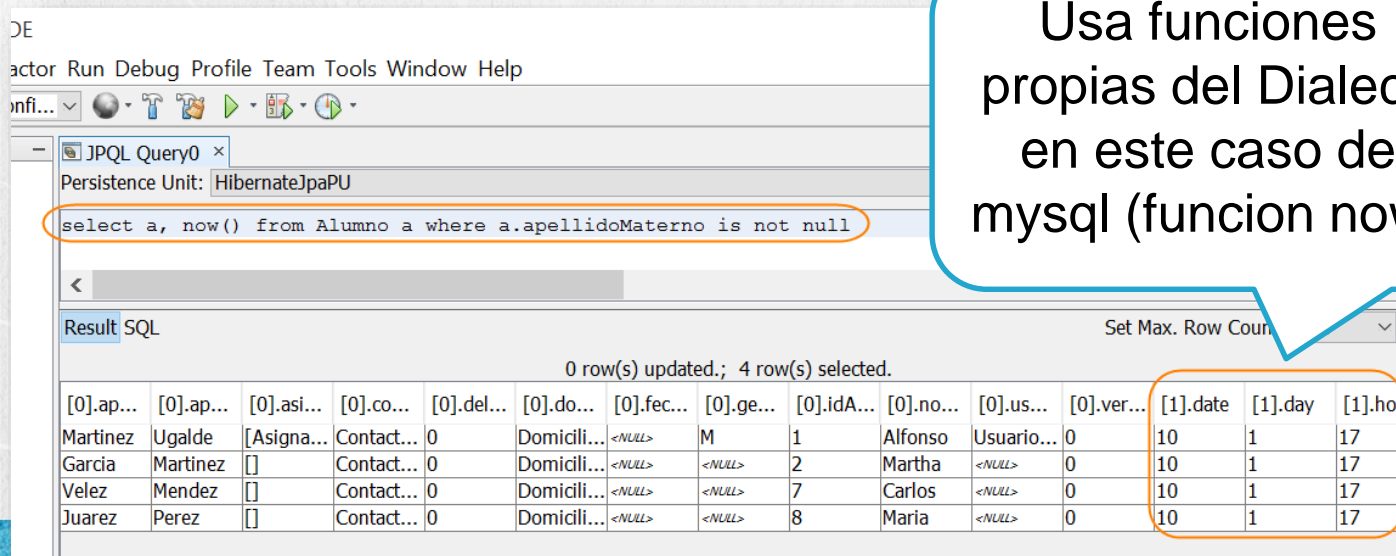
A callout bubble points to the query text, stating: 'Suponiendo que el parámetro proporcionado es %M%'.

PASO 20. EJECUTAMOS LA CONSULTA JPQL

Ejecutamos la siguiente consulta:

```
select a, now() from Alumno a where a.apellidoMaterno is not null
```

Resultado: Regresa la fecha now de mysql, similar a Date en Java.



IDE Screenshot showing the JPQL Query and its results.

Query: `select a, now() from Alumno a where a.apellidoMaterno is not null`

Result SQL: 0 row(s) updated.; 4 row(s) selected.

| [0].ap... | [0].ap... | [0].asi... | [0].co... | [0].del... | [0].do... | [0].fec... | [0].ge... | [0].idA... | [0].no... | [0].us... | [0].ver... | [1].date | [1].day | [1].ho |
|-----------|-----------|------------|------------|------------|-------------|------------|-----------|------------|-----------|------------|------------|----------|---------|--------|
| Martinez | Ugalde | [Asigna... | Contact... | 0 | Domicili... | <NULL> | M | 1 | Alfonso | Usuario... | 0 | 10 | 1 | 17 |
| Garcia | Martinez | [] | Contact... | 0 | Domicili... | <NULL> | <NULL> | 2 | Martha | <NULL> | 0 | 10 | 1 | 17 |
| Velez | Mendez | [] | Contact... | 0 | Domicili... | <NULL> | <NULL> | 7 | Carlos | <NULL> | 0 | 10 | 1 | 17 |
| Juarez | Perez | [] | Contact... | 0 | Domicili... | <NULL> | <NULL> | 8 | Maria | <NULL> | 0 | 10 | 1 | 17 |

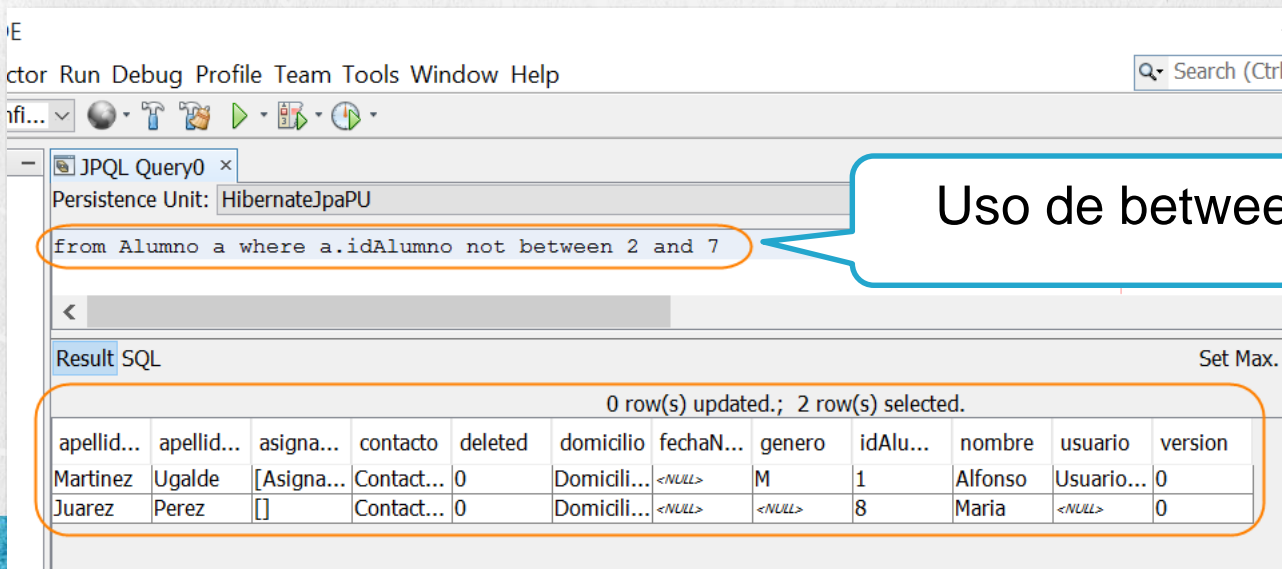
Usa funciones propias del Dialect, en este caso de mysql (funcion now)

PASO 21. EJECUTAMOS LA CONSULTA JPQL

Ejecutamos la siguiente consulta:

```
from Alumno a where a.idAlumno not between 2 and 5
```

Resultado: Regresa los alumnos cuyo idAlumno NO esté entre 2 y 7



The screenshot shows an IDE window titled "JPQL Query0". The query text is "from Alumno a where a.idAlumno not between 2 and 7". A callout bubble points to the "between" keyword with the text "Uso de between". Below the query, the "Result SQL" tab is active, showing "0 row(s) updated.; 2 row(s) selected." and a table of results.

| apellid... | apellid... | asigna... | contacto | deleted | domicilio | fechaN... | genero | idAlu... | nombre | usuario | version |
|------------|------------|------------|------------|---------|-------------|-----------|--------|----------|---------|------------|---------|
| Martinez | Ugalde | [Asigna... | Contact... | 0 | Domicili... | <NULL> | M | 1 | Alfonso | Usuario... | 0 |
| Juarez | Perez | [] | Contact... | 0 | Domicili... | <NULL> | <NULL> | 8 | Maria | <NULL> | 0 |

CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx

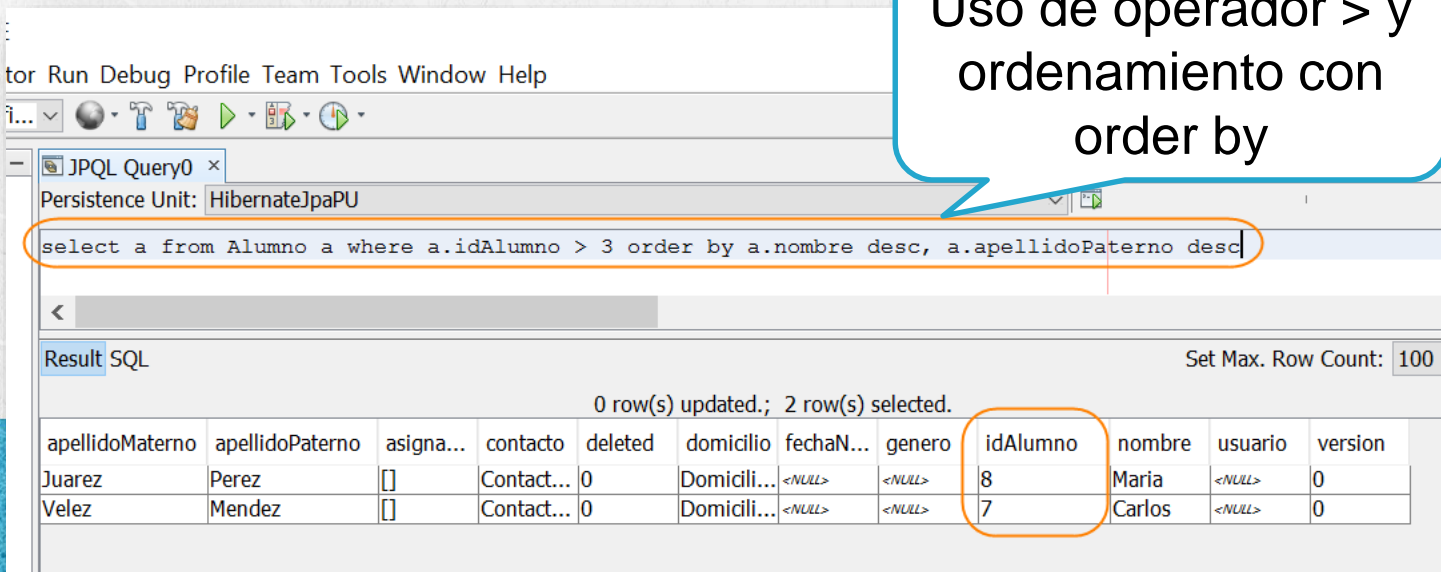
PASO 22. EJECUTAMOS LA CONSULTA JPQL

Ejecutamos las siguiente consulta:

```
select a from Alumno a where a.idAlumno > 3 order by a.nombre desc,  
a.apellidoPaterno desc
```

Resultado: Regresa los alumnos cuyo idAlumno sea mayor a 3 y ordenados por nombre y apellidoPaterno

Uso de operador > y
ordenamiento con
order by



The screenshot shows an IDE window titled "JPQL Query0" with a persistence unit of "HibernateJpaPU". The query entered is "select a from Alumno a where a.idAlumno > 3 order by a.nombre desc, a.apellidoPaterno desc". Below the query editor, the results are displayed in a table. The table has 12 columns: apellidoMaterno, apellidoPaterno, asigna..., contacto, deleted, domicilio, fechaN..., genero, idAlumno, nombre, usuario, and version. Two rows are selected, corresponding to idAlumno 8 and 7. The idAlumno column is circled in orange in the original image.

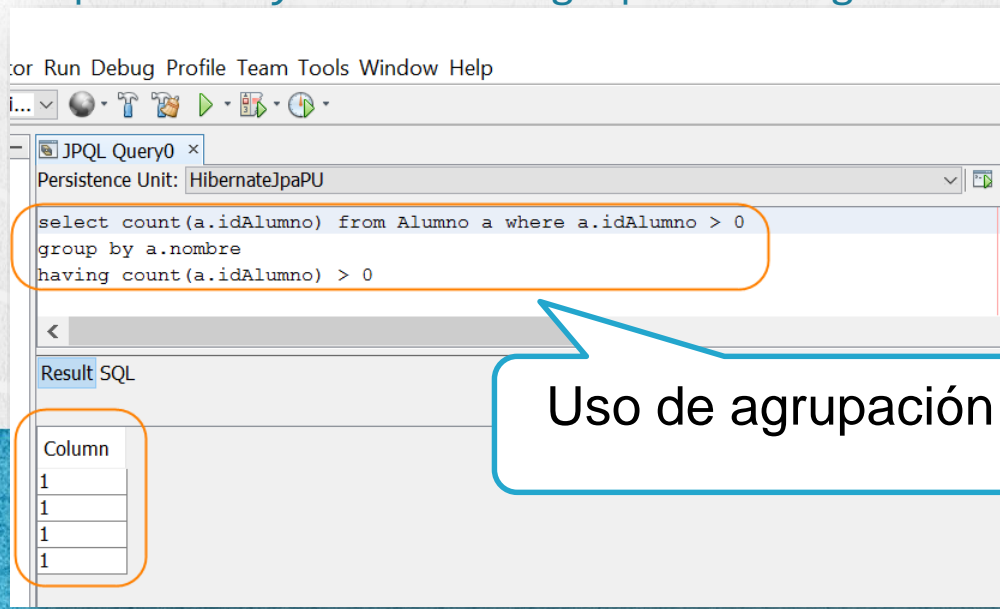
| apellidoMaterno | apellidoPaterno | asigna... | contacto | deleted | domicilio | fechaN... | genero | idAlumno | nombre | usuario | version |
|-----------------|-----------------|--------------------------|------------|---------|-------------|-----------|--------|----------|--------|---------|---------|
| Juarez | Perez | <input type="checkbox"/> | Contact... | 0 | Domicili... | <NULL> | <NULL> | 8 | Maria | <NULL> | 0 |
| Velez | Mendez | <input type="checkbox"/> | Contact... | 0 | Domicili... | <NULL> | <NULL> | 7 | Carlos | <NULL> | 0 |

PASO 23. EJECUTAMOS LA CONSULTA JPQL

Ejecutamos la siguiente consulta:

```
select count(a.idAlumno) from Alumno a where a.idAlumno > 0  
group by a.nombre  
having count(a.idAlumno) > 0
```

Resultado: Regresa el conteo de alumnos agrupándose por nombre, donde el idAlumno es mayor que cero y conteo de grupo contenga al menos 1 elemento.

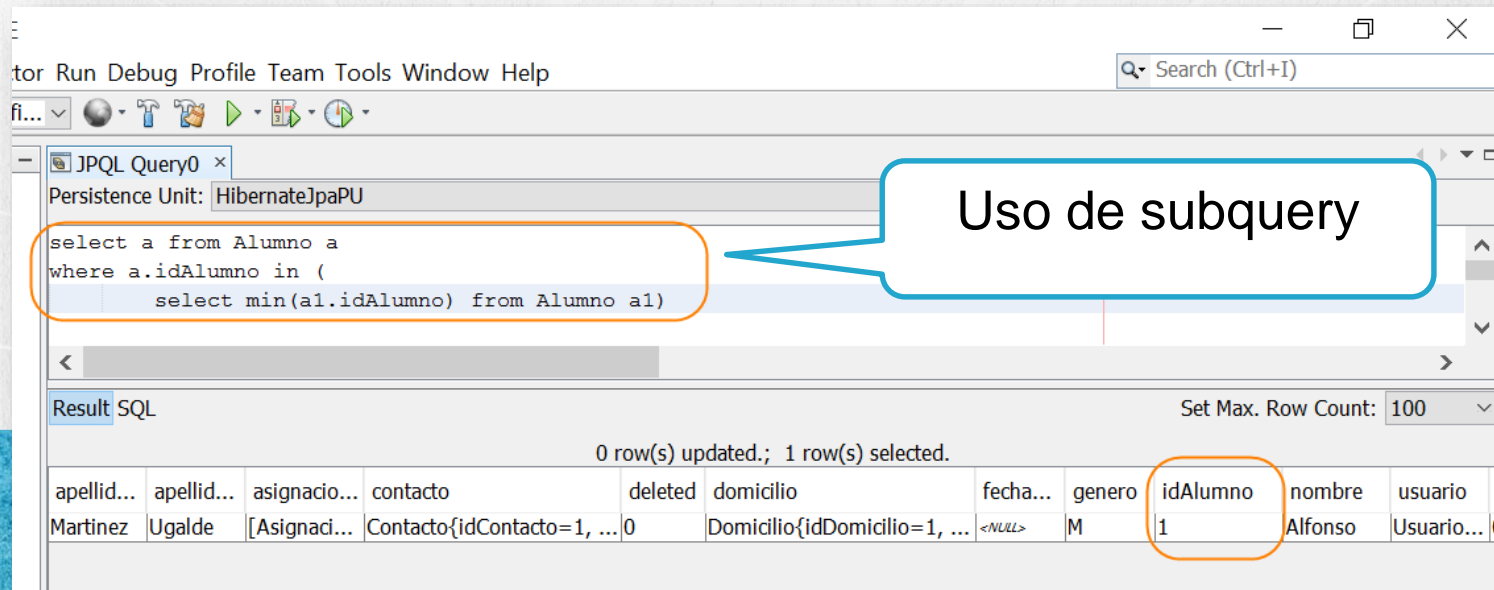


PASO 24. EJECUTAMOS LA CONSULTA JPQL

Ejecutamos las siguiente consulta:

```
select a from Alumno a
where a.idAlumno in ( select min(a1.idAlumno) from Alumno a1 )
```

Resultado: Regresa el alumno cuyo id es el menor de los idAlumno. el soporte de esta funcionalidad depende de la base de datos utilizada



The screenshot shows an IDE window titled "JPQL Query0". The query is: `select a from Alumno a where a.idAlumno in (select min(a1.idAlumno) from Alumno a1)`. The subquery `select min(a1.idAlumno) from Alumno a1` is highlighted with an orange box. A blue callout bubble points to this subquery with the text "Uso de subquery". Below the query editor, the "Result SQL" tab is active, showing "0 row(s) updated.; 1 row(s) selected." and a table with 10 columns: `apellid...`, `apellid...`, `asignacio...`, `contacto`, `deleted`, `domicilio`, `fecha...`, `genero`, `idAlumno`, `nombre`, and `usuario`. The first row of data is: `Martinez`, `Ugalde`, `[Asignaci...`, `Contacto{idContacto=1, ...`, `0`, `Domicilio{idDomicilio=1, ...`, `<NULL>`, `M`, `1`, `Alfonso`, and `Usuario...`. The `idAlumno` value `1` is highlighted with an orange box.

| apellid... | apellid... | asignacio... | contacto | deleted | domicilio | fecha... | genero | idAlumno | nombre | usuario |
|------------|------------|--------------|----------------------------|---------|------------------------------|----------|--------|----------|---------|------------|
| Martinez | Ugalde | [Asignaci... | Contacto{idContacto=1, ... | 0 | Domicilio{idDomicilio=1, ... | <NULL> | M | 1 | Alfonso | Usuario... |

PASO 25. EJECUTAMOS LA CONSULTA JPQL

Ejecutamos las siguiente consulta:

```
from Alumno a join a.domicilio d join a.contacto c
```

Resultado: Regresa los alumnos sin regresar sus relaciones como es domicilio o contacto.

ConsultasHQLyJPQL - NetBeans IDE

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

JPQL Query0 x

Persistence Unit: HibernateJpaPU

```
from Alumno a
join a.domicilio d
join a.contacto c
```

Uso de join con lazy loading

Result SQL

Set Max. Row Count: 100

0 row(s) updated.; 4 row(s) selected.

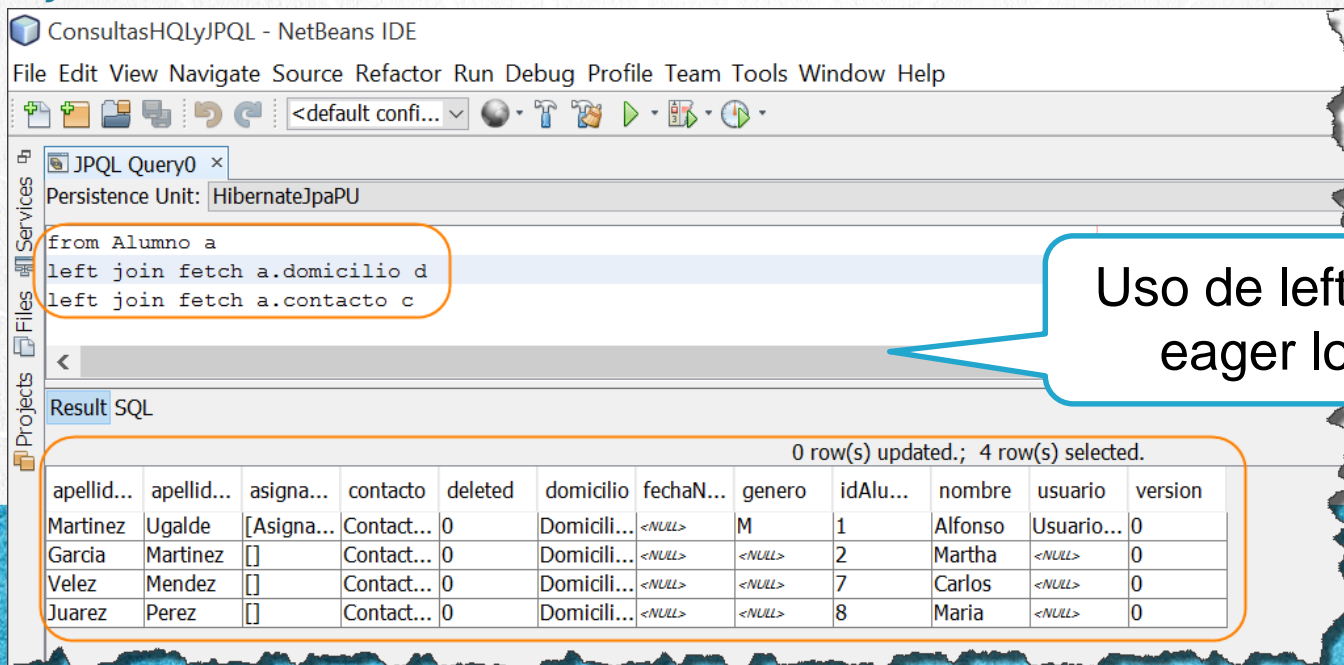
| [0].ge... | [0].idA... | [0].no... | [0].us... | [0].ver... | [1].alu... | [1].calle | [1].cp | [1].del... | [1].est... | [1].idD... | [1].ins... | [1].mu... | [1].no... | [1].noI... | [1].pais | [1].ver... | [2].alu... | [2]. |
|-----------|------------|-----------|------------|------------|------------|-----------|--------|------------|------------|------------|-------------|-----------|-----------|------------|----------|------------|------------|--------|
| M | 1 | Alfonso | Usuario... | 0 | Alumn... | Londres | 112233 | 0 | estado | 1 | [Instruc... | municipio | 1 | 1 | pais | 0 | [Alumn... | <NULL> |
| <NULL> | 2 | Martha | <NULL> | 0 | Alumn... | Allende | <NULL> | 0 | <NULL> | 2 | [] | <NULL> | 115 | A-101 | <NULL> | 0 | [Alumn... | <NULL> |
| <NULL> | 7 | Carlos | <NULL> | 0 | Alumn... | Jupiter | <NULL> | 0 | <NULL> | 9 | [] | <NULL> | <NULL> | <NULL> | <NULL> | 0 | [Alumn... | <NULL> |
| <NULL> | 8 | Maria | <NULL> | 0 | Alumn... | Darwin | <NULL> | 0 | <NULL> | 10 | [] | <NULL> | <NULL> | <NULL> | <NULL> | 0 | [Alumn... | <NULL> |

PASO 26. EJECUTAMOS LA CONSULTA JPQL

Ejecutamos la siguiente consulta:

```
from Alumno a left join fetch a.domicilio d left join fetch a.contacto c
```

Resultado: Regresa los alumnos cargando también las relaciones como son domicilio y contacto.



The screenshot shows the NetBeans IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, and Help. Below the menu is a toolbar with various icons. The main editor window displays a JPQL query in a tab titled "JPQL Query0". The query is: `from Alumno a left join fetch a.domicilio d left join fetch a.contacto c`. The query is highlighted with a blue selection box. Below the query, the persistence unit is set to "HibernateJpaPU". The "Result SQL" section shows the query results. The results are displayed in a table with 12 columns: apellido..., apellido..., asigna..., contacto, deleted, domicilio, fechaN..., genero, idAlu..., nombre, usuario, and version. The table contains 4 rows of data. A status bar at the bottom indicates "0 row(s) updated.; 4 row(s) selected.".

ConsultasHQLyJPQL - NetBeans IDE

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

JPQL Query0 x

Persistence Unit: HibernateJpaPU

```
from Alumno a
left join fetch a.domicilio d
left join fetch a.contacto c
```

Result SQL

0 row(s) updated.; 4 row(s) selected.

| apellido... | apellido... | asigna... | contacto | deleted | domicilio | fechaN... | genero | idAlu... | nombre | usuario | version |
|-------------|-------------|------------|------------|---------|-------------|-----------|--------|----------|---------|------------|---------|
| Martinez | Ugalde | [Asigna... | Contact... | 0 | Domicili... | <NULL> | M | 1 | Alfonso | Usuario... | 0 |
| Garcia | Martinez | [] | Contact... | 0 | Domicili... | <NULL> | <NULL> | 2 | Martha | <NULL> | 0 |
| Velez | Mendez | [] | Contact... | 0 | Domicili... | <NULL> | <NULL> | 7 | Carlos | <NULL> | 0 |
| Juarez | Perez | [] | Contact... | 0 | Domicili... | <NULL> | <NULL> | 8 | Maria | <NULL> | 0 |

Uso de left join con eager loading

PASO 27. CONSULTA DE QUERIES HQL/JPQL

Si desean pueden descargar todas las consultas que hemos visto hasta el momento del siguiente link:

<http://icursos.net/cursos/Hibernate/Leccion08/ConsultasHQL-JPQL/Consultas-HQL-JPQL.txt>



Experiencia y Conocimiento para tu vida



CONCLUSIÓN DEL EJERCICIO

Con este ejercicio hemos revisado varias de las consultas que podemos ejecutar con HQL/JPQL.

Desde consultas simples, con parámetros, ordenamiento, agrupaciones, lazy loading y eager loading, entre varios ejemplos más.

A continuación ejecutaremos estas consultas pero desde código Java.

CURSO ONLINE

HIBERNATE & JPA

Por: Ing. Ubaldo Acosta



Experiencia y Conocimiento para tu vida



CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx