Formación JPA

<?xml version="1.0"?><DocumentFileOSQ xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"> <GraphicCharterDefinitionId>0</GraphicCharterDefinitionId> <TemplateBaseTypeId>0</TemplateBaseTypeId> <CompanyId>1</CompanyId> <ConfidentialId>0</ConfidentialId> <ConfidentialDescription /> <CountryId>20</CountryId> <PageSizeId>1</PageSizeId> <PageOrientationId>1</PageOrientationId> <PrePrintedStationary>false</PrePrintedStationary> <Project>GEMIS</Project> <Reference>20170124-170150-esgomez</Reference> <TemplateType>3</TemplateType> <CultureId>fr-FR</CultureId> <LanguageId>2</LanguageId> <Customer>Centro de Servicio Valencia</Customer> <DocumentDate>2018-12-04T12:02:38.1672589+01:00</DocumentDate> <Saved>true</Saved> <IsValid>true</IsValid> <FirstPageCover>false</FirstPageCover> <IsNew>false</IsNew> <CurrentVersion>1.00</CurrentVersion> <DocumentType>Solution Requirements Specification</DocumentType> <DocumentTypeId>-1</DocumentTypeId> <Entity /> <HasDistributionList>false</HasDistributionList> <HasForeword>false</HasForeword> <Recipient>Customer Stakeholders (including at least the IT PM and the Stakeholder and User manager)</Recipient> <Title>Spécifications des besoins de la solution - GEMIS</Title> <Status>2</Status> <StatusDescription>Projet</StatusDescription> <SetEdition>false</SetEdition> <SetVersion>false</SetVersion> <TemplateEditor>EN\_eMREQ-SRS</TemplateEditor> <TemplateVersionMajor>1.2</TemplateVersionMajor> <TemplateVersionMinor>4</TemplateVersionMinor> <TemplateYear>2016</TemplateYear> <TemplateState>1</TemplateState></DocumentFileOSQ>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  | Centre de service Valence | | |  |
|  |  | GEMIS | | |  |
|  |  | Formación JPA - GEMIS | | |  |
|  |  | Version 1.0 Martes 20 Noviembre 2018 | | |  |
|  |  | Statut: Initiation | | |  |
|  |  |  | |  |  |

Historique

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version** | **Date** | | **Origine de la mise à jour** | **Rédigée par** | **Validée par** |
| 1.0 | 21/11/2018 | | Première version du document | Alba Bermejo Solís  Adrián Colmena Mateos  Emilio Guillem Simón |  |
|  | |
|  |  | |  |  |  |
|  |  | |  |  |  |

Contenidos

[1. Capítulo 4 4](#_Toc531688287)

[1.1. Cláusula WHERE 4](#_Toc531688288)

[1.1.1. Parámetros de entrada 4](#_Toc531688289)

[1.1.2. Operadores 4](#_Toc531688290)

[1.1.3. Expresiones BETWEEN 5](#_Toc531688291)

[1.1.4. Expresiones LIKE 5](#_Toc531688292)

[1.1.5. Subconsultas 5](#_Toc531688293)

[1.1.6. Expresiones IN 6](#_Toc531688294)

[1.1.7. Expresiones de colección 7](#_Toc531688295)

[1.1.8. Expresión EXISTS 8](#_Toc531688296)

[1.1.9. Expresiones ANY, ALL y SOME 8](#_Toc531688297)

[1.2. Herencia y polimorfismo 9](#_Toc531688298)

[1.2.1. Selección entre subclases 9](#_Toc531688299)

[1.2.2. *Downcasting* 9](#_Toc531688300)

[1.3. Expresiones escalares 10](#_Toc531688301)

[1.3.1. Valores literales 10](#_Toc531688302)

[1.3.2. Funciones 10](#_Toc531688303)

# Capítulo 4

## Cláusula WHERE



**WHERE** permite especificar condiciones para limitar los valores devueltos en una consulta. Prácticamente todo lo visto para esta cláusula se puede aplicar a **HAVING**.

Esta cláusula funciona de manera sencilla. Simplemente limita la consulta escribiendo WHERE y después la condición.

### Parámetros de entrada

Los parámetros que se usen para las condiciones pueden ser tanto el nombre o la posición que ocupa.

* Mediante el nombre. Se escribe “:nombre\_del\_parámetro” en la consulta. Distingue entre mayúsculas y minúsculas. Véase el siguiente ejemplo:

SELECT e

FROM Employee e

WHERE e.salary > :sal

Devuelve aquellos salarios de los empleados en la tabla Empleado que sean mayores del valor del registro “sal”

* Mediante su posición. Para ello se escribe el número de la variable precedido por un signo de interrogación tal que: “?1”, “?2”, etc. Véase un ejemplo análogo:

SELECT e

FROM Employee e

WHERE e.salary > ?1

### Operadores

La forma de expresar en JPQL las condiciones, proviene en gran parte del lenguaje SQL. La diferencia clave entre las expresiones condicionales en JPQL y SQL, es que JPQL al ser dinámico permite identificar y acceder a la dirección de memoria de las distintas variables mientras se evalúa una expresión.

Los operadores son también similares a los que se usan en la cláusula de WHERE en SQL.

* Operador de navegación (.)
* Aritméticos (+, -, \*, /)
* Lógicos (AND, OR, NOT)
* Comparación (=, >, <, >=, <=, <>, [NOT] BETWEEN, [NOT] LIKE, [NOT] IN, , IS [NOT] NULL, IS [NOT] EMPTY, [NOT] MEMBER [OF])

### Expresiones BETWEEN

Se utilizan para delimitar el rango de valores en el que se desea que encuentre un resultado. Para ello se pueden utilizar valores numéricos, *strings* y fechas. Véase el siguiente ejemplo.

SELECT e

FROM Employee e

WHERE e.salary BETWEEN 40000 AND 45000

El resultado de esta consulta serán aquellos empleados cuyo salario este en ese rango incluyendo 40000 y 45000. Una forma análoga con operadores de comparación básicos sería:

SELECT e

FROM Employee e

WHERE e.salary >= 40000 AND e.salary <= 45000

El operador BETWEEN también puede ser negado con el operador NOT.

### Expresiones LIKE

Sirven para comparar textos (*string*). Tienen dos operadores que se escriben dentro del *string*, uno (\_) que equivale a cualquier carácter y otro (%) que equivale a cualquier cadena de texto. Véase el ejemplo:

SELECT d

FROM Departament d

WHERE d.name LIKE '\_\_Ent%'

El resultado de esta consulta serán aquellos departamentos, que tengan dos caracteres al principio (nótese que hay dos \_ unidas), luego contengan el *string* 'Ent' y a continuación tengan cualquier cadena de texto. Un ejemplo sería iVEntas o SrEntas, pero no COmpras.

### Subconsultas

Pueden ser usadas combinándolas con WHERE o HAVING. Una subquery, es una consulta SELECT que irá, entre paréntesis, dentro de una condición. En este ejemplo:

SELECT e

FROM Employee e

WHERE e. salary = (SELECT MAX(emp.salary)

FROM Employee emp)

El resultado será el empleado con el mayor salario entre todos los empleados.

La posibilidad de referirse a una variable de la consulta principal en una subconsulta permite a las dos estar relacionadas, como en el siguiente ejemplo:

SELECT e

FROM Employee e

WHERE EXISTS (SELECT 1

FROM Phone p

WHERE p.employee = e AND p.type = 'Cell')

El resultado será aquellos empleados que tengan un número de teléfono móvil. Más adelante se verá la palabra reservada *EXISTS.*

### Expresiones IN

Permite saber si un valor pertenece a una lista de textos o números. El siguiente ejemplo mostrará aquellos empleados que viven en Nueva York o California:

SELECT e

FROM Employee e

WHERE e.address.state IN ('NY', 'CA')

Estas expresiones pueden utilizar subconsultas a su vez, donde la lista será el resultado de esa subconsulta. El siguiente ejemplo devuelve aquellos empleados que trabajan en departamentos que contribuyen con proyectos cuyo nombre empieza por QA.

SELECT e

FROM Employee e

WHERE e.departament IN (SELECT DISTINCT d

FROM Departament d JOIN d.employee de JOIN

de.projects p

WHERE p.name LIKE 'QA%')

La expresión IN puede ser negada con el operador NOT. En este ejemplo se seleccionan aquellos números de teléfono que no son ni de la oficina ni de casa.

SELECT p

FROM Phone p

WHERE p.type NOT IN ('Office', 'Home')

### Expresiones de colección

El operador IS EMPTY es el equivalente de IS NULL pero para colecciones. Puede ser negado también (IS NOT EMPTY) para ver que una colección contenga al menos un valor.

El siguiente ejemplo mostrará los empleados que sean directores de al menos otro empleado:

SELECT e

FROM Employee e

WHERE e.directs IS NOT EMPTY

Para traducir IS EMPTY a SQL se deben de utilizar subconsultas. Para ello, el ejemplo anterior quedaría como:

SELECT m

FROM Employee m

WHERE (SELECT COUNT(e)

FROM Employee e

WHERE e.manager = m) > 0

El operador MEMBER OF muestra si un valor es miembro de una colección. También puede ser negado (NOT MEMBER OF). La siguiente consulta mostrará aquellos empleados que están asignados a un determinado proyecto.

SELECT e

FROM Employee e

WHERE :project MEMBER OF e.projects

También en este caso para traducirlo a SQL sería necesario utilizar subconsultas. El mismo ejemplo en JPQL pero utilizando subconsultas:

SELECT e

FROM Employee e

WHERE :project IN (SELECT p

FROM e.projects p)

### Expresión EXISTS

La condición EXISTS devuelve verdadero si una subconsulta devuelve cualquier línea. El operador EXISTS también puede ser negado con el operador NOT. La siguiente consulta selecciona a todos los empleados que no tienen un teléfono móvil:

SELECT e

FROM Employee e

WHERE NOT EXISTS (SELECT p

FROM e.phones p

WHERE p.type = 'Cell')

### Expresiones ANY, ALL y SOME

Se usan junto con los comparadores básicos: =, >, <, >=, <= y <>. All devuelve *true* cuando todos los elementos de la lista cumplen la condición, va comparando la condición a la izquierda con todos los elementos de la derecha. Para que la condición sea efectiva, todos los elementos de la lista deben cumplir esa condición.

SELECT e

FROM Employee e

WHERE e.directs IS NOT EMPTY AND

e.salary < ALL(SELECT d.salary

FROM e.directs d)

Esta consulta devuelve aquellos jefes cuyo salario es menor que todos los empleados que trabajan para ellos.

ANY trabaja de manera análoga, pero si encuentra al menos un elemento de la lista que cumple la condición ya devuelve *true*. Tiene un funcionamiento muy parecido al de las expresiones IN. Usando el mismo ejemplo que en las expresiones IN, pero con ANY:

SELECT e

FROM Employee e

WHERE e.departament = ANY (SELECT DISTINCT d

FROM Departament d JOIN d.employee de JOIN

de.projects p

WHERE p.name LIKE 'QA%')

## Herencia y polimorfismo

JPA también trabaja con herencia entre entidades, por lo que varias subclases de una entidad pueden ser el resultado de una misma consulta.

Por ejemplo, Proyect es la clase madre para QualityProject y DesignProject. Si se utiliza una variable de identificación directamente de la entidad Proyect, el resultado de la consulta incluirá una mezcla de objetos Java tanto de Proyect como de QualityProject y DesignProject, y con el resultado se puede hacer *cast* según sea necesario.

### Selección entre subclases

Si se quiere re*string*ir el resultado de una consulta a una subclase particular, después de la cláusula FROM puede utilizarse esa subclase en vez de la entidad raíz. Sin embargo, si el resultado quiere ampliarse a más de una subclase pero no a todas, la cláusula WHERE unida a TYPE puede incluir las subclases que se quieran.

En este ejemplo sólo se devolverán los proyectos de calidad y de diseño:

SELECT p

FROM Project p

WHERE TYPE(p) = DesignProject OR TYPE(p) = QualityProject

Después del type, no se han puesto ' ', ya que JPQL toma los nombres de las subclases como entidades no como *strings.* Estas consultas también pueden hacerse dinámicas, y que tome el valor de esa subclase dado por el programador:

SELECT p

FROM Project p

WHERE TYPE(p) = :projectType

### *Downcasting*

Puede accederse a un atributo de una subclase pero cuando la consulta trata con la clase madre, hay que usar un *cast* al que se demonina *downcasting.* Esta técnica consiste en hacer que una consulta que se refiere a una clase sea aplicada a una subclase de esta específica, usando el operador TREAT.

En este ejemplo, el resultado será todos los proyectos de diseño y los de calidad con un índice de calidad mayor de 4.

SELECT p

FROM Project p

WHERE TREAT(p AS QualityProject).qaRating > 4

OR TYPE(p) = DesignProject

La sintaxis de la expresión empieza con TREAT seguido con su argumento entre paréntesis. El argumento será el alias de la clase madre, la palabra reservada AS y el nombre de la subclase.

## Expresiones escalares

Una expresión escalaras es un valor literal, una secuencia aritmética, una función, una expresión *type* o una expresión *case* que tiene como resultado un valor escalar (número). Pueden ser usadas en un SELECT o como condición en una expresión WHERE o HAVING.

### Valores literales

En JPQL pueden usarse *strings*, valores numéricos, *booleans*, *enums*, entidades y fechas.

En el siguiente ejemplo se usa un *enum* llamado *PhoneType*.

SELECT e

FROM Employee e JOIN e.phoneNumbers p

WHERE KEY(p) = com.acme.PhoneType.Home

Para fechas, se utiliza el mismo formato que JBDC, tal que:

{d 'yyyy-mm-dd'} e.g. {d '2009-11-05'}

{t 'hh-mm-ss'} e.g. {t '12-45-52'}

{ts 'yyyy-mm-dd hh-mm-ss.f'} e.g. {ts '2009-11-05 12-45-52.325'}

### Funciones

La tabla a continuación resume la sintaxis para las distintas funciones soportadas por JPA.

|  |  |
| --- | --- |
| **Función** | **Devuelve** |
| ABS (número) | El valor absoluto del número en el argumento con el mismo tipo (*int, float* o *double*) |
| CONCAT(*string1*, *string2*) | Un nuevo *string* que es la concatenación de los dos en el argumento |
| CURRENT\_DATE | La fecha actual definida en la base de datos del servidor |
| CURRENT\_TIME | La hora actual definida en la base de datos del servidor |
| CURRENT\_TIMESTAMP | El *timestamp* actual definido en la base de datos del servidor |
| INDEX(alias) | La posición de una entidad en una lista ordenada |
| LENGTH(*string*) | El número de caracteres en un *string* |
| LOCATE(*string1*, *string2* [,start]) | La posición del *string1* en el *string2,* pudiendo empezar por la posición indicada por *start*. El resultado es 0 si el *string* no se encuentra |
| LOWER(*string*) | El *string* del argumento en minúsculas |
| MOD(num1, num2) | El módulo del cociente entre num1 y num2, como un *integer* |
| SIZE(colección) | El número de elementos de un colección, devolviendo 0 si está vacía |
| SQRT(número) | La raíz cuadrada del número en el argumento como un *double* |
| SUBSTRING(*string*, start, end) | Una parte del *string,* empezando en la posición indicada en starthasta la posición end. La posición de un *string* empieza en 1 |
| UPPER(*string*) | El *string* del argumento en mayúsculas |
| TRIM([[LEADING|TRAILING|BOTH] [*char*] FROM] *string* | Elimina los caracteres LEADING y/o TRAILING de un *string.* Si no se usa TREADING, LEADING o BOTH, ambos caracteres se eliminan. El carácter por defecto de TRIM son los espacios |

La función SIZE requiere especial atención especial porque es una notación abreviada para una subconsulta agregada. Por ejemplo, en el siguiente ejemplo se devuelve aquellos departamentos con sólo dos empleados:

SELECT d

FROM Department d

WHERE SIZE(d.employees) = 2