**Cláusula GROUP BY**

La cláusula GROUP BY define las expresiones de agrupación sobre las que se agregarán los resultados. Una expresión de agrupación debe ser una expresión de trayectoria de un solo valor o una variable de identificación, la entidad no debe tener ningún estado serializado ni campos de objetos grandes. La siguiente consulta cuenta el número de empleados en cada departamento:

SELECT d.name, COUNT(e)

FROM Department d JOIN d.employees e

GROUP BY d.name

Se debe tener en cuenta que la misma expresión de campo utilizada en la cláusula SELECT se repite en la cláusula GROUP BY. Todas las expresiones no agregadas deben ser listadas de esta manera. Se puede aplicar más de una función agregada:

SELECT d.name, COUNT(e), AVG(e.salary)

FROM Department d JOIN d.employees e

GROUP BY d.name

Esta variación de la consulta calcula el salario medio de todos los empleados de cada departamento, además de contar el número de empleados del departamento. También se pueden utilizar múltiples expresiones de agrupación para desglosar aún más los resultados:

SELECT d.name, e.salary, COUNT(p)

FROM Department d JOIN d.employees e JOIN e.projects p

GROUP BY d.name, e.salary

Ambas expresiones de agrupación, el nombre del departamento y el salario del empleado, deben aparecer tanto en la cláusula SELECT como en la cláusula GROUP BY. Para cada departamento, esta consulta cuenta el número de proyectos asignados a los empleados en función de su salario.

En ausencia de una cláusula GROUP BY, las funciones agregadas se aplicarán a todo el resultado establecido como un solo grupo. Por ejemplo, la siguiente consulta devuelve el número de empleados y su salario medio en toda la empresa:

SELECT COUNT(e), AVG(e.salary)

FROM Employee e

**Cláusula HAVING**

La cláusula HAVING define un filtro que se aplicará después de agrupar los resultados de la consulta. Tiene un funcionamiento parecido a la cláusula WHERE. La diferencia es, que en la cláusula HAVING, es que las expresiones condicionales se limitan en su mayoría a campos de estados o a campos de asociación de un solo valor incluidos en el grupo.

Las expresiones condicionales de la cláusula HAVING también pueden utilizar funciones agregadas. En muchos aspectos, el uso principal de la cláusula HAVING es restringir los resultados basados en los valores totales de los resultados. La siguiente consulta utiliza esta técnica para recuperar todos los empleados asignados a dos o más proyectos:

SELECT e, COUNT(p)

FROM Employee e JOIN e.projects p

GROUP BY e

HAVING COUNT(p) >= 2

**UPDATE Queries (Consultas de actualización)**

Las consultas de actualización proporcionan un equivalente a la sentencia SQL UPDATE pero con expresiones condicionales JP QL. La forma de una consulta de actualización es la siguiente:

UPDATE <entity name> [[AS] <identification variable>]

SET <update\_statement> {, <update\_statement>}\*

[WHERE <conditional\_expression>]

Cada instrucción UPDATE consta de una expresión de trayectoria de un solo valor, la asignación de los valores correspondientes a la expresión anterior por medio de un SET y una condición. Las opciones de expresión para la sentencia de asignación están ligeramente restringidas en comparación con las expresiones condicionales regulares. El lado derecho de la asignación debe resolverse con una expresión literal y simple que resuelva un tipo básico, una expresión de función, una variable de identificación o un parámetro de entrada. El tipo de resultado de esa expresión debe ser compatible con la ruta de asociación simple o el campo de estado persistente en el lado izquierdo de la asignación.

En el siguiente ejemplo muestra cómo realizar una consulta de actualización al cambiar el salario de los empleados que ganan 55000 dólares a 60000.

UPDATE Employee e

SET e.salary = 60000

WHERE e.salary = 55000

La cláusula WHERE de una instrucción UPDATE funciona igual que una instrucción SELECT

y puede utilizar la variable de identificación definida en la cláusula UPDATE en las expresiones. Una consulta de actualización un poco más compleja sería otorgar un aumento de $5,000 a los empleados que trabajaron en un proyecto en particular:

UPDATE Employee e

SET e.salary = e.salary + 5000

WHERE EXISTS (SELECT p

FROM e.projects p

WHERE p.name = 'Release2')

Se puede modificar más de una propiedad de la entidad de destino con una única instrucción UPDATE. Por ejemplo, la siguiente consulta actualiza la central telefónica para empleados en la ciudad de Ottawa y cambia la terminología del tipo de teléfono a oficina de empresa:

UPDATE Phone p

SET p.number = CONCAT('288', SUBSTRING(p.number, LOCATE('-', p.number),

4)),

p.type = 'Business'

WHERE p.employee.address.city = 'Ottawa' AND

p.type = 'Office'

**DELETE Queries (Consultas de borrado)**

La consulta de eliminación proporciona la misma capacidad que la instrucción SQL DELETE, pero con expresiones condicionales JP QL. La forma de borrar una consulta es la siguiente:

DELETE FROM <entity name> [[AS] <identification variable>]

[WHERE <condition>]

El siguiente ejemplo elimina todos los empleados que no están asignados a un departamento:

DELETE FROM Employee e

WHERE e.department IS NULL

La cláusula WHERE para una instrucción DELETE funciona igual que para una instrucción SELECT. Todas las expresiones condicionales están disponibles para filtrar el conjunto de entidades a eliminar. Si no se proporciona la cláusula WHERE, se eliminan todas las entidades del tipo dado.

También se eliminarán las instancias de las subclases de las entidades que cumplan los criterios de la consulta de eliminación. Sin embargo, las consultas de eliminación no respetan las reglas de cascada. Ninguna otra entidad que no sea el tipo al que se hace referencia en la consulta y sus subclases podrá ser eliminada, incluso si la entidad tiene relaciones con otras entidades con la función de eliminación en cascada habilitado.