AddGroupBy Refactoring

Santiago Ojeda y Martín Esquercia
Grupo 9

Problema

- Necesitamos agregar a una consulta SQL el statement GROUP BY con su respectivo parámetro
  - Ejemplo tenemos la siguiente query
    - `SELECT P.name,P.edad FROM PERSONA P;`
  - Query resultante seteando P.name,P.edad como variables del GROUP BY :
    - `SELECT P.name,P.edad FROM PERSONA P GROUP BY P.name,P.edad;`

- Validaciones correspondientes:
  - Que la palabra GROUP BY no exista dentro de la query
  - Que sea una query válida
  - Que luego de la transformación siga siendo una query válida y que contenga la sentencia Group BY
  - Que los parámetros que se setean al menos tenga alguno contenido en la cláusula SELECT

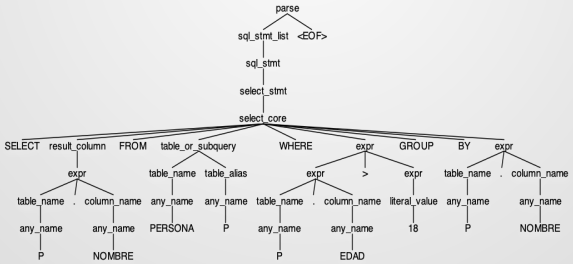
Santiago Ojeda y Martín Esquercia
Grupo 9

Problema

Algunos de los problemas que nos encontramos fueron el no saber a priori como recorrer la consulta sin caer en el recorrido de la query como si fuera un simple String.

Para hacerlo de manera más eficiente, la cátedra nos brindó la opción de hacerlo mediante Antlr y el patrón Visitor, el cual nos permitió ir recorriendo como si fuera un árbol la query con sus expresiones y statements.

- Ejemplo de un árbol creado con el parse de antlr:



query: SELECT P.NOMBRE FROM PERSONA P WHERE P.EDAD >18 GROUP BY P.NOMBRE

Santiago Ojeda y Martín Esquercia
Grupo 9

Solución

Luego que entendimos como funcionaba el patrón y los métodos que necesitábamos utilizar para hacer dicho recorrido, creamos ciertas clases las cuales nos permitieron hacer las validaciones necesarias como así hacer la transformación:

1. **GroupByRefactoring** *(hereda de clase padre Refactoring y contiene los métodos que nos ayudaran para validar las pre,post condiciones y hacer también la transformación, mediante un template method se heredan ciertos métodos abstractos que luego redefinimos con nuestra lógica)*
2. **GetOrderByVisitor** *(Devuelve la cláusula Order by con sus respectivos parámetros de ordenamiento)*
3. **GroupByVisitor** *(Genera la transformación parcial de la query agregando el statement Group by con sus respectivos parámetros previamente seteados)*
4. **ExistsGroupByVisitor** *(Valida que no exista el group by, con este Visitor podemos evitar hacer una transformación cuando ya existe dicha clausula)*
5. **ExistsOrderByVisitor** *(Valida si existe la cláusula Order By)*
6. **ParameterFromSelectVisitor** *(Devuelve la lista de parámetros que contiene el Select en la query)*
7. *Tambien agregamos una clase* **GroupByRefactoringTest** *para hacer las pruebas unitarias correspondientes a los casos que fuimos encontrando importantes.*

Santiago Ojeda y Martín Esquercia
Grupo 9

Conclusión

- Gracias al patrón Visitor y al a gramática provista por ANTLR sobre SQLite pudimos generar una transformación aplicando lo necesario e intentando mantener las responsabilidades en cada clase con un propósito único.
- Visitor es un patrón que permite de una manera sencilla ejecutar acciones en una familia de clases.
- Herramientas utilizadas:
    - Java
    - JUnit v4
    - VSCode
    - Antlrv4
    - Hamcrest

Santiago Ojeda y Martín Esquercia
Grupo 9