

Using Distinct Instead GroupBy Refactor

Fernández Gisela, Martínez Leandro Agustín

Using Distinct Instead Group By Refactoring


Definición:

El siguiente refactoring realiza una manipulación específica en el árbol de análisis gramatical generado por ANTLR para consultas de SQLite, ya que la gramática otorgada por la cátedra es de SQLite.

Este transforma una cláusula GROUP BY por una cláusula DISTINCT eliminando las expresiones que le siguen al GROUP BY.

Esta manipulación del árbol cambiará la sintaxis de la consulta pero los resultados seguirán siendo equivalentes.

```
SELECT departamento,  
id_empleado  
FROM empleados  
WHERE salario > 50000 AND  
(id_empleado > 100 OR  
id_empleado < 50)  
GROUP BY departamento,  
id_empleado;
```



```
SELECT DISTINCT departamento,  
id_empleado  
FROM empleados  
WHERE salario > 50000 AND  
(id_empleado > 100 OR  
id_empleado < 50);
```

Using Distinct Instead Group By Refactoring

- Precondiciones (entrada)
 - La consulta SQLite de ser válida y no tener errores de sintaxis
 - La consulta debe contener GROUP BY
 - La consulta no debe contener DISTINCT
 - La consulta no debe contener funciones de agregación
 - Las columnas que se encuentran en el SELECT deben ser la mismas que se encuentran en el GROUP BY y respetando el mismo orden
- Postcondiciones (salida)
 - La consulta resultante debe ser una consulta SQLite válida y no tener errores de sintaxis
 - La consulta debe contener DISTINCT
 - La consulta no debe contener GROUP BY
 - La consulta no debe tener funciones de agregación

Using Distinct Instead Group By Refactoring

Implementación

- UsingDistinctInsteadGroupByVisitor
 - Reimplementa visitSelect_core(...)
- UsingDistinctInsteadGroupBy
- ConditionVisitor
 - Reimplementa visitSelect_core(...)
 - Reimplementa visitFunction_name(...)
- UsingDistinctInsteadGroupByTest

Conclusiones

- UsingDistinctInsteadGroupByVisitor reimplementa visitSelect_core(...) para modificar el contexto de un subárbol SELECT y sus cláusulas que permite recorrer y modificar sus hijos tales como GROUP, BY, DISTINCT.
- Para analizar las pre y poscondiciones decidimos utilizar una clase visitor que funciona para ambos en lugar de implementar dos clases visitor para cada uno.
- El refactoring funciona para consultas que no contengan cláusulas SELECTs anidadas.
- Decidimos no tener en cuenta el orden de las columnas ya que la información es la misma.
- Los resultados de una consulta SQL antes y después de refactorizar arroja los mismos resultados, lo que quiere decir que ambas consultas son equivalentes.
- Durante la implementación nos resultó más complicado trabajar con el chequeo de las pre condiciones que con la transformación de la consulta en si.