Using Limit with Order By

Carla Renfiges, Nicolas de ROse

Limit with Order By

Definición:

Usando el ORDER BY La cláusula con grandes conjuntos de resultados puede ser ineficiente, ya que requiere que la base de datos ordene todas las filas devueltas por la consulta.

Una buena practica es usando LIMIT con ORDEN BY

Para mejorar el rendimiento cuando se clasifican grandes conjuntos de resultados, es mejor utilizar el LIMIT cláusula con ORDER BY, para que la base de datos sólo ordene las filas necesarias.

```
SELECT nombre, apellido
FROM persona
ORDER BY nombre;
```



```
SELECT nombre, apellido
FROM persona
ORDER BY nombre
LIMIT 10;
```

Limit with Order By

- **Precondiciones (entrada)**
  - **Aplicable unicamente a “consultas”**
  - **La consulta debe ser válida (parsea ok)**
  - **La consulta debe tener un ORDER BY**
  - **La consulta puede tener o no un LIMIT. En caso de no tener: se agrega un limit con valor por defecto en 10 (tambien se puede setear el valor del limit). En caso de tener no se hace la transformacion ya que la consulta esta completa.**

- **Postcondiciones (salida)**
  - **La consulta debe ser válido (parsea ok)**
  - **La consulta final debe tener un LIMIT**

Limit with Order By

- **Implementación**
  - **CheckPreVisitor: devuelve un booleando verificando si existe la clausila ORDER BY en la consulta.**
  - **CheckPreVisitorSelect: verifica si el string recibido es de una consulta SQL.**
  - **LimitWithOrderByVisitor: re-implementamos el metodo visitTerminal(...) donde por cada texto de la hoja del arbol lo guardamos en una variable de tipo StringBuilder, aplicando un espacio cuando sea correcto y cuando se llega a la final de la ocnsulta SQL se le agrega el string LIMIT, el valor de la variable limit y el punto final de la consulta.**
  - **LimitWithOrderBy: subclase de Refactoring.**
  - **LimitWithOrderByTest: los test correspondiente para el refactoring que nos toco.**
  - **VisitorLimit: verifica que la consulta SQL final contenga un LIMIT.**

Limit with Order By

- **Conclusiones**

- **Utilizamos dos visitor para las pre condiciones, uno para verificar si existe la clausula ORDER BY en la consulta mediante el metodo de visitOrder_by_stmt(...), y otro para verificar que el string que recibo sea una consulta SQL mediante el metodo visitSql_stmt(..).**
- **Utilizamos un visitor para la postcondicion, el cual verificaba que la consulta final contenga un LIMIT mediante el metodo visitLimit_stmt(...).**
- **Para la transformacion hicimos un visitor que re-implementa el metodo visitTerminal(...) para recorrer los nodos terminales del arbol y hacer la transformacion donde corresponde.**
- **Una clase para los test del refactoring, teniendo en cuenta cuando llega un string que no es una conuslta SQL, cuando llega una consulta valida (con limit, sin limit y subconsultas), y cuando no es una consulta valida (sin order by).**
- **Y por ultimo la clase LimitWithOrderBy que es subclase de Refactoring he implementa los 3 metodos abstractos de esa clase: checkPreconditions(...), checkPostconditions(...) y transform(...).**