Análisis de operaciones SQL

Insertar compra correcta:

Hibernate: select cliente0_.nif as nif1_0_0_, cliente0_.apellidos as apellidos2_0_0_, cliente0_.ciudad as ciudad3_0_0_, cliente0_.cp as cp4_0_0_, cliente0_.direccion as direccion5_0_0_, cliente0_.nombre as nombre6_0_0_ from HR.Cliente cliente0_ where cliente0_.nif=?

Hibernate: select grupo0_.idgrupo as idgrupo1_3_0_, grupo0_.activo as activo2_3_0_, grupo0_.estilo as estilo3_3_0_, grupo0_.nombre as nombre4_3_0_ from HR.Grupo grupo0_ where grupo0_.idgrupo=?

Hibernate: select concierto0_.idconcierto as idconcierto1_2_, concierto0_.ciudad as ciudad2_2_, concierto0_.fecha as fecha3_2_, concierto0_.IDGRUPO as idgrupo7_2_, concierto0_.nombre as nombre4_2_, concierto0_.precio as precio5_2_, concierto0_.tickets as tickets6_2_ from HR.Concierto concierto0_ where concierto0_.fecha=? and concierto0_.IDGRUPO=?

Hibernate: select max(compra0_.idcompra) as col_0_0_ from HR.Compra compra0_

Hibernate: insert into HR.Compra (NIF, IDCONCIERTO, N_TICKETS, idcompra) values (?, ?, ?, ?)

Hibernate: update HR.Concierto set ciudad=?, fecha=?,IDGRUPO=?,nombre=?, precio=?, tickets=? where idconcierto=?

Se realizan 6 operaciones, 4 selects, 1 insert y 1 update.

Las 4 selects corresponden a los métodos que buscan el cliente, el grupo y el concierto y al método que obtiene el siguiente id de compra. En concreto:

La primera select corresponde al método findbyId, de clienteDAO. Realiza una select buscando el cliente con el NIF que se nos pasa por parámetro.

La segunda select corresponde al método findbyId de grupoDAO. Realiza una select buscando el grupo con el id que se nos pasa por parámetro.

La tercera select corresponde al método findByDateGroup de conciertoDAO. Realiza una select buscando el concierto con fecha y grupo que se nos pasan por parámetro.

La cuarta select corresponde al método nextId de compraDAO. Realiza una select que obtiene el máximo id de compras.

El insert inserta una nueva compra una vez realizadas todas las comprobaciones.

El update modifica el número de tickets del concierto una vez realizadas todas las comprobaciones.

Insertar compra con grupo incorrecto:

Hibernate: select cliente0_.nif as nif1_0_0_,cliente0_.apellidos as apellidos2_0_0_, cliente0_.ciudad as ciudad3_0_0_, cliente0_.cp as cp4_0_0_, cliente0_.direccion as direccion5_0_0_, cliente0_.nombre as nombre6_0_0_ from HR.Cliente cliente0_ where cliente0_.nif=?

Hibernate: select grupo0_.idgrupo as idgrupo1_3_0_, grupo0_.activo as activo2_3_0_, grupo0_.estilo as estilo3_3_0_, grupo0_.nombre as nombre4_3_0_ from HR.Grupo grupo0_ where grupo0_.idgrupo=?

Se realizan 2 operaciones, 2 selects.

La primera select corresponde al método findbyId, de clienteDAO. Realiza una select buscando el cliente con el NIF que se nos pasa por parámetro.

La segunda select corresponde al método findbyId de grupoDAO. Realiza una select buscando el grupo con el id que se nos pasa por parámetro. No lo encuentra, por lo que el método comprar finaliza lanzando un error.

Dado que en el código primero se comprueba si existe el cliente y una vez realizada esa comprobación se comprueba si existe el grupo, en este caso se realizan dos selects. Si intercambiásemos el orden de estas dos operaciones, en este caso solamente se ejecutaría una operación. En cambio, en otros casos que se quiera comprobar si existe el cliente se ejecutarían 2 operaciones. Por lo tanto, sería buena práctica estudiar que comprobación vamos a realizar más veces y realizar primero esa consulta. De esta manera, ahorraremos consultas hacia la base de datos y obtendremos un mejor rendimiento.

Insertar compra con cliente incorrecto:

Hibernate: select cliente0_.nif as nif1_0_0_, cliente0_.apellidos as apellidos2_0_0_, cliente0_.ciudad as ciudad3_0_0_, cliente0_.cp as cp4_0_0_, cliente0_.direccion as direccion5_0_0_, cliente0_.nombre as nombre6_0_0_ from HR.Cliente cliente0_ where cliente0_.nif=?

Se realiza 1 operación, 1 select.

La select corresponde al método findbyId, de clienteDAO. Realiza una select buscando el cliente con el NIF que se nos pasa por parámetro. No lo encuentra, por lo que el método comprar finaliza lanzando un error.

En este caso, al realizar la prueba con un cliente incorrecto, se realiza una única select. Nos hemos “ahorrado” consultas hacia la base de datos al ser la primera comprobación que se realiza. Como se ha mencionado en el apartado anterior, es conveniente realizar un estudio sobre la frecuencia de cada caso que vamos a tener, de esta manera podemos realizar primero la consulta que comprueba el caso más frecuente.

Insertar compra con concierto incorrecto:

Hibernate: select cliente0_.nif as nif1_0_0_, cliente0_.apellidos as apellidos2_0_0_, cliente0_.ciudad as ciudad3_0_0_, cliente0_.cp as cp4_0_0_, cliente0_.direccion as direccion5_0_0_, cliente0_.nombre as nombre6_0_0_ from HR.Cliente cliente0_ where cliente0_.nif=?

Hibernate: select grupo0_.idgrupo as idgrupo1_3_0_, grupo0_.activo as activo2_3_0_, grupo0_.estilo as estilo3_3_0_, grupo0_.nombre as nombre4_3_0_ from HR.Grupo grupo0_ where grupo0_.idgrupo=?

Hibernate: select concierto0_.idconcierto as idconcierto1_2_, concierto0_.ciudad as ciudad2_2_, concierto0_.fecha as fecha3_2_, concierto0_.IDGRUPO as idgrupo7_2_, concierto0_.nombre as nombre4_2_, concierto0_.precio as precio5_2_, concierto0_.tickets as tickets6_2_ from HR.Concierto concierto0_ where concierto0_.fecha=? and concierto0_.IDGRUPO=?

Se realizan 3 operación, 3 selects.

La primera select corresponde al método findbyId, de clienteDAO. Realiza una select buscando el cliente con el NIF que se nos pasa por parámetro.

La segunda select corresponde al método findbyId de grupoDAO. Realiza una select buscando el grupo con el id que se nos pasa por parámetro.

La tercera select corresponde al método findByDateGroup de conciertoDAO. Realiza una select buscando el concierto con fecha y grupo que se nos pasan por parámetro. No lo encuentra, por lo que el método comprar finaliza lanzando un error.

En este caso, al realizar la prueba con un concierto incorrecto, se realizan 3 selects. Por lo tanto, no nos hemos “ahorrado” ninguna operación. Al igual que en los apartados anteriores, es conveniente realizar un estudio sobre las frecuencias de los distintos casos para obtener un mejor rendimiento. Sin embargo, en este caso no tendría sentido obtener el concierto si no hemos comprobado el grupo antes, por lo tanto, como mucho podríamos ahorrarnos la operación que comprueba el cliente.

Insertar compra con tickets incorrectos:

Hibernate: select cliente0_.nif as nif1_0_0_, cliente0_.apellidos as apellidos2_0_0_, cliente0_.ciudad as ciudad3_0_0_, cliente0_.cp as cp4_0_0_, cliente0_.direccion as direccion5_0_0_, cliente0_.nombre as nombre6_0_0_ from HR.Cliente cliente0_ where cliente0_.nif=?

Hibernate: select grupo0_.idgrupo as idgrupo1_3_0_, grupo0_.activo as activo2_3_0_, grupo0_.estilo as estilo3_3_0_, grupo0_.nombre as nombre4_3_0_ from HR.Grupo grupo0_ where grupo0_.idgrupo=?

Hibernate: select concierto0_.idconcierto as idconcierto1_2_, concierto0_.ciudad as ciudad2_2_, concierto0_.fecha as fecha3_2_, concierto0_.IDGRUPO as idgrupo7_2_, concierto0_.nombre as nombre4_2_, concierto0_.precio as precio5_2_, concierto0_.tickets as tickets6_2_ from HR.Concierto concierto0_ where concierto0_.fecha=? and concierto0_.IDGRUPO=?

Se realizan 3 operación, 3 selects.

La primera select corresponde al método findbyId, de clienteDAO. Realiza una select buscando el cliente con el NIF que se nos pasa por parámetro.

La segunda select corresponde al método findbyId de grupoDAO. Realiza una select buscando el grupo con el id que se nos pasa por parámetro.

La tercera select corresponde al método findByDateGroup de conciertoDAO. Realiza una select buscando el concierto con fecha y grupo que se nos pasan por parámetro.

En este caso, se intenta comprar con número de tickets incorrecto, por lo tanto, una vez obtenido el concierto, se lanza un error al ver que no hay tantos tickets disponibles, y no se llega a ejecutar la última select. Al igual que en los casos anteriores, podríamos ahorrarnos operaciones si modificamos el orden de las consultas que buscan el cliente, grupo y concierto. Sin embargo, para obtener el número de tickets disponibles, necesitamos conocer el concierto, por lo tanto, como mucho podríamos ahorrarnos la operación que comprueba el cliente.

Información completa con grafos de entidades:

Select grupo0_.idgrupo as idgrupo1_3_0_, conciertos1_.idconcierto as idconcierto1_2_1_, compras2_.idcompra as idcompra1_1_2_, cliente3_.nif as nif1_0_3_, grupo0_.activo as activo2_3_0_, grupo0_.estilo as estilo3_3_0_, grupo0_.nombre as nombre4_3_0_, conciertos1_.ciudad as ciudad2_2_1_, conciertos1_.fecha as fecha3_2_1_, conciertos1_.IDGRUPO as idgrupo7_2_1_, conciertos1_.nombre as nombre4_2_1_, conciertos1_.precio as precio5_2_1_, conciertos1_.tickets as tickets6_2_1_, conciertos1_.IDGRUPO as idgrupo7_2_0__, conciertos1_.idconcierto as idconcierto1_2_0__, compras2_.NIF as nif3_1_2_, compras2_.IDCONCIERTO as idconcierto4_1_2_, compras2_.N_TICKETS as n_tickets2_1_2_, compras2_.IDCONCIERTO as idconcierto4_1_1__, compras2_.idcompra as idcompra1_1_1__, cliente3_.apellidos as apellidos2_0_3_, cliente3_.ciudad as ciudad3_0_3_, cliente3_.cp as cp4_0_3_, cliente3_.direccion as direccion5_0_3_, cliente3_.nombre as nombre6_0_3_ from HR.Grupo grupo0_ left outer join HR.Concierto conciertos1_ on grupo0_.idgrupo=conciertos1_.IDGRUPO left outer join HR.Compra compras2_ on conciertos1_.idconcierto=compras2_.IDCONCIERTO left outer join HR.Cliente cliente3_ on compras2_.NIF=cliente3_.nif

Se realiza 1 operación, 1 select.

En este caso, se realiza una única select que recupera toda la información vinculada a los grupos, concierto, clientes y compras. Esto se consigue realizar en una única operación dado que hemos utilizado un grafo de entidades. El grafo de entidades carga en una única operación toda la información. De esta manera, evitamos el problema “N+1 consultas”, ahorrándonos un gran número de consultas. Si no lo realizamos con el grafo de entidades, tendríamos una operación por cada entidad o relación. Se realizaría una operación para cargar los grupos, otra operación por cada grupo para cargar los conciertos, otra operación por cada concierto para cargar las compras y otra operación por cada compra para cargar el cliente. Como vemos, esto generaría un gran número de operaciones hacia la base de datos. Por lo tanto, utilizar el grafo de entidades en este caso está justificado. Gracias a esto, generamos y obtenemos la información en una única operación, ahorrándonos un gran número de operaciones que se realizarían si no utilizásemos el grafo de entidades.

Desactivar grupo correcto:

Hibernate: select grupo0_.idgrupo as idgrupo1_3_0_, grupo0_.activo as activo2_3_0_, grupo0_.estilo as estilo3_3_0_, grupo0_.nombre as nombre4_3_0_ from HR.Grupo grupo0_ where grupo0_.idgrupo=?

Hibernate: select concierto0_.idconcierto as idconcierto1_2_,concierto0_.ciudad as ciudad2_2_,concierto0_.fecha as fecha3_2_, concierto0_.IDGRUPO as idgrupo7_2_, concierto0_.nombre as nombre4_2_, concierto0_.precio as precio5_2_, concierto0_.tickets as tickets6_2_ from HR.Concierto concierto0_ where concierto0_.IDGRUPO=?

Hibernate: select compras0_.IDCONCIERTO as idconcierto4_1_0_, compras0_.idcompra as idcompra1_1_0_, compras0_.idcompra as idcompra1_1_1_, compras0_.NIF as nif3_1_1_, compras0_.IDCONCIERTO as idconcierto4_1_1_, compras0_.N_TICKETS as n_tickets2_1_1_, cliente1_.nif as nif1_0_2_, cliente1_.apellidos as apellidos2_0_2_, cliente1_.ciudad as ciudad3_0_2_, cliente1_.cp as cp4_0_2_, cliente1_.direccion as direccion5_0_2_, cliente1_.nombre as nombre6_0_2_ from HR.Compra compras0_ left outer join HR.Cliente cliente1_ on compras0_.NIF=cliente1_.nif where compras0_.IDCONCIERTO=?

Hibernate: update HR.Grupo set activo=?, estilo=?, nombre=? where idgrupo=?

Hibernate: delete from HR.Compra where idcompra=?

Hibernate: delete from HR.Compra where idcompra=?

Hibernate: delete from HR.Compra where idcompra=?

Hibernate: delete from HR.Compra where idcompra=?

Hibernate: delete from HR.Concierto where idconcierto=?

Se realizan 9 operaciones, 3 selects, 1 update y 5 deletes.

La primera select corresponde al método findbyId de grupoDAO. Realiza una select buscando el grupo con el id que se nos pasa por parámetro.

La segunda select corresponde al método findbyGroup de conciertoDAO. Realiza una select buscando los conciertos de ese grupo.

La tercera select obtiene todas las compras relacionadas con un concierto. Estos datos se necesitan para borrar los registros de compra asociados a un concierto del grupo.

El update es el que desactiva el grupo, modifica el campo activo del grupo. Los deletes borran los registros de compras y conciertos relacionados con el grupo. En concreto, se borran 4 registros de compra y un registro de concierto.

La transacción desactivar ha sido modificada para reducir el número de consultas. Antes se generaban 13 consultas, generando 7 selects en vez de los 3 selects que se generan ahora. Esta disminución de consultas se ha obtenido eliminando el método que buscaba las compras de un grupo concreto y utilizando solamente el método que busca conciertos de un grupo concreto. A partir de los conciertos obtenidos con ese método, para cada uno de ellos se obtienen las compras de ese concierto. De esta manera, estos datos se logran conseguir en una única

consulta (la tercera select), ahorrando 4 consultas de más que se realizaban antes de modificar la transacción.

Podemos observar que para cada eliminación de un registro, se genera una operación para la base de datos. Esto es así ya que estamos utilizando el método remove del entity manager. Otra opción sería utilizar bulk deletes para generar una única operación de borrado que eliminase todas las compras o todos los conciertos. Es decir, solamente se generaría un delete para eliminar todas las compras y otro delete para eliminar todos los conciertos. En este caso no se han implementado bulk deletes ya que se está utilizando el patrón de diseño DAO y en los bulk deletes las operaciones se realizan directamente contra la base de datos, no actualizan las instancias de las entidades persistentes en memoria. Los bulk deletes deben utilizarse con mucho cuidado. Por otra parte, pueden resultar interesantes para disminuir el número de operaciones que se generan.

Desactivar grupo incorrecto:

Hibernate: select grupo0_.idgrupo as idgrupo1_3_0_, grupo0_.activo as activo2_3_0_, grupo0_.estilo as estilo3_3_0_, grupo0_.nombre as nombre4_3_0_ from HR.Grupo grupo0_ where grupo0_.idgrupo=?

Se realiza 1 operación, 1 select.

La select corresponde al método findbyId de grupoDAO. Realiza una select buscando el grupo con el id que se nos pasa por parámetro. No lo encuentra, por lo que el método desactivar finaliza lanzando un error.

En este caso, el grupo es incorrecto, y dado que solamente necesitamos comprobar si existe el grupo para seguir ejecutando la transacción, se genera una única operación hacia la base de datos.