Tips De Concurrencia Y Optimización

OCC

El control de concurrencia optimista (en inglés Optimistic concurrency control o OCC) es un método de control de concurrencia que se aplica a sistemas transaccionales, tales como sistemas de gestión de bases de datos relacionales y memoria transaccional de software. El OCC asume que múltiples transacciones se pueden completar frecuentemente sin interferir entre sí. Mientras se ejecutan, las transacciones utilizan recursos de datos sin adquirir bloqueos en esos recursos. Antes de hacer el commit, cada transacción verifica que ninguna otra transacción ha modificado los datos que ha leído. Si la comprobación revela modificaciones en conflicto, la transacción que iba a hacer commit hace un rollback y se puede reiniciar.

El OCC se utiliza generalmente en entornos con baja contención de datos. Cuando los conflictos son poco frecuentes, las transacciones se pueden completar sin el coste de la gestión de bloqueos y sin tener transacciones esperando a que se borren los bloqueos de otras transacciones, dando lugar a un mayor rendimiento que otros métodos de control de concurrencia. Sin embargo, si la contención de recursos de datos es frecuente, el coste de reiniciar las transacciones repetidamente perjudica el rendimiento de manera significativa; comúnmente se piensa que otros métodos de control de concurrencia tienen un mejor rendimiento en estas condiciones. Sin embargo, los métodos basados en bloqueos ("pesimistas") también pueden ofrecer un rendimiento pobre porque los bloqueos pueden limitar drásticamente la concurrencia efectiva incluso cuando se evitan los deadlocks.

Fases del OCC:

Más en concreto, las transacciones del OCC implican estas fases:

Inicio: Grabar un timestamp que marca el inicio de la transacción.
Modificar: Leer los valores de la base de datos y tentativamente escribir cambios.
Validar: Comprobar si otras transacciones han modificado datos que esta transacción ha utilizado (leído o escrito). Esto incluye las transacciones que se han completado con posterioridad al tiempo de inicio de esta transacción y, opcionalmente, las transacciones que aún están activas en el momento de la validación.
Commit/Rollback: Si no hay conflicto, hacer que todos los cambios surtan efecto. Si hay un conflicto, resolverlo, típicamente abortando la transacción, aunque otros sistemas de resolución son posibles. Se debe tener cuidado para evitar un bug time TOCTTOU * , especialmente si esta fase y la anterior no se realizan como una única operación atómica.

Uso en la Web

La naturaleza sin estado de HTTP hace que el bloqueo no sea factible para interfaces de usuarios web. Es común que un usuario empiece a editar un registro y, a continuación, salga sin seguir un enlace de "cancelar" o " cerrar sesión". Si se utilizan bloqueos, otros usuarios que intentan editar el mismo registro deben esperar hasta que el bloqueo del primer usuario termine.

Ejemplo de USO:

```
class OCCActiveRecord extends CActiveRecord{

  public $checksum = null;

  public function afterFind(){

    /* obtiene un checksum de todos los valores de los atributos despues de leer un registro de la db */

    $this->checksum = md5(implode('', $this->getAttributes(false)));
```

```
    parent::afterFind();
  }

    public function rules(){

/* usa un validador existente para asegurar que el registro usa el
mismo checksum sigue estando en la db. Si fue modificado, los
checksums van a ser distintos  */

 return array(
     array('id', 'exist', 'message'=>'Este registro fue modificado
despues de que usted realizo una lectura', 'on'=>'update',
'criteria'=>array('condition'=>'md5(concat('.implode(',',$this-
>attributeNames()).'))=:checksum',
'params'=>array('checksum'=>$this->checksum)))
 );
    }
}

// Solo funciona con MySQL AFAIK.
```

Otra forma de ver esto es usando una extensión: PcBaseArModel.

Es un modelo "Active Record" que agrega la capacidades al CActiveRecord de Yii's

Introducción & Features

Esta clase es usada como clase base para todas las clases Active Records en un proyecto Yii. Agrega algunos features amables y útiles:

- Timestamping automático cuando un registro es creado y actualizado.
- Eliminación y actualización "Seguro" del registro usando "Optimistic Locking".
  - También un método de actualización seguro usando un numero de intentos en caso de fallos.

Metodos convenientes para atributos de registros para trimming. Por ejemplo,el atributo 'title' necesita ser trimmeado ya que es muy largo (es un requisito común ya que los breadcrumb (hilo de Ariadna*) tienen estado real de pantalla limitado).

Puede decir si un objeto esta 'sucio'.

CUANDO USAR ACTIVE RECORD Y CUANDO LOS DAOs DE PHP

Obtener data de la base de datos es usualmente un cuello de botella en la performance de una aplicación WEB. A pesar de que el uso de cacheo puede aliviar el “performance hit”, no resuelve completamente el problema. Cuando la base de datos contiene mucha cantidad de datos y la data cacheada es invalida, obtener los últimos datos puede ser prohibitivamente caro sin uso apropiado y diseño de la base de datos.

El indexado en base de datos es usado inteligentemente en una base de datos. La indexación puede hacer que las consultas SELECT sean más rapidas pero puede alentar los INSERT, UPDATE o DELETE .

Para consultas complejas es recomendable crear una vista y no resolver consultas de codigo PHP y preguntarle al DBMS que parsee repetitivamente.

Ahora, no hay que sobrecargar a Active Record. Active Record es bueno para modelar datos orientada a objetos , de hecho degrada performance debido al hecho que necesita crear uno o varios objetos para representar cada columna del resultado de la consulta. Para datos de aplicaciones intensivos, usar DAOs o APIs de la BD en bajo nivel es una mejor opción.

REFERENCIAS

*TOCTTOU o TOCTOU, (viene de Time of check to time of use), es la clase de bug de software causado por cambios en un sistema

entre el chequeo de una condición (como una credencial de seguridad) y el uso de los resultados de tal chequeo . Ese es un ejemplo de una condición de carrera (Race condition*).

* Race condition : Múltiples procesos se encuentran en condición de carrera si el resultado de los mismos depende del orden de su ejecución. Si los procesos que están en condición de carrera no son correctamente sincronizados, puede producirse una corrupción de datos, que puede ser aprovechada por exploits locales para vulnerar los sistemas. Análogamente, en circuitos electrónicos se da una condición de carrera cuando la salida de un sistema o subsistema depende del orden en que se hayan activado o desactivado sus componentes. Una condición de carrera se da principalmente cuando varios procesos acceden al mismo tiempo y cambian el estado de un recurso compartido (por ejemplo una variable), obteniendo de esta forma un valor no esperado de ese recurso.

* Hilo de Ariadna: Una técnica de navegación usada en muchas interfaces gráficas de usuario, además de páginas web. Aunque su diseño puede adoptar infinidad de variantes, en términos generales consiste en una línea de texto en la que se indica el recorrido seguido y la forma de regresar. Permite que el usuario conozca la ruta de su ubicación en directorios y subdirectorios, y navegue a través de ella.