TP - Concurrencia-Performance-Usuarios : 24/06/2010

Introducción a las bases de datos

UNQ

1. Concurrencia y transacciones

Lo que sigue es parte del esquema de la BD que se armó para una aplicación de reserva de herramientas en un pañol. Vamos a usar MySQL.

`herramienta<idHerramienta, tipo, marca, nomPersonaQueLaReservo>`
`persona<nomPersona, fnac>`

Si una herramienta no está reservada, el `nomPersonaQueLaReservo` está en `null`.

Las herramientas siempre se reservan de a pares (p.ej. una lijadora y una agujereadora, un martillo y un punzón, un martillo y una agujereadora, etc.), por esto se armó el siguiente stored procedure

```
reservar(idHerr1 int, idHerr2 int, nomPersonaQueQuiereReservar varchar(45))
begin
  start transaction;
  declare nomPersonaRes1, nomPersonaRes2 varchar(45);
  select nomPersonaQueLaReservo from herramienta where idHerramienta = idHerr1
    into nomPersonaRes1;
  select nomPersonaQueLaReservo from herramienta where idHerramienta = idHerr2
    into nomPersonaRes2;
  if (nomPersonaRes1 is not null and nomPersonaRes2 is not null) then
    update herramienta set nomPersonaQueLaReservo = nomPersonaQueQuiereReservar
     where idHerramienta = idHerr1;
    update herramienta set nomPersonaQueLaReservo = nomPersonaQueQuiereReservar
     where idHerramienta = idHerr2;
  end if;
  commit;
end
```

A partir de esta situación

1. Indicar un intercalado en el que el valor que lee el primer `select` para alguna de las transacciones incluidas es distinto según si elegimos nivel de aislamiento `READ COMMITTED` o `REPEATABLE READ`, inventando los datos que hagan falta.

2. Hay una posible anomalía de concurrencia. Indicar de qué se trata, mostrarla mediante un intercalado inventando los datos que hagan falta, e indicar una forma sencilla de evitarla modificando el stored procedure.

3. Ya sea con el stored procedure original o con el modificado, pueden darse deadlocks. Armar un intercalado que entre en deadlock en alguna de las dos versiones (original o modificada), indicando cuál se está tomando.

4. Cambiar el sp para evitar la posibilidad de deadlock. **Ayuda** considerar que si una sentencia SQL debe lockear varias filas, lockea todas o ninguna.
   En este ítem "se te tiene que ocurrir", si no sale, ¡¡pasar al ejercicio siguiente!!.

2. BD para los tres ejercicios que siguen

Los tres ejercicios que siguen (que es lo que queda de parcial) se refieren todos a la siguiente BD armada para una sala enorme de bingos en la que también se reciben apuestas para carreras de caballos del mundo entero, tanto presenciales, telefónicas o por Internet.

```
cajero<idCajero, nomCajero>
jugadaBingo<idJugada, fecha, hora>
numeroQueSalioBingo<idJugada, nroOrden, numeroQueSalio>
tarjeta<idTarjeta, idJugada, importe, idCajero>
numeroEnTarjeta<idTarjeta, numero>
caballo<nomCaballo, importeMaximoApuestasPorCarrera>
carrera<nomCarrera, hipodromo, fecha, hora>
participanteEnCarrera<nomCarrera, nomCaballo, numero>
apuesta<nomCarrera, nomCaballo, nomApostador, importe, idCajero>
```

3. Programación, concurrencia y transacciones

El stored procedure para agregar una apuesta es muy sencillo

```
apostar(laCarrera varchar(45), elCaballo varchar(45), elApostador varchar(45),
        elImporte decimal(12,2), elCajero int)
begin
  insert into apuesta(nomCarrera, nomCaballo, nomApostador, importe, idCajero)
         values(laCarrera, elCaballo, elApostador, elImporte, elCajero);
end
```

Dada esta situación

1. Como está el sp, no hay ninguna validación que impida superar el importe máximo de apuestas por carrera de un caballo. Agregar la validación al sp apostar. Por ahora no hacer ningún lockeo.
2. Si la validación que hicieron es correcta, no se podrá superar el límite de apuestas ... salvo anomalías de concurrencia. Mostrar un intercalado que provoque que se supere el límite, inventando los datos que hagan falta.
3. Agregar lo que haga falta para evitar las anomalías de concurrencia detectadas, suponiendo que el motor de BD que estamos usando lockea por conjunto de filas y no por condición. Mostrar que el schedule que provocaba la anomalía ya no es posible. Ayuda: vale agregar operaciones al sp que sirvan solamente para lockear.

4. Performance

Nos pasan la siguiente información sobre volumen de datos estimado

- En la casa trabajan 300 cajeros.
- Se manejan apuestas para carreras de 10000 hipódromos en todo el mundo.
- Cada hipódromo hace 40 reuniones por año, en cada reunión se hacen 10 carreras.
- Para cada carrera se recibe un promedio de 30 apuestas.

Nos dicen que la operación más habitual que se va a hacer sobre esta base es la siguiente consulta, que es el detalle de las apuestas hechas por un apostador

```
select nomCarrera, fecha, hora, nomCaballo, idCajero, nomCajero, importe
from apuesta
  natural join carrera
  natural join cajero
where nomApostador = XXX
order by fecha, hora
```

Se pide

1. indicar las tablas que intervienen en esta consulta, de mayor a menor según el volumen estimado, incluyendo el volumen anual de cada una.
2. indicar qué índices pueden brindar un aceleramiento significativo sobre esta consulta, si te parece que alguno brinda más aceleramiento que otro indicarlo.
   Incluir todos los índices que creas que son convenientes, incluso aquellos que el motor de MySQL hace solo.

5. Usuarios

5.1. Análisis de usuarios existentes

Tenemos los siguientes usuarios, para cada uno se indica qué permisos tiene

- haras: insert, update, delete, select sobre tabla caballo; select sobre tabla carrera
- encargado bingo: insert, delete, update, select sobre tablas jugadaBingo, numeroQueSalioBingo, tarjeta, numeroEnTarjeta.
- cajero: insert, delete, update, select sobre tabla apuesta.

Las operaciones que tiene que hacer cada usuario son:

- haras: cargar caballos
- encargado bingo: cargar las jugadas, registrar los números que salieron, saber qué tarjetas hay para una jugada y su composición.
- cajero: cargar apuestas y tarjetas de bingo, incluyendo (claro) qué números van en cada una.

En todos los casos donde se dice “cargar” o “registrar” se incluye: agregar, corregir errores, borrar, ver qué se cargó.
Indicar qué permisos le faltan o sobran a cada usuario.

5.2. Creación de usuarios nuevos

Hay que crear el usuario “encargado recaudación” que tiene que poder hacer estas operaciones:

- cargar cajeros (misma aclaración sobre qué entendemos por “cargar” que en el punto anterior).
- ver la recaudación de cada cajero, teniendo en cuenta tanto apuestas como bingo. Lo que tiene que poder ver es exactamente (ni más ni menos) tres columnas para cada operación: id cajero, fecha e importe.
- Modificar el máximo de apuestas por día permitido para un caballo.

Indicar qué permisos hay que darle a este usuario, definiendo las vistas que se necesite crear.