Control de Replicación

BD Replicada=Todo o parte de ella es copiada y las copias son mantenidas en N sitios[1].
Problema=Mantener consistente las copias.
+Disponibilidad
+Confiabilidad
+Read performance
-Write performance

Control de Replicación

Clasificación según si las copias son o no idénticas todo el t:

* Replicación Sincrónica
  - Copias sincronizadas todo el t
  - Oculta items lockeados o que estan siendo modificados
  - Todas las T's ven los mismos valores

* Replicación Asincrónica
  - Copias no sincronizadas todo el t
  - Pueden verse <> valores

Replicación Sincrónica (RS)

Todo cambio será aplicado en TODAS las copias.

Ej. usando 2PC, tabla EMP en sitio 1,2,3

```
Begin T1:
    Update EMP set Sal=Sal*1.05 where Eno = 100;
End T1;
```

T1 es cambiada a:

```
Begin T1:
    Begin T11:
        Update EMP set Sal=Sal*1.05 where Eno=100;
    End T11;
    Begin T12:
        Update EMP set Sal=Sal*1.05 where Eno=100;
    End T12;
    Begin T13:
        Update EMP set Sal=Sal*1.05 where Eno=100;
    End T13;
End T1;
```

Corre en Sitio1

Corre en Sitio2

Corre en Sitio3

Replicación Sincrónica (RS)

Una vez que T1 hace commit, las 3 copias son actualizadas Ok, mientras tanto, el salario del Eno 100 esta lockeado y las otras T's no pueden ver un valor inconsistente del salario.
El lock se quita cuando T1 hizo commit y revela el nuevo salario del Eno 100 en TODAS las copias.

Problema:

-Performance

-Caída de Sitio durante T1 → bloqueo de T1 hasta que el Sitio este on-line → aplicable en arquitectura con servidores primarios y secundarios

Replicación Asincrónica (RA)

Puede haber copias no actualizadas con respecto a las T's que estan corriendo → proceso de sincronización → dependiente de la aplicación (cómo y cuándo sincronizar)

<> Enfoques:

- Copia Primaria (Primary Copy / Store & Forward)
- N Sitios Primarios
- N Sitios Pares (peers)

RA – Copia Primaria

Sitio primario → Publicador
Sitio secundario → Suscriptor
T's corren en primario → decide orden de serialización. T's encoladas en secundario en forma batch, DBA decide frecuencia, puede hacerse rollout de copias ya actualizadas.
Fallo en primario=catástrofe
Suele utilizarse un sitio backup (hot standby) o servidor primario “flotante”.

RA – N Sitios Primarios

N sitios primarios
1 sitio secundario
T's aplicadas a primarios luego de haber arribado a sitio secundario.
T's arribadas a sitios primarios se encolan en sitio secundario.
Sitio secundario genera orden de serialización y aplicación en TODOS los sitios primarios

RA – N Sitios Pares (Peer)

Todos los sitios son pares.
La T se aplica donde arribó. Hace replicación simétrica en donde todas las copias son iguales.
Provee Software para comparar copias y herramienta de sincronización e identificación de diferencias entre copias.

Alg. Repl. Master-Slave (C. Ellis)

1 Alg. Maestro
    Corre en 1 sitio
    Responsable de la detección de conflictos
N Alg. Esclavos
    Corre en N sitios

Consta de 2 fases:
    1) Aceptación/Rechazo T
    2) Aplicación T

Master-Slave Fase 1 Centralizado

1 T entra en 1 sitio → TM local envía petición al esclavo para que ejecute T. Esclavo pregunta a maestro si tiene Ok para correr T. Maestro mantiene cola con todas las peticiones pendientes que autorizó y aun no estan committed.
Si nueva petición no esta en conflicto con las T's pendientes, maestro envia “ACK+”, sino envia “ACK-”
Si esclavo recibe “ACK+” → comienza fase de aplicación de T
Si esclavo recibe “ACK-” → rechaza T

Master-Slave Fase 2 Centralizado

Cuando esclavo (1) recibe “ACK+” de maestro, hace broadcast enviado “UPD” a todos los otros esclavos. Los esclavos corren la T y envian “ACK” a (1). (1) espera hasta recibir todos los “ACK”, luego (1) envia “DONE” al maestro y al TM notificando que la T ha sido aplicada Ok. El maestro quita a T de la cola de pendientes.

Sistema de votación centralizado → sobrecarga de maestro → cuello de botella

Alg. Votación Distribuido

1 T debe ser aceptada antes de ser aplicada.
Todos los sitios actúan como pares, no hay maestro.
Todos tienen que tener todos los Ok's para correr la T, cualquier objeción de cualquier sitio, causará el rechazo de la T. La aprobación debe ser unánime, sino la T se rechaza.

Master-Slave Fase 1 Distribuido

Cada sitio espera el Ok de todos los demás.
Cada sitio usa unas reglas de votación basado en prioridades para resolver conflictos. Hay 3 casos de conflictos, para una Ti que todo esclavo debe resolver:
Caso 1: El sitio esclavo no esta trabajando sobre 1 T que haya sido iniciada en el sitio → el esclavo vota Ok para aplicar Ti.
Caso 2: El sitio esclavo esta trabajando en Tj que fue iniciada en su sitio y Tj no esta en conflicto con Ti → el esclavo vota Ok para aplicar Ti.
Caso 3: El sitio esclavo esta trabajando en Tj y Tj esta en conflicto con Ti. Una debe ser rechazada / abortada. Si usa la edad de la T como prioridad → la más nueva es rechazada.

Master-Slave Fase 2 Distribuido

Se puede iniciar la fase 2 cuando se recibe el Ok de todos los esclavos. En esta fase, el esclavo aplica la T a todas las copias enviando un nuevo broadcast para que los otros esclavos corran la T. Los esclavos que reciben el broadcast corren la T y envían ACK al emisor. El esclavo emisor espera hasta recibir todos los ACK de todos los esclavos, cuando ello ocurre, el esclavo emisor notifica al TM que la T ha sido aplicada Ok. Durante esta fase, el esclavo emisor pospone la votación de nuevas T's hasta que se ha completado la aplicación de su propia T.

Alg. De Consenso Mayoritario

En el caso anterior, puede ocurrir que más de un esclavo detecte el mismo conflicto, lo cual no es necesario. Ahora, cuando 1 T entra, ésta pueda recolectar la mitad + 1 de los Ok's de los otros esclavos antes de ser aplicada.
Cuando 2 T's están en conflicto, el sitio que lo detecta debe resolverlo.
Ciclo de vida de la T = 2 fases: idem anterior.
Se utilizan ts para T y para los items de datos.
ts(Ti) = Site ID+local clock
ts(X) = ts de la ultima Ti que lo actualizó.
Ti colecta todos los items que necesita leer y escribir, junto con sus respectivos ts's y ello se pasa a una lista de actualización (update list, UL) y la votación comienza.
La UL lleva la cuenta de los votos Ok que va colectando.

Alg. De Consenso Mayoritario
Regla de Votación

Cuando 1 sitio recibe una UL de Ti para votación, compara los ts's de todos los items Vs ts's de todos los items de su copia local, luego el sitio aplicará las siguientes reglas:

- Si el ts de cualquier item es <> ts local → vota REJECT → la copia local ha sido escrita por otra Tj

- Si el ts de todos los items es = ts local, pero hay conflicto con una Tj de mayor prioridad (ts(Tj) > ts(Ti)) → vota REJECT → impide que una T de menor prioridad corra mientras una T de alta prioridad esta corriendo

- Si ts de todos los items es = ts local, pero hay conflicto con una Tj de menor prioridad (ts(Tj) < Ts(Ti)) → vota DEFER (posponer) y pone a Ti en su lista de pospuestas y hace forward de la Ti y su UL a otro sitio.

-Sino el sitio vota Ok para Ti, incrementa los Ok's en UL, agrega Ti a su lista de Ti pendientes.

Alg. De Consenso Mayoritario Regla de Resolución

Si un sitio vota Ok y la mayoría es alcanzada → el sitio acepta Ti y hace broadcast de la aceptación a todos los sitios.
Si un sitio vota REJECT → el sitio hace broadcast de REJECT a todos los sitios.
Sino hace forward de la Ti.

Alg. De Consenso Mayoritario
Regla de Aplicación de T

Una vez que se ha tomado la decisión de aceptar o rechazar la T, la decisión fue por broadcast a todos los sitios.
Una vez que el sitio recibe el mensaje, deberá hacer:
-Si la Ti es aceptada → el sitio quita a Ti de su cola de pendientes, aplica Ti y rechaza todas las Tj conflictivas que fueron demoradas a causa de Ti.
-Si la Ti es rechazada → el sitio quita a Ti de su cola de pendientes y reconsidera todas las Tj conflictivas que fueron demoradas a causa de la Ti.

Bibliografía

- [1] Saeed K. Rahimi, Frank S. Haug, "Distributed Database Managment Systems: A Practical Approach", Wiley-IEEE Computer Society Press., 2010, Chapter 7 Replication Control