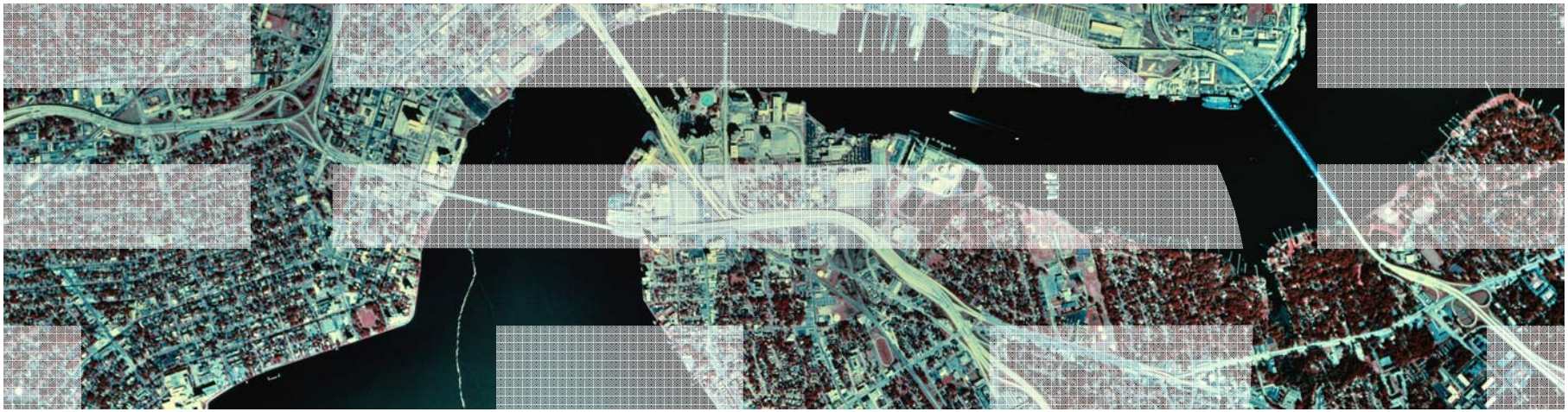
Charla con los Expertos Informix Flexible Grid

Edgar Sanchez, Ingeniero de Soporte Avanzado de Informix




Informix
Flexible Grid

En resumen, cuales son los componentes de alta disponibilidad que ofrece Informix?

- Cluster, conocido tambien como MACH11
  - SDS
  - HDR secundario
  - RSS
- Enterprise Replication
  - Flexible Grid
- Connection Manager
  - Con cluster
  - Con ER
  - Mejoras en CSDK 3.70.xC3
- ifxclone
- cdr start sec2er

IBM Informix Cluster o “MACH11”

- El clúster Informix es un conjunto de servidores Informix que trabajan juntos de una manera estrechamente coordinada para ofrecer la funcionalidad de OLTP y OLAP. Un clúster se compone de un primario y uno o varios servidores secundarios. Hay tres tipos de secundarios:

Tipo	Número máximo	Introducido inicialmente en la versión
Shared Disk Secondary (SDS)	ilimitado	11.10
HDR	1	6.0
Remote Standalone Secondary (RSS)	ilimitado	11.10

IBM Informix Cluster o “MACH11” (cont.)

- El número, tipo y rol de servidores secundarios de un clúster puede cambiar con el tiempo (por ejemplo, RSS puede convertirse en el HDR, el HDR puede convertirse en el primario).
- Todos los servidores de un clúster deben ejecutar la misma versión de Informix, p.e., 11.50.FC8 para Linux x86 (64 bits). Esto da como resultado que el Sistema Operativo de cada secundario tenga exactamente o casi la misma versión que el Primario
- Algunos parámetros del ONCONFIG deben ser igual para todos los servidores del clúster, pero algunos pueden ser diferentes (p.e., número de CPU VPs , tamaño del buffer pool,etc).

IBM Informix Cluster o “MACH11” (cont.)

- Todos los servidores secundarios reciben una copia completa de todos los datos “logged” en el primario (en el caso de servidores SDS estos comparten los mismos espacios en disco que el primario). Datos Unlogged (p.e. raw table) no son soportados.
- Cada secundario puede ser configurado individualmente para acceso de sólo lectura o de actualizaciones. En Informix 11.50 las actualizaciones estaban limitadas a solo DML. En 11.70 las actualizaciones ahora incluyen la mayoría de las sentencias DDL. Todas las actualizaciones se coordinan con el primario mediante un algoritmo de bloqueo optimista.

IBM Informix Cluster o “MACH11” (cont.)

- Un servidor secundario soporta muchos de los niveles de aislamiento que el primario, sin embargo Repeatable Read no está soportado.
- En el evento de un fallo que afecta el primario, un secundario puede ser promovido para ser el nuevo primario, y el clúster puede continuar sus actividades.
- La detección de fallas y promoción puede ocurrir manual o automáticamente por medio de Connection Manager.

IBM Informix Cluster or “MACH11” (cont.)

- Un cluster puede ser configurado para que no tolere ninguna pérdida de datos en el caso de un fallo, esto requiere Bases de Datos con “unbuffered logging” entre otras cosas.
- El cluster puede ser configurado para perder algunos datos potencialmente (Bases de Datos con “Buffered Logging”), cuando se configura de esta forma se puede tener un aumento de rendimiento.

Servidores Secundarios SDS

- El servidor de disco compartido secundario (SDS) reside en una máquina separada de la primaria, pero tiene acceso al mismo espacio en disco. Compartir el disco se consigue normalmente mediante el uso de un sistema de archivos de clúster (por ejemplo, IBM General Parallel File System) o un administrador concurrente de volumen lógico para dispositivos ‘raw’.
- Sólo el primario escribe en el disco; un SDS sólo lee del disco. En el caso de un fallo en el primario, es típicamente mejor promover un SDS a ser el nuevo primario porque un SDS es más cercano al estado del primario antes que este encuentre el problema.

Servidores Secundarios SDS (cont.)

- Por esta razón, es una buena práctica para la tecnología SDS soportar I/O Fencing, por ejemplo, la capacidad de prevenir escrituras en el disco de un participante determinado. Los unicos datos que sólo el primario tiene que el SDS no tiene en esencia son los logical log buffers unflushed.
- Las principales ventajas de un SDS son sus costos más bajos de hardware, es decir no se compra el disco que se requiere y su tiempo de configuración es mas rápido sin la necesidad de restaurar un Backup.

Servidores Secundarios HDR

- El tipo de servidor secundario HDR ha estado presente desde la versión de Informix 6. No comparte ningún hardware con el primario. A medida que el primario procesa transacciones, este envía la información de los logical logs al HDR por la red
- El HDR puede ser configurado para ser sincrónico o asincrónico con el primario.

  - Sincrónico significa que cada vez que un logical log buffer es escrito a disco en el primario este también es enviado al HDR.

  - Asincrónico significa que puede haber un retraso limitado (en segundos) entre el flush del primario y el envío al HDR.

Servidores Secundarios HDR (cont.)

- El HDR por su diseño no se le permite quedarse demasiado atrás de la posición del logical log del primario. De hecho, cada checkpoint en el primario no se completará hasta que la información correspondiente al logical log ha sido enviado al HDR (o el primario considere que el HDR esta en un estado de desconexión).
- En el caso de un fallo en el primario si no hay un SDS disponible, el HDR es el mejor candidato para convertirse en el nuevo primario.

Servidores Secundarios RSS

- Un servidor de RSS es similar a un HDR con algunos requisitos relajados. Al igual que HDR, un RSS no comparte hardware con el primario, se debe restaurar del backup para inicializarlo, y recibe los registros lógicos por la red.
- A diferencia de HDR, a un RSS se le permite estar muy por detrás de la posición actual del logical log del primario. Esto se puede configurar a través de RSS_FLOW_CONTROL. Esta característica le permite a un RSS ser ubicado más alejado del primario que un HDR, con un enlace de red con mayor latencia y el ancho de banda en ocasiones menor también.

Servidores Secundarios RSS (cont.)

- RSS tiene algunas capacidades adicionales únicas. A través de DELAY_APPLY se puede decir al RSS de retrasar la aplicación de los logical logs que ha recibido por un período de tiempo determinado (por ejemplo, 4 horas). Por medio de STOP_APPLY la aplicación de los logical logs puede ser detenida, lo que permite una copia de seguridad externa y descarga de datos.
- En el caso de un fallo en el primario, un RSS primero puede ser promovido a HDR y después al primario. Sin embargo, si el RSS no ha recibido todos los logicos logs que el primario escribió a disco (debido al desfase permitido), entonces las transacciones ‘committed’ se perderan.

Enterprise Replication

- Enterprise Replication (ER) es una tecnología de replicación en Informix que ha existido desde la versión 7.
- En resumen, ER utiliza replicación asincrónica, basada en transacciones.
- ER se compone de dos o más nodos Informix y una o más replicas.
- Un nodo es un servidor Informix independiente o en un clúster.

Enterprise Replication (cont.)

- Una réplica es un convenio entre los nodos, en el cual los datos en un conjunto dado de columnas, de una tabla dada, en una base de datos dada, en un nodo dado, se envía a un segundo nodo y se aplica a un conjunto dado de columnas, de una tabla dada, en una determinada base de datos.
- Después que una réplica está definida e iniciada, una transacción commited que cambia los datos sobre el primer nodo, será enviada a todos los demás nodos participantes en una forma asíncronica , de forma rápida, y a continuación es committed.

Enterprise Replication (cont.)

- ER ofrece una gran variedad de flexibilidad que incluye:
  - Número de nodos en un dominio: de 2 a cientos
  - Número de nodos que participan en una réplica
  - Réplica una sola celda de datos (es decir, una columna de una fila) a toda la tabla
  - Definir las réplicas para una tabla hasta todas las tablas de las bases de datos en el nodo

    Determinar si un nodo recibe o envía y recibe datos de una réplica
  - Un dominio puede incluir nodos de diferentes versiones de Informix (por ejemplo, un nodo es de 11,7, otro nodo es 11,5)
  - Un dominio puede incluir nodos en diferentes plataformas (por ejemplo, un nodo es AIX en Power, otro es Linux en Intel)

Enterprise Replication (cont.)

- ER tiene una variedad de usos que incluyen:
  - Replicar datos de un sistema de producción a un sistema donde se pueden ejecutar reportes
  - La consolidación de la información desde varias ubicaciones a un sitio central
  - La difusión de información desde un sitio central a varios lugares remotos
  - Particionar la carga de trabajo del cliente entre varios servidores
  - Proporcionar un sitio de recuperación ante desastres

Connection Manager

- Connection Manager (CM) es el ejecutable *oncmsm* suministrado con el Cliente SDK. En resumen lo que hace CM es que supervisa continuamente un determinado conjunto de servidores de Informix para conocer el estado online/offline, carga de trabajo, etc, y redirecciona una conexión de cliente al servidor apropiado en función de un conjunto de reglas.
- Estas reglas se pueden modificar mientras el CM está en ejecucion. Para que un cliente pueda aprovechar CM, el SQLHOSTS o información de conexión debe ser actualizado con el host y el puerto en el cual CM escucha, y no en el servidor.

Connection Manager (cont.)

- Desde la perspectiva del cliente, CM sólo está implicado cuando el cliente hace una conexión, aunque el CM puede ser configurado para ser proxy o enrutar todo el tráfico del cliente a través de sí mismo.
- El uso de CM para redirigir los clientes permite que la carga de trabajo sea dividida entre los servidores y permite a un servidor ser puesto fuera de línea - por ejemplo, para el mantenimiento o actualización - transparente para los clientes.

Connection Manager con un Cluster

- Desde su comienzo, el CM ha apoyado el trabajo con un clúster. CM está configurado con información acerca de cómo conectar por lo menos el clúster primario, se conecta, aprende acerca de los secundarios en el clúster, se conecta a ellos, y luego monitorea continuamente el clúster. Si un servidor queda fuera de línea el CM no dirigirá las nuevas conexiones allí. Si la carga de trabajo de un servidor es alta, y el CM a sido ha configurado para ello, puede dirigir las nuevas conexiones a otros servidores.
- CM también se puede configurar para arbitrar failover. Si el principal es inalcanzable, por ejemplo, debido a una caída, un fallo de hardware, o interrupción de la red, el CM identificara los servidores secundarios que sobreviven, elige al mejor candidato, e instruye este servidor para hacer la transición a ser el primario del cluster. Es una buena práctica el utilizar al menos dos CMs, ya que teniendo un solo CM representa un punto único de falla.

Connection Manager con ER

- A partir de Client SDK 3.70.xC1, CM se puede utilizar para redirigir clientes entre los nodos de ER.
- Primero, el CM se configura con el nombre de un conjunto de réplicas o Flexible Grid y una lista de los nodos participantes.
- A continuación, el CM se conecta a cada nodo y continuamente monitorea el estado en línea/fuera de línea, carga de trabajo, número de fallos, cuando se aplican transacciones replicadas, y la latencia de replicación extremo a extremo.
- Estos dos últimos puntos requieren que la Calidad de Datos esté activada.
- CM toma todos los factores basados en toda esta información para tomar la decisión de hacia dónde dirigir el próximo cliente. Calidad de los datos es una característica solo de nodos 11.70 o superiores. Carga de trabajo es soportada en nodos con 11.10 o superiores.

Que es el Informix Flexible Grid?

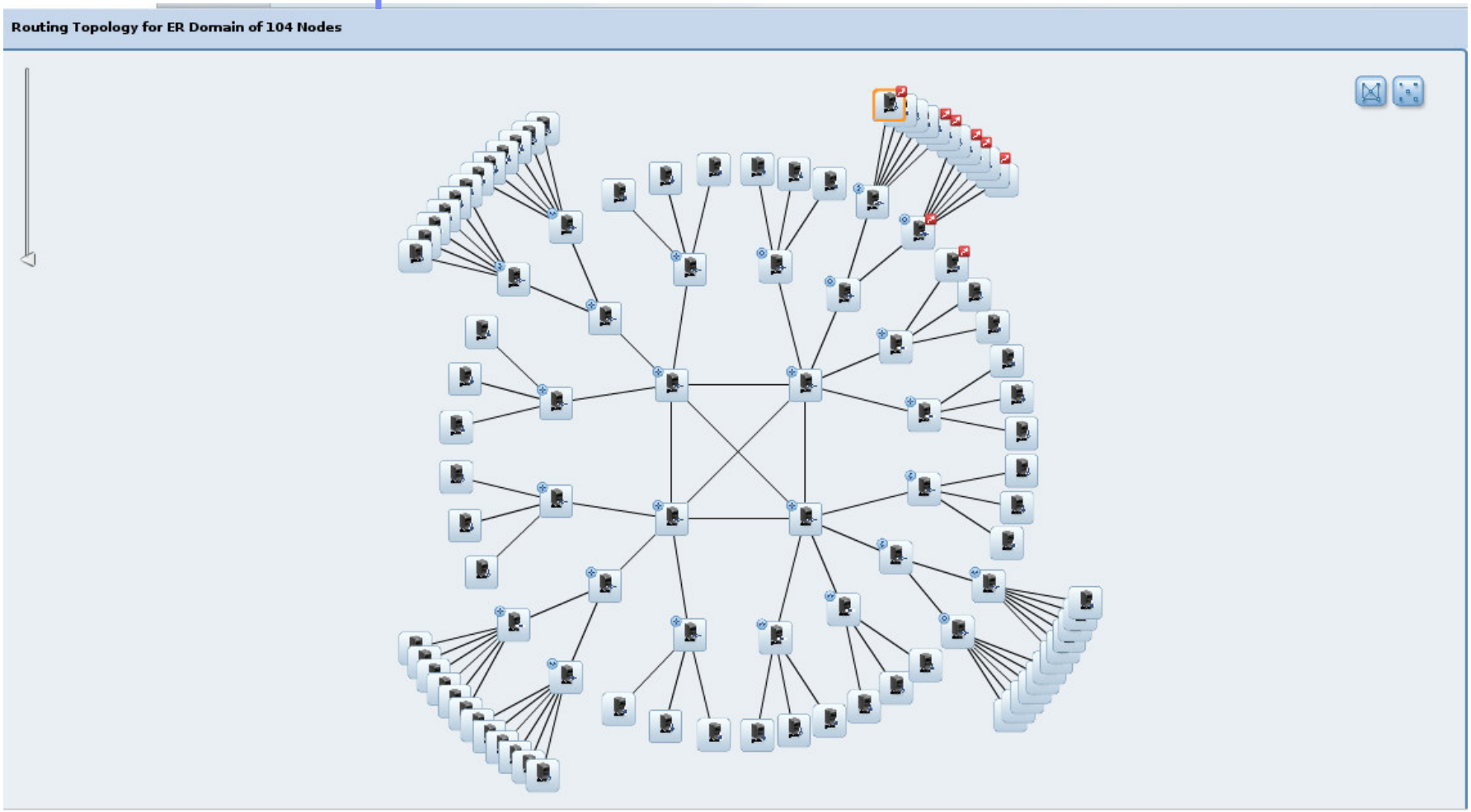
- El Flexible Grid es una solución escalable de alta disponibilidad multi-nodo en un ámbito global
- El Flexible Grid proporciona un medio de replicar DDLs en varios nodos

  CREATE TABLE, ALTER TABLE, CREATE INDEX, CREATE PROCEDURE ...
- El Flexible Grid proporciona un medio de replicar la ejecución de una sentencia en lugar de sólo los resultados de la ejecución

Que es el Flexible Grid (cont.)

- El Flexible Grid proporciona la capacidad de crear automáticamente la replicación ER como parte de una sentencia DDL como CREATE TABLE, ALTER TABLE o DROP TABLE. No se requiere de administración adicional.
- El Connection Manager puede ser utilizado con el Flexible Grid
- Nuevo con la versión de Informix 11.70

Que tan simple es escalar con Flexible Grid?



Representación en OAT de un cliente que participo del BETA de Informix Flexible Grid

Requisitos para el GRID

- Enterprise Replication debe estar corriendo

- Los servidores deben estar en Informix 11.70 (Panther)
  - Pueden existir servidores pre-Informix 11.70 dentro del dominio de ER, pero no pueden ser parte del GRID

Que es necesario para crear un Grid?

1. Crear ER
2. Definir el Grid
   Definir los nodos dentro del Grid

   cdr define grid <grid name> list of nodes
3. Autorizar el Grid
   Definir los nodos y usuarios autorizados para ejecutar commands del Grid

   cdr enable grid –grid=<grid_name> --user=<name>

Realizando las operaciones de DDL GRID

Con el fin de realizar operaciones DDL a nivel del Grid,primero deberá establecer el contexto del Grid en un nodo del Grid y como usuario habilitado del Grid. Esto se hace mediante la ejecución del procedimiento incorporado en :

ifx_grid_connect(<gridName>, <autoRegister>, <tag>);

Donde autoRegister se establece en 1 si quiere registrar el DDL con ER o cero si no desea el registro automático. "Tag" es una etiqueta opcional asociada con cualquier comando Grid. El tag se puede utilizar para hacer más fácil monitorear el éxito/fracaso de las operaciones del Grid.

Esto se puede realizar desde desde el Stored Procedure ***sysdbopen().***

Ejemplo de la propagación de DDL con auto-registro

```
execute procedure ifx_grid_connect('grid1', 1, 'tag1');
```

Podría ser en sysdbopen ()

```
create database tstdb with log;
create table tab1 (
        col1    int primary key,
        col2    int,
        col3    char(20)) lock mode row;
create index idx1 on tab1 (col2);
create procedure loadtab1(maxnum int)
define tnum int;
for tnum = 1 to maxnum
        insert into tab1 values
                (tnum, tnum * tnum, 'mydata');
end for:
end procedure;
```

Se ejecuta en todos los nodos dentro del "grid1" del GRID y la tabla tab1 será una tabla replicada

Realizando las operaciones de DDL GRID (cont.)

- Las operacion de DDL se llevaran a cabo en los nodos destino dentro del Grid
  - Dentro de la misma base de datos como en el origen
  - Por el mismo uso como estaba en el origen
  - Utilizando el mismo locale como en el origen

Comentarios

Como se mencionó, la ejecución de ifx_grid_connect podría ser incluida en el procedimiento almacenado sysdbopen().

Esto eliminaría la necesidad de tener que cambiar todos los scripts existentes para tomar ventaja del ambiente GRID. No hay ningún problema con el contexto que queda en el GRID ya que lo único que afecta es la propagación automática de registro DDL con ER.

Utilice ifx_grid_disconnect () para removerse de un contexto GRID.

Caso practico 1

Susana tiene un dominio de ER de unos 20 nodos. Ella tiene que purgar unos 4 millones de registros de una tabla replicada. Ella podía confiar en la funcionalidad básica de ER para hacer esto, pero ella está preocupada por el impacto de la red que tal operación tendría.

Caso practico 2

Jaime tiene que añadir 5 filas a una tabla en cada uno de los servidores de los cuales él es responsable. Esta tabla no se replica y contiene la clasificación de puestos para algunas nueva posiciones que su empresa está creando. Hay varios cientos de servidores en los que esto tiene que ser hecho . Le preocupa que pueda tardar una eternidad para completar esta tarea.

Case practico 3

Se le ha pedido a Samuel agregar una nueva tabla en las bases de datos en las cuales el es responsable. Debido al tamaño potencial de la tabla, el quiere aislar todo en un dbspace para sí mismo. El único problema es que él necesita crear en primer lugar ese dbspace en cada uno de los 500 nodos en los cuales él es responsable.

Ejecución de las sentencias dentro de un Grid.

- Estos tres casos describen situaciones en las que la replicación normal no es muy exitosa. Para hacer frente a estas situaciones, podemos apoyar la replicación de la ejecución de sentencias dentro del Grid.

Ejecución de una instrucción basada en Grid

- Una instrucción individual se puede ejecutar como una declaración de red, así como un procedimiento almacenado/función.

  execute procedure ifx_grid_execute('grid1',
  'delete from tab1 where mod(col1,2) = 1');

  – La ejecución es replicada, no los resultados de la ejecución.

  – La tabla 'tab1' puede ser una tabla raw o incluso contenida en una base de datos sin log de transacciones.

  – Por defecto, los resultados no serán replicados por ER

Ejecución de procedimientos basada en Grid

- Además de propagación DDL, podemos realizar la ejecución de un procedimiento, función, o instruccion como una operacion Grid.

execute procedure ifx_grid_procedure(‘grid1’,
‘myprocedure()’,’tag2’);

  - Esto causaría la ejecución de ‘myprocedure()’ en todos los nodos dentro del Grid "grid1”.
  - El comando sería "etiquetado" con "tag2“
  - Por defecto, los resultados del procedimiento no serán replicados por ER.

Ejecución de una función basada en Grid

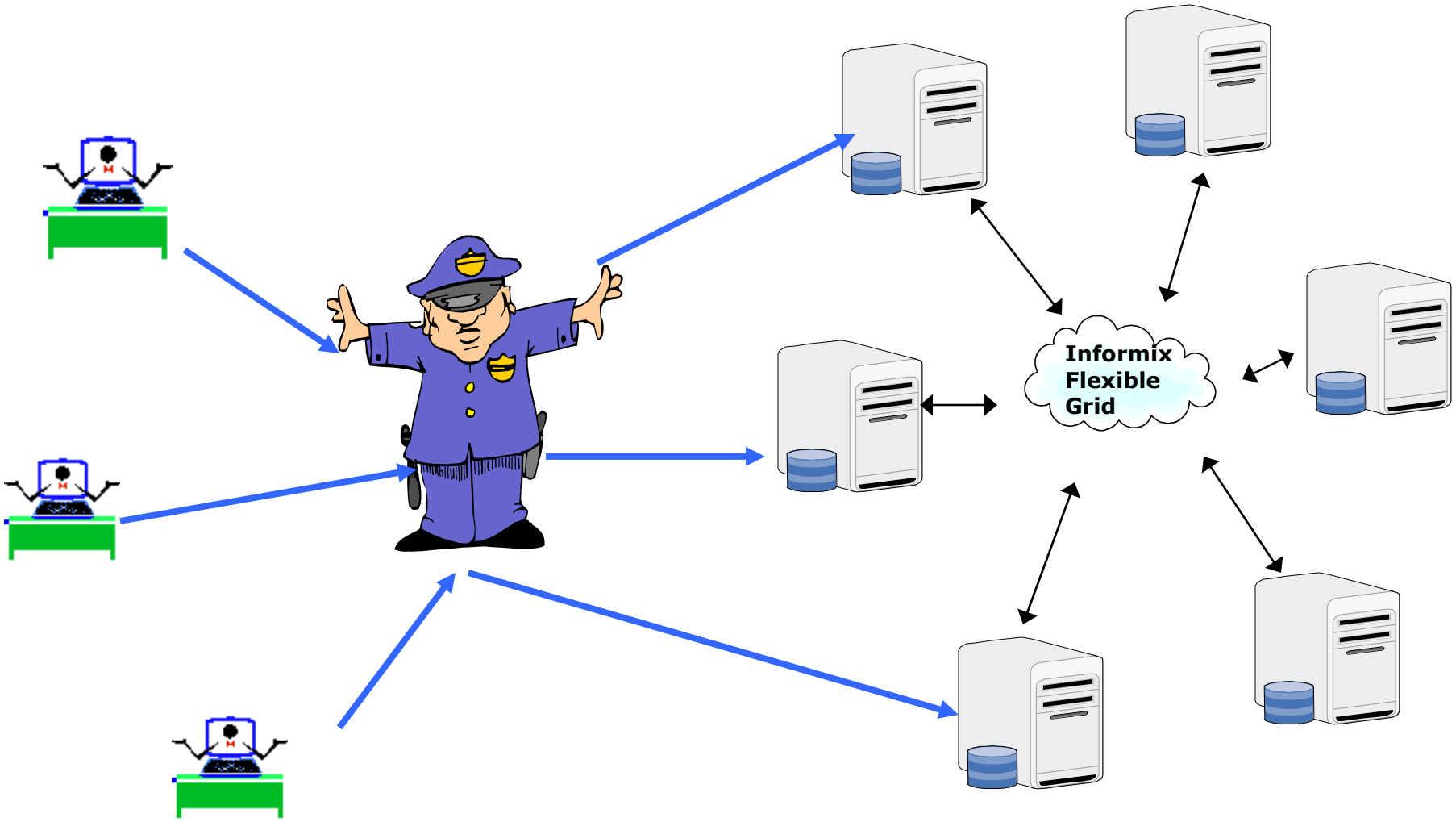
- La única diferencia entre una función y un procedimiento es que una función tendrá un valor de retorno.
  - El retorno se guarda en la base de datos syscdr y puede ser vista desde **cdr list grid**

    database sysadmin;
    execute function ifx_grid_function(‘grid1’,
    ‘task(“create dbspace”,”dbsp3”, “/db/chk/chk3”,”8G”,”0”)’);

  - Lo anterior creara el nuevo dbspace ‘dbsp3’ de 8 GB en todos los nodos dentro grid.

  - Por defecto, los resultados de la función no serán replicados por ER.

Quiero usar Connection Manager para dirigir un cliente a un nodo del Grid



CM = Connection Manager

Connection Manager & Grid

- Redirecciona los clientes de acuerdo a:
  - Orden de los nodos que se especifique
  - Una politica como:
    - Fallas para aplicar datos replicados
    - Latencia para recibir datos replicados
    - Carga de trabajo en un nodo
    - Ubicación
- SLA es a nível de Grid
- Activar la mediciín de la calidad de los datos para el fracaso & latencia

  cdr define qod –start

Ejemplo de configuración del archivo de CM

```
NAME   cm1
TYPE    REPLSET          # Este CM es para ER no para cluster
NODES list1=(grp_1+grp_2)
NODES list2=(grp_3+grp_4)+(grp_1+grp_2)
SLA  oltp=replset grid1 list1 <policy=WORKLOAD * 10 + LATENCY>
SLA  current=replset grid1 list1 <policy=LATENCY>
SLA  report =replset grid1 list2 <policy=FAILURE+WORKLOAD>
LOGFILE /tmp/cm1.log
```

Preferencia usando carga de trabajo con algo de latencia

Funcionalidades asociadas con Flexible Grid

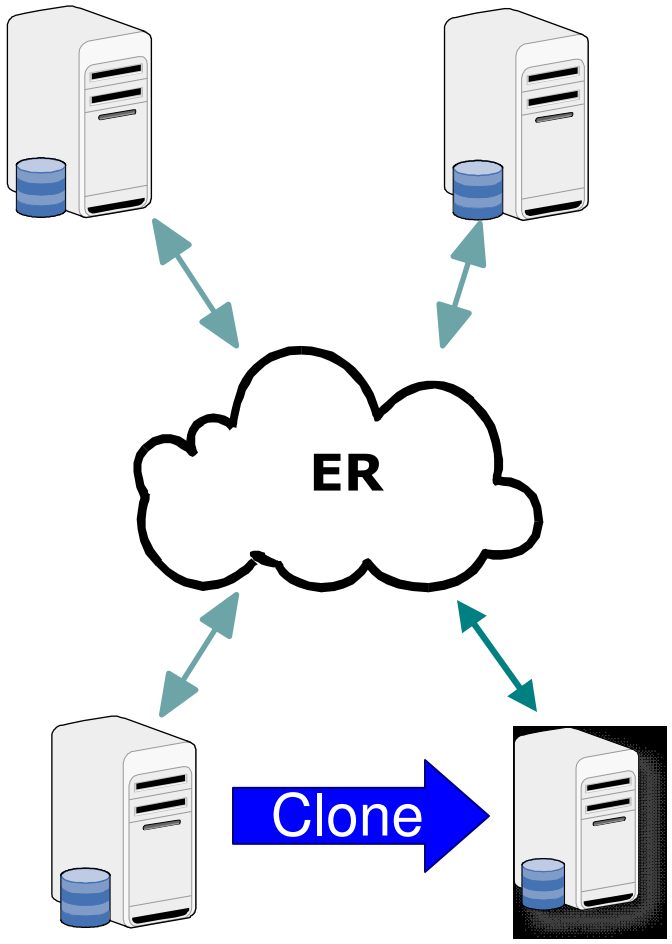
- Ifxclone
  - Permite a un nuevo nodo ser creado rápidamente a partir de una instancia existente
- Calidad de los datos
- ERKEY
  - Permite replicar sin necesidad de un primary key
- Conversion de RSS-a-ER
  - Permite convertir dinámicamente un nodo de RSS en un nodo de ER sin necesidad de check/sync

Funcionalidades asociadas con Flexible Grid

- HDR/RSS Rolling Upgrade
  - Se puede convertir dinámicamente un nodo de HDR o RSS en un nodo de ER y después de la conversión restablecer HDR / RSS mediante ifxclone.
- ifx_get_erstate()/ifx_set_erstate()
  - Permite al usuario obtener el estado de la replicación y dinámicamente activarla o desactivarla desde dentro de un transacción.
- DDRBLOCK Configurable
  - Permite al usuario definir el comportamiento de un potencial log wrap.

ifxclone

- La capacidad de clonar una instancia desde un solo comando
- Los resultados pueden ser en un clon stand-alone, un RSS secundario, o un nodo de ER
- Si la disposición final es ser un nodo de ER, el registro ER se clona también
  - No es necesario Sync/Check



Seguimiento a la calidad de los Datos

- Se hace seguimiento al número de fallas que ha encontrado una réplica desde el ultimo check/repair y monitorea la latencia para determinar la calidad de los datos

- Usos
  - Connection Manager se utiliza como parte de un SLA
  - Permite saber lo que tiene que ser verificado/reparado

Replicando sin un Primary Key

- Nueva sintaxis para create/alter table

  create table….. with erkey;
  alter table … add erkey;

- Añade nuevas columnas shadow las cuales ER va a utilizar como una clave.
- Crea automáticamente un índice único en las columnas erkey

Creación de una réplica con el ERKEY

- Nuevo parametro adicionado a define replicate --erkey or –K
- Create/Realize Template automaticamente utilizara ERKEY si este existe.
- Las tablas creadas como una tabla de Grid automáticamente tienen ERKEY

Procedimiento almacenado rss2er

- El procedimiento almacenado rss2er () se encuentra en la base de datos syscdr.
- Cuando se ejecuta, convertirá el servidor RSS secundario en un servidor de ER.
- El secundario heredará las reglas de replicación que el principal tenía.
- No requiere un ‘cdr check’ o ‘cdr sync’

cdr check/start sec2er

- Convierte un par HDR/RSS en un par de ER
- Crea automáticamente replicación ER entre el servidor primario y un secundario
- Divide un par de servidores HDR/RSS en servidores independientes estándar
- Puede ser utilizado en Informix 11.50

Rolling Upgrade

- Al tomar ventaja de 'sec2er cdr start' e 'ifxclone', es posible realizar una actualización gradual de un par HDR/RSS para que el tiempo de inactividad planeado no sea necesario durante una migración de servidores.
- Pasos Basicos
  1. Ejecutar ‘cdr start sec2er’
  2. Limitar la aplicación a sólo uno de los nodos
  3. Migrar el servidor en el cual las aplicaciones no se están ejecutando
  4. Mover las aplicaciones al servidor migrado
  5. Utilice ifxclone para volver a RSS/HDR

Dinámicamente encendiendo o apagando ER

- Para activar ER snooping – ejecute
  execute procedure ifx_set_erstate(1) or ifx_set_erstate('on')

- Para desactivar ER snooping – ejecute
  execute procedure ifx_set_erstate(0) or ifx_set_erstate('off')

- Para obtener el estado actual - ejecute
  execute function ifx_get_erstate();
  - Si retorna 1 significa que ER va a hacer snoop de los logs para esta transacción

Ejemplo de encendido de replicación ER por medio de la ejecución de un procedimiento

```
Execute procedure ifx_grid_connect('grid1');


create procedure myproc()
execute procedure ifx_set_erstate('on');
execute procedure create_summary_report();
end procedure;


execute procedure ifx_grid_disconnect();

execute procedure ifx_grid_procedure('grid1','myproc()');
```

Configuración de DDRBLOCK

- Ahora puede especificar el comportamiento si un log wrap está a punto de ocurrir con el parámetro de configuración CDR_LOG_LAG_ACTION.
  - **logstage**
    - Stages logs en un formato comprimido en el directorio de log staging
  - **dlog**
    - Crea dynamic logs
  - **Ignore**
    - Simplemente ignora un potencial log wrap - espera por el verdadero wrap
  - **shutdown**
    - Apaga ER si un log wrap pendiente está por ocurrir
  - **ddrblock**
    - el comportamiento actual

Monitoreo del estado de los comandos del Grid

- Utilice 'cdr list grid' para ver el estado de las operaciones grid ejecutadas previamente
  - Las opciones incluyen:
    --source=<source_node>
    --verbose
    --nacks
    --acks
    --pending

- Despliega la salida de cualquiera de las funciones grid que pueden haber sido ejecutadas

Ejemplo de ‘cdr list grid –v’

```
Grid Node
------------------------ ------------------------
grid1 cdr1*
cdr2
cdr3
Details for grid grid1


Node:cdr1 Stmtid:1 User:mpruet Database:tstdb 2010-05-27 15:21:57
Tag:test
create database tstdb with log
ACK cdr1 2010-05-27 15:21:57
ACK cdr2 2010-05-27 15:21:58
PENDING cdr3


Node:cdr1 Stmtid:2 User:mpruet Database:tstdb 2010-05-27 15:21:57
Tag:test
create table tab1 (col1 int, col2 int)
ACK cdr1 2010-05-27 15:21:57
ACK cdr2 2010-05-27 15:21:58
PENDING cdr3
```

Ejemplo de ‘cdr list grid –v’ (cont.)

```
Node:cdr1 Stmtid:2 User:mpruet Database:tstdb 2010-05-27
15:21:57
Tag:test
create table tab1 (col1 int, col2 int)
ACK cdr1 2010-05-27 15:21:57
ACK cdr2 2010-05-27 15:21:58
NACK cdr3 2010-05-27 15:39:21 SQLERR:-310 ISAMERR:0
Grid Apply Transaction Failure
```

- Se puede utilizar el procedimiento integrado para volver a ejecutar comandos grid fallidos

  execute procedure ifx_grid_redo(‘grid1’,’cdr1’,’cdr3’,’test’);

  Vuelve a ejecutar los comandos con la etiqueta de 'test', los cuales fallaron en cdr3 y que provenían de cdr1 dentro del Grid grid1