Aspectos del Mecanismo de Replicación de Microsoft SQL Server 7.0

Gustavo Larriera

Julio, 1998.

Resumen. En este artículo se describen las características del modelo de replicación de datos utilizado en Microsoft SQL Server 7. Se muestran las diferentes posibilidades de replicación disponibles, el modelo utilizado y consideraciones de diseño. Se describen los agentes utilizados para llevar a cabo los mecanismos de replicación. Adicionalmente, se explican los mecanismos de replicación de SQL Server 7 desde y hacia fuentes de datos heterogéneos.

Palabras clave: *Replicación; bases de datos; SQL Server; distribución; bases de datos distribuidas.*

Contenido

1 Introducción

Durante el diseño de SQL Server 7 (versión denominada "Sphynx") se han realizado diversas consideraciones en lo que respecta al mecanismo de replicación de datos, extendiendo las escasas funcionalidades disponibles en la versión previa 6.5. Fundamentalmente se ha hecho énfasis en disponer de varias alternativas de replicación y mecanismos para replicar hacia y desde diferentes fuentes de datos heterogéneas, así también como disponer de mecanismos de control de consistencia transaccional en un ambiente de datos distribuídos.

2 Consideraciones previas

Como estrategia global de diseño, SQL Server ha considerado disponer de varios enfoques de replicación, a los efectos de que puedan ser utilizados de acuerdo al tipo de contexto de replicación requerido. Cada enfoque tiene diversas ventajas específicas, y pueden mirarse desde tres puntos de vista: consistencia transaccional, autonomía de sitio y particionamiento de datos para minimizar conflictos.

La **consistencia transaccional** ha sido cubierta mediante los siguientes niveles:

- **Immediate Guaranteed Consistency (IGC).** Disponible en la versión 6.x como "tightly consistency", consiste en que todos los sitios vean a los datos actualizados en el momento exacto. Esto se implementa mediante commits en 2 fases y es una solución apropiada para redes fiables y con pocos nodos.
- **Latent Guaranteed Consistency (LGC).** Disponible en la versión 6.x como "loose consistency", consiste en que todos los sitios vean a los datos actualizados, en algún punto en el tiempo, pero con posibilidad de retraso. Esto posibilita que no todos los sitios tengan exactamente los mismos datos en un momento dado.

La **autonomía de sitio** se refiere a si las operaciones en un sitio se ven o no afectadas por las operaciones realizadas en otro. El uso de Merge Replication permite que los sitios trabajan en forma autónoma, según se describe más abajo. Finalmente, el **particionamiento de datos** es mas bien una cuestión de diseño lógico de los datos, donde las tablas se particionan horizontal y/o verticalmente para replicarlas en los sitios correspondientes y minimizar los conflictos de acceso a datos en el contexto replicado.

3 Modelo de Replicación

El modelo utilizado en SQL Server se basa en la metáfora de "publicadores/distribuidores/suscriptores" que fue introducida en las versiones 6.x En dicho modelo, un SQL Server que participa de un ambiente de replicación, cumple al menos uno de los roles de: servidor publicador (emite datos replicados), servidor distribuidor (distribuye datos replicados) y servidor suscriptor (recibe datos replicados). SQL Server provee ahora tres tipos de replicación que se adaptan a diferentes contextos de aplicación y no son excluyentes entre sí:

- **Snapshot Replication.** Este tipo de replicación es el mecanismo más simple de todos, donde el publicador replica datos tal como están en la base de datos en un momento dado. La publicación se puede realizar en forma cronogramada o por demanda. El funcionamiento es sencillo: el publicador simplemente envia una replica de todos los datos hacia los suscriptores, en vez de solamente enviar los datos que fueron alterados desde el último snapshot realizado.
- **Transactional Replication.** Este tipo de replicación realiza un monitoreo de los cambios a los datos que son realizados en el publicador (inserciones, borrados y modificaciones de transacciones que dieron COMMIT). Dichos cambios son propagados a los suscriptores en forma cronogramada o en forma continua, de forma tal que se aproxima a una transacción en tiempo real. Este mecanismo garantiza una consistencia transaccional en snetido laxo: todos los suscriptores tarde o temprano reciben los datos como si se hubiesen alterado en un único sitio.
- **Merge Replication.** Este tipo de replicación permite que los diferentes servidores actúan con alto nivel de independencia y desconectados entre sí. En un momento dado se provoca la consolidación o merging de los datos que fueron alterados en cada sitio. Los posibles conflictos, tanto a nivel de tupla como de atributo, que surjan se resuelven automáticamente mediante un algoritmo basado en el uso de prioridades y timestamps.

3.1 Replicación Snapshot y Transactional

En su forma más simple, la replicación Snapshot y Transactional se basan en un modelo de replicación en una sola dirección, desde un único publicador hacia los suscriptores. Esta ha sido la forma habitual de trabajo disponible en la versión 6.x de SQL Server. En los casos en que se desee flujo de datos desde los suscriptores hacia el publicador, se dispone de la opción **Immediate Updating Subscribers (IUS)**, que se configura cuando el artículo es creado. Una modificación al artículo realizada en un suscriptor, puede verse reflejada inmediatamente en el publicador mediante el uso automático de un protocolo de *commit en 2 fases (2PC).* Una vez que el dato actualizado es aceptado en el publicador, puede ser propagado a los demás suscriptores que participan en el contexto de replicación.

En la nueva versión, se dispone adicionalmente de la opción **Queued Updating Subscriber (QUS),** donde los datos modificados en una réplica son aplicados usando 2PC a los datos locales y también al servicio de cola disponible en NT Server, denominado Microsoft Message Queue. Esta nueva opción no requiere que el publicador con los datos originales esté en línea. Una vez en línea, los mensajes son sacados de cola y si no se detectan conflictos, los datos son commiteados. Si se detectan conflictos en una transacción, las transacciones subsiguientes son rechazadas y esta situación se notifica al suscriptor a cargo de la réplica.

3.2 Replicación Merge

Cuando se usa este tipo de replicación, SQL Server realiza 3 cambios en el esquema de la base de datos. En primer lugar, establece una columna de identidad especial que permite identificar a cada tupla a través de sus múltiples copias existentes en el ambiente de replicación. Si una tabla ya dispone de una columna con dichas características (por ejemplo, una columna de tipo ROWGUIDCOL) dicha columna es automáticamente utilizada por SQL Server. En caso de no existir tal columna, automáticamente se crea una columna de nombre 'rowguid' (de tipo ROWGUIDCOL) en la tabla base y se la indexa. En segundo lugar, SQL Server instala triggers para monitorear los cambios en los datos y llevar un registro de los mismos en un conjunto especial de tablas del sistema. Los cambios monitoreados pueden ser a nivel de tupla o de atributo. Finalmente, SQL Server agrega un conjunto de tablas del sistema, encargadas monitorear los datos alterados, sincronizarlos y detectar y resolver los conflictos. Estas tablas utilizan columnas 'rowguid' para establecer joins a las tablas base involucradas.

La resolución de los conflictos se puede realizar mediante un mecanismo basado en prioridades. En este caso, cada publicación tiene un número de prioridad asignado, que combinado con timestamps, permite resolver quién es el ganador de un conflicto. Adicionalmente, es posible diseñar a medida una táctica de resolución de conflictos, utilizando objectos del COM (Common Object Model) y/o procedimientos almacenados, que pueden ser invocados por el agente encargado de resolver conflictos (Merge Agent). Los agentes disponibles se mencionan más abajo.

4 Agentes de Replicación

A los efectos de implementar el funcionamiento de la replicación, se dispone de 4 *agentes* o servicios, que están disponibles en los servidores participantes en la replicación. Todos los agentes se ejecutan bajo el **SQL Server Agent** y pueden administrarse a través de la herramienta Enterprise Manager. En el cuadro siguiente se describen las funcionalidades de cada agente y en qué tipo de servidor se ejecutan.

AGENTE	**FUNCIONALIDADES**	**Se ejecuta en:**
Snapshot Agent	• Prepara/inicializa el esquema, datos publicados y procedimientos. • Almacena el snapshot en el distribuidor. • Recolecta información de sincronización en la base de distribución.	• Distribuidor
Log Reader Agent	• Mueve las transacciones marcadas para replicación, desde el log de transacciones del publicador a la base de distribución. • Cada BD publicada usando replicación transaccional dispone de su propio Log Reader.	• Distribuidor
Distribution Agent	• Mueve las transacciones y snapshots de la base de distribución hacia los suscriptores.	• Distribuidor (en suscripciones push) • Suscriptor (en suscripciones pull)
Merge	• Mueve y reconcilia los cambios a los datos,	• Distribuidor (en

Agent	ocurridos después de la inicialización de la replicación. Los datos se pueden mover en ambas direcciones o en una sola. • Puede embeberse y controlarse desde una aplicación, usando controles ActiveX.	suscripciones push iniciadas del lado del publicador) • Suscriptor (en suscripciones pull iniciadas del lado del suscriptor)

5 Replicación con Datos Heterogéneos

SQL Server soporta replicación desde y hacia datos heterogéneos, mediante el uso de drivers ODBC de 32-bit y OLE DB. En forma nativa, se soportan mecanismos de replicación con datos Access, Oracle y DB2. Adicionalmente, se soporta cualquier otro servidor compatible a nivel de ODBC o que cumpla los requisitos de suscriptor OLE DB. Todas las interfaces de programación de la replicación están abiertas y documentadas para uso de los desarrolladores.

El único modo de garantizar una consistencia fuerte en un entorno lo másparecido posible a una base de datos distribuída, se logra utilizando el Distributed Transaction Coordinator, disponible para soportar updates distribuidos desde la versión 6.x.

Finalmente, se dispone de herramientas para importación y exportación de datos usando los servicios de transformación de datos (Data Transformation Services, DTS). Los DTS permiten mover y transformar datos dsede y hacia proveedores que usen OLE DB, ODBC y archivos en formato ASCII delimitado.

6 Conclusiones

La replicación es un mecanismo utilizado para propagar y diseminar datos en un ambiente distribuído, con el objetivo de tener mejor performance y confiabilidad, mediante la reducción de dependencia de un sistema de base de datos centralizado. Dada la diversidad de contextos donde se aplican mecanismos de replicación, SQL Server dispone de una gama de posibilidades, en vez de utilizar una única forma de replicar datos. Cada uno de los tipos de replicación se adapta en mayor o menor medida y pueden utilizarse en forma combinada para un caso específico. Los tipos de replicación disponibles permiten moverse desde contextos donde los sitios trabajan en forma completamente unos de otros, hasta contextos donde se requiere una alta consistencia transaccional. En el primer caso, se puede utilizar Merge Replication. En el último, se utiliza el Distributed Transaction Coordinator. En los puntos intermedios, se puede elegir entre Snapshot Replication, Transactional Replication y Immediate Updating Subscribers.

Bibliografía

- Replication for Microsoft SQL Server Version 7.0. Microsoft, 1997.
- Microsoft SQL Server 6.5 System Administration. Microsoft, 1996.
- Inside SQL Server 6.5. Microsoft, 1996.
