بسم الله الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

آموزش تکرار سازی(Replication) (قسمت دوم)

مدرس : مهندس افشین رفوآ

رمز فایل :tahlildadeh.com

تراکنشی

replicationتراکنشی به طور عمده توزیع یک طرفه تراکنش ها از منتشر کننده به مشترک را امکان پذیر می کند. از replication های تراکنشی می توان با پیکره بندی ها و optionهای مختلف استفاده کرد.

این بخش تمام معماری ها و ساختار هایی که می توان در انها از replicationتراکنشی استفاده کرد را بر می شمرد و برخی از عملیات های درونی replicationتراکنشی را توضیح می دهد.

(Change tracking)

replicationتراکنشی با استفاده از دو عامل replication مدیریت می شود :عامل Log Reader و عامل توزیع . (Distribution agent)عامل Log Reader مسئول انتقال تغییرات از log تراکنش ها روس منتشر کننده به پایگاه داده توزیع است. عامل توزیع مسئول انتقال دسته های تغییرات از پایگاه داده توزیع به هر مشترک.

عاملLog Reader

عامل Log Reader کارهای زیر را طی هر چرخه انجام می دهد:

۱. به پایگاه داده توزیع متصل وreplication watermark ، آخرین شماره سری Log (LSN)طی چرخه قبلی، را از جدول MSlogreader_history می گیرد.
۲. به log تراکنش منتشر کننده متصل و آخرین LSN را می یابد.
۳. از LSN بعدی شروع به خواندن می کند تا به قدیمی ترین تراکنش باز برسد.
٤. تراکنش ها را در جداول MSrepl_commands و MSrepl_transactions پایگاه داده توزیع، می نویسد. ترتیب تراکنش ها دقیقا همان ترتیبی است که روی منتشر کننده commit شده است.
٥. Replication watremarkرا در جدول MSlogreader_history جلو می برد.
٦.) Replicated flagنشانگر replicate شدن) موجود در log تراکنش ها را برای تمام تراکنش هایی که با موفقیت به پایگاه داده توزیع نوشته شده، setمی کند.
۷. اطلاعات خطا و تاریخچه (history)را در پایگاه داده توزیع ثبت می کند.

عامل توزیع(Distribution Agent)

عامل توزیع در هر چرخه کارهای زیر را انجام می دهد:

۱. به پایگاه داده توزیع متصل شده و آخرین تراکنش اعمال شده به مشترک را از جدول MSdistribution_historyمی یابد.
۲. تمام تراکنش هایی که برای یک مشترک در حالت معلق (pending) مانده را جمع می کند.
۳. تمام تراکنش ها را دسته دسته می کند.
٤. به مشترک متصل شده و دسته های تراکنش ها را اعمال می کند.
٥. مدخل جدول MSdistribution_history را با شماره سری آخرین تراکنش که اِعمال شد آپدیت می کند.
٦. اطلاعات خطا و تاریخچه (history)را در پایگاه داده توزیع ثبت می کند.

نکته: پاکسازی پایگاه داده توزیع

عامل توزیع مستقیما مدخل های جدول توزیع که با موفقیت به تمام مشترکین نوشته شده را پاکسازی نمی کند. یک کار جدا (به نام عامل پاکسازی((cleanup agent)، به طور دوره ای اجرا می شود و تراکنش هایی که به تمام مشترکین فرستاده شده را حذف می کند. این کار برای بالابردن کارایی مجزا شده است و عامل توزیع را قادر می سازد تا سریعتر از انچه عامل Log Reader تراکنش ها را به پایگاه داده توزیع می نوشت، تراکنش ها رابه مشترکین ارسال کند. این معماری پردازشی تضمین می کند که عامل توزیع حین کار، حتی وقتی تعداد مشترکین خیلی زیاد باشد، ایجاد گلوگاه (bottleneck) نمی کند.

از انجا که موتور replication تضمین می کند تمام تراکنش های نوشته شده در منتشر کننده توسط مشترکین دریافت می شود، فشار زیادی بر پایگاه داده منتشر شده وجود دارد که باید مد نظر داشته باشید.

اگر پایگاه داده تان در مدل Simple recovery است، بخش راکد log در هر checkpoint حذف می شود.

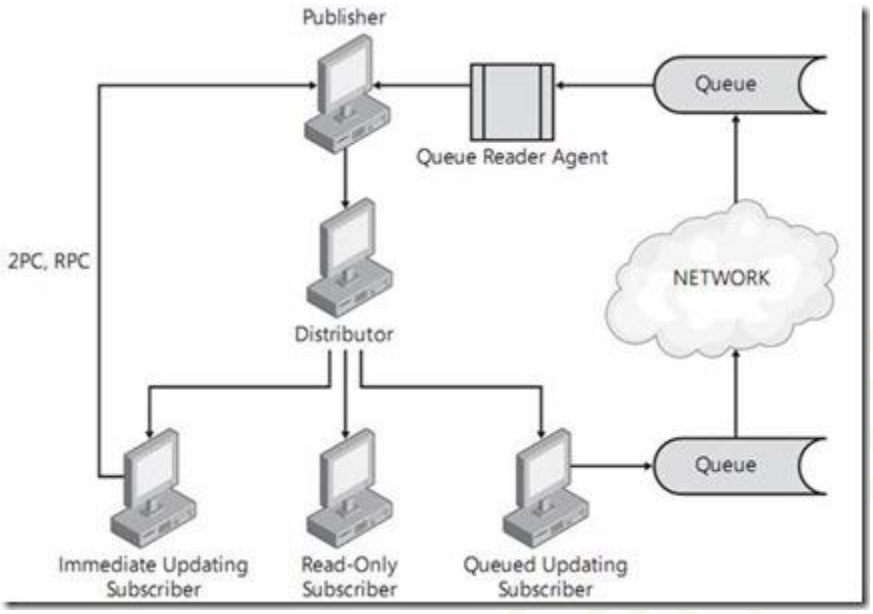
معمولا، پشتیبان گیری از log تراکنش در پایان فرایند پشتیبان گیری بخش های راکد log را حذف می کند. موتور پشتیبان گیری با شروع از اول log و ادامه تا رسیدن به قدیمی ترین تراکنش باز، قسمت های راکد را حذف می کند. بعد از اینکه به قدیمی ترین تراکنش باز رسید، فرایند پشتیبان گیری متوقف می شود .برای تشخیص تراکنش های باز از تراکنش هایcommited ، پروسه پشتیبان گیری یک flagیک بیتی را در هر رکورد از log تراکنش می خواند که نشانگر commited بودن یا نبودن ان تراکنش است.

موتور replication باید تضمین کند که تمام تراکنش های منتشر کننده به مشترک می رسد. اما پروسه پشتیبانی گیری از log تراکنش و یا پروسه checkpointدر مدل Simple Recovery ممکن است با این پروسه تداخل ایجاد کند .اگر SQL Server اجازه دهد بخش راکدlog ، قبل از اینکه عامل Log Reader بتواند تراکنش ها را در پایگاه داده توزیع بنویسد، حذف شود، تراکنش ها ممکن است گم شود. به همین خاطر، هنگامی که یک پایگاه داده، در یک replication تراکنشی مشارکت دارد، یک flag دوم درون یک رکورد log تراکنش فعال می شود .پشتیبان گیری از log تراکنش، در این حالت هم، بخش راکد log را پردازش می کند، اما مجاز نیست تا وقتی که هر دوflag ، commited , replicated ست (set)نشده باشد، رکوردی را پاک کند. بنابراین، اگر عامل Log Readerتراکنش ها را به پایگاه داده توزیع ننویسد، logتراکنش روی منتشر کننده بزرگ می شود حتی اگر پشتیبان گیری از log تراکنش، و یا checkpoint گیری از پایگاه داده در مدلSimple Recovery، انجام شود. نتیجه اینکه replication تراکنشی با هر مدل recoveryسازگار است.

همین فرایند بنیادین برای پایگاه داده توزیع رخ می دهد. تراکنش از پایگاه داده توزیع حذف نمی شود تا وقتی که به موفقیت به تمام مشترکین نوشته شود. هنگامی که یک مشترک آفلاین است و یا در دسترس نیست ،پایگاه داده توزیع بزرگ می شود.

(Transactional Options)

همان طور که در تصویر زیر می بینید، می توان replication تراکنشی را با دو گزینه پیکره بندی کنید، گزینه آپدیت صفی مشترک و آپدیت بلادرنگ مشترک.



گزینه آپدیت بلافاصله مشترک(Immediate Updating Subsciber)

آپدیت بلافاصله مشترک، اجازه می دهد تغییراتی در مشترک رخ دهد و به منتشر کننده انتقال داده شود. سپس این تغییرات توسط موتور تراکنشی کشف شده و به تمام مشترکین دیگر فرستاده می شود، ولی به مشترکی که منبع تغییر بود دوباره اعمال نمی شود. برای جلوگیری از ارسال مجدد تغییرات به مشترک اولی،هر جدول که در replicationمشارکت دارد یک ستون از نوع timestamp نیاز دارد.

پروسه اعمال تغییرات در مشترکی که با گزینه آپدیت بلادرنگ مشترک پیکره بندی شده به این صورت است:

- برنامه کاربردی تراکنشی را در مشترک صادر می کند.
- triggerاجرا می شود.
- Triggerاز Microsoft distributed Transaction Coordinator (MS DTV) می خواهد تا به منتشر کننده متصل و تراکنش را مجددا صادر کند.
- تراکنش روی منتشر کننده با موفقیت تمام (commit)می شود.
- triggerروی مشترک با موفقیت تمام (commit) می شود.
- تراکنش با موفقیت روی مشترک تمام (commit) می شود.

مشکل اصلی در معماری های با درجه دسترس پذیری بالا(high-availability) ، با آپدیت بلادرنگ مشترک، این است که تغیرات باید به منتشر کننده اعمال شود. اگر منتشر کننده در دسترس نباشد، تراکنش توزیع شده شکست می خورد .(fail)چون تراکنش توزیع شده از طریق یک Trigger اجرا می شود، تراکنش اغاز کننده هم شکست می خورد و Rollback می شود. بنابراین، آپدیت بلادرنگ مشترک یک replication ناسازگار برای محیط های دسترس پذیری بالا است.

گزینه آپدیت صفی مشترک(Queued Updating subscriber)

گزینه آپدیت صفی مشترک هم اجازی می دهد تغییراتی در مشترک رخ داده و به منتشر کننده انتقال یابد، اما مکانیسم ان با گزینه آپدیت بلادرنگ مشترک کاملا متفاوت است.

فرایند اعمال تغییرات به مشترک با این گزینه به این شکل است:

- برنامه یک تراکنش را صادر می کند.
- triggerآغاز می شود.
- Triggerتراکنش را در یک صف (جدولی از پایگاه داده) قرار می دهد.
- Triggerبا موفقیت تمام می شود.
- تراکنش با موفقیت تمام می شود.

عامل صف خوان (Queue Reader Agent) در فواصل معین، صف را به منتشر کننده انتقال و تمام تراکنش ها را صادر می کند. تغییراتی که به منتشر کننده انتقال می یابد دوباره به مشترک اولی که تراکنش را آغاز کرد، اعمال نمی شود. برای این کار به یک ستون timestamp در جدول نیاز است.

بدلیل پردازش آسنکرون (نا همزمان) یک آپدیت صف بندی شده، ممکن است ناسازگاری داده رخ بدهد. هنگام استفاده از گزینه آپدیت صفی مشترک در محیط های با دسترس پذیری بالا، برای کم کردن ناسازگاری داده ها، تغییرات فقط طی یک حالت شکست(failure) ، تغییر به مشترک اعمال می شود. این تضمین می کند که تغییرات روی یک کپی از داده ها انجام می شود، به این صورت با استفاده از روش های متداول برنامه های کاربردی، می توان تغییرات ناسازگار را کنترل کرد.

برتری گزینه آپدیت صفی مشترک این است که هنگامی که منتشر کننده دوباره آنلاین می شود، به طور خودکار میتوان تمام تغییراتی که در زمان عدم دسترسی به منتشر کننده، برروی مشترک انجام شده را به ان منتشر کننده اعمال کرد . تمام برنامه ها و پردازش هایی که در معماری های دیگر برای fail back لازم است در اینجا حذف شده است، چون آپدیت صفی مشترک به طور توکار قابلیت up to date کردن منتشر کننده را بعد از یک Failoverداراست. تنها گام مورد نیاز برای fail back به منتشر کننده، repointکردن برنامه های کاربردی است.

نکته: یک publication تراکنشی را می توان با هر دو گزینه آپدیت بلادرنگ مشترک و آپدیت صفی مشترک پیکره بندی کرد. از گزینه آپدیت صفی مشترک می توان در هنگام عدم دسترسی به منتشر کننده به عنوان یک مکانیسم failover استفاده کرد.

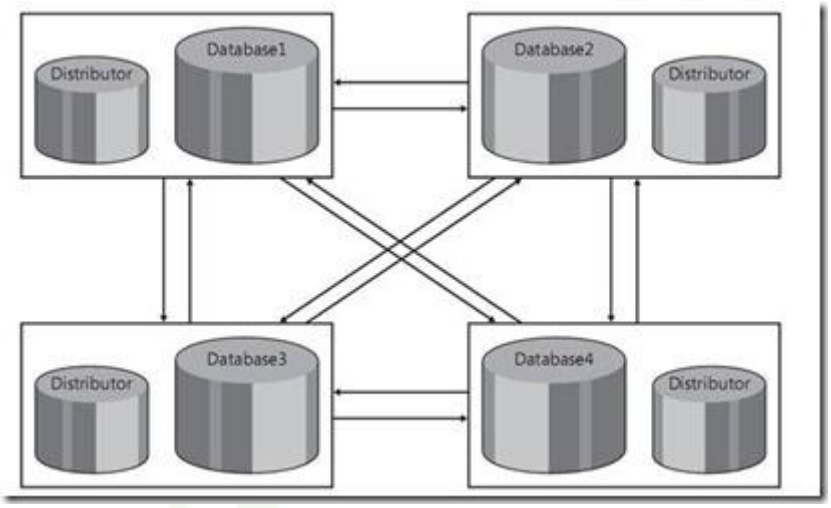
علاوه بر عملیات یک طرفه معمولی، که پیش فرض است، می توان Replication تراکنشی را بر اساس دو معماری متداول دیگر نیز پیکره بندی کرد. این معماری ها را با هیچ مکانیسمی در موتور replication اشتباه نگیرید. این معماری ها، Replicationتراکنشی یک طرفه معمولی را بدون هیچ گزینه اپدیتی، پیاده سازی می کنند. پیاده

سازی انجام شده باعث جریان تغییرات بین منتشر کننده و مشترک می شود، با این حال همچنان یک موتور تراکنشی است.

موتور تراکنشی ناسازگاری های داده را کشف و حل نمی کند. بنابراین، اگر ممکن است تغییراتی را در منتشرکننده یا مشترک انجام دهید که منجر به ناسازگاری داده ها شود، معماری های دوطرفه و نظیر به نظیر Replication تراکنشی را نمی توانید پیاده کنید.

Replicationنظیر به نظیر

replicationنظیر به نظیر معماری است که در SQL Server 2005 معرفی شد و فقط در نسخه Enterprise موجود است. اصل این معماری این است که replication تراکنشی می تواند داده ها را بین دو یا چند نظیر (peer)جا به جا کند . دیاگرامی از معماری نظیر به نظیر در زیر آمده است.



replicationنظیر به نظیر پیاده سازی replication تراکنشی است. ایده اصلی این است که شما می توانید با استفاده از replication تراکنشی یک مجموعه از جداول را برداشته و از پایگاه داده ۱ به پایگاه داده ۲ Replicateکنید. می توانید با همان مجموعه داده روی پایگاه داده ۲، publicationبسازید و ان را با استفاده از replication تراکنشی به پایگاه داده ۱ replicateکنید.

در عمل، تمام پایگاه های داده ای که در معماری نظیر به نظیر مشارکت دارند، تمام تغییرات را به تمام پایگاه های داده دیگر Replicate می کنند . برای جلوگیری از اینکه تراکنش ها در این معماری بدون پایان بچرخند، باید در جدول هایی که در replication مشارکت دارند یک ستون rowguid داشته باشید، که موتور را قادر می سازد پایگاه داده آغازگر تراکنش را تشخیص دهد.

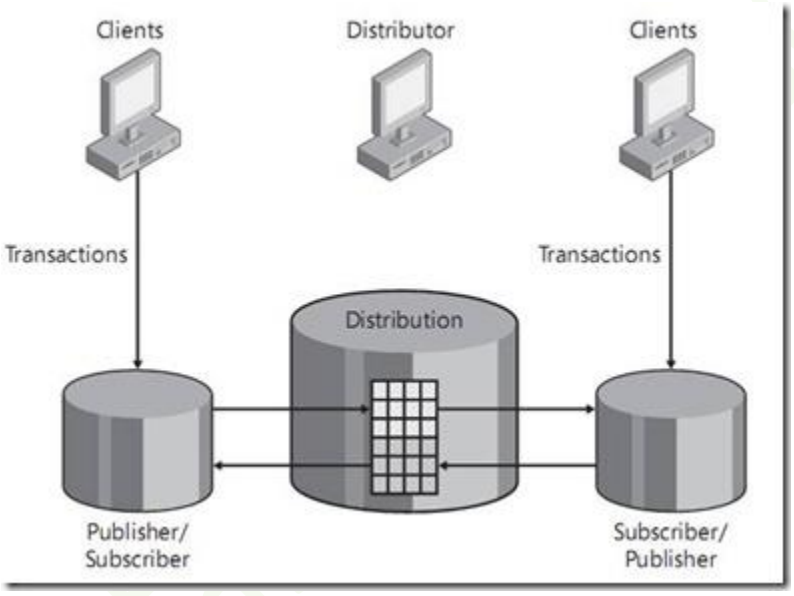
replicationنظیر به نظیر شرایط سفت و سختی دارد که باید بر آورده شود:

- هر نظیر باید توزیع کننده خود را داشته باشد.

- ساختار جدول روی تمام نظیر ها باید یکی باشد.
- گزینه های آپدیت بلافاصله و صفی موجود نمی باشد.
- ناسازگاری داده ها رخ نمی دهد.

replicationدو طرفه

پیکره بندی replication دوطرفه کمی با replication نظیر به نظیر متفاوت است. این معماری یک نوع replicationتراکنشی است. همانطور که در تصویر زیر امده مجموعه جداولی که از پایگاه داده ۱ به پایگاه داده ۲ replicate می شود همان مجموعه جداولی است که از پایگاه داده ۲ به پایگاه داده ۱ replicate می شود.



برای جلوگیری از چرخیدن (looping) تراکنش ها بین دو پایگاه داده، باید به هر) subscription اشتراک)، یک پارامتر @loopback_detection بیفزایید.

هر چند replication دو طرفه را می توان به عنوان زیر مجموعه ای از replicationنظیر به نظیر انجام داد، هنگام پیاده کردن این replication کارایی بالاتری بدست می اورید.

در این معماری، نیازی نیست هر منتشرکننده یک توزیع کننده داشته باشد و جداولی که در replication مشارکت دارند نیازی به ستون rowguid ندارند .ساختار جداول هم می تواند متفاوت باشد، هر چند من با این نمونه برخورد نداشته ام.

ناسازگاری داده ها کنترل نمی شوند بنابراین باید با استفاده از کد نویسی یک معماری و ساختار دوطرفه پیاده کنید.نمی توان این معماری را با استفاده از GUI پیاده کرد تا کمپوننت ها را تولید کند.

مانیتورینگ

معمولا از Replication Monitor برای مانیتور کردن معماری های replicationاستفاده می شود. درون Replication Monitor، شما می توانید آماری از وضعیت تمام publication ها، Subscriptionها و عامل ها را ببینید . همچنین می توانید اطلاعات تاریخچه وضعیت (status history) و خطا ها را به منظور عیب یابی، ببینید.

یکی از بزرگترین مشکلات محیط های replication در نسخه های قبلی SQL Server مربوط به گلوگاه (bottleneck)بود. با استفاده از یک روال ذخیره شده سیستمی (System-store procedure) به نام sp_browsereplcmdsکه در پایگاه داده توزیع وجود داشت، تعیین تعداد تراکنش هایی که موتور replicationدر پردازش انها عقب افتاده بود، سر راست بود. با این حال، غیر ممکن بود بتوان زمانی که طول می کشید تا موتور replication عقب افتادگی را جبران کند)تراکنش های عقب افتاده را انجام دهد)، را تعیین کرد، چون اطلاعات زمانی بین محیط نگهداری نمی شد.

درحالی که تغییرات توسط عامل Log Reader و عامل توزیع انتقال می یابد، موتور، آمار سرعت انتقال داده و مدت زمان انتقال داده را نگهداری می کند . با استفاده از این امار، Replication Monitorمی تواند پیوسته اطلاعاتی را نمایش دهد که به شما نشان می دهد چند تراکنش دیگر باید به مشترکین فرستاده شود، بعلاوه مدت زمان تخمینی جبران عقب افتادگی.

هر چند امار Replication Monitor اطلاعات وضعیتی (status) خوبی را نشان می دهد، جزئیات دقیقتری را درمورد میزان عقب افتادگی مشترکین به دست نمی دهد Replication Monitor . برای تاخیر تنها یک رقم نشان می دهد، Administratorنمی تواند مشخص کند که گلوگاه (bottleneck) در عامل Log Reader است , یا اینکه عامل توزیع سعی در جبران عقب افتادگی دارد. به همین خاطر tracer token ها معرفی شدند تا این جزئیات را فراهم آورند.

tracer tokenیک تراکنش مخصوص است که برای موتور replication صادر می شود Tracer token . مانند دیگر تراکنش ها به log تراکنش ها فرستاده می شود .عاملینreplication ، tracer tokenرا مانند دیگر تراکنش ها، در ساختار)معماری) انتقال می دهند. انچه tracer token را ویژه می کند، موتور replication تراکنش مخصوص را شناخته و امار زمانی ان را حین انتقال در ساختار معماری ثبت می کند. با استفاده از یک tracer token، می توانید زمان دقیقی که طول کشید تا ان token به پایگاه داده توزیع انتقال یابد و یا مدت زمانی که طول کشید تا ان به هر مشترک فرستاده شود، را داشته باشید؛ همچنین می توانید مجموع تاخیرهای کلی از منتشر کننده تا مشترک را بدست اورید.

با این اطلاعات، حالا می توانید تمام گلوگاه ها را در معماری replication ایزوله و رفع کنید.

اعتبارسنجی

موتور replication تضمین می کند که تراکنش هایی که از منتشر کننده به مشترک انتقال می یابد به همان ترتیبی است که صادر شد. از انجا که منتشر کننده و مشترک معمولا پایگاه های داده ای بر روی نمونه های مختلفی از SQL Server هستند که می توانند از لحاظ فیزیکی جدا باشند، یک سوال همواره مطرح است: از کجا بدانیم دو پایگاه داده سنکرون هستند؟

موتور replication مکانیسمی برای اعتبارسنجی سنکرون سازی دارد. دو روال ذخیره شده سیستمی فراهم شده تا اعتبار سنجی را انجام دهد sp_publication_validation: و sp_article_validation.

sp_publication_validation روال sp_article_validation را برای تمام Article های درون Publication ، اجرا می کند.

هر دوی این روال ها می توانند با دو روش اعتبارسنجی را انجام دهند:

۱. فقط تعداد سطرها
۲. تعداد سطرها و checksum باینری

روش پیش فرض اعتبارسنجی فقط تعداد سطرها را می شمرد. این روش فقط برسی می کند تعداد سطرها بین منتشرکننده و مشترک یکی باشد. اگر محتوای سطرها متفاوت باشد با این روش نمی توان ان را کشف کرد. با این حال، از انجا که پایگاه های داده در replication مشارکت دارند، به شدت نامحتمل است که جداول دارای تعداد سطرهای برابر ولی محتوای متفاوت باشند.

گسترده ترین روش اعتبارسنجی شمارش تعداد سطرها و انجام یک checksum باینری است. تعداد سطرهای جداول منتشرکننده و مشترک تطبیق داده می شوند . علاوه بر ان، یک checksum باینری محاسبه و مقایسه می شود. این روش تمام اختلافات تعداد سطور و محتوای ان ها را کشف می کند. با این حال، عملیاتی پر هزینه است که باعث سربار (overhead) در پردازش می شود. به خاطر حجم سربار، از این روش هر از چند گاه باید استفاده کرد.

اعتبار سنجی ممکن است باعث سربار زیاد، مخصوصا در محیط هایی که تعداد مشترکین یک publication زیاد است و یا publication ها دارای تعداد زیادی article هستند، بشود. معمولا روش تعداد سطور را روزانه و روش تعداد سطور و checksum باینری به صورت هفتگی انجام می شود.

replication تراکنشی

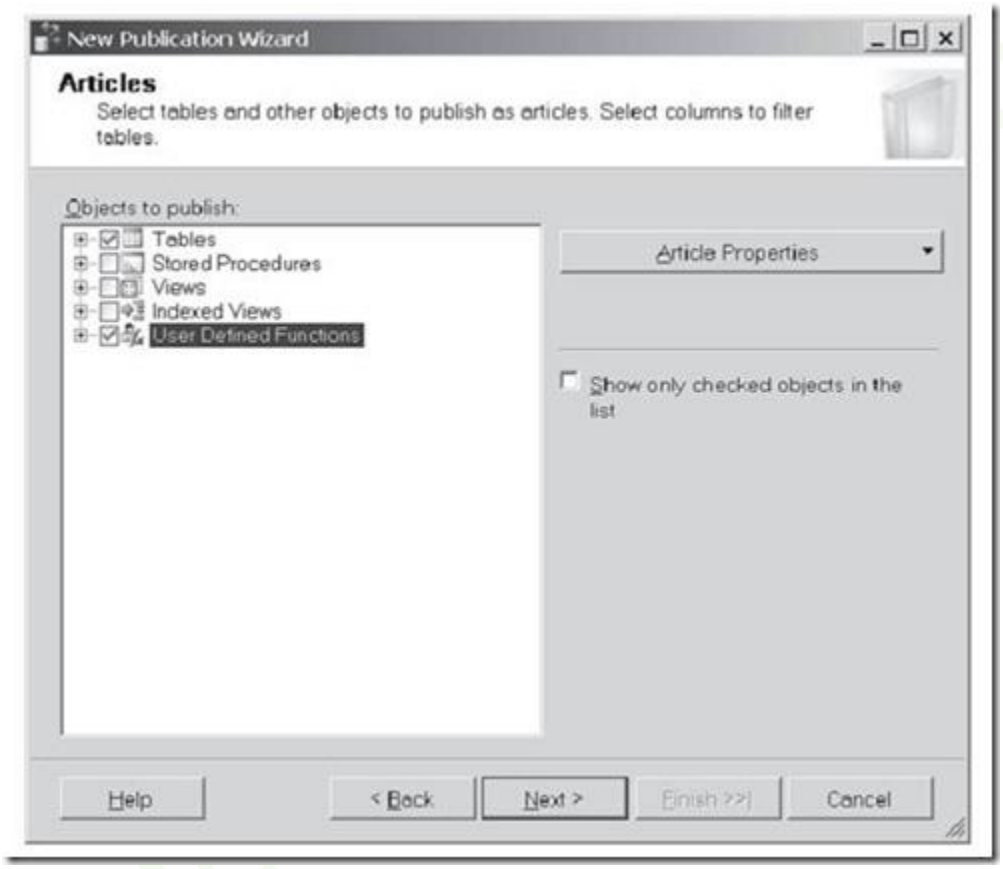
در اینجا به طور عملی، با استفاده از پایگاه داده AWTransactional ، یک replication تراکنشی را پیکره بندی می کنیم.

تمرین ۱: ساخت publication

یک publication می سازیم.

۱. SSMS را باز و به نمونه ای که برای replication استفاده می کنید، متصل شوید.
۲. پایگاه داده ای به نام AWTranSubscriber روی همان نمونه ای که پایگاه داده AWTransactional را دارد بسازید.

۳. اگر لازم است گره Replication را بسط دهید، روی Local Publications کلیک راست کرده و New Publicationرا انتخاب کنید. روی Next کلیک کنید.
٤. پایگاه داده AWTransactional را انتخاب کنید و روی Next کلیک کنید.
٥. Transcational Publicationرا انتخاب و روی Next کلیک کنید.
٦. مانند شکل زیر، تمام جدول ها و توابع کاربرـ تعریف (UDF) را انتخاب و روی Next کلیک کنید.



نکته UDF :ها

به این خاطر UDF ها را در publication قرار می دهیم که جداولی در پایگاه داده AdventureWorks وجود دارد که دارای محدودیت هایی (constraint)هستند که با UDF تعریف می شوند. اگر انها ساخته نشوند و یا به publication اضافه نشوند، اِعمال snapshot با شکست مواجه می شود.

۷. فیلتری اعمال نمی کنیم پس روی Next کلیک کنید.
۸. تیک گزینه "Create a Snapshot Immediately and Keep Snapshot available to Initialize Subscriptions"را زده و روی Next کلیک کنید.
۹. Security settings (تنظیمات امنیتی) را انتخاب کنید.

۱۰. گزینه های "Run under SQL Server Agent Service Account" و "By Impersonating Process Account"را انتخاب و روی Next کلیک کنید.

تذکر: تنظیمات امنیتی عامل

توصیه می شود عاملین Replication را تحت اکانت SQL Server Agent servcie اجرا نکنید. اکانت های عامل replication باید دارای مینیمم مجوزها (permission)در محیط باشند، اکانت SQL Server Agent serviceان مجوزها را افزایش می دهد.

۱۱. روی Next کلیک کنید. مطمئن شوید گزینه "Create the Publication" انتخاب شده است. روی Nextکلیک کنید.
۱۲. به publication یک نام داده و روی Finish کلیک کنید.
۱۳. بعد از New Publication Wizard ، publicationرا ساخت، روی Close کلیک کنید.

تمرین ۲: ساختSubscription

۱. در این تمرین، یک Subscription برای publication ی که در بالا ساختید، می سازیم.
۲. Local Publicationsرا بسط دهید، روی publication که الان ساختید کلیک راست کنید، New Subscriptionsرا انتخاب کنید. در New Subscription Wizard روی Next کلیک کنید.
۳. مطمئن شوید publication شما انتخاب شده و روی Next کلیک کنید.
٤. مطمئن شوید گزینه "Run All Agents At the Distribute" انتخاب شده باشد و روی Next کلیک کنید.
٥. چک باکس کنار نمونه تان را انتخاب کنید. از منوی پایین افتادنی Subscription Database گزینه Database AWTranSubscriberرا انتخاب کنید.
٦. در صفحهDistribution Agent Security ، دکمه ... کنار مشترکتان (subscriber) در پنجره Subscription Propertiesرا انتخاب کنید.
۷. گزینه های "Run under SQL Server Agent Service Account" و هر دو گزینه پایین "By Impersonating the Process Account"، Connect to Distributer وConnect to Subscriberرا انتخاب کنید Ok . و Next را کلیک کنید.
۸. در صفحهSynchronization Schedule ، مطمئن شوید Agent Schedule برابر Continuously)مقدار پیش فرض) است و روی Next کلیک کنید.
۹. مطمئن شوید Subscription Properties برابر Initialize Immediately است و روی Next کلیک کنید.
۱۰. مطمئن شوید گزینه Create Subscription(s) انتخاب شده باشد و روی Next و بعد finish کلیک کنید.
۱۱. بعد از ساخته شدن Subscription روی Close کلیک کنید.

تمرین ۳: استفاده ازReplication Monitor

در این تمرین، از Replication Monitor برای مشاهده اطلاعاتی درباره publicationو subscription استفاده و کمی با tracer token کار می کنیم.

۱. درObject Explorer ، روی Replication کلیک راست کرده و Launch Replication Monitor را انتخاب کنید.
۲. درReplication Monitor ، بخش ها و تب های مختلف را برسی کنید و اطلاعاتی که الان درباره publicationو subscription ای که ساختیم، نمایش می دهد، را ببینید. با کلیک راست برروی هر مدخل می توان به ویژگی های ان شی و جزئیات بیشتر دست پیدا کرد.
۳. چند Tracer token ارسال و نتایج را ببینید.

· Replicationتراکنشی از عامل Log Reader برای انتقال تراکنش ها از log تراکنش های منتشر کننده به پایگاه داده توزیع استفاده می کند. عامل توزیع، سپس تراکنش ها را از پایگاه داده توزیع به تک تک مشترکین انتقال می دهد.

· Replicationتراکنشی داده را در یک جهت توزیع می کند – از منتشر کننده به مشترک. از دو گزینه، آپدیت بلافاصله مشترک و آپدیت صفی مشترک، می توان برای پیکره بندی استفاده کرد. این گزینه ها تراکنش ها را قادر می سازد تا بر مشترک اجرا و به منتشر کننده انتشار یابد.

· Replicationتراکنشی با ۳ معماری مختلف قابل پیکره بندی است. معماری پیش فرض داشتن یک منتشر کننده با یک یا چند مشترک است. به عنوان جایگزین می توان Replication تراکنشی را با معماری نظیر به نظیر و دوطرفه پیکره بندی کرد.

· Administratorمی تواند با استفاده ازtracer token ، آمار زمانی درون replicationرا از منتشر کننده به مشترک، یا از توزیع کننده به مشترک بدست اورد. از این امار برای مانیتور کردن تاخیر های نقطه به نقطه استفاده می شود.