بسم الله الرحمن الرحیم

# آموزشگاه تحلیل داده

## تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

## آموزش تکرار سازی(Replication) (قسمت اول)

مدرس : مهندس افشین رفوآ

رمز فایل :tahlildadeh.com



هدف اصلی از replication توزیع داده از یک پایگاه داده master به یک یا چند پایگاه داده ثانویه است. از انجا که replication یک کپی از داده را به شکل سنکرون با کپی اصلی پایگاه داده master نگهداری می کند، از این تکنولوژی می توان برای بالا بردن دسترس پذیری (availability) برنامه ها استفاده کرد.

در این بخش مروری ابتدایی بر موتور replication ، به علاوه ی گزینه هایی که برای توزیع داده می توانید استفاده کنید، خواهیم داشت. همچنین توضیح خواهیم داد چگونه می توان از مکانیسم های توزیع داده استفاده کرد تا راهکاری (solution) همواره در دسترس با زمان تاخیر به شدت کوتاه را فراهم اوریم که بتواند در صورت نیاز بهfailback ، زمان خاموشی و عدم دسترسی برنامه را به کمترین حد برساند.

برای ادامه به ابزار زیر نیاز داریم:

- یک نسخه ازMicrosoft SQL Server 2008 ، نسخهEnterprise ، استاندارد و یا.Developer
- پایگاه داده AdventureBooks باید نصب شود
- دو پایگاه داده با نام های AWTransactionalوAWMergeکه کپی پایگاه داده AdventureBooksاست.

توجه: اکانت های سرویس(SERVICE ACCOUNTS)

موتور replication برای امنیت بیشتر از اکانت های نام دار (named accounts) استفاده می کند، بنابر این اکانت سرویس که SQL Server و SQL Server Agent تحت ان اجرا می شوند باید اکانت local و یا اکانت دامین باشد . برای این اکانت سرویس نمی توان از localsystem استفاده کرد.

## replication

Replication به عنوان یک مکانیسم توزیع داده طراحی شده است. در پایین ترین سطح، تغییراتی که به یک پایگاه داده اعمال می شود به یک یا چند) target هدف) توزیع می شود. هسته موتورreplication ، برای پیاده سازی های خیلی انعطاف پذیر طراحی شده است، اما معماری هسته را می توان به کار بست تا قابلیت دسترس پذیری (availibility) یک پایگاه داده را افزایش داد، چون یک کپی اضافه از داده ها، به صورت سنکرون با کپی اصلی نگه داری می شود.

این بخش درباره کمپوننت های مختلفی است که می توانید در replication پیکره بندی کنید همراه با معماری هسته موجود در موتور.replication داده ای که قرار است) replicate کپی) شود توسط سه کمپوننت مرکزی تعریف می شود.

## Article

Articleبنیادی ترین عنصر سازنده replication است و درشت ترین (granular)سطح توزیع داده را تعریف می کند. یک Article را می توان بر اساس یک جدول، view، روال ذخیره شده (stored procedure)و تابع تعریف کرد.

مربوط ترین نوع Article برای دسترس پذیری بالا بر اساس جدول تعریف می شود . Article مجموعه داده ای را مشخص می کند که SQL Server به یک یا چند پایگاه داده replicate می کند.

## ( Publicationانتشارات)

یک publication درشت ترین سطح در معماری replication است Publication . به معنای گروه بندی Article های تعریف کننده مجموعه replication است.

## Filter

از این لحاظ که replication قادر است فقط بخشی از پایگاه داده را کپی کند، در میان تکنولوژی های با قابلیت دسترسی بالا (high-availability) بی نظیر است. با اعمال فیلتر به Article می توان مجموعه داده ای که قرار است replicateشود را محدود کرد.

Articleها را می توان بر اساس سطر یا ستون فیلتر کرد.

داده ای که درون replication جابجا می شود را Publication یا مجموعه داده می گویند. این داده همیشه درون context زیرمجموعه داده ای است که بر اساس Article های تعریف شده و سطر و ستون های فیلتر شده، برای Replication مشخص شده است. این ویژگی یکتای Replication است که ما را قادر می سازد دسترس پذیری به فقط یک بخش از پایگاه داده را بالا ببریم.

یک فیلتر ستونی یک زیر مجموعه از ستون های جدول را مشخص می کند. فیلتر ستونی اجازه replicate کردن داده را می دهد ولی داده های حساس را می توان خارج کرد.

یک فیلتر سطری تعداد سطرهایی که قرار است replicate شود را محدود می کند. از سه نوع فیلتر سطری می توان استفاده کرد:

- فیلتر سطری استاتیک هنگامی که Article تولید می شود پیش تعریف می شود و Articleرا، سوای Subscriber، محدود به یک زیر مجموعه از داده ها می کند . یک مثال از فیتلر سطری استاتیک در زیر امده است :

```
1: WHERE State = 'TX'
```

- فیلتر سطری پویا، فقط در Replication ادغامی (merge) موجود است، امکان تعریف فیلتری را فراهم می اورد که بر روی یک Article ثابت نیست. در زمان پروسه سنکرون سازی، فیلتر دوباره، بر اساس اطلاعات مشترک (Subscriber) محاسبه می شود تا یک publication بتواند مجموعه متفاوتی از داده را به هر مشترک توزیع کند. یک نمونه از این فیلتر ها در زیر امده است :

```
1: WHERE UserName = suser_sname()
```

- فیلترjoin ، فقط برای Replication ادغامی، امکان فیلتر کردن یک جدول را براساس رابطه اش با یک جدول) parent والد) فراهم می اورد. برای نمونه، ممکن است جدولی شامل مشتری ها، سفارش هایشان و جزئیات ان سفارش ها داشته باشید. اگر جدول مشتری های شما دارای فیلتری باشد که مجموعه داده ها را به یک حالت خاص محدود کند، شما احتمالا می خواهید جدول های سفارش ها و جزئیات سفارش ها را نیز به همان شکل فیلتر کنید. با استفاده از فیلتریjoin ، می توانید جدول مشتری ها را بر اساس یک حالت فیلتر و سپس جداول سفارش ها و جرئیات سفارش ها را بر اساس زیر مجموعه ی جدول مشتری ها که replicate می شود، فیلتر کنید .

نکته

هر چند Replication توانایی اعمال فیلترها به Article ها را داراست، این قابلیت در معماری های برای دسترسی بالا به کار نمی رود. یک معماری دسترسی بالا بیشتر درگیر نگهداری یک کپی کامل و منسجم از داده ها بر روی یک نمونه مجزای SQL Server است.

## Replication

با سه نقش (role) مختلف می توان پایگاه داده – و به تبع ان نمونه هایی (instance) که پایگاه داده را میزبانی می کند – را پیکره بندی کرد:

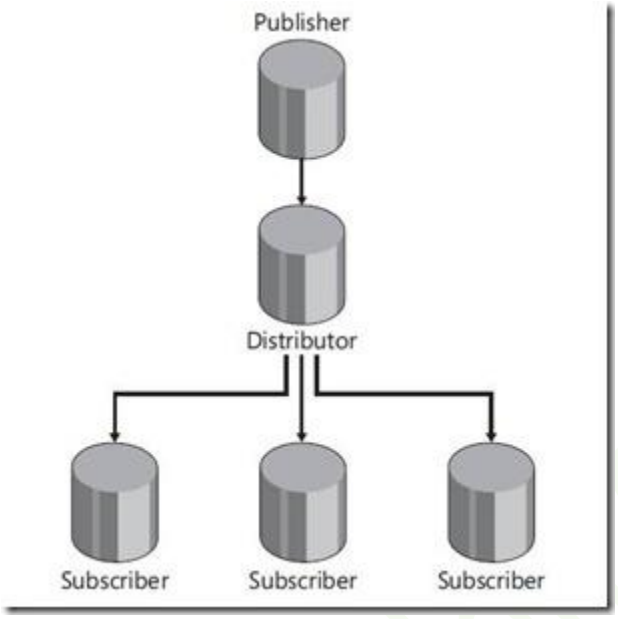
- Publisher (منتشر کننده) کپی داده master را درون یک معماری Replication نگهداری می کند. نمونه ای (instance) که میزبان پایگاه داده Publisher است را با publication ی که مجموعه داده های Replication را مشخص می کند، پیکره بندی می کنیم.
- Subscriber (مشترک) پایگاه داده ای است، که تغییرات را از موتور Replication که بر اساس publication تعریف شده، دریافت می کند.
- Distributer (توزیع کننده) موتور اصلی درون معماری Replication است .پایگاه داده توزیع (distribution database)، برروی نمونه ای ذخیره شده که به عنوان Distributer پیکره بندی شده، نصب شده است. در تمام معماری های Replication، Distributer جایی است که تمام عاملان Replication (agent) به طور پیش فرض، در حال اجرا هستند.

یک نمونه از SQL Server را می توان به عنوان توزیع کننده پیکره بندی کرد. یک پایگاه داده را می توان به عنوان منتشر کننده، مشترک و یا هر دو پیکره بندی کرد.

توپولوژی Replication یک دیاگرام جریان فرایند (process flow diagram) را فراهم می اورد که نحوه جریان داده درون معماری Replication را توصیف می کند.

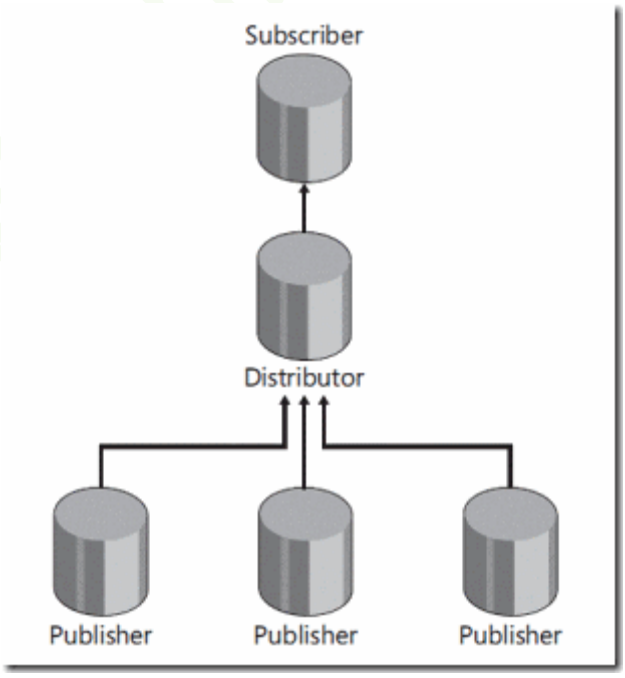
### توپولوزی منتشر کننده مرکزی

یک توپولوژی منتشر کننده مرکزی شامل یک منتشر کننده است که یک یا چند مشترک دارد. منتشر کننده مرکزی حاوی کپی master از داده ها است و برای بر پا سازی معماری Replication به کار می رود. در این توپولوژی، تغییرات داده معمولا در یک نقطه، یعنی منتشر کننده، رخ می دهد و سپس، همانطور که در شکل زیر امده، به یک یا چند مشترک جریان می یابد. منتشر کننده مرکزی متداول ترین توپولوژی مورد استفاده در Replication است.




## توپولوژی مشترک مرکزی

توپولوژی مشترک مرکزی شامل یک مشترک است که بیش از یک منتشر کننده دارد . همانطور که در تصویر پایین امده است، تغییرات به چند منتشر کننده نوشته شده و سپس در یک مشترک منسجم و یکی می شود. توپولوژی مشترک مرکزی معمولا برای منسجم و یکی کردن چندین پایگاه داده و یا به عنوان پایگاه داده گزارشگر مرکزی استفاده می شود.



## توپولوزی های دیگر

در منابع مختلف توپولوژی های Replication مختلفی مستند شده است، به طور ساده همه انها نمونه ای تغییر یافته از توپولوژی های منتشر کننده مرکزی و مشترک مرکزی هستند. یکی از این توپولوژی ها "توپولوژی دوطرفه" است که در واقع دو منتشر کننده مرکزی است که روی هم قرار گرفته و واقعا یک توپولوژی نیست؛ بلکه یک پیاده سازی ساختاری از Replication تراکنشی است. دو توپولوژی دیگر شامل یک منتشر کننده مرکزی همراه با یک توزیع کننده از راه دور (remote distributer) و یک مشترک مرکزی همراه با توزیع کننده از راه دور است. این توپولوژی ها نیز به ترتیب یک منتشر کننده مرکزی و مشترک مرکزی هستند؛ محل قرارگیری توزیع کننده یک موضوع مربوط به پیاده سازی فیزیکی است و در دیاگرام جریان فرایند business قرار نمی گیرد.

## Replication (Replication Agents)

هنگام اغاز کار باReplication ، بیشتر مردم از نحوه برخورد موتور Replicationبا حالت های مختلف عدم موفقیت (failure) گیج می شوند. روی هم رفته، SQL Serverنمی داند چگونه یک تراکنش را time out کند و یا یک عملیات را دوباره انجام دهد.

نکته اصولی که باید درباره Replication بدانیم این است که ان جزئی از موتور هسته SQL Server نیست . Replicationبا استفاده از یک سری فایل های اجرایی به نام عاملین Replication (replication agents) ، خارج از موتور SQL Serverکار می کند. به این صورت موتور Replication در واقع مانند برنامه های دیگری است که به SQL Server متصل و داده را پردازش می کند. به عنوان یک برنامه کاربردی، موتورReplication مانند هر برنامه دیگری که باید اتصالی Object Linking and Embedding database (OLE DB) باSQL Serverبرقرار کند، محدود است و مانند انها واکنش نشان می دهد.

## عامل Snapshot (Snapshot agent)

عامل Snapshot در واقع همان snapshot.exe است. این عامل مسئول استخراج داده و شمائی (schema) است که قرار است از منتشر کننده به مشترک فرستاده شود Snapshot.exe . در Replication ادغامی، تراکنشی و snapshot استفاده می شود.

## عامل Log Reader

عاملLog Reader ، همانlogread.exe ، فقط در Replication تراکنشی استفاده می شود. از ان برای استخراج تراکنش های commit شده از log تراکنشی موجود در منتشر کننده که باید replicate شود، استفاده می شود. بعد از استخراج تراکنش های commit شده، عامل Log Reader اطمینان حاصل می کند که تمام تراکنش ها، دقیقا به همان ترتیبی که در منتشر کننده صادر شده، دوباره در پایگاه داده توزیع نوشته می شود. بسیار حیاتی است که این تراکنش ها به همان ترتیب به مشترک اعمال شود.

## عامل توزیع (Distribution Agent)

عامل توزیع، distrib.exe، با Replication های تراکنشی و snapshot استفاده می شود. عامل توزیع دو کار انجام می دهد: اعمال کردن snapshot ها و ارسال تراکنش ها. عامل توزیع مسئول اعمال تمام snapshot های تولید شده در Replication تراکنشی و snapshot ، به تمام مشترک ها است. ان همچنین مسئول اعمال تمام تراکنش هایی که عامل Log Reader در پایگاه داده توزیع نوشته، به تمام مشترکین است.

## عامل ادغام (Merge Agent)

عامل ادغام، replmerg.exe، برای Replication ادغامی به کار می رود. عامل ادغام snapshot تولید شده هنگام initialize شدن مشترک را، اعمال می کند . عامل ادغام همچنین مسئول مبادله تراکنش ها بین منتشر کننده و مشترک است.

## عامل صف خوان (Queue Reader Agent)

عامل صف خوان، qrdrsvc.exe، فقط زمانی که گزینه آپدیت صف بندی شده (Queued Updating) فعال باشد مورد استفاده قرار می گیرد. عامل صف خوان مسئول انتقال صف روی مشترک به منتشر کننده است.

## (Agent Profile)

تمام عاملین Replication دارای پارامترهای زیادی برای پیکره بندی است که رفتار ان ها را تغییر می دهد. ۱۲ تا از متداول ترین گزینه ها با هم ترکیب و واحدی به نام پروفایل عامل تشکیل شده است. برخی از گزینه های متداول تری که می توانید پیکره بندی کنید در زیر امده است:

- Polling Interval – مشخص می کند عامل چند وقت یکبار دنبال تراکنش جدیدی برا replicate کردن بگردد. مقدار پیش فرض ۵ ثانیه است.
- Query timeout – مشخص می کند عامل چقدر منتظر اتمام یک پرس و جو بماند. مقدار پیش فرض ۱۸۰۰ ثانیه است.
- Login timeout - مشخص می کند عامل چقدر منتظر ساخت اتصال (connection) بماند. مقدار پیش فرض ۱۵ ثانیه است.

موتور replication سه روش متفاوت برای replicate کردن داده ها دارد: snapshot replication، replication تراکنشی و replication ادغامی.

## Snapshot replication

Snapshot replication تمام مجموعه داده ها را گرفته و در هر چرخه از موتور replication ان ها را ارسال می کند. این داده یک کپی کامل از داده ها است که به مشترک اعمال می شود. تمام تراکنش هایی که روی منتشر کننده رخ داده، گرفته و دفعه بعد که snapshot اجرا می شود، به مشترک فرستاده می شود.

Snapshot replication از عامل Snapshot و عامل توزیع استفاده می کند . هنگامی که snapshot اغاز می شود، عامل Snapshot شماء (schema) را استخراج و داده ها را به پوشه snapshot ، BCP (bulk copy program) می کند (به صورت دسته ای کپی می کند.)

توجه: پوشه snapshot

پوشه snapshot دایرکتوری است که هنگام پیکره بندی replication مشخص می کنید، و از ان به عنوان مکان پیش فرض snapshot استفاده می شود. هنگام ساخت یک publication ، برای ان publication می توانید محلی متفاوت را به عنوان پوشه snapshot مشخص کنید.

بعد از اینکه شماء و تمام داده ها استخراج شد، عامل Snapshot خاموش می شود (بسته می شود). از انجا عامل توزیع ادامه می دهد و تمام snapshot ها را به تمام مشترکین (Subscribers) اعمال می کند. طی این فرایند، جدول های موجود حذف و بر اساس اسکریپت های شماء موجود در پوشه snapshot دوباره ساخته می شوند؛ سپس با استفاده از کپی دسته ای (BCP) ، داده ها به درون جدول های تک تک مشترکین کپی می شود.

توجه: اِعمال یک snapshot

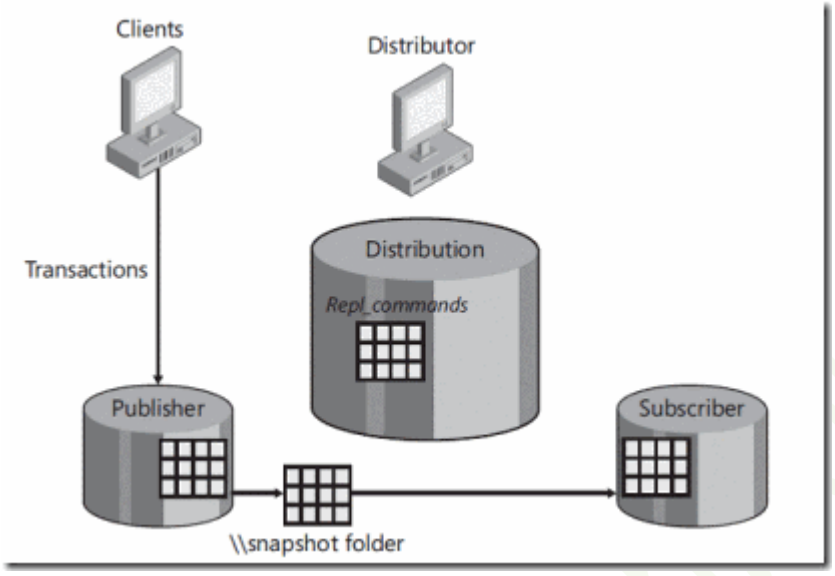
به طور پیش فرض، یک snapshot ساختار جدول، کلید اصلی، اندیس clustered ، constraint های یکتا و داده ها را به مشترک (Subscriber) اعمال می کند . بقیه اشیاء مرتبط با جدول مانند check constraint ها و foreign key constraint ها فرستاده نمی شود. با تغییر ویژگی های article (article properties) می توانید این رفتار پیش فرض را تغییر دهید.

هنگام اعمال یک snapshot چهار گزینه وجود دارد. گزینه پیش فرض حذف شی موجود و دوباره سازی ان است. می توانید انتخاب کنید که جدول موجود بدون تغییر بماند، داده های مشابه داده های snapshot ارسال شده، حذف شود، و یا ساختار جدول دست نخورد اما کوچک (truncate) شود تا فقط داده های snapshot را بپذیرد.

در اینجا ما از تنظیمات پیش فرض استفاده می کنیم.

یک دیاگرام از فرایند انتقال داده با snapshot replication در زیر امده است.

Snapshot replication تمام داده های مشترک را جایگزین می کند. از ان معمولا برای بالا بردن دسترس پذیری استفاده نمی شود، چون تراکنش هایی که بین اِعمال های snapshot به مشترک، صادر می شود، ارسال نمی شود.




## Replicationتراکنشی

هنگام اغاز Replicationتراکنشی، یک snapshot اولیه به مشترک (Subscriber)اعمال می شود تا مطمئن شویم که دو پایگاه داده با هم سنکرون هستند. با ادامه صدور تراکنش ها برای منتشر کننده، موتور replicationانها را به مشترک اعمال می کند. جریان افزایش (incremental) تراکنش ها از منتشر کننده به مشترک، replicationتراکنشی را گزینه ای مناسب برای نگهداری یک کپی ثانویه از پایگاه داده برای دسترس پذیری بیشتر و یا برای offload کردن عملیات های گزارش گیری، کرده است. متعول ترین پیکره بندی برای replicationتراکنشی در محیط های سرور_به_سرور است.

می توانید replication تراکنشی را در دو حالت اختیاری پیکره بندی کنید تا امکان انجام تراکنش بر روی مشترک فراهم شود. این دو حالت عبارتند از آپدیت بلادرنگ مشترکین و آپدیت صفی مشترکین. علاوه بر ارسال تراکنش ها از منتشر کننده به مشترک، replicationتراکنشی را می توان در دو معماری به کار برد replication : تراکنشی دوطرفه و replication تراکنشی نظیر به نظیر . (peer to peer)

## تراکنش ادغامی

تراکنش ادغامی به طور عمده برای پردازش های سیار و نامتصل (disconnected)طراحی شده است. در این روشreplication ، منتشر کننده و مشترک همیشه متصل نیستند.

همانند replication تراکنشی، برای اطمینان از سنکرون بودن، یک snapshot اولیه به مشترک اعمال می شود، و سپس تغییراتی که بعد رخ می دهد به مشترک ارسال می شود. بر خلاف replication تراکنشی، replication

ادغامی طراحی شده تا، به طور پیش فرض، اِعمال تغییرات هم در منتشر کننده و هم در مشترک را امکان پذیر سازد. موتور ادغام طی هر چرخه عامل، تغییرات ان دو را مبادله می کند.

نکته: چرخه عامل(Cycle of Agent)

Replicationرا می توان در دو حالت پیوسته (continuous) یا زمانبندی شده (scheduled)اجرا کرد. در حالت زمانبندی شده، عامل replication به صورت دوره ای اجرا می‌شود. در حالت پیوسته، موتور replication هممواره در حال اجراست. در هر دو حال، موتور replication در چرخه ی زیر کار می کند: برسی وجود تغییری برای replicate کردن، انتقال تغییرات به مشترک، تایید وصول (receipt)تغییرات. به این فرایند فرایند چرخه عامل replication می گویند . در حالت زمانبندی شده، چرخه عامل واضح تر است چون عامل شروع می شود، کاری را انجام می دهد و بعد بسته می‌شود. این چرخه در حالت پیوسته خیلی واضح نیست چون عامل هیچ گاه بسته نمی شود، در عوض به محض تکمیل یک چرخه، چرخه بعدی را اغاز می کند. برای درک بسیاری از حالت هایی که ممکن است پیش بیاید – مهمترین انها ناسازگاری داده ها – (data conflict) فهمیدن مفهوم چرخه عامل بسیار مهم است.

## (Data Conflicts)

در تمام محیط هایی که امکان پردازش توزیع شده تراکنش ها وجود دارد ممکن است به ناسازگاری داده ها بر بخوریم. وقتی می توان یک داده را در مکان های مختلف تغییر داد، مکانیسمی باید طراحی شود تا پردازش تغییرات را مدییریت کند.

برنامه هایی که تغییرات یک پایگاه داده را پردازش می کنند روشی برای پردازش تغییرات ناسازگار دارند – یا تمام تغییرات را با اخرین تغییر رونویسی (overwrite) می کنند و یا با رد کردن تغییرات به کاربر اطلاع می دهند که داده از اخرین زمان دستیابی، تغییر کرده است. هر چند این فرایند ها در سطح برنامه های کاربردی عملی اند ولی در محیط های توزیع شده کمکی نمی کنند چون هر برنامه روی یک کپی محلی (local) از داده ها کار می کند . ناسازگاری داده ها فقط زمانی رخ می دهد که موتور replication سعی می کند تمام تغییرات را سنکرون کند.

ناسازگاری داده ها تنها بین رخه های عامل replication رخ می دهد، بنابراین احتمال رخداد ان مینیمم است. بعد از اتمام یک چرخه عامل، می توان از مکانیسم های آشکار ساز (detection mechanisms) عادی درون برنامه های کاربردی استفاده کرد، چون تمام مجموعه داده ها برای برنامه به شکل محلی (local)است.

ناسازگاری داده ها فقط در replication ادغامی، replicationتراکنشی با آپدیت صفی مشترکین، replication تراکنشی دوطرفه و replication تراکنشی نظیر به نظیر رخ می دهد، چون در این حالات، تغییرات را می توان هم روی منتشر کننده و هم روی مشترکین پردازش کرد.

## انواع ناسازگاری ها

3نوع ناسازگاری ممکن است رخ دهد:

۱. افزودن کلید اصلی (primary key) تکراری، زمانی رخ می دهد که دو کاربر روی منتشر کننده و مشترک، یک کلید اصلی را وارد می کنند.
۲. آپدیت ناسازگار، زمانی رخ می دهد که دو کاربر یک سطر داده را، یکی از روی منتشر کننده و دیگری از روی مشترک تغییر می دهند.
۳. آپدیت کردن سطری که وجود ندارد، زمانی رخ می دهد که یک کاربر یک سطر داده را از یک طرف معماری replicationآپدیت و دیگری همان سطر را از طرف دیگر حذف می کند.

## حلال ناسازگاری ها

موتور replication موظف است یک کپی سازگار و منسجم ار داده ها را بین منتشر کننده و مشترک نگه داری کند. ناسازگاری داده ها مانع بزرگی بر سر نگهداری انسجام و سازگاری داده ها است replication . ادغامی و replicationتراکنشی با آپدیت صفی مشترکین دارای مکانیسمی برای کشف و حل ناسازگاری ها هستند.

بخشی که کار حل ناسازگاری ها را انجام می دهد را حلال ناسازگاری ها (Conflict Resolver) می گویند SQL . Serverدارای چندین حلال ناسازگاری است. دو نمونه متداول، که برای

replicationادغامی و replication تراکنشی با آپدیت صفی مشترکین قابل استفاده است، عبارتند از:

- منتشر کننده همیشه برنده است .
- مشترک همیشه برنده است

اگر طوری حلال ناسازگاری را تنظیم کنید که همیشه منتشر کننده برنده باشد، تغییرات روی منتشر کننده همواره تغییرات روی مشترک را override می کند. در این حالت، تغییری که روی مشترک رخ داده، در منتشر کننده دور انداخته شده و در یک جدول ناسازگاری ها ثبت (log) می شود و تغییری که روی منتشر کننده رخ داده به مشترک ارسال می شود. این، تغییرات رخ داده روی مشترک را بی اثر می کند.

اگر طوری حلال ناسازگاری را تنظیم کنید که همیشه مشترک برنده باشد، تغییرات روی مشترک همواره تغییرات روی منتشر کننده را override می کند.

استفاده از هر روش تضمین می کند که یک کپی سازگار و منسجم از داده ها در معماری replication نگهداری می شود. با این حال، یک وضعیت خطیر ممکن است پیش بیاید. تغییراتی که روی منتشر کننده و مشترک انجام شده بود تراکنش هایی معتبر بوده و commit شده بودند. ممکن است یک کاربر دیگر اطلاعات را گرفته و بر اساس ان داده ها تصمیم تجاری مهمی گرفته باشد. سپس موتور replication داده ها را مبادله می کند، یک ناسازگاری کشف و داده ها overwrite می شود .از دید تجاری، تصمیم ان کاربر ممکن است الان نامعتبر باشد.

برای اینکه بتوانیم یک کپی منسجم و سازگار از داده ها را در معماری مان نگهداری کنیم، این ناسازگاری ها باید کشف و حل شوند. با این حال، به عهده طراح برنامه است که برای حفظ جامعیت (integrity) تصمیم های تجاری،

تضمین کند تضاد و ناسازگاری در محیط های با پردازش توزیعی رخ نمی دهد. تضاد و ناسازگاری داده ها باید در شرکت شما نامتعارف و غیر قابل قبول باشد.

تذکر: تراکنش هایی که تا کمترین حد ثبت (log) می شوند

اگر پایگاه داده در یک replication شرکت دارد، باید به شدت مواظب مدل های Bulk-logged و Simple recovery باشید. وقتی یک پایگاه داده در مدل های Bulk-logged و Simple recovery قرار داده می شود، تراکنش هایی اجرا می شود که به کمترین حد ثبت می شوند. این نوع تراکنش ها فقط تخصیص و آزادسازی صفحه را در log تراکنش ثبت می کنند؛ انها trigger ها را فعال نمی کنند.

5تراکنش از این نوع عبارتند از:

۱. CREATE INDEX
۲. TRUNCATE TABLE
۳. BULK INSERT
٤. BCP
٥. SELECT...INTO

Replication فقط با ۳ تا از اینها برخورد دارد TRUNCATE TABLE – ، BULK INSERT، – BCP چون انها بر داده جدول ها تاثیر می گذارند. اگر یک پایگاه داده در مدل های Bulk-logged و Simple recovery قرار داده شود و یکی از این دستور ها اجرا شود، موتور Replication نمی تواند تغییرات را بیابد چون replication تراکنشی به تراکنش های ثبت شده در log تراکنش ها، و replication ادغامی به trigger ها متکی اند.

## (Publishing)

در این تمرین، انتشار را روی نمونه SQL Server تان پیکره بندی می کنید.

۱. SQL Server Management Studio (SSMS) را باز کنید و در Object Browser به نمونه SQL Server تان متصل شوید .
۲. روی گره Replication کلیک راست کرده و Configure Distribution را انتخاب کنید. روی Next کلیک کنید .
۳. اولین گزینه را انتخاب کنید تا نمونه تان به عنوان توزیع کننده خودش کار کند Next . را کلیک کنید .
٤. مقدار پوشه snapshot را پیش فرض گذاشته، روی Next کلیک کنید .
٥. مانند تصویر زیر، نام و مکان پایگاه داده توزیع را مقادیر پیش فرض گذاشته و روی Next کلیک کنید .


۶. مطمئن شوید نمونه SQL Server تان به عنوان منتشر کننده (publisher) انتخاب شده است Next . را بزنید .

۷. بررسی کنید Configure Distribution تیک خورده باشد. روی Next کلیک کنید .

۸. روی Finish کلیک کنید تا انتشار فعال شود، بعد Close را بزنید .

بررسی کنید یک پایگاه داده به نام Distribution در نمونه SQL Server تان ساخته شده باشد.

با ترکیب یک یا چند article و تشکیل یک publication می توان مجموعه داده ای برای انتقال توسط موتور Replicationتعریف کرد.

یک پایگاه داده می تواند نقش منتشر کننده، مشترک و یا هر دو را بازی کند.

3نوع replication وجود دارد snapshot :، تراکنشی و ادغامی.

تمام کارهای موتور replication را پنج عامل انجام می دهند: عامل Snapshot، عاملLog Reader ، عامل توزیع، عامل ادغام و عامل صف خوان.(Queue Reader)

در حالاتی که امکان اعمال تغییر هم در منتشر کننده و هم در مشترک وجود دارد، ممکن است ناسازگاری و تضاد داده ها رخ دهد.