معماری SQL server

- اهداف
- مبانی معماری SQL Server
- ویژگی‌های SQL Server
- معماری پایگاه داده
- معماری سرویس‌دهنده

۱-۳- اهداف

تمام وجوه و اجزاء SQL Server برای افزایش اثر بخشی آن به عنوان یک پایگاه داده بر مبنای SQL، سرویس‌دهنده/ سرویس‌گیرنده و رابطه‌ای با همدیگر کار می‌کنند. مباحث مطرح شده در این فصل چگونگی همکاری اجزاء مختلف SQL Server برای پردازش مؤثر داده‌ها را تشریح می‌کند. همچنین ساختار SQL Server و مفاهیم اصلی آن بررسی خواهد شد.

۲-۳- مبانی معماری SQL Server

SQL Server، پایگاه داده‌ای بر مبنای زبان پرس و جوی ساخت یافته، سرویس‌دهنده/ سرویس‌گیرنده و رابطه‌ای است. هر یک از این مفاهیم، یک قسمت اساسی معماری SQL Server را توصیف می‌کنند.

پایگاه داده

پایگاه داده از جهت مکانی برای ذخیره سازی داده‌ها مشابه فایل داده‌ای می‌باشد. پایگاه داده همانند فایل داده‌ای، اطلاعات را مستقیماً به کاربر ارائه نمی‌دهد. در عوض، کاربر با اجرای یک برنامه کاربردی که به داده‌های پایگاه داده دسترسی می‌یابد داده‌ها را با قالبی قابل فهم استخراج می‌کند.

سیستم‌های پایگاه داده، قوی‌تر از فایل داده‌ای هستند. داده‌ها در این سیستم‌ها از سازمان بهتری برخوردار هستند. در پایگاه داده‌ای که خوب طراحی شده باشد پدیده افزونگی داده‌ها وجود ندارد. قسمت‌هایی از داده که به هم مربوط هستند در یک ساختار یا رکورد واحد گروه‌بندی می‌شوند و می‌توان بین این ساختارها و رکوردها ارتباط‌هایی را تعریف کرد.

وقتی که با فایل‌های داده‌ای کار می‌کنیم، برای برخورد با ساختار ویژه هر یک از آنها باید برنامه‌ای تهیه شود. در عوض، پایگاه داده حاوی یک کاتالوگ است که برنامه‌های کاربردی برای تعیین سازماندهی داده‌ها از آن استفاده می‌کنند.

نوعاً یک پایگاه داده دارای دو جزء می‌باشد: فایل‌هایی که حاوی پایگاه داده

فیزیکی هستند و نرم افزار سیستم مدیریت پایگاه داده (DBMS) که برنامه‌های کاربردی برای دستیابی به داده‌ها از آن استفاده می‌کنند. DBMS مسئول پیاده‌سازی ساختار پایگاه داده می‌باشد، شامل:

- نگهداری ارتباط‌های بین داده‌ها در پایگاه داده.
- تضمین این که داده‌ها به درستی ذخیره شده‌اند و قواعدی که ارتباط بین داده‌ها را تعریف می‌کنند خدشه‌دار نشده‌اند.
- ترمیم تمام داده‌ها به حد شناخته شده‌ای از ثبات استحکام در مواقع بروز اشکال در سیستم.

پایگاه داده رابطه‌ای

راه‌های مختلفی برای سازماندهی داده‌ها در یک پایگاه داده وجود دارد، اما پایگاه‌های داده رابطه‌ای یکی از مؤثرترین آنهاست. سیستم‌های پایگاه دادهٔ رابطه‌ای در واقع کاربردی از تئوری مجموعهٔ ریاضی در مسئله سازماندهی مؤثر داده‌ها می‌باشد. در یک پایگاه داده رابطه‌ای داده‌ها در جدول‌ها جمع‌آوری می‌شوند (در تئوری رابطه‌ای از جدول به رابطه اطلاق می‌شود).

جدول، کلاس‌هایی از شیءها را که برای یک سازمان اهمیت دارند، نمایش می‌دهد. به عنوان مثال، یک شرکت ممکن است دارای یک پایگاه داده باشد که یک جدول برای کارمندان، یک جدول برای مشتریان و یک جدول برای فروشگاه‌ها می‌باشد. هر جدول شامل ستون و سطر است (در تئوری رابطه‌ای به ترتیب صفت و تاپل نامیده می‌شوند). هر ستون نشان دهنده برخی از صفات شیءای است که توسط جدول نمایش داده می‌شود. به عنوان مثال، جدولی که حاوی اطلاعات کارمندان است دارای ستون‌هایی برای نام و نام خانوادگی، شماره کارمندی، حقوق و عنوان شغل می‌باشد. هر سطر، نشان دهنده یک نمونه از شیءای است که توسط جدول نمایش داده شده است. به عنوان مثال، یک سطر در جدول اطلاعات کارمندان نشان دهنده کارمندی با شماره کارمندی ۱۲۳۴۵ است.

هنگام سازماندهی داده‌ها در جدول‌ها، معمولاً راه‌های بسیار مختلفی برای تعریف جدول‌ها وجود دارد. تئوری پایگاه داده رابطه‌ای فرآیندی به نام نرمال‌سازی را تعریف

می‌کند که تضمین می‌نماید مجموعه جدول‌هایی که در یک پایگاه داده تعریف شده است داده‌ها ر ا به طور مؤثر سازماندهی خواهد کرد.

سرویس‌دهنده / سرویس‌گیرنده

در یک سیستم سرویس‌دهنده / سرویس‌گیرنده، سرویس‌دهنده یک کامپیوتر بزرگ در یک واحد مرکزی است که یک منبع اطلاعاتی مورد استفاده بسیاری از افراد را مدیریت می‌کند. اگر فردی بخواهد از این منبع اطلاعاتی استفاده کند، می‌تواند از طریق شبکه بوسیله کامپیوتر یا سرویس‌گیرنده خود به سرویس‌دهنده متصل شود. نمونه‌هایی از سرویس‌دهنده‌ها به صورت زیر است:

- سرویس‌دهنده‌های چاپ: چاپگرهای مورد استفاده یک واحد را مدیریت می‌کنند.
- سرویس‌دهنده‌های فایل: فایل‌های بزرگ مورد استفاده یک واحد را با استفاده از دیسک‌های بزرگ ذخیره می‌کنند.
- سرویس‌دهنده‌های E-mail: سیستم E-mail یک شرکت را اجرا می‌کنند.

در یک معماری پایگاه داده سرویس دهنده/ سرویس‌گیرنده فایل‌های پایگاه داده و نرم‌افزار DBMS بر روی یک سرویس‌دهنده قرار دارند. یک جزء ارتباطی وجود دارد به طوری که برنامه‌های کاربردی می‌توانند بر روی سرویس‌گیرنده‌های مجزا اجرا شده و از طریق یک شبکه با سرویس دهنده پایگاه داده ارتباط برقرار کنند. جزء ارتباطی SQL Server ارتباط بین یک برنامه کاربردی که روی سرویس دهنده اجرا می‌شود و SQL Server را نیز میسر می‌سازد.

برنامه‌های کاربردی سرویس‌دهنده معمولاً قادر به کار با چندین سرویس‌گیرنده به طور همزمان می‌باشند. SQL Server می‌تواند همزمان با هزاران برنامه کاربردی سرویس‌گیرنده کار کند. سرویس‌دهنده دارای ویژگی‌هایی است که از بروز اشکالات منطقی وقتی کاربر سعی می‌کند داده‌های مورد استفاده کاربران دیگر را همزمان خوانده یا تغییر دهد، جلوگیری می‌کند.

در حالی که SQL Server برای کار به عنوان یک سرویس‌دهنده در یک شبکه

سرویس‌دهنده / سرویس‌گیرنده طراحی شده است، مستقیماً قادر به کار به عنوان یک پایگاه داده واحد بر روی سرویس گیرنده نیز می‌باشد. قابل تنظیم بودن و سادگی استفاده از SQL Server به آن امکان می‌دهد که بر روی یک سرویس‌گیرنده، بدون مصرف بیش از حد منابع به طور مؤثر کار کند.

برای کار با داده‌ها در یک پایگاه داده، باید از مجموعه‌ای از دستورات و عبارات که توسط نرم‌افزار DBMS تعریف شده است، استفاده کرد. زبان‌های مختلفی برای استفاده در پایگاه‌های داده رابطه‌ای وجود دارد که معمول‌ترین آنها SQL است.

۳-۳- ویژگی‌های SQL Server

SQL Server دارای ویژگی‌هایی است که به صورت زیر طبقه‌بندی می‌شوند:

- سهولت نصب و استفاده: بسیاری از پایگاه‌های داده‌ای که قادر به جوابگویی تمام نیازهای پردازشی یک مجموعه هستند، اغلب پیچیده و مدیریت آنها مشکل است. SQL Server شامل بسیاری از ابزار و ویژگی‌هایی است که توانایی شما را در نصب، مدیریت و استفاده از پایگاه داده سهولت می‌بخشند.

SQL Server به طور خودکار و پویا در هنگام اجرا خود را با شرایط موجود هماهنگ می‌کند. اگر کار بیشتری در حال انجام باشد به طور پویا منابع بیشتری بدست می‌آورد (مانند حافظه). وقتی میزان کار کاهش می‌یابد SQL Server منابع را آزاد کرده و به سیستم باز می‌گرداند.

SQL Server همچنین می‌تواند اندازهٔ پایگاه داده را همچنانکه داده‌ها درج شده و یا حذف می‌شوند به طور پویا افزایش یا کاهش دهد. میزان تنظیم پویا در هنگام نصب SQL Server می‌تواند توسط مدیران پایگاه داده کنترل شود. پایگاه داده کوچکی که توسط افرادی استفاده می‌شود که آشنایی زیادی با پایگاه داده‌ها ندارند می‌توانند با تنظیمات پیش فرض اجرا شود. در این حالت، تنظیمات مربوطه به طور پویا انجام می‌شود. پایگاه داده بزرگی که نظارت آن بر عهده مدیران با تجربه است می‌تواند طوری تشکیل شود که مدیران کنترل کاملی بر تنظیمات مربوط داشته باشند. SQL Server ابزارهای

متعددی را برای مدیریت سیستم به مدیران پایگاه داده ارائه می‌دهد (مانند SQL Server Enterprise Manager).

- قابلیت تنظیم: موتور پایگاه داده در SQL Server یک سرویس‌دهنده نیرومند است که می‌تواند پایگاه‌های داده بزرگی را که توسط هزاران کاربر دستیابی می‌شوند مدیریت نماید.

قفل‌گذاری در SQL Server به صورت پویا انجام می‌شود. وقتی یک پرس و جو به تعداد اندکی از سطرها در یک جدول بزرگ ارجاع می‌کند، بهترین راه برای به حداکثر رساندن دستیابی همزمان به داده‌ها، قفل‌گذاری سطرها است. اما اگر یک پرس و جو به بیشترین یا تمام سطرهای یک جدول ارجاع کند، بهترین راه برای به حداکثر رساندن دستیابی همزمان به داده‌ها، قفل‌گذاری کل جدول است تا سربار قفل‌گذاری به حداقل برسد. SQL Server دستیابی همزمان به داده‌ها را با انتخاب مناسب سطوح قفل‌گذاری برای هر جدول در هر پرس و جو به حداکثر می‌رساند.

بهینه‌ساز پرس و جو در SQL Server دارای روش‌های دستیابی نوینی است که پردازش پرس و جو را سرعت می‌بخشند.

SQL Server به طور کامل از جامعیت پایگاه‌های داده خود حفاظت می‌کند. تمام بهنگام‌سازی‌ها در تراکنش‌ها انجام می‌شود. تراکنش‌ها یا به تمامی تأیید شده و یا لغو می‌شوند. اگر یک سرویس دهنده با اشکال مواجه شود، تمام تراکنش‌های تکمیل نشده هنگام شروع به کار مجدد سرویس‌دهنده، از تمام پایگاه‌های داده SQL Server لغو می‌شوند.

- مجتمع بودن سیستم: SQL Server با تولیدات و محیط‌های دیگری مجتمع می‌باشد تا یک محل با ثبات و امن ذخیره‌سازی داده‌ها برای سیستم‌های اینترنت و اینترانت را تشکیل دهد.

۴-۳- معماری سرویس دهنده / سرویس‌گیرنده

SQL Server طوری طراحی شده است که در محیط‌های زیر به طور مؤثر کار کند:

- سیستم پایگاه داده سرویس‌دهنده / سرویس‌گیرنده.

- سیستم پایگاه داده **Desktop**.

سیستم‌های سرویس‌دهنده / سروس گیرنده طوری ساخته شده‌اند که پایگاه داده بر روی یک کامپیوتر مرکزی به نام سرویس‌دهنده قرار می‌گیرد و بین چندین کاربر به اشتراک گذاشته می‌شود. کاربران از طریق یک برنامه کاربردی سرویس‌گیرنده یا سرویس‌دهنده، به سرویس‌دهنده دسترسی می‌یابند. ذخیره سازی و مدیریت داده‌ها در یک مکان مرکزی دارای مزایای زیر است:

- هر قلم داده در یک مکان مرکزی ذخیره می‌شود و تمام کاربران می‌توانند با آن کار کنند.
- قواعد امنیتی فقط یک بار بر روی سرویس دهنده تعریف می‌شود و به طور یکسان به تمام کاربران اعمال می‌شود. این کار در پایگاه داده توسط محدودیت‌ها، رویه‌های ذخیره شده و **Trigger**ها انجام می‌شود.
- سرویس‌دهنده پایگاه داده رابطه‌ای ترافیک شبکه را با بازگرداندن فقط داده‌هایی که برنامه کاربردی به آن نیاز دارد، بهینه می‌کند. به عنوان مثال، اگر یک برنامه کاربردی که با یک سرویس‌دهنده فایل کار می‌کند بخواهد لیست اسامی خاصی را نمایش دهد، می‌بایست کل فایل مربوط به کارمندان را واکشی نماید. اگر برنامه کاربردی با یک سرویس دهندهٔ پایگاه داده رابطه‌ای کار می‌کند، این دستور را ارسال می‌کند:

```
SELECT first_name, last_name
FROM employees
WHERE emp_title = 'Sales Representative'
    AND emp_state = 'OR'
```

پایگاه داده رابطه‌ای صرفاً اسامی مربوطه را باز می‌فرستد و نیازی ندارد که اطلاعات مربوط به تمام کارمندان را ارسال کند.

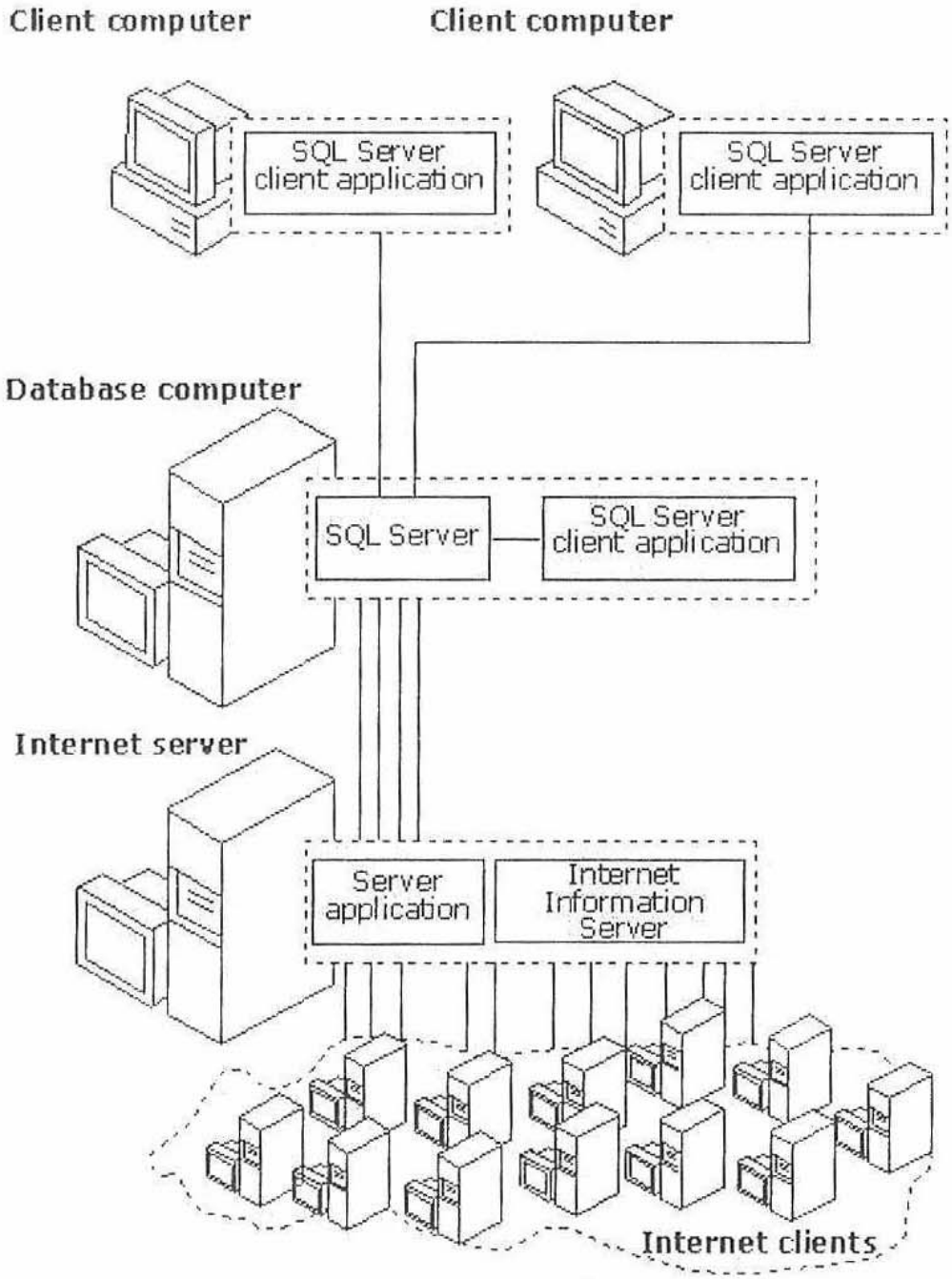
- هزینه‌های سخت افزاری به حداقل می‌رسد. چون داده‌ها روی

سرویس‌گیرنده‌ها ذخیره نمی‌شوند، نیازی به فضای زیاد دیسک ندارند. همچنین برای مدیریت محلی داده‌ها نیازی به ظرفیت بالای پردازش ندارند. سرویس‌دهنده نیز نیازی ندارد تا از قدرت نمایشی بالایی برخوردار باشد. سرویس‌دهنده می‌تواند طوری تنظیم شود تا ظرفیت ورودی/خروجی دیسک برای واکشی داده‌ها بهینه شود. سرویس‌گیرنده‌ها نیز می‌توانند طوری تنظیم شوند تا قالب بندی و نمایش داده‌های واکشی شده بهینه شود.

- فعالیت‌های مربوط به نگهداری سیستم مانند تهیه نسخه پشتیبان و بازیابی داده‌ها به راحتی صورت می‌گیرد، زیرا می‌توانند بر روی یک سرویس‌دهنده مرکزی تمرکز یابند.

در سیستم‌های بزرگ سرویس‌دهنده / سرویس‌گیرنده هزاران کاربر ممکن است به یک SQL Server به طور همزمان متصل باشند. SQL Server دارای حفاظت کاملی برای این گونه محیط‌ها می‌باشد. SQL Server همچنین منابع موجود مانند حافظه، شبکه و ورودی/ خروجی دیسک را به طور مؤثر به کاربران متعدد تخصیص می‌دهد.

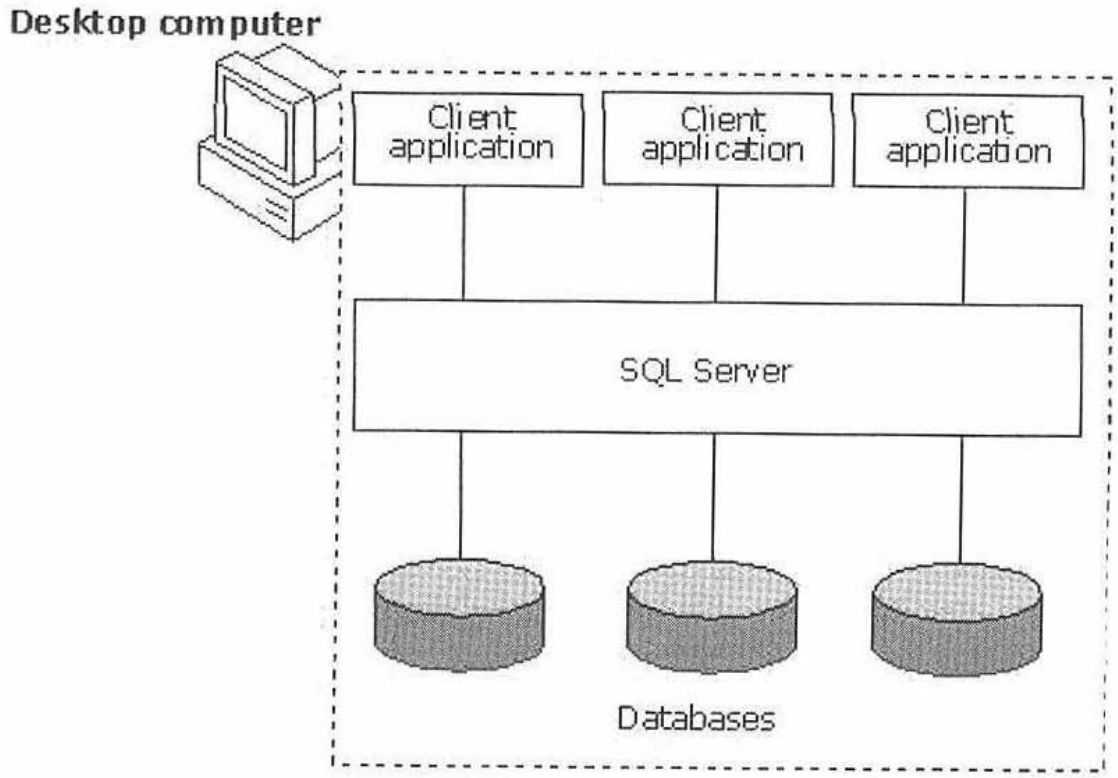
برنامه‌های کاربردی SQL Server می‌تواند بر روی همان کامپیوتری که SQL Server در آن قرار دارد اجرا شوند. این برنامه‌ها برای اتصال به SQL Server به جای استفاده از شبکه، از اجزاء ارتباطی بین فرآیندی ویندوز مانند حافظهٔ مشترک استفاده می‌کنند. این امر به SQL Server امکان می‌دهد تا بر روی سیستم‌های کوچکی که داده‌ها لازم است به طور محلی ذخیره شوند، مورد استفاده قرار گیرد.



«شکل ۱-۳: سیستم پایگاه داده سرویس‌دهنده / سرویس‌گیرنده»

با این که **SQL Server** به طور مؤثر به عنوان یک سرویس‌دهنده کار می‌کند، درعین حال می‌تواند در برنامه‌های کاربردی که لازم است پایگاه داده به طور محلی در سرویس‌گیرنده ذخیره شود، مورد استفاده قرار گیرد. **SQL Server** می‌تواند به طور پویا خود را طوری پیکربندی نماید تا با استفاده از منابع موجود بر روی سرویس‌گیرنده بدون نیاز به مدیر پایگاه داده اجرا شود.

وقتی سرویس‌گیرنده‌ها از پایگاه‌های داده محلی **SQL Server** استفاده می‌کنند، یک کپی از موتور پایگاه دادهٔ **SQL Server** بر روی سرویس‌گیرنده اجرا می‌شود و تمام پایگاه‌های داده **SQL Server** را بر روی سرویس‌گیرنده مدیریت می‌کند.



«شکل ۲-۳: سیستم پایگاه داده Desktop»

loginها

در بیشتر مواقع، برنامه‌های کاربردی برای اتصال به SQL Server به دو فقره اطلاع نیاز دارند:

- نام شبکه‌ای سرویس‌دهنده‌ای که SQL Server بر روی آن اجرا می‌شود.
- شناسه login.

شناسه‌های login، شناسه‌هایی هستند که دستیابی به سیستم SQL Server را کنترل می‌کنند. SQL Server اتصال را برقرار نمی‌کند مگر این که اطمینان حاصل کند شناسه login که شما مشخص کرده‌اید مجاز می‌باشد. این اطمینان از صحت login، تأیید صلاحیت نامیده می‌شود.

دو نوع تأیید صلاحیت وجود دارد که هر کدام دارای کلاس مختلفی از شناسه login می‌باشد:

- تأیید صلاحیت توسط SQL Server: عضوی از نقش سرویس‌دهنده sysadmin، ابتدا تمام loginها و کلمات عبور مجاز را برای SQL Server مشخص می‌کند. این اطلاعات ربطی به حساب‌های کاربر ویندوز یا شبکه ندارند. هنگام اتصال به SQL Server، هم login و هم کلمه عبور باید مشخص شود.

- تأیید صلاحیت توسط ویندوز NT: عضوی از نقش سرویس‌دهنده sysadmin، ابتدا تمام حساب‌ها و گروه‌های ویندوز NT را که می‌توانند به SQL Server متصل شوند، برای SQL Server مشخص می‌کند. با استفاده از این روش، هنگام اتصال به SQL Server لازم نیست شناسه login و کلمه عبور مشخص شود. دستیابی به SQL Server بوسیله حساب یا گروه ویندوز NT کنترل می‌شود که این حساب یا گروه هنگام Logon شدن به سیستم عامل ویندوز روی سرویس‌گیرنده تأیید صلاحیت می‌شود. هنگام اتصال به SQL Server، نرم افزار سرویس‌گیرنده، یک اتصال مطمئن ویندوز NT به SQL Server را درخواست می‌کند. تا وقتی که سرویس‌گیرنده با استفاده از یک حساب مجاز ویندوز NT با موفقیت logon نشده باشد، ویندوز NT اتصال مطمئن را بوجود نمی‌آورد. خواص یک اتصال مطمئن شامل گروه و حساب‌های کاربر ویندوز NT بر روی سرویس‌گیرنده است که اتصال را بوجود آورده است. تا وقتی که ویندوز NT ابتدا صلاحیت کاربر را تأیید نکرده باشد اتصال مطمئن بوجود نمی‌آید، لذا SQL Server لازم نیست برای تأیید صلاحیت حساب‌های شما کاری انجام دهد. SQL Server اطلاعات حساب کاربر را از خواص اتصال مطمئن بدست می‌آورد و آنها را با حساب‌های ویندوز NT که به عنوان loginهای مجاز SQL Server تعریف شده‌اند، مطابقت می‌دهد. اگر SQL Server تطابقی پیدا کند، اتصال را قبول می‌کند. در واقع شما با گروه یا حساب کاربر ویندوز NT خود در SQL Server شناسایی می‌شوید.

اگر SQL Server بر روی ویندوز NT اجرا شود، اعضای نقش sysadmin می‌توانند یکی از حالات تأیید صلاحیت زیر را مشخص کنند:

- تأیید صلاحیت توسط ویندوز NT: تأیید صلاحیت فقط توسط ویندوز NT انجام می‌شود. کاربران نمی‌توانند شناسهٔ login برای SQL Server مشخص کنند.
- حالت مخلوط: اگر کاربر هنگام اتصال به SQL Server یک شناسه login مشخص کند، تأیید صلاحیت توسط SQL Server انجام می‌شود. اگر شناسه login مشخص نشود، توسط ویندوز NT تأیید صلاحیت می‌شود.

حالت‌های فوق می‌توانند در Enterprise Manager مشخص شوند.

وقتی SQL Server بر روی ویندوز ۹۵/۹۸ اجرا می‌شود، تأیید صلاحیت توسط ویندوز NT را پشتیبانی نمی‌کند. کاربران هنگام اتصال باید یک login مربوط به SQL Server را مشخص کنند. وقتی SQL Server بر روی ویندوز NT اجرا می‌شود، سرویس‌گیرنده‌های ویندوز ۹۵/۹۸ می‌توانند با استفاده از تأیید صلاحیت توسط ویندوز NT به آن متصل شوند.

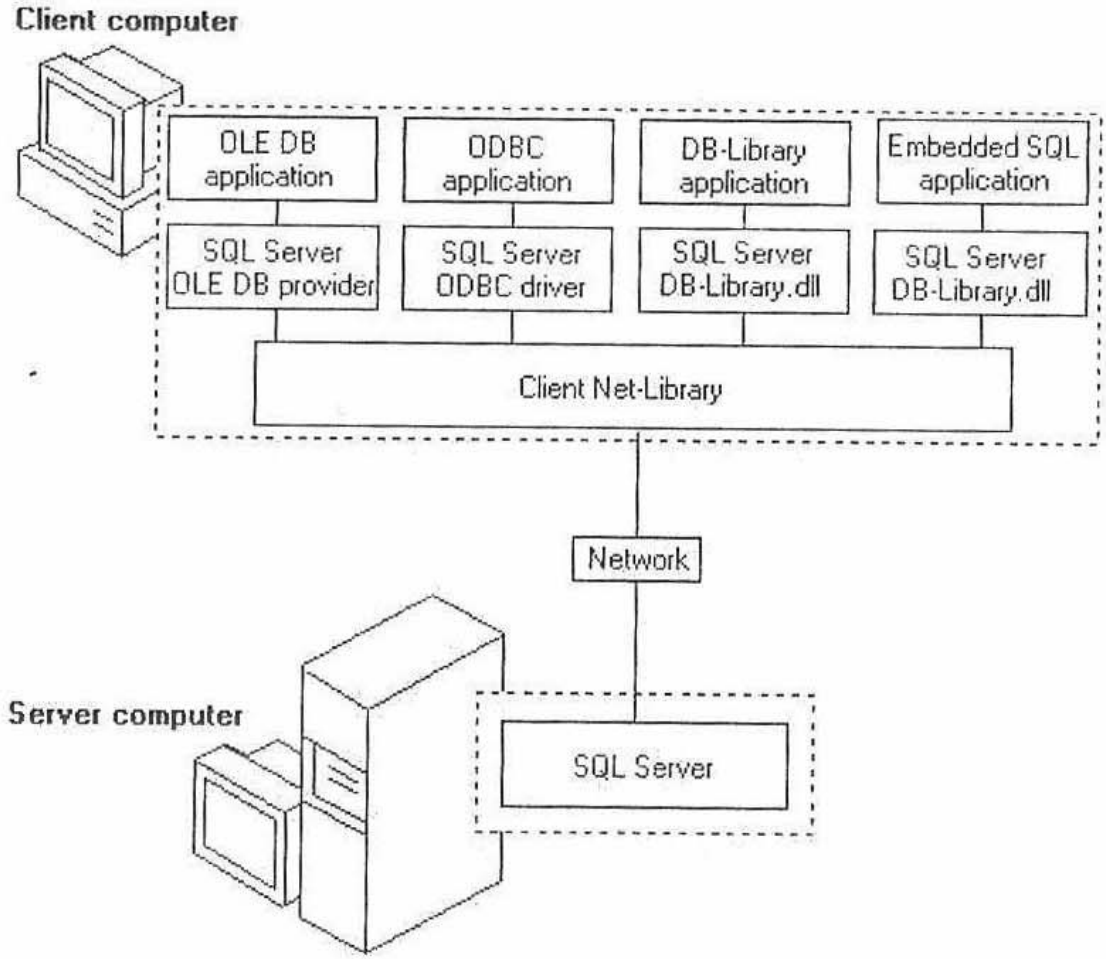
یکی از خواص login، پایگاه داده پیش فرض است. وقتی که کاربر به SQL Server متصل می‌شود، این پایگاه داده پیش فرض برای اتصال به عنوان پایگاه داده جاری در نظر گرفته می‌شود مگر این که درخواست اتصال، پایگاه داده دیگری را به عنوان پایگاه داده جاری مشخص کند.

اجزاء سرویس‌گیرنده

کاربران مستقیماً به SQL Server دسترسی ندارند، در عوض برای دستیابی به داده‌ها از یک برنامه کاربردی که برای این منظور تهیه شده است استفاده می‌کنند. این برنامه‌های کاربردی شامل برنامه‌های کمکی همراه SQL Server و یا برنامه‌هایی است که توسط برنامه نویسان نوشته شده است. برنامه‌های کاربردی که برای دستیابی به SQL Server نوشته می‌شوند، در رابط برنامه‌نویسی کاربردی پایگاه داده (API) تولید می‌شوند. یک API پایگاه داده از دو قسمت تشکیل شده است:

- دستوراتی که به پایگاه داده ارسال می‌شوند. این دستورات در SQL Server به زبان Transact-SQL شهرت دارند.
- مجموعه‌ای از توابع یا رابط‌ها و متدهای شئ‌گرا که برای ارسال دستورات به پایگاه داده و پردازش نتایج بازگشتی توسط پایگاه داده استفاده می‌شوند.

APIهای پایگاه داده به چهار دسته کلی OLEDB، ODBC، DB-Library و Embedded SQL طبقه‌بندی می‌شوند (شرح جزئیات مربوط به این APIها خارج از حیطه این کتاب است).



«شکل ۳-۳: اجزاء سرویس‌گیرنده»

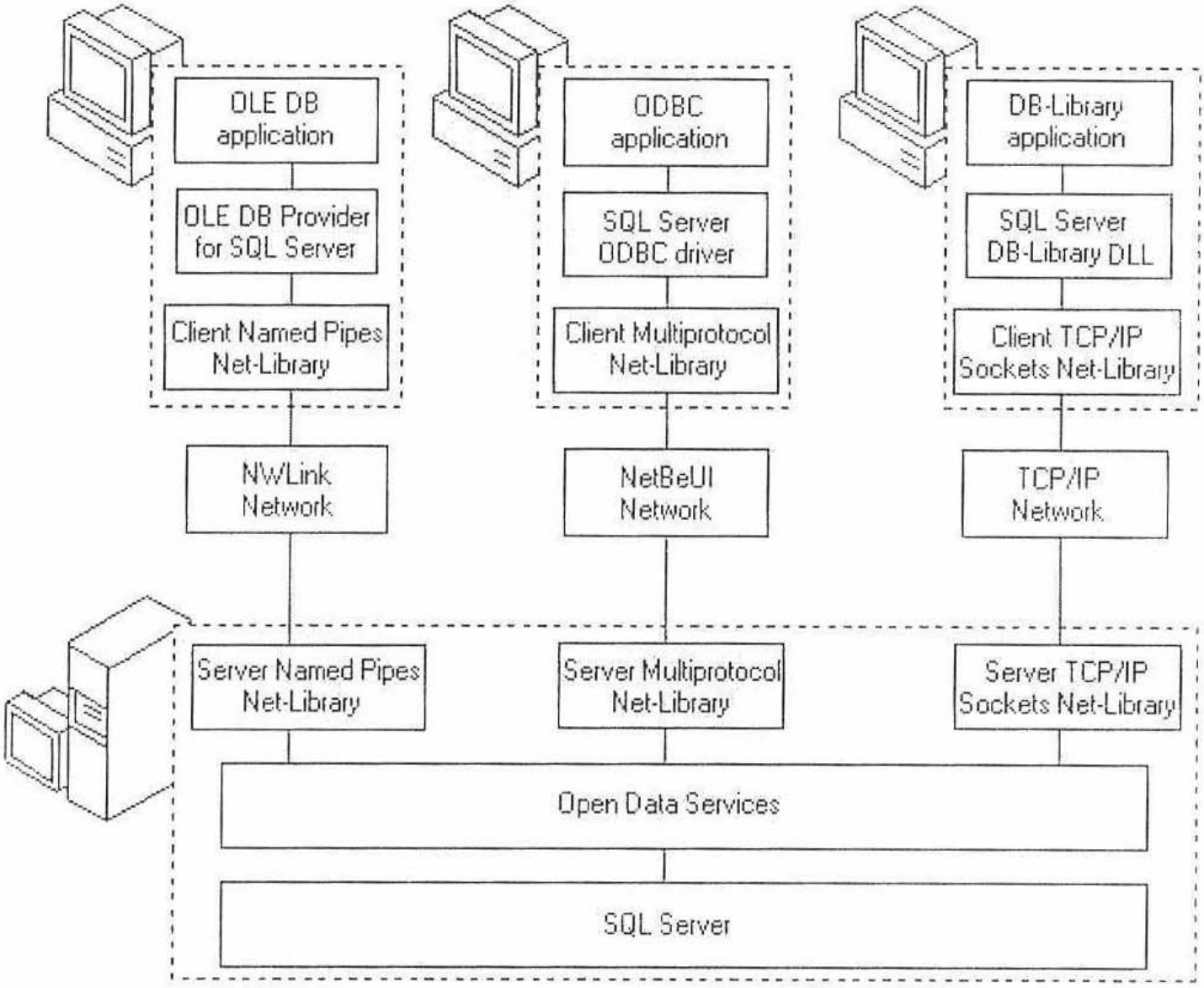
اجزاء ارتباطی

SQL Server روش‌های متعددی برای ارتباط بین برنامه‌های کاربردی سرویس‌گیرنده و سرویس‌دهنده دارد. وقتی برنامه کاربردی و **SQL Server** بر روی یک کامپیوتر قرار دارند، اجزاء ارتباطی بین فرآیندی (IPC) ویندوز مانند حافظه مشترک، مورد استفاده قرار می‌گیرند. وقتی برنامهٔ کاربردی بر روی یک سرویس‌گیرنده مجزا قرار دارد، یک IPC شبکه برای ارتباط با **SQL Server** مورد استفاده قرار می‌گیرد. یک IPC دارای دو جزء است:

- رابط برنامه‌نویسی کاربردی (API) : API مجموعه‌ای از توابع است که برای استفاده از IPC فراخوانی می‌شوند.
- پروتکل: پروتکل، قالب اطلاعات ارسالی بین دو جزء ارتباطی از طریق IPC

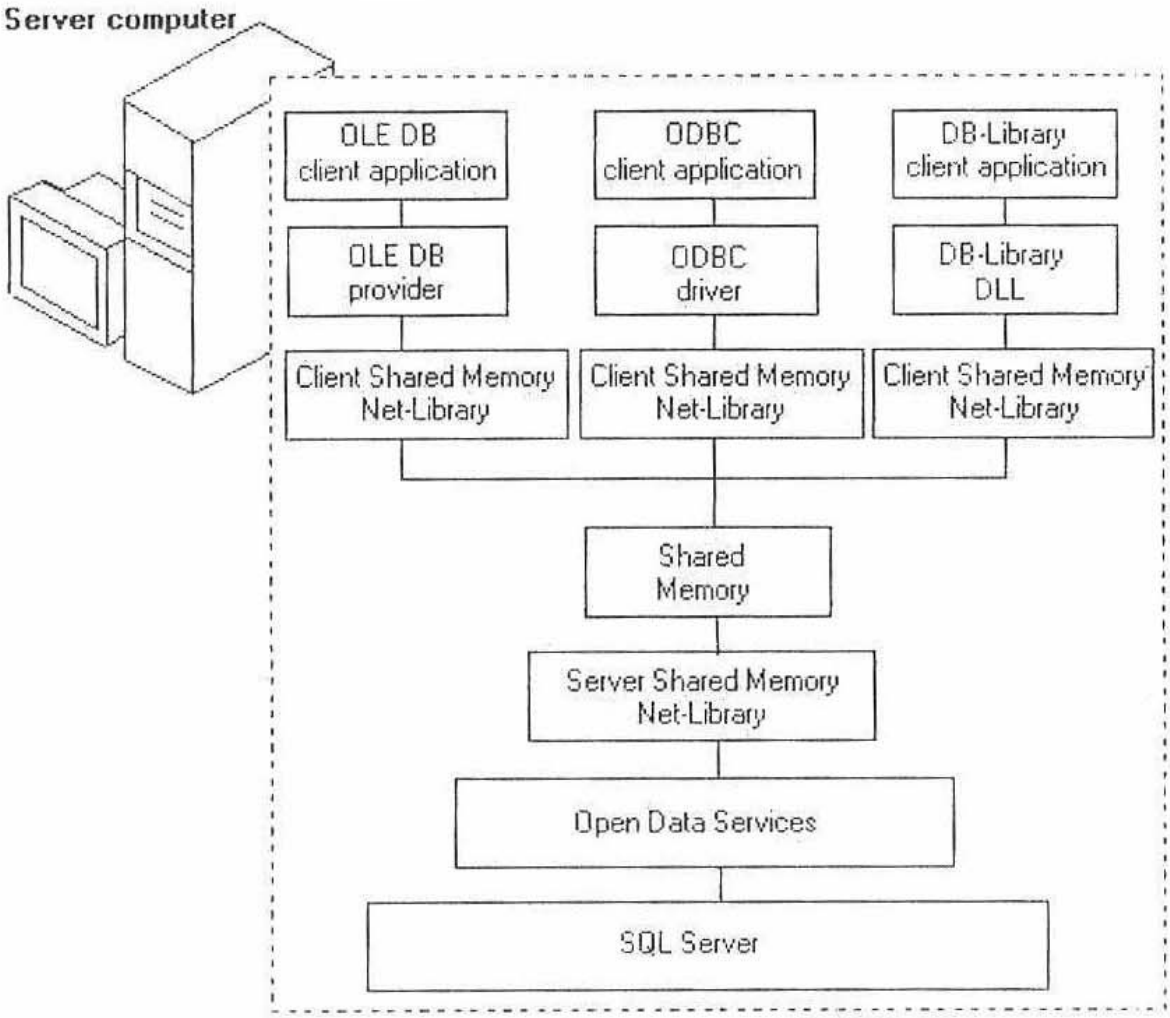
را تعریف می‌کند. در حالتی که از **IPC** شبکه استفاده می‌شود، پروتکل قالب **packet**های ارسالی بین دو کامپیوتر با استفاده از **IPC** را تعریف می‌کند.

اگر برنامه کاربردی بر روی کامپیوتر دیگری جدا از **SQL Server** اجـرا شـود، مسیر ارتباطی به شکل زیر خواهد بود:



«شکل ۴-۳: اجزاء ارتباطی هنگامی که برنامه کاربردی و SQL Server بر روی دو کامپیوتر مجزا قرار دارند»

اگر برنامه کاربردی بر روی همان کامپیوتری اجرا شود که ویندوز ۹۵/۹۸ را اجرا می‌کند و **SQL Server** نیز در آن قرار دارد، مسیر ارتباطی به صورت زیر خواهد بود:



«شکل ۵-۳ : اجزاء ارتباطی هنگامی که برنامه کاربردی و SQL Server بر روی کامپیوتر قرار دارند»

اگر SQL Server بـــر روی ویـنـدوز NT اجـرا می‌شـود، NET-Library بـرای ارتباط‌های محلی مورد استفاده قرار می‌گیرد. برای اتصال‌های محلی بدون کـــارت شـبکه، ویندوز NT از زیر سیستم فایل برای پیاده سازی اتصال استفاده می‌کند.

اجزاء سرویس‌دهنده

اجزای (سرویس‌های) اساسی سرویس‌دهنده SQL Server عبارتند از:

- **سرویس‌های داده باز (Open Data Services).**
- **QL Server (سرویس MSSQLServer).**
- **کارگزار SQL Server (سرویس SQLServerAgent).**
- **سرویس جستجوی مایکروسافت.**
- **هماهنگ کننده تراکنش توزیع شده (سرویس MS DTC).**

بر روی ویندوز ۹۵/۹۸، اجزاء سرویس‌دهنده بـه صـورت سـرویس پیاده‌سـازی نمی‌شوند، زیرا مفهوم سرویس در ویندوز ۹۵/۹۸ پشتیبانی نشده است. سرویس جستجوی مایکروسافت فقط بر روی ویندوز NT سرویس‌دهنده موجود است و بر روی ویـنـدوز NT ایستگاه یا ویندوز ۹۵/۹۸ قابل استفاده نمی‌باشد.

اجزای سرویس‌دهنده از چند طریق می‌توانند متوقف شده یا شروع به کار نمایند:

- ویندوز NT می‌تواند هر یک از سرویس‌ها را هنگام شـروع بـه کـار سیسـتم عامل به طور خودکار راه‌اندازی نماید.
- با استفاده از Service Control Manager در SQL Server می‌تـوان یـک سرویس را شروع و یا متوقف کرد.
- با استفاده از Enterprise Manager در SQL Server می‌توان یک سـرویس را شروع و یا متوقف کرد.
- در ویندوز NT می‌توان با استفاده از دستورات net start و net stop در خط فرمان، هر یک از سرویس‌ها را شروع و یا متوقف کرد.

سرویس‌های داده باز به خودی خود شروع و یا متوقف نمی‌شوند. این سرویس‌ها هنگام شروع و یا توقف SQL Server شروع شده و یا متوقف می‌شوند.

سرویس داده باز

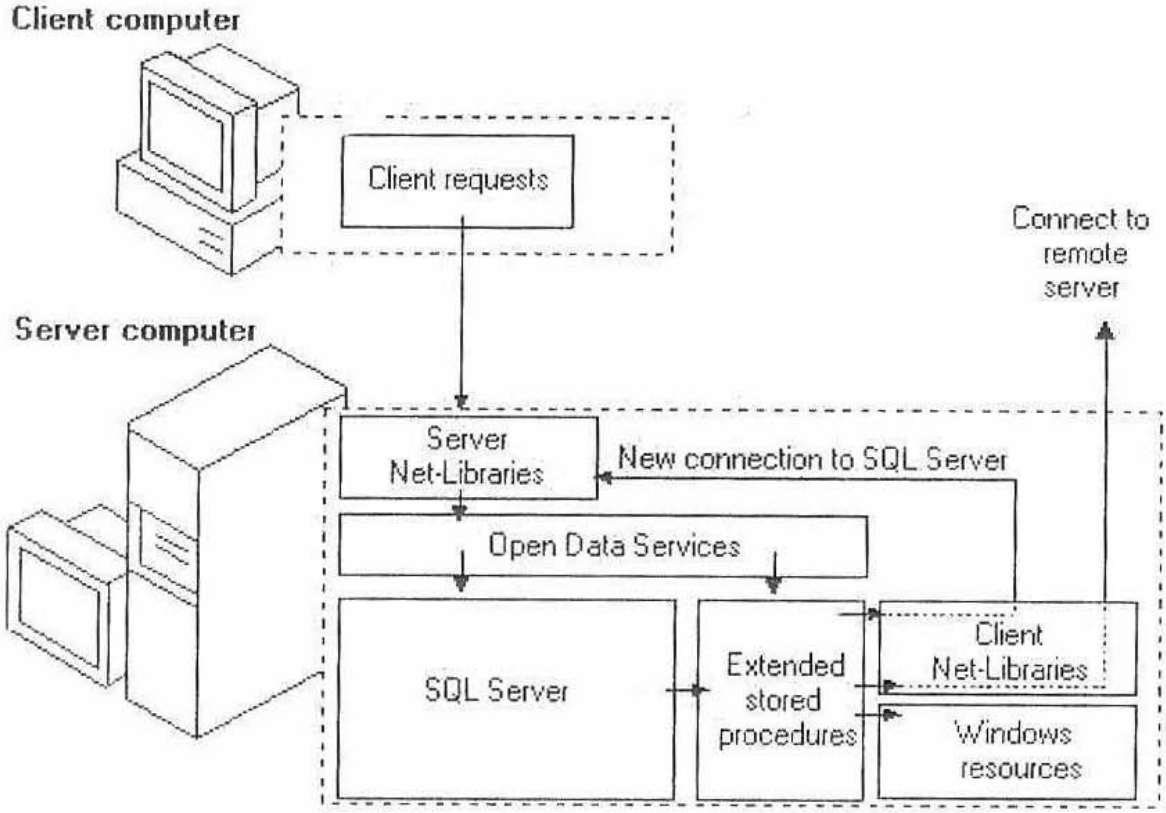
سرویس دادهٔ باز رابطی بین کتابخانه‌های شـبکه‌ای سـرویس‌دهنده و برنامـه‌های کاربردی سرویس‌دهنده است. این سرویس‌ها امکان می‌دهند که :

- رویه‌های ذخیره شده‌ای تولید کنیـد تـا قـدرت Transact-SQL و SQL Server بسط داده شود.
- یک برنامه کاربردی سرویس دهنده تولید کنید تا دستورات Transact-SQL را کـه برنامه‌های کاربردی سرویس‌گیرنده به آن ارسال می‌کنند، دریافت کرده و پردازش نماید.

سرویس‌های داده باز بر روی سرویس‌دهنده اجرا می‌شوند. کتابخانه‌های شبکه‌ای سرویس‌دهنده، درخواست‌ها را از سرویس‌گیرنده دریافت و به سرویس‌های دادهٔ باز ارسال می‌کنند. در آنجا، این درخواست‌ها به رویدادهایی تبدیل می‌شوند و به برنامه‌های کاربردی

سرویس‌دهنده ارسال می‌شوند. برنامه‌های کاربردی سرویس‌دهنده مجدداً پاسخ‌ها را به سرویس‌گیرنده‌های SQL Server باز پس می‌فرستند.

نوع اساسی برنامه‌های کاربردی سرویس‌های داده باز، رویه‌های ذخیره شده نوسعه یافته هستند. این ویژگی SQL Server امکان می‌دهد توابعی به زبان C و C++ نوشته شود و مستقیماً از دستورات Transact -SQL فراخوانی شوند.



«شکل ۶-۳: ساختار سرویس‌های داده باز».

سرویس MSSQLServer

SQL Server به عنوان یک سرویس به نام MSSQLServer در ویندوز NT اجرا می‌شود. SQL Server را می‌توان به عنوان یک فایل اجرایی نیز در ویندوز NT اجرا نمود، اما معمولاً به صورت یک سرویس اجرا می‌شود.

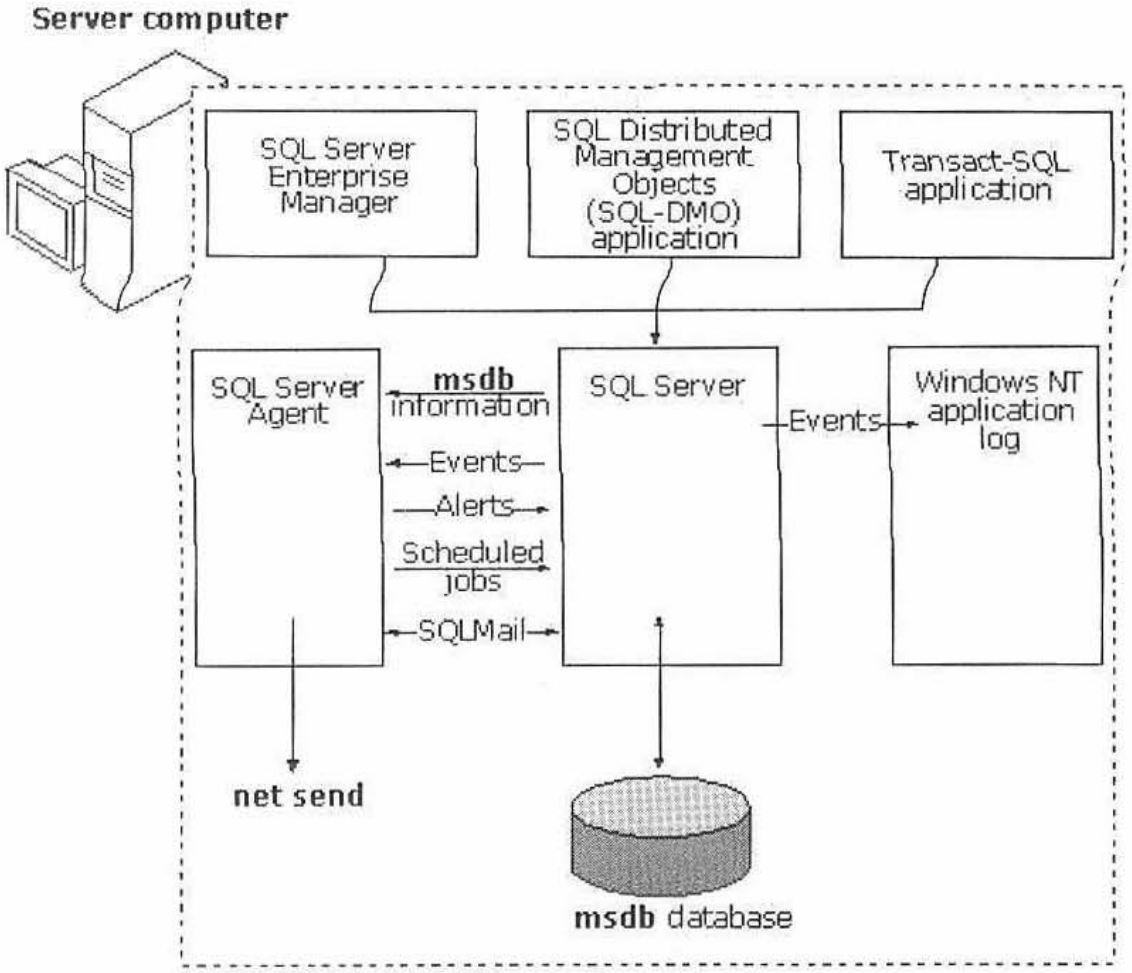
این سرویس تمام فایل‌های تشکیل دهندهٔ پایگاه‌های داده بر روی سرویس‌دهنده را مدیریت می‌نماید. همچنین تمام دستورات Transact-SQL را که از برنامه‌های کاربردی سرویس‌گیرنده ارسال می‌شود، پردازش می‌کند. این سرویس

رویه‌های ذخیره شده موجود در سرویس‌دهنده‌های دور را نیز اجرا می‌نماید. این سرویس، منابع کامپیوتر را به طور مؤثر به کاربران مختلف تخصیص می‌دهد. علاوه بر این، قواعد و محدودیت‌هایی را که در رویه‌های ذخیره شده و Triggerها تعریف شده‌اند اعمال می‌کند، سازگاری داده‌ها را تضمین می‌کند و از بروز خطاهای منطقی مانند دستیابی همزمان کاربران به یک قلم داده جلوگیری می‌کند.

سرویس SQLServerAgent

این سرویس دارای ویژگی‌هایی است که زمان‌بندی نوبه‌ای فعالیت‌ها در SQL Server را امکان پذیر می‌کند و یا در مواقع بروز اشکال در سرویس‌دهنده، به مدیران سیستم اخطار می‌دهد. اجزاء کارگزار SQL Server که این قابلیت‌ها را پیاده‌سازی می‌کنند عبارتند از:

- Jobها: شئ‌های تعریف شده‌ای هستند که شامل یک یا چند گام برای اجرا می‌باشند. گام‌ها دستورات Transact - SQL هستند که می‌توانند اجرا شوند. Jobها می‌توانند زمان‌بندی شوند.
- Alertها: عملیاتی هستند که هنگام وقوع رویدادهایی خاص (مثل خطاها) باید اجرا شوند.
- اپراتورها: اشخاصی هستند که از طریق حساب خود در شبکه شناسایی می‌شوند.



«شکل ۷-۳ : اجزاء اصلی مورد استفاده برای تعریف Jobها، Alertها و اپراتورها»

تعریف این اجزاء توسط SQL Server در پایگاه داده سیستمی msdb ذخیره می‌شود. وقتی سرویس SQLServerAgent شروع به کار می‌کند، پرس و جویی بر روی جدول‌های سیستمی موجود در پایگاه داده msdb اجرا کرده تا تعیین نماید چه Jobها و Alertهایی باید فعال شوند.

سرویس جستجوی مایکروسافت

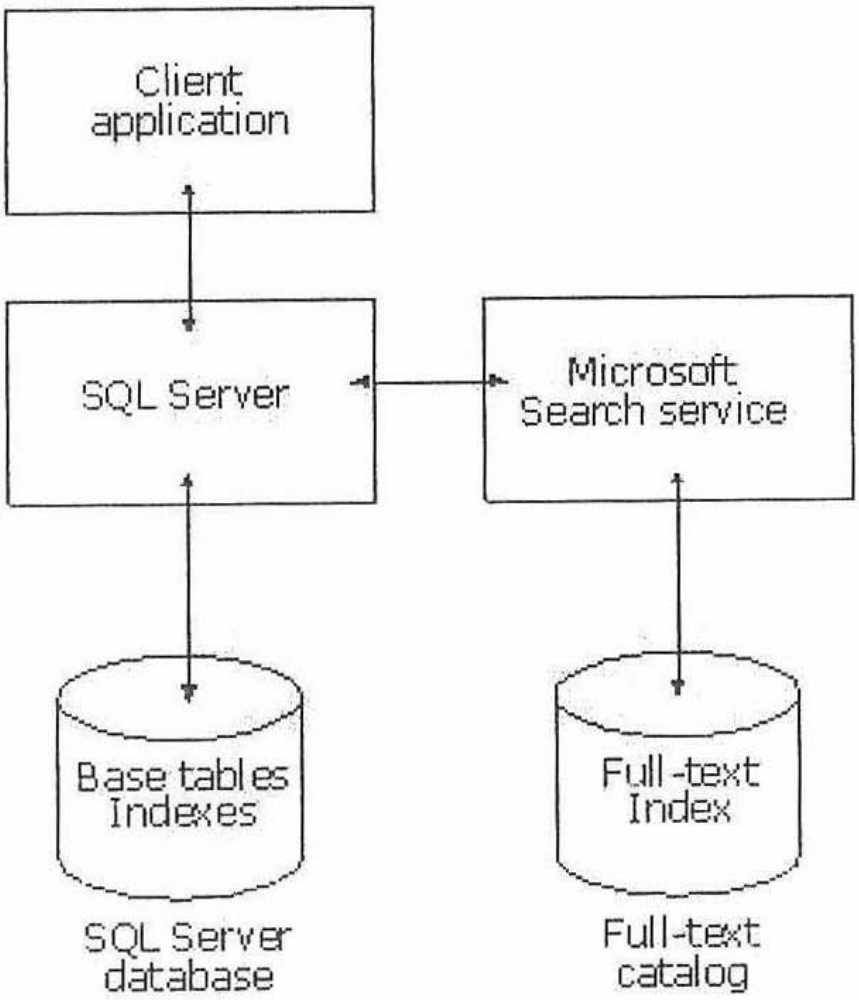
این سرویس، یک موتور شاخص گذار و جستجوی تمام متن می‌باشد.

نسخه‌های اولیه SQL Server فقط قابلیت‌های جستجوی ابتدایی زیر را پشتیبانی می‌کردند:

- **جستجو برای یک مقدار کاراکتری که مساوی با، کوچکتر از یا بزرگتر از یک ثابت کاراکتری باشد.**
- **جستجو برای یک مقدار کاراکتری که شامل یک الگوی رشته‌ای است.**

این جستجوها فقط بر روی ستون‌های Char و Varchar در یک پایگاه داده می‌تواند انجام شود.

استفاده از سرویس جستجوی مایکروسافت به SQL Server 7.0 اجازه می‌دهد که جستجوهای پیچیده‌تری را بر روی ستون‌های رشته‌ای انجام دهد.



«شکل ۸-۳: سرویس جستجوی مایکروسافت»

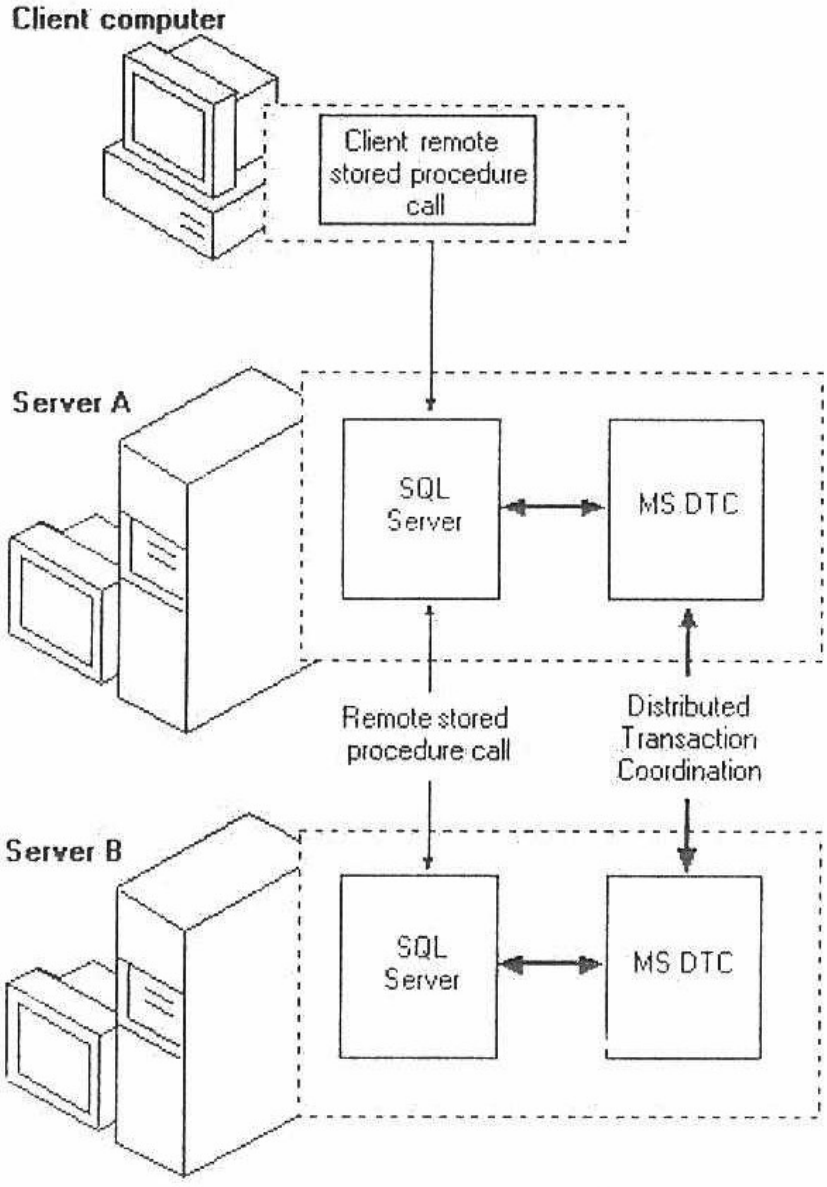
سرویس MS DTC

این سرویس، یک مدیر تراکنش است که به برنامه‌های کاربردی سرویس‌گیرنده اجازه می‌دهد چند منبع مختلف داده‌ای را در یک تراکنش دخالت دهند. سرویس MS DTC تأیید تراکنش توزیع شده را در بین تمام سرویس‌دهنده‌ها هماهنگ می‌کند.

SQL Server به روش‌های زیر می‌تواند در یک تراکنش توزیع شده شرکت کند:

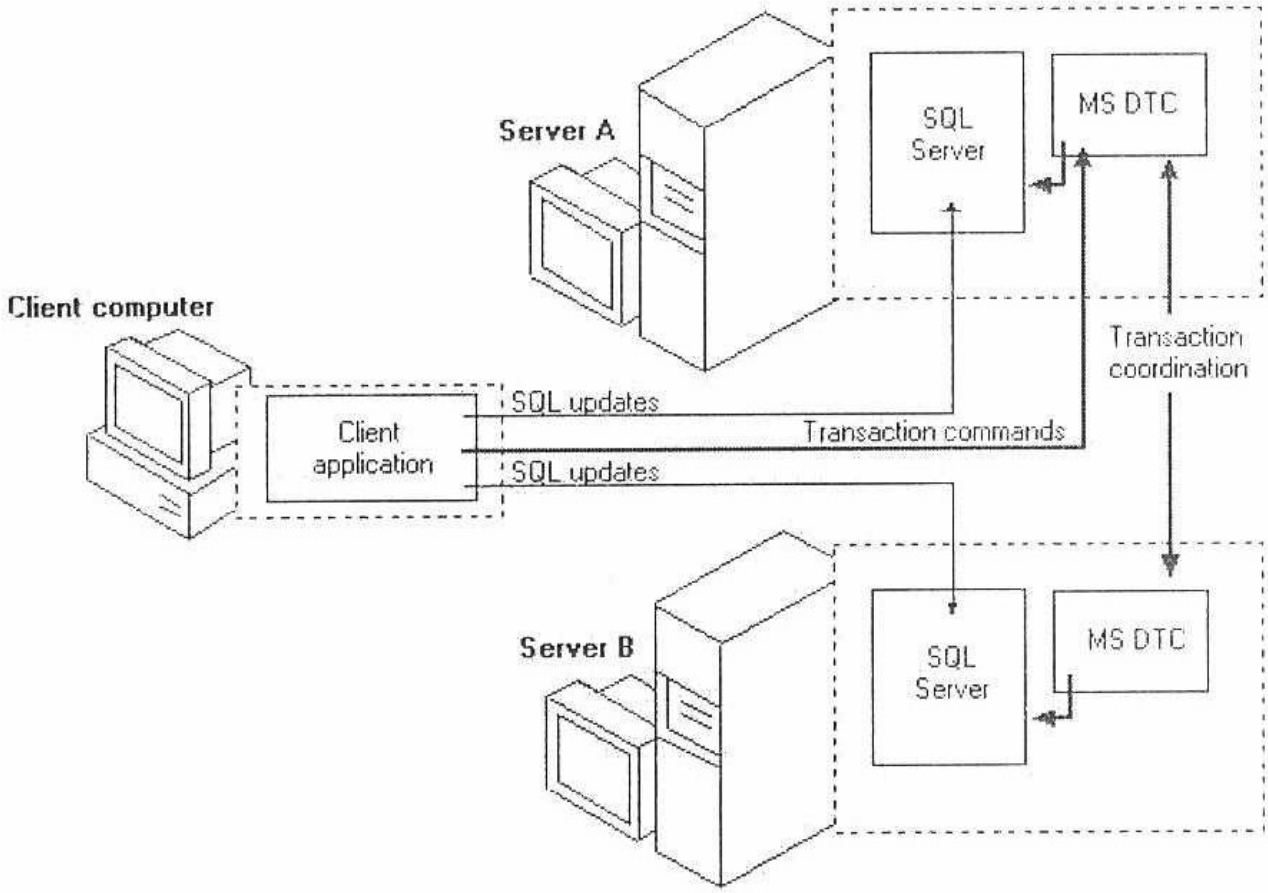
- فراخوانی رویه‌های ذخیره شده روی سرویس‌دهنده‌های دوری که SQL Server را اجرا می‌کنند.
- ارتقاء تراکنش محلی به یک تراکنش توزیع شده به طور خودکار یا صریح و ارجاع به سرویس‌دهنده‌های دور در تراکنش.
- اجرای بهنگام سازی‌های توزیع شده که داده‌های چندین منبع داده‌ای OLE DB را بهنگام‌سازی می‌کند.

سرویس MS DTC تکمیل شدن مناسب تراکنش توزیع شده را هماهنگ می‌کند تا تضمین نماید که تمام بهنگام‌سازی‌ها روی تمام سرویس‌دهنده‌ها یا دائمی شوند و یا در موقع بروز خطا، تماماً حذف شوند.



«شکل ۹-۳: سرویس MS DTC»

برنامه‌های کاربردی SQL Server می‌توانند برای شروع یک تراکنش توزیع شده به طور سریع سرویس MS DTC را فراخوانی کنند. سپس یک یا چند سرویس دهنده‌ای که SQL Server را اجرا می‌کنند، می‌توانند در تراکنش توزیع شده شرکت کرده و تکمیل تراکنش را با استفاده از سرویس MS DTC به طور مناسب هماهنگ نمایند.

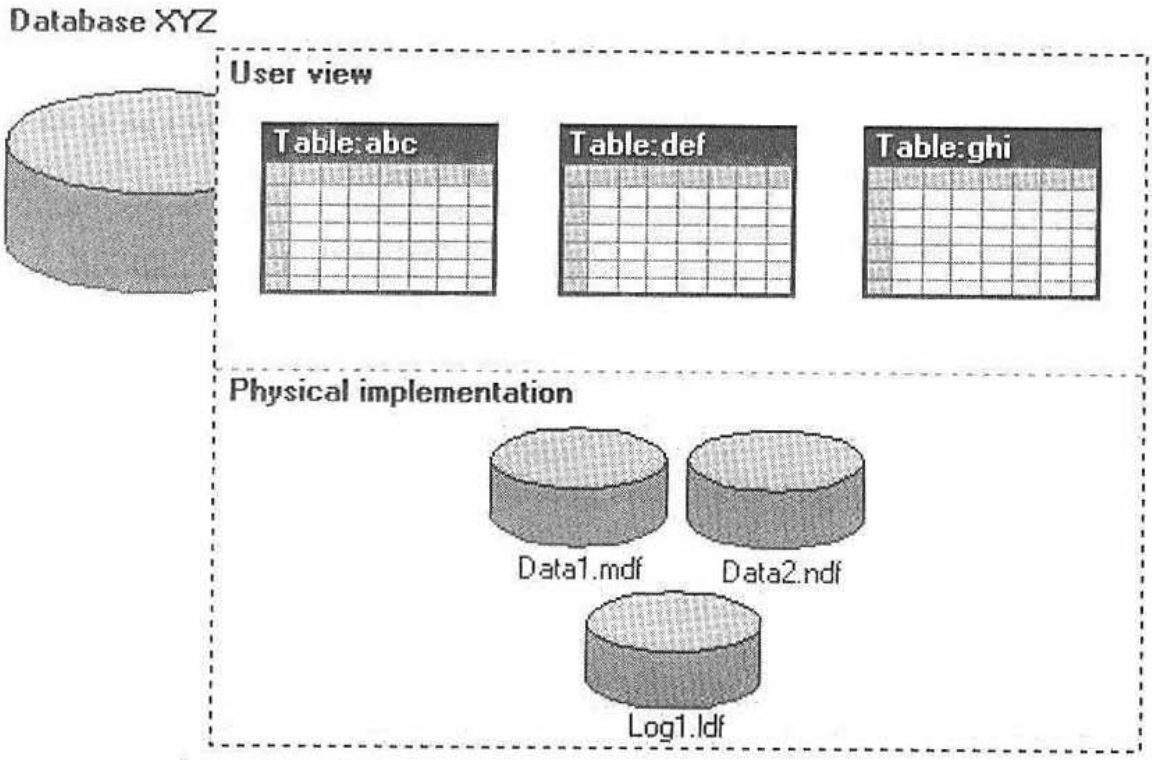


«شکل ۱۰-۳: فراخوانی صریح سرویس MS DTC»

۵-۳- معماری پایگاه داده

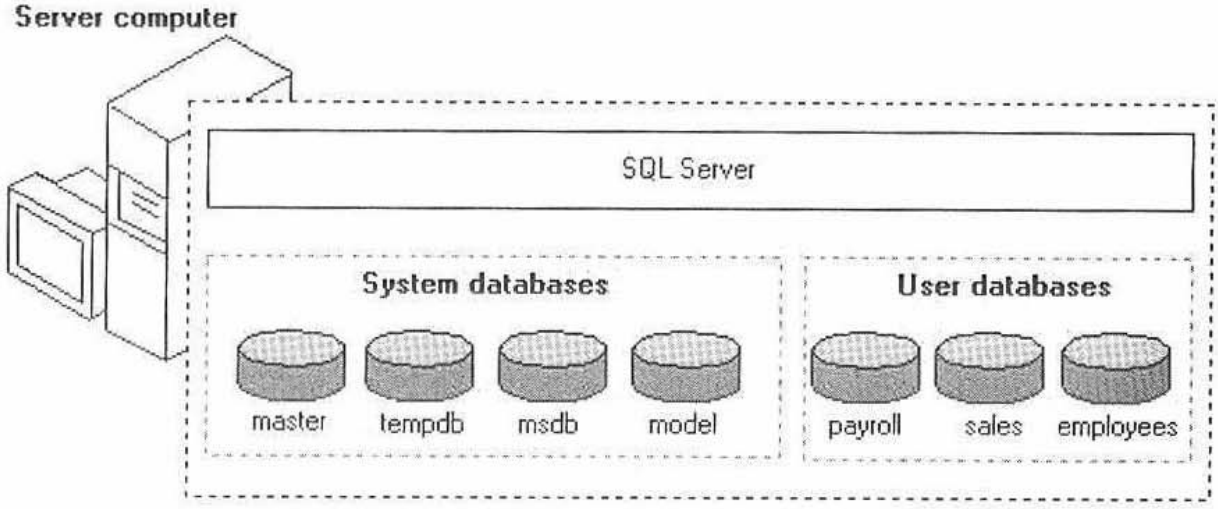
داده‌های SQL Server در پایگاه‌های داده ذخیره می‌شود. داده‌های درون یک پایگاه داده در واحدهایی منطقی که برای کاربر قابل رؤیت هستند سازماندهی می‌شوند. پایگاه داده از نظر فیزیکی به صورت دو یا چند فایل بر روی دیسک پیاده‌سازی می‌شود.

وقتی از پایگاه داده استفاده می‌کنید، در واقع شما با واحدهای منطقی مانند جدول‌ها، دیدها، رویه‌ها و کاربران کار می‌کنید. پیاده سازی فیزیکی فایل‌ها کاملاً شفاف است. نوعاً فقط مدیر پایگاه داده نیاز به کار با پیاده‌سازی فیزیکی دارد.



«شکل ۱۱-۳: اجزاء فیزیکی و منطقی پایگاه داده»

SQL Server دارای چهار پایگاه داده سیستمی (master، model، tempdb و msdb) و یک یا چند پایگاه داده مربوط به کاربران است. برخی از سازمان‌ها فقط دارای یک پایگاه داده کاربر هستند که شامل تمام داده‌های آنها است. برخی از سازمان‌ها نیز دارای پایگاه‌های داده مختلفی برای هر گروه در سازمان هستند. به عنوان مثال، یک سازمان می‌تواند یک پایگاه داده برای فروش، یک پایگاه داده برای حقوق و یک پایگاه داده برای مدیریت اسناد داشته باشد.



«شکل ۱۲-۳: پایگاه‌های داده موجود در SQL Server»

لازم نیست که چند نسخه از SQL Server را اجرا کنید تا کاربران بتوانند به پایگاه‌های داده یک سرویس‌دهنده دسترسی یابند. SQL Server قادر به جوابگویی به هزاران کاربر در پایگاه‌های داده مختلف به طور همزمان بر روی یک سرویس‌دهنده

می‌باشد. SQL Server تمام پایگاه‌های داده موجود در سرویس‌دهنده را برای تمام کاربرانی که به سرویس‌دهنده متصل هستند، با توجه به جوازهایشان قابل دسترسی می‌سازد.

هنگام اتصال به SQL Server، اتصال شما با یک پایگاه داده مشخص روی سرویس‌دهنده در ارتباط می‌باشد. این پایگاه داده، پایگاه دادهٔ جاری نامیده می‌شود. شما معمولاً به پایگاه داده‌ای که به عنوان پیش فرض توسط مدیر سیستم تعریف شده است متصل هستید مگر این که با استفاده از گزینه‌های اتصال APIهای پایگاه داده، پایگاه داده دیگری را مشخص کنید.

اجزاء منطقی پایگاه داده

داده‌های درون یک پایگاه داده SQL Server در چند شئ مختلف ذخیره شده‌اند. این شئ‌ها پس از اتصال به پایگاه داده، برای کاربر قابل رؤیت هستند.

در SOL Server اجزاء زیر به صورت شئ تعریف شده‌اند:

محدویت‌ها	جدول‌ها
پیش فرض‌ها	Triggerها
شاخص‌ها	نوع‌های داده‌ای تعریف شده توسط کاربر
کلیدها	دیدها
رویه‌های ذخیره شده	

نوع‌های داده‌ای و ساختار جدول

تمام داده‌های پایگاه داده SQL Server در شئ‌هایی به نام جدول قرار دارند. هر جدول دلالت بر شئ‌ای دارد که برای کاربر با معنا می‌باشد. به عنوان مثال، در یک پایگاه داده مدرسه، جدول‌ها می‌توانند جدول کلاس، جدول معلم و جدول دانش آموز باشد.

جدول‌های SQL Server دارای دو جزء اساسی هستند:

- ستون: نشان دهنده بعضی از صفت‌های شئ‌ای است که توسط جدول مدل‌سازی شده است.
- سطر: نشان دهنده نمونه‌ای از یک شئ است که توسط جدول مدل‌سازی شده است.

به عنوان مثال، جدول قطعات در زیر دارای سطرها و ستون‌هایی برای هر قطعه است:

parts table		
ID	color	weight
AB123	Blue	10.5
CD456	Red	8.0
EF789	Green	9.25
GH012	Yellow	8.0
IJ341	Blue	1.0

«شکل ۱۳-۳: جدول parts»

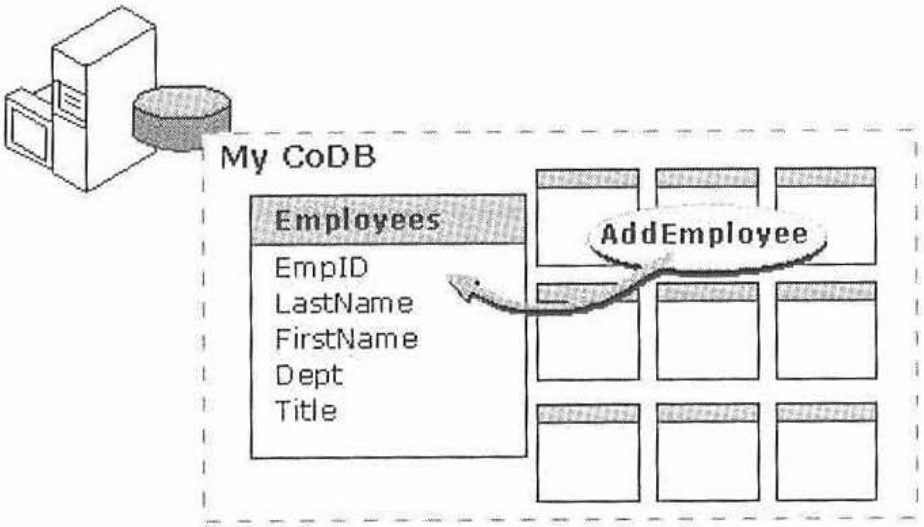
یکی از خواص ستون، نوع داده‌ای آن است که نوع مقادیری را که ستون نگهداری می‌کند تعریف می‌نماید. **SQL Server** دارای چند نوع داده مبنایی به صورت زیر است:

binary	bit	char	datetime	decimal
float	image	int	money	nchar
ntext	nvarchar	numeric	real	smalldatetime
smallint	smallmoney	sysname	text	timestamp
tinyint	varbinary	varchar	uniqueidentifier	

میدان، مجموعه‌ای از تمام مقادیر مجاز در یک ستون است. به عنوان مثال، میدان رنگ قطعه هم شامل نوع داده‌ای مانند **char(6)** و هم شامل رشته‌های کاراکتری مجاز در ستون مانند **Red**، **Blue**، **Green** و غیره است.

جدول‌ها، کنترل‌های متعددی دارند (محدودیت‌ها، قاعده‌ها، **Trigger**ها، پیش فرض‌ها) که صحت داده‌ها را تضمین می‌کنند. به عنوان مثال، می‌توانید پایگاه داده‌ای به نام **MyCoDb** برای مدیریت داده‌های شرکت خود ایجاد کنید. می‌توانید جدولی به نام **Employees** برای ذخیره اطلاعات کارمندان ایجاد کنید. جدول می‌تواند ستون‌هایی به نام **EmpId**، **LastName**، **FirstName**، **Dept** و **Title** داشته باشد. می‌توانید رویه‌ای به نام **AddEmployee** ایجاد کنید تا مقادیر داده‌ای مربوط به یک کارمند جدید را دریافت کرده

و عملیات مربوط به درج سطر در جدول **Employees** را انجام دهد.



«شکل ۱۴-۳: پایگاه داده MyCoDb»

ستون‌ها می‌توانند مقادیر **NULL** را قبول کرده و یا رد نمایند. **NULL** یک مقدار ویژه در پایگاه داده است که مفهوم "مقدار ناشناخته" را در بر دارد. **NULL** با مقادیر جای خالی یا صفر تفاوت دارد. جای خالی در واقع یک کاراکتر مجاز و صفر یک عدد مجاز است در حالی که **NULL** صرفاً نشان دهنده این باور است که ما نمی‌دانیم این مقدار چیست. **NULL** با رشته به طول صفر نیز تفاوت دارد.

SQL Server داده‌های مربوط به تعریف مشخصات سرویس‌دهنده و تمام جدول‌های خود را در مجموعه ویژه‌ای از جدول‌ها به نام جدول‌های سیستمی ذخیره می‌کند. کاربران نباید مستقیماً جدول‌های سیستمی را بهنگام‌سازی کرده و یا روی آنها پرس و جوهایی اجرا کنند. فقط **SQL Server** باید در پاسخ به دستورات مدیریتی که توسط کاربران صادر می‌شود به جدول‌های سیستمی ارجاع کند.

SQL Server دارای جدول موقت می‌باشد. نام این گونه جدول‌ها با علامت # شروع می‌شود. اگر یک جدول موقت هنگام قطع اتصال کاربر حذف نشود، به طور خودکار توسط **SQL Server** حذف می‌شود. جدول‌های موقت در پایگاه داده جاری ذخیره نمی‌شوند بلکه در جدول سیستمی **tempdb** ذخیره می‌شوند. دو نوع جدول موقت وجود دارد:

- جدول‌های موقت محلی که فقط دارای یک علامت # در ابتدای نامشان هستند. این جدول‌ها فقط برای اتصالی که آنها را ایجاد کرده است قابل رؤیت هستند.
- جدول‌های موقت عمومی که دارای دو علامت # در ابتدای نامشان هستند.

این جدول‌ها برای تمام اتصال‌ها قابل رؤیت هستند. اگر این جدول‌ها توسط اتصالی که آنها را ایجاد کرده است قبل از قطع اتصال صریحاً حذف نشوند، به طور خودکار پس از این که تمام اتصال‌های دیگر، ارجاع خود به این جدول‌ها را پایان دادند، بلافاصله حذف می‌شوند. اگر اتصالی که یک جدول موقت را ایجاد کرده است قطع گردد، از آن پس دیگر هیچ اتصال جدیدی نمی‌تواند به جدول موقت ارجاع کند.

دیدهای SQL

دید را می‌توان به عنوان یک جدول مجازی یا پرس و جوی ذخیره شده در نظر گرفت. داده‌هایی که از طریق یک دید قابل دستیابی هستند به عنوان یک شئ مجزا در پایگاه داده ذخیره نمی‌شوند. چیزی که در پایگاه داده ذخیره می‌شود یک دستور SELECT است. مجموعه نتیجه دستور SELECT، جدول مجازی حاصل از دید را تشکیل می‌دهد. از دید برای انجام کارهای زیر استفاده می‌شود:

- محدود کردن کاربر به سطرهای ویژه‌ای از یک جدول.
- محدود کردن کاربر به ستون‌های ویژه‌ای از یک جدول.
- الحاق ستون‌هایی از جدول‌های مختلف به طوری که همانند یک جدول باشند.
- بدست آوردن اطلاعات جمعی به جای پرداختن به جزئیات.

جدول‌هایی که دستور SELECT موجود در تعریف دید به آنها ارجاع می‌کند، جدول مبنا نامیده می‌شوند.

پس از تعریف دید، می‌توانید همانند یک جدول به آن ارجاع کنید. در تعریف دید می‌توان به یک دید دیگر ارجاع کرد.

دیدهای SQL Server قابل بهنگام سازی هستند در صورتی که عمل بهنگام سازی فقط بر یکی از جدول‌های مبنایی که توسط دید ارجاع می‌شود، تأثیر داشته باشد.

رویه‌های ذخیره شده

رویه ذخیره شده، مجموعه‌ای از دستورات Transact-SQL است که در یک طرح اجرایی کامپایل شده‌اند.

رویه‌های ذخیره شده به چهار طریق داده‌ها را باز می‌گردانند:

۱- پارامترهای خروجی که می‌توانند داده یا یک متغیر کرزر را باز گردانند.

۲- کدهای بازگشتی که همواره مقادیر صحیح هستند.

۳- برای هر دستور **SELECT** درون رویه ذخیره شده یا رویه‌های ذخیره شده دیگری که در این رویه قرار دارند، یک مجموعه نتیجه بازی گرداند.

۴- کرزر عمومی که نمی‌تواند خارج از رویه ذخیره شده ارجاع شود.

نمونه‌ای از یک رویه ذخیره شده را در زیر مشاهده می‌کنید:

```
IF (@QuantityOrdered < (SELECT QuantityOnHand
FROM Inventory
WHERE PartID = @PartOrdered) )
BEGIN
-- SQL statements to update tables and process order.
END
ELSE
BEGIN
-- SELECT statement to retrieve the IDs of alternate items
-- to suggest as replacements to the customer.
END
```

برنامه‌های کاربردی لازم نیست که تمام دستورات **SQL** درون رویه را ارسال کنند. در عوض فقط یک دستور **EXECUTE** یا **CALL** که شامل نام رویه و مقادیر پارامترها است ارسال می‌شود.

رویه‌های ذخیره شده کاربران را از جزئیات جدول‌های درون پایگاه داده بی‌نیاز می‌کنند. اگر مجموعه‌ای از رویه‌های ذخیره شده تمام کارهای مورد نیاز کاربران را پشتیبانی کند کاربران هرگز مجبور نخواهند بود که مستقیماً به جدول‌ها دسترسی یابند. یک نمونه از این نحوه استفاده از رویه‌های ذخیره شده، رویه‌های ذخیره شده سیستمی **SQL Server** است که کاربران را از ارتباط مستقیم با جدول‌های سیستمی بی‌نیاز می‌کند. نام این رویه‌های ذخیره شده با _sp شروع می‌شود. این رویه‌های ذخیره شده تمام وظایف مدیریتی لازم برای اجرای **SQL Server** را پشتیبانی می‌کند. در واقع، شما می‌توانید با استفاده از

دستورات مدیریتی Transact - SQL (مثل CREAT TABLE) یا رویه‌های ذخیره شده سیستمی، SQL Server را مدیریت نمایید و هرگز نیاز نخواهید داشت که مستقیماً جدول‌های سیستمی را بهنگام سازی کنید.

اجرای یک رویه ذخیره شده بهتر از اجرای یک دستور SQL می‌باشد زیرا لازم نیست SQL Server به طور کامل یک طرح اجرا را کامپایل نماید و فقط باید بهینه سازی طرح ذخیره شده رویه را تکمیل نماید.

SQL Server دارای رویه‌های ذخیره شده موقت نیز می‌باشد که هنگام قطع اتصال حذف می‌شوند. رویه‌های ذخیره شده موقت در tempdb ذخیره می‌شوند. از این رویه‌ها بیشتر در مواقعی استفاده می‌شود که برنامه‌های کاربردی، دستورات Transact-SQL پویا تولید می‌کنند و چندین بار اجرا می‌شوند. به جای این که دستورات SQL Server چند بار کامپایل شوند، می‌توان یک رویه ذخیره شده ایجاد کرد که در اولین اجرای خود کامپایل شده و سپس طرح از پیش کامپایل شده چندین بار اجرا شود. استفاده بیش از حد از رویه‌های ذخیره شده موقت، منجر به ایجاد تداخل در جدول‌های سیستمی tempdb می‌شود.

در زیر، مثال ساده‌ای بیان شده است که سه روش بازگرداندن داده‌ها توسط رویه‌های ذخیره شده را تشریح می‌کند:

۱- ابتدا یک دستور SELECT اجرا می‌شود که مجموعه نتیجه‌ای را از فروشگاه‌های جدول Sales باز می‌گرداند.

۲- سپس یک دستور SELECT اجرا می‌شود که یک پارامتر خروجی را مقداردهی می‌کند.

۳- در انتها، یک دستور RETURN به همراه یک دستور SELECT وجود دارد که یک عدد صحیح را باز می‌گرداند:

```
USE Northwind
GO
DROP PROCEDURE OrderSummary
GO
```

```
CREATE PROCEDURE OrderSummary @MaxQuantity INT OUTPUT AS
-- SELECT to return a result set summarizing
-- employee sales.
SELECT Ord.EmployeeID, SummSales = SUM(OrDet.UnitPrice *
OrDet.Quantity)
FROM Orders AS Ord
   JOIN [Order Details] AS OrDet ON (Ord.OrderID = OrDet.OrderID)
   GROUP BY Ord.EmployeeID
   ORDER BY Ord.EmployeeID
-- SELECT to fill the output parameter with the
-- maximum quantity from Order Details.
SELECT @MaxQuantity = MAX(Quantity) FROM [Order Details]
-- Return the number of all items ordered.
RETURN (SELECT SUM(Quantity) FROM [Order Details])
GO
-- Test the stored procedure.
-- DECLARE variables to hold the return code
code
-- and output parameter.
DECLARE @OrderSum INT
DECLARE @LargestOrder INT
-- Execute the procedure, which returns
-- the result set from the first SELECT.
EXEC @OrderSum = OrderSummary @MaxQuantity = @LargestOrder
OUTPUT
-- Use the return code and output parameter.
PRINT 'The size of the largest single order was: ' +
CONVERT(CHAR(6), @LargestOrder)
PRINT 'The sum of the quantities ordered was: ' +
CONVERT(CHAR(6), @OrderSum)
```

```
GO

EmployeeID SummSales
----------- -------------------------
1 202,143.71
2 177,749.26
3 213,051.30
4 250,187.45
5 75,567.75
6 78,198.10
7 141,295.99
8 133,301.03
9 82,964.00
The size of the largest single order was: 130
The sum of the quantities ordered was: 51317
```

محدودیت‌ها

محدودیت‌ها روشی را ارائه می‌دهند که **SQL Server** جامعیت یک پایگاه داده را اعمال کند. محدودیت‌ها قواعد مربوط به مقادیر مجاز در ستون‌ها را تعریف می‌کنند و مکانیزم استانداردی برای اعمال جامعیت به شمار می‌آیند. پنج نوع محدودیت وجود دارد:

۱- **NOT NULL** مقرر می‌کند که ستون، مقادیر **NULL** را قبول نکند.

۲- محدودیت **CHECK**، جامعیت میدان را با محدود کردن مقادیری که می‌تواند در یک ستون قرار گیرد، اعمال می‌کند. این محدودیت یک عبارت منطقی را مشخص می‌کند که به تمام مقادیری که می‌خواهند در ستون وارد شوند، اعمال می‌شود. تمام مقادیری که این عبارت منطقی برای آنها **FALSE** ارزیابی شود رد می‌شوند. برای هر ستون می‌توانید چند محدودیت **CHECK** مشخص کنید. مثال زیر نشان می‌دهد که چگونه محدودیت **chk_id** تضمین می‌کند که فقط اعدادی در محدوده مشخص شده وارد شوند:

```
CREATE TABLE cust_sample
(
cust_id int PRIMARY KEY,
cust_name char(50),
cust_address char(50),
cust_credit_limit money,
CONSTRAINT chk_id CHECK (cust_id BETWEEN 0 and 10000 )
)
```

۳- محدودیت **UNIQUE** منحصربفرد بودن مقادیر در مجموعه‌ای از ستون‌ها را اعمال می‌کند. هیچ دو سطری از جدول نمی‌توانند دارای مقادیر غیر **NULL** و یکسان در ستون‌های مشخص شده در محدودیت **UNIQUE** باشند.

۴- محدودیت **PRIMARY KEY** مجموعه‌ای از ستون‌ها را مشخص می‌کند که مقادیر آنها یک سطر از جدول را به طور منحصربفرد شناسایی می‌کند. هیچ دو سطری از جدول نمی‌تواند کلید اصلی یکسان داشته باشد. ستون‌های موجود در کلید اصلی مقادیر **NULL** نمی‌پذیرند. یک جدول ممکن است دارای ترکیب‌های مختلفی از ستون‌ها باشد که یک سطر را به طور منحصربفرد شناسایی می‌کنند. هر ترکیب یک کلید کاندید به شمار می‌آید. یکی از این کلیدهای کاندید را می‌توان به عنوان کلید اصلی انتخاب کرد.

اگر یک محدودیت **PRIMARY KEY** بر روی بیش از یک ستون تعریف شده باشد، مقادیر موجود در یک ستون ممکن است تکراری باشند، اما هر ترکیب از مقادیر تمام ستون‌های **PRIMARY KEY** باید منحصربفرد باشند.

Primary Key

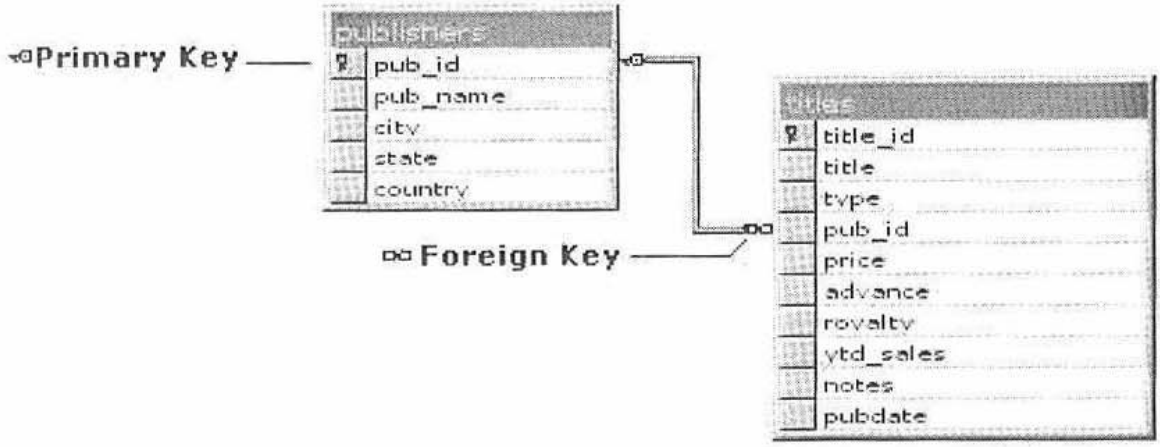
au_id	title_id	au_ord	royaltyper
172-32-1176	PS3333	1	100
213-46-8915	BU1032	2	40
213-46-8915	BU2075	1	100
238-95-7766	PC1035	1	100
267-41-2394	BU1111	2	40

titleauthor table

« شکل ۱۵-۳ : کلید اصلی ترکیبی»

۵- محدودیت **FOREIGN KEY** ارتباط بین جدول‌ها را ایجاد می‌کند. کلید خارجی در یک جدول به یک کلید کاندید در جدول دیگر اشاره می‌کند. نمی‌توان سطری را در جدول درج کرد که مقدار کلید خارجی آن دارای مقدار متناظر در کلید کاندید نباشد. همچنین نمی‌توان سطری از یک جدول را حذف کرد اگر مقادیری از

کلید خارجی به کلید کاندید آن ارجاع می‌کنند. در مثال زیر، ستون **part_nmbr** یک کلید خارجی است.



« شکل ۱۶-۳ : کلید خارجی»

محدودیت‌ها می‌توانند محدودیت ستونی و یا محدودیت جدولی باشند. محدودیت ستونی به عنوان قسمتی از تعریف ستون مشخص می‌شود و فقط بر آن ستون اعمال می‌شود. محدودیت جدولی مستقل از تعریف ستون مشخص می‌شود و می‌تواند بر بیشتر از یک ستون از جدول اعمال شود. محدودیت‌های جدولی در مواقعی به کار می‌روند که بیشتر از یک ستون باید در محدودیت شامل شود. به عنوان مثال، اگر یک جدول دارای دو یا چند ستون در کلید اصلی باشد، برای شامل کردن دو ستون در کلید اصلی باید از محدودیت جدولی استفاده کرد. به عنوان مثال، جدولی را در نظر بگیرید که رویدادهای واقع شده در یک ماشین در یک کارخانه را ثبت می‌کند. فرض کنید رویدادهایی از انواع مختلف می‌توانند در یک زمان ایجاد شوند اما دو رویدادی که در یک زمان واقع می‌شوند نمی‌توانند از یک نوع باشند.

این امر می‌تواند در جدول با شامل کردن ستون‌های **type** و **time** در یک کلید اصلی دو ستونی اعمال شود:

```
CREATE TABLE factory_process
(event_type int,
event_time datetime,
event_site char(50),
event_desc char(1024),
CONSTRAINT event_key PRIMARY KEY (event_type, event_time) )
```

قاعده‌ها

قاعده‌ها بیشتر برای سازگاری با نسخه‌های قبلی در نظر گرفته شده است و بعضی از همان کارهای محدودیت **CHECK** را انجام می‌دهند. فقط یک قاعده می‌توان برای یک ستون مشخص کرد. قاعده‌ها خارج از تعریف ستون و به صورت شیءهایی مجزا ایجاد می‌شوند و سپس به ستون مرتبط می‌شوند. در مثال زیر، یک قاعده تعریف شده است که همان کار محدودیت **CHECK** را انجام می‌دهد:

```
CREATE RULE id_chk AS @id BETWEEN 0 and 10000
GO
CREATE TABLE cust_sample
(
cust_id int PRIMARY KEY,
cust_name char(50),
cust_address char(50),
cust_credit_limit money,
)
GO
sp_bindrule id_chk, 'cust_sample.cust_id'
GO
```

پیش فرض‌ها

پیش فرض مشخص می‌کند که اگر هنگام درج یک سطر مقداری برای ستون تعیین نشود، چه مقداری برای آن مورد استفاده قرار گیرد. برای اعمال کردن پیش فرض دو روش وجود دارد:

۱- با استفاده از کلمه کلیدی **DEFAULT** در دستور **CREATE TABLE** یک عبارت ثابت را به عنوان مقدار پیش فرض برای ستون مشخص می‌کنیم.

۲- با استفاده از دستور **CREATE DEFAULT** یک شئ پیش فرض ایجاد می‌کنیم و سپس آن را با استفاده از رویه ذخیره شده سیستمی **sp_binddefault** به

ستون(های) مورد نظر مرتبط می‌کنیم. این روش بیشتر برای سازگاری با نسخه‌های قبلی در نظر گرفته شده است.

در مثال زیر، یک جدول با استفاده از هر دو روش ایجاد شده است:

```
USE pubs
GO
CREATE TABLE test_defaults
(keycol smallint,
process_id smallint DEFAULT @@SPID, --Preferred default definition
date_ins datetime DEFAULT getdate(), --Preferred default definition
mathcol smallint DEFAULT 10 * 2, --Preferred default definition
char1 char(3),
char2 char(3) DEFAULT 'xyz') --Preferred default definition
GO
/* Illustration only, use DEFAULT definitions instead.*/
CREATE DEFAULT abc_const AS 'abc'
GO
sp_bindefault abc_const, 'test_defaults.char1'
GO
INSERT INTO test_defaults(keycol) VALUES (1)
GO
SELECT * FROM test_defaults
GO
Default bound to column.
(1 row(s) affected)
keycol process_id date_ins mathcol char1 char2
------ ---------- -------------------------- ------- ----- -----
1 7 Oct 16 1997 8:34PM 20 abc xyz
(1 row(s) affected)
```

Triggerها کلاس ویژه‌ای از رویه‌های ذخیره شده هستند که وقتی دستورات **UPDATE**، **INSERT** یا **DELETE** روی یک جدول صادر می‌شود، به طور خودکار اجرا می‌شوند. البته در صورت امکان بهتر است به جای **Trigger** از محدودیت‌ها و مقادیر پیش فرض استفاده شود.

جدول‌ها می‌توانند چند Trigger داشته باشند. برای تعریف Trigger از دستور CREATE TRIGGER استفاده می‌شود. در این دستور می‌توان با استفاده از عبارات FOR UPDATE، FOR INSERT یا FOR DELETE، Trigger را با یکی از دستورات پردازش داده‌ها مرتبط کرد. SQL Server امکان می‌دهد که چند Trigger را برای یک عمل مشخص روی جدول تعریف نمود.

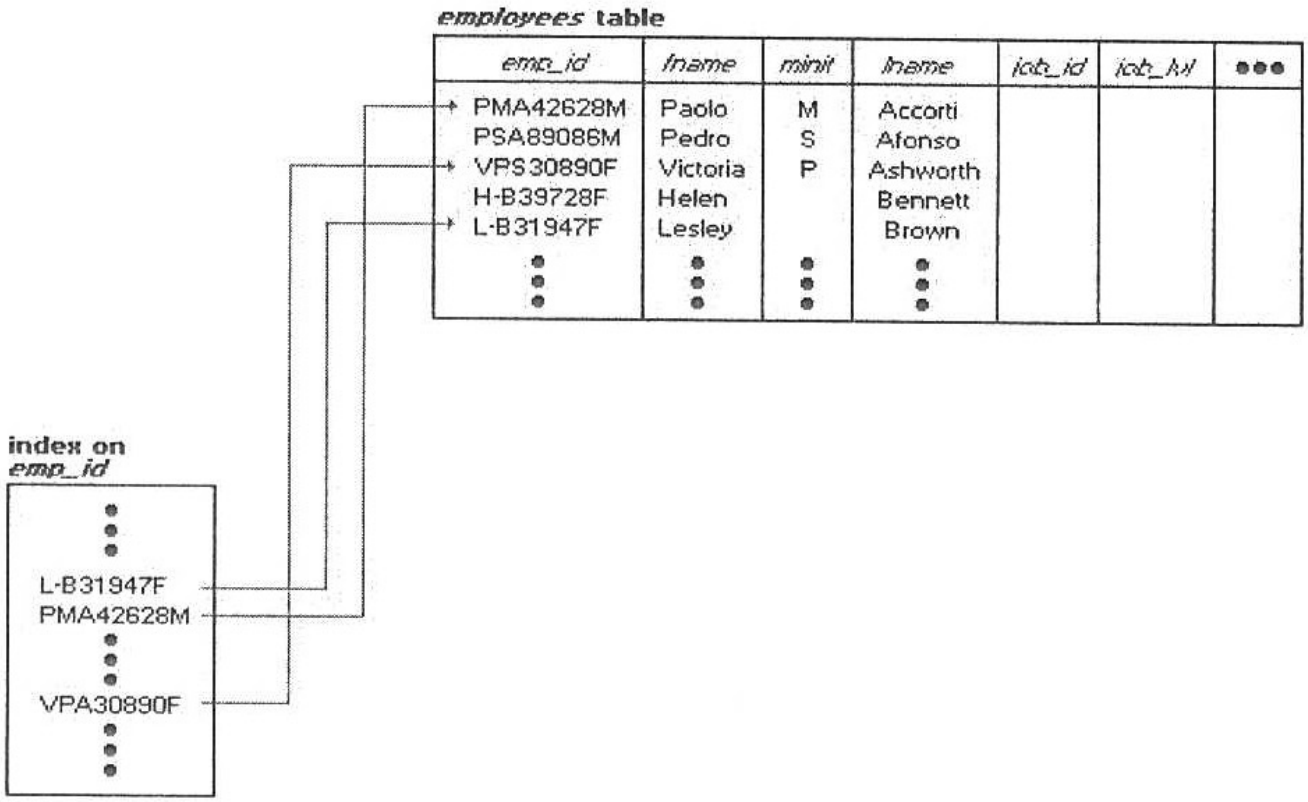
Triggerها حاوی دستورات Transact - SQL هستند. Triggerها همانند رویه‌های ذخیره شده، مجموعه نتیجه تولید شده توسط دستورات SELECT درون خود را باز می‌گردانند، لذا اجرای دستور SELECT در Triggerها توصیه نمی‌شود زیرا عموماً کاربران هنگام اجرای دستورات پردازش داده انتظار دیدن مجموعه نتیجه را ندارند.

Trigger پس از تکمیل اجرای دستوری که باعث فعال شدنش شده است، اجرا می‌شود. اگر اجرای دستور با خطا همراه باشد، Trigger مربوط به آن اجرا نمی‌شود.

شاخص‌های جدول

شاخص در SQL Server ساختاری مرتبط با یک جدول است که واکشی سطرهای یک جدول را تسریع می‌کند. شاخص شامل کلیدهایی است که از یک یا چند ستون جدول ساخته شده است. این کلیدها در ساختاری ذخیره شده‌اند که به SQL Server امکان می‌دهد سطرهای مرتبط با مقادیر شاخص را سریع‌تر و کارآتر جستجو نماید.

به عنوان مثال، در شکل زیر، جدول employee دارای یک شاخص روی ستون emp_id می‌باشد. چگونگی ذخیره هر مقدار emp_id و اشاره به سطرهای داده‌ای در جدول با هر یک از مقادیر مشاهده می‌شود.



«شکل ۱۷-۳: چگونگی عملکرد یک شاخص»

اگر جدولی بدون شاخص ایجاد شود، سطرهای داده‌ای به ترتیب خاصی ذخیره نمی‌شوند. این ساختار، **heap** نامیده می‌شود. دو نوع شاخص در **SQL Server** وجود دارد:

۱- خوشه‌ای: شاخص‌های خوشه‌ای سطرهای جدول را بر اساس مقادیر کلید مرتب کرده و ذخیره می‌کنند. چون سطرهای داده‌ای به ترتیب بر اساس کلید شاخص خوشه‌ای ذخیره می‌شوند، برای جستجوی سطرها کارآیی خوبی دارند. فقط یک شاخص خوشه‌ای می‌تواند برای هر جدول وجود داشته باشد، زیرا سطرها به خودی خود فقط به یک ترتیب می‌توانند مرتب باشند.

۲- غیر خوشه‌ای: شاخص‌های غیر خوشه‌ای دارای ساختاری هستند که به طور کامل از سطرهای داده‌ای مجزا می‌باشند. سطرها به ترتیب کلید خوشه‌ای ذخیره نمی‌شوند. پایین‌ترین سطرهای یک شاخص غیرخوشه‌ای شامل مقادیر کلید شاخص غیر خوشه‌ای است و هر کلید، اشاره‌گرهایی به سطرهایی دارد که شامل مقدار کلید هستند.

شاخص‌ها می‌توانند منحصربفرد باشند، در این صورت هیچ دو سطری نمی‌تواند مقدار یکسانی برای کلید شاخص داشته باشد.

عامل سرریز خاصیتی از شاخص **SQL Server** است که کنترل می‌کند شاخص

پس از ایجاد شدن، تا چه حد متراکم شود. عامل سرریز پیش فرض معمولاً دارای کارآیی خوبی است اما در بعضی اوقات بهتر است که تغییر کند. اگر جدول قرار است درج‌ها و بهنگام سازی‌های زیادی داشته باشد، ایجاد شاخص با عامل سرریز پایین فضای بیشتری را برای کلیدهای آتی باقی می‌گذارد. اگر جدول، یک جدول فقط خواندنی و بدون تغییر است، ایجاد شاخص با عامل سرریز بالا اندازه فیزیکی شاخص را کاهش می‌دهد. عامل سرریز فقط وقتی اعمال می‌شود که شاخص ایجاد شده باشد. در حالی که کلیدها درج و حذف می‌شوند، شاخص در نهایت در یک چگالی مشخص تثبیت می‌شود.

تصمیم‌گیری در مورد این که چه مجموعه‌ای از شاخص‌ها کارآیی را بهتر می‌کند بستگی به ترکیب پرس و جوها در سیستم دارد.

کاربرها

شناسه کاربر، یک کاربر را درون یک پایگاه داده شناسایی می‌کند. تمام جوازها و مالکیت شئ‌ها در پایگاه داده توسط حساب کاربر کنترل می‌شود. حساب‌های کاربر، متعلق به پایگاه داده خاص هستند. حساب کاربر **xyz** در پایگاه داده **sales** با حساب کاربر **xyz** در پایگاه داده **inventory** متفاوت می‌باشد. شناسه‌های کاربر بوسیله اعضای نقش پایگاه داده **db_owner** تعریف می‌شوند.

یک شناسهٔ **login** به خودی خود به یک کاربر جواز دستیابی به شئ‌های درون پایگاه داده را نمی‌دهد. قبل از این که کسی بتواند پس از اتصال بوسیله شناسه **login**، به شئ‌های درون پایگاه داده دسترسی یابد، آن شناسه **login** باید با یک شناسه کاربر در پایگاه داده مرتبط شود. اگر یک شناسه **login** صریحاً با یک شناسه کاربر در پایگاه داده مرتبط نشده باشد، به طور خودکار با شناسه کاربر **guest** مرتبط می‌شود. اگر پایگاه داده دارای حساب کاربر **guest** نباشد، **login** نمی‌تواند به پایگاه داده دسترسی یابد مگر این که با یک حساب کاربر مجاز مرتبط شود.

وقتی یک شناسه کاربر تعریف می‌شود، با یک شناسهٔ **login** مرتبط می‌شود. به عنوان مثال، عضوی از نقش **db_owner** می‌تواند شناسه **login** ویندوز **NT** مانند **NETDOMAIN\joe** را با شناسه کاربر **abc** در پایگاه داده **sales** و شناسه کاربر **def** در پایگاه داده **employee** مرتبط نماید.

یک کاربر در پایگاه داده بوسیله شناسه کاربر شناسایی می‌شود نه شناسه login. به عنوان مثال، sa یک حساب login است که به طور خودکار به حساب کاربر ویژه dbo در هر پایگاه داده نگاشته می‌شود.

حساب guest، یک حساب کاربر ویژه در پایگاه داده SQL Server است. اگر کاربر دستور USE را برای دستیابی به یک پایگاه داده صادر کند که در آن با یک حساب کاربر مرتبط نشده است، در عوض به کاربر guest مرتبط می‌شود.

نقش‌ها

نقش‌ها ابزاری قدرتمند هستند که به شما اجازه می‌دهند کاربران را در یک واحد جمع‌آوری کرده و به آنها جوازهایی را اعطاء نمایید. جوازهایی که به یک نقش اعطاء شده و یا از آن باز پس گرفته می‌شود به اعضای نقش نیز اعمال می‌شود. می‌توانید یک نقش بوجود آورید که نشان‌دهنده کارهای انجام شده بوسیله کلاس خاصی از کارکنان در سازمان شما باشد و جوازهای مناسب را به آن نقش اعطاء کنید. هنگامی که کارمند دیگری به این مجموعه اضافه می‌شود، به سادگی می‌توانید او را به عنوان عضو جدید نقش به آن اضافه کنید. وقتی آن شخص از مجموعه خارج می‌شود می‌توانید او را از نقش حذف کنید، در این صورت مجبور نخواهید بود به طور مکرر جوازهایی را به اشخاص اعطاء کرده و یا از آنها باز پس بگیرید.

اگر شما مجموعه‌ای از نقش‌ها را بر اساس کارهای مختلف تعریف کنید و به هر نقش جوازهای لازم برای انجام آن کار را انتساب کنید، مدیریت جوازها در پایگاه داده تسهیل خواهد شد. از این پس به راحتی می‌توانید کاربران را بین نقش‌ها جابجا کنید و دیگر نیازی به مدیریت انفرادی جوازهای کاربران نیست. اگر یک کار تغییر کند، بهتر آن است که جوازها را یک بار برای نقش تغییر دهید و این تغییرات به طور خودکار به تمام اعضای نقش اعمال می‌شود.

در SQL Server 7.0 کاربران می‌توانند به چند نقش متعلق باشند.

چند نقش ثابت در SQL Server 7.0 تعریف شده است. کاربران می‌توانند به این نقش‌ها اضافه شوند تا جوازهای مدیریتی مربوط را دریافت کنند. نقش‌های زیر در

سطح سرویس‌دهنده می‌باشند:

نقش ثابت سرویس‌دهنده	شرح
sysadmin	می‌تواند هر فعالیتی را در SQL Server انجام دهد.
Serveradmin	می‌تواند گزینه‌هایی را در سطح سرویس‌دهنده تنظیم نماید، سرویس‌دهنده را پایین بیاورد.
Setupadmin	می‌تواند سرویس‌دهنده‌های پیوندی و رویه‌های راه‌اندازی را مدیریت نماید.
Securityadmin	می‌تواند loginها و جوازهای CREATE DATABASE را مدیریت نماید. همچنین می‌تواند گزارش‌های خطا را بخواند.
Processadmin	می‌تواند فرآیندهایی را که در SQL Server اجرا می‌شوند مدیریت نماید.
dbcreator	می‌تواند پایگاه‌های داده را ایجاد کرده و تغییر دهد.
diskadmin	می‌تواند فایل‌های روی دیسک را مدیریت نماید.

می‌توانید با استفاده از رویه ذخیره شده سیستمی sp_helpsrvrole لیستی از نقش‌های ثابت سرویس‌دهنده و بوسیلهٔ sp_srvrolepermission جوازهای ویژه هر نقش را مشاهده کنید.

هر پایگاه داده دارای مجموعه‌ای از نقش‌های ثابت پایگاه داده هستند. حوزه هر یک از این نقش‌ها خاص یک پایگاه داده است. به عنوان مثال اگر Database1 و Database2 هر دو دارای شناسه کاربر به نام userx باشند، اضافه کردن userx مربوط به Database1 به نقش پایگاه داده db_owner، هیچ تأثیری بر userx مربوط به Database2 ندارد.

نقش ثابت پایگاه داده	شرح
db_owner	تمام جوازها را در پایگاه داده دارا می‌باشد.
db_accessadmin	می‌تواند شناسه‌های کاربر را اضافه یا حذف کند.
db_securityadmin	می‌تواند تمام جوازها، مالکیت شئ‌ها، نقش‌ها و اعضای نقش را مدیریت نماید.
db_ddladmin	می‌تواند تمام دستورات DDL به جز GRANT، REVOKE

	و DENY را اجرا نماید.
db_backupoperator	می‌تواند دستورات CHECKPOINT،DBCC و BACKUP را اجرا نماید.
db_datareader	می‌تواند تمام داده‌های هر یک از جدول‌های کاربران در پایگاه داده را بخواند.
db_datawriter	می‌تواند داده‌های هر یک از جدول‌های کاربران در پایگاه داده را تغییر دهد.
db_denydatareader	می‌تواند جوازهای SELECT را روی هر شئ باز پس بگیرد.
db_denydatawriter	می‌تواند جوازهای INSERT، UPDATE و DELETE را روی هر شئ باز پس بگیرد.

می‌توانید با استفاده از رویه ذخیره شده سیستمی sp_helpdbfixedrole لیستی از جوازهای ثابت پایگاه داده و بوسیله sp_dbfixedrolepermission جوازهای ویژه هر نقش را مشاهده نمایید.

هر کاربر در پایگاه داده به نقش public تعلق دارد. اگر می‌خواهید که تمام کاربران پایگاه داده جوازهای ویژه‌ای داشته باشند، این جوازها را به نقش public اعطاء کنید.

گروه ها

در SQL Server 7.0 مفهوم گروه وجود ندارد. نقش‌ها، جایگزین گروه‌ها در نسخه‌های اولیه SQL Server شده است. اما شما می‌توانید امنیت SQL Server را در سطح گروه ویندوز NT مدیریت کنید.

اگر از sp_grantlogin استفاده کرده و نام یک گروه ویندوز NT را مشخص کنید، تمام اعضای گروه می‌توانند با استفاده از تأیید صلاحیت توسط ویندوز NT به SQL Server متصل شوند. پس از اینکه گروه برای اتصال تأیید صلاحیت شد، می‌توانید از sp_grantdbaccess برای مرتبط کردن اعضای گروه با یک شناسه کاربر در پایگاه داده استفاده کنید. از دو روش می‌توانید استفاده کنید:

۱- می‌توانید گروه را با یک شناسه کاربر در پایگاه داده مرتبط کنید. در این حالت،

تمام اعضای گروه با آن شناسه کاربر مرتبط می‌شود.

۲- می‌توانید یک حساب کاربر در گروه ویندوز NT را با یک شناسه کاربر در پایگاه داده مرتبط کنید.

اعضای یک گروه دارای جوازهایی هستند که به کاربر مرتبط با گروه اعطاء شده است مگر این که حساب ویندوز NT آنها با یک کاربر خاص مرتبط شده باشد. اگر عضوی از گروه، یک شئ ایجاد کند نام مالک شئ نام حساب ویندوز NT خواهد بود نه نام گروه.

مالک‌ها و جوازها

هر شئ در SQL Server متعلق به یک کاربر می‌باشد. مالک با یک شناسه کاربر پایگاه داده شناسایی می‌شود. وقتی یک شئ برای اولین بار ایجاد می‌شود، تنها شناسه کاربری که می‌تواند به شئ دسترسی یابد شناسه کاربر مالک یا ایجاد کننده است. اگر کاربران دیگر بخواهند به شئ دسترسی یابند، مالک باید جوازهایی را به آنها اعطاء نماید.

برای جدول‌ها و دیدها، مالک می‌تواند جوازهای INSERT، UPDATE، DELETE، SELECT، REFRENCES یا ALL را اعطاء نماید. قبل از این که کاربر بتوانند دستورات INSERT، UPDATE، DELETE یا SELECT را روی یک جدول اجرا کند باید دارای جوازهای مربوطه باشد. جواز REFRENCES به مالک یک جدول دیگر امکان می‌دهد که از ستون‌های جدول شما در محدودیت REFRENCES FOREIGN KEY در تعریف ستون‌های جدول خودش استفاده نماید.

مالک یک رویه ذخیره شده می‌تواند جواز EXECUTE را برای رویه ذخیره شده اعطاء نماید. اگر مالک یک جدول مبنا بخواهد از دستیابی مستقیم کاربران به جدول جلوگیری کند، می‌تواند جوازهایی را برای دیدها یا رویه‌های ذخیره شده‌ای که به جدول ارجاع می‌کنند، اعطاء نماید اما هیچ جوازی را روی خود جدول اعطاء ننماید. این کار، شالوده مکانیزم‌های SQL Server برای تضمین این امر است که کاربران داده‌هایی را که صلاحیت دستیابی به آنها را ندارند، نمی‌بینند.

کاربران می‌توانند جوازهای دستورات را نیز دریافت کنند. بعضی از دستورات مانند CREATE TABLE و CREATE VIEW فقط باید توسط کاربران خاصی اجرا

شوند. (در این حالت، کاربر dbo). اگر کاربر dbo بخواهد اجازه ایجاد جدول یا دید را به دیگری نیز اعطاء نماید، می‌تواند جوازهایی را برای اجرای این دستورات به آن کاربر بدهد.

پایگاه‌های داده و داده‌های سیستمی

SQL Server دارای چهار پایگاه داده سیستمی است:

۱- پایگاه داده master تمام اطلاعات در سطح سیستم را برای SQL Server ثبت می‌کند، مانند حساب‌های login و تمام تنظیمات پیکربندی سیستم. master پایگاه داده‌ای است که وجود پایگاه‌های داده دیگر و مکان فایل‌های اصلی را که حاوی اطلاعات اولیه برای کاربران پایگاه داده هستند ثبت می‌کند.

۲- tempdb تمام جدول‌های موقت و رویه‌های ذخیره شده موقت را نگهداری می‌کند. tempdb یک منبع عمومی است. جدول‌های موقت و رویه‌های ذخیره شده تمام کاربرانی که به سیستم متصل هستند در این پایگاه داده ذخیره می‌شود. هر بار که SQL Server شروع به کار می‌کند tempdb دوباره ایجاد می‌شود. هنگام قطع اتصال، تمام جدول‌ها و رویه‌های ذخیره شده موقت حذف می‌شوند.

۳- model قالبی برای تمام پایگاه‌های داده‌ای است که در سیستم ایجاد می‌شود. وقتی یک دستور CREATE DATABASE صادر می‌شود، اولین قسمت از پایگاه داده با کپی کردن محتوای پایگاه داده model ایجاد می‌شود. سپس قسمت‌های باقیمانده از پایگاه داده با صفحه‌های خالی پر می‌شود.

۴- msdb توسط کارگزار SQL Server برای زمان‌بندی alertها و jobها و ثبت اپراتورها مورد استفاده قرار می‌گیرد.

هر پایگاه داده در SQL Server شامل جدول‌های سیستمی است که داده‌های مورد نیاز اجزاء SQL Server را ثبت می‌کند. موفقیت عملیات در SQL Server بستگی به جامعیت اطلاعات در جدول‌های سیستمی دارد. بنابراین کاربران نمی‌توانند مستقیماً اطلاعات جدول‌های سیستمی را بهنگام‌سازی کنند.

معماری فیزیکی پایگاه داده

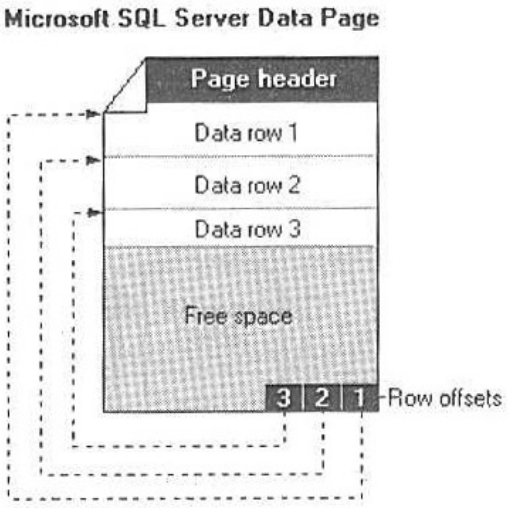
SQL Server 7.0 در روش ذخیره سازی فیزیکی داده‌ها پیشرفت‌های قابل توجهی نموده است. این تغییرات بیشتر برای کاربران معمولی SQL Server شفافیت دارند اما بر مدیریت پایگاه داده SQL Server نیز تأثیر دارند.

صفحه‌ها و Extentها

واحد اصلی ذخیره‌سازی داده‌ها در SQL Server، صفحه است. در SQL Server 7.0 اندازه صفحه‌ها ۸ کیلو بایت است. ۹۶ بایت اول هر صفحه، مربوط به سر صفحه است که اطلاعات سیستمی مانند نوع صفحه، میزان حافظه آزاد در صفحه و شناسه شیء‌ای که مالک صفحه است را ذخیره می‌کند. شش نوع صفحه در فایل‌های داده‌ای یک پایگاه داده وجود دارد:

- داده. شامل سطرهای داده‌ای با تمام داده‌ها به جز داده‌های نوع text، ntext و image است.
- شاخص. مدخل‌های شاخص را در بردارد.
- متن/ تصویر. شامل داده‌های نوع text، ntext و image است.
- نگاشت تخصیص عمومی. اطلاعاتی در مورد Extentهای تخصیص یافته را در بردارد.
- فضای آزاد صفحه. اطلاعاتی در مورد فضای آزاد موجود در صفحه‌ها را در بردارد.
- نگاشت تخصیص شاخص. حاوی اطلاعاتی در مورد Extentهایی است که توسط یک جدول یا شاخص مورد استفاده قرار گرفته است.

سطرهای داده‌ای در صفحه بلافاصله بعد از سر صفحه به طور ترتیبی قرار گرفته‌اند. در انتهای صفحه، جدول آفست سطر قرار دارد. این جدول برای هر سطر در صفحه شامل یک مدخل می‌باشد که هر مدخل مشخص می‌کند که سطر مزبور چند بایت از ابتدای صفحه فاصله دارد.



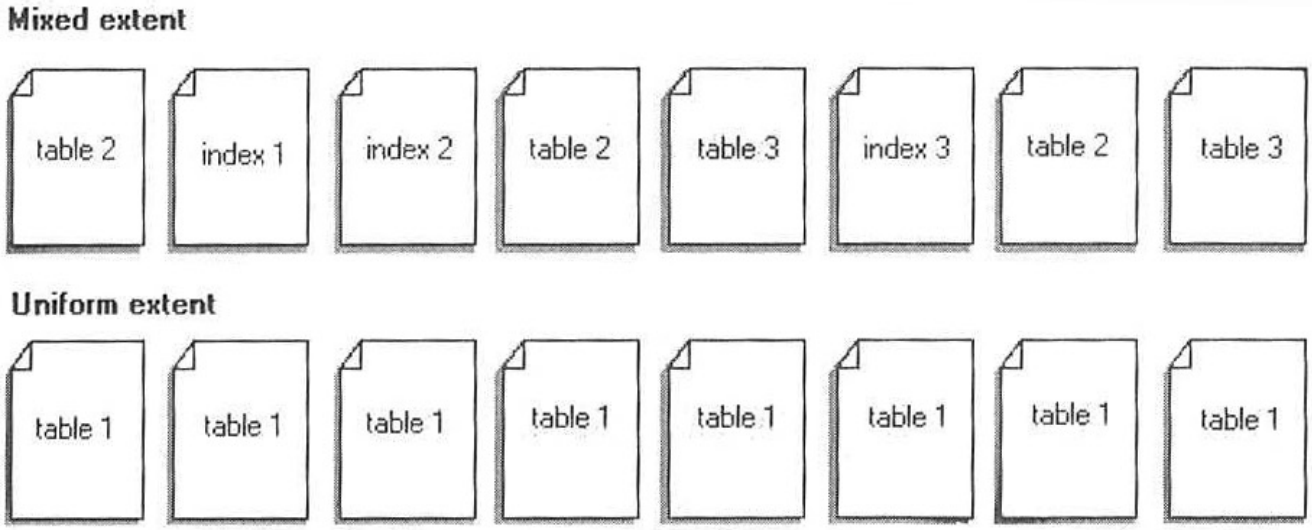
«شکل ۱۸-۳: صفحه داده‌ای SQL Server»

سطرها نمی‌توانند در صفحه‌ها زنجیره شوند. در SQL Server 7.0 حداکثر مقدار داده‌ای که می‌تواند در یک سطر قرار گیرد، بدون احتساب داده‌های نوع text، ntext و image ، ۸۰۶۰ بایت است.

Extentها واحد اصلی تخصیص فضا در جدول و شاخص‌ها هستند. هر Extent معادل هشت صفحه پیوسته یا ۶۴ کیلو بایت است. SQL Server برای تخصیص بهتر فضا، کل Extent را به جدول‌هایی که داده کمی دارند اختصاص نمی‌دهد. SQL Server 7.0 دارای دو نوع Extent است:

- Extentهای یکنواخت که متعلق به یک شئ واحد می‌باشند. تمام هشت صفحه موجود در Extent فقط توسط شئ مالک می‌توانند مورد استفاده قرار گیرند.
- Extentهای مخلوط که حداکثر بوسیله هشت شئ به اشتراک گذاشته می‌شوند.

صفحه‌هایی که به یک جدول یا شاخص جدید اختصاص داده می‌شود، از Extentهای مخلوط هستند. وقتی جدول یا شاخص به حدی رشد کند که دارای هشت صفحه شود، Extentهای آن به نوع یکنواخت تبدیل می‌شود.



«شکل ۱۹-۳: Extentهای مخلوط و یکنواخت»

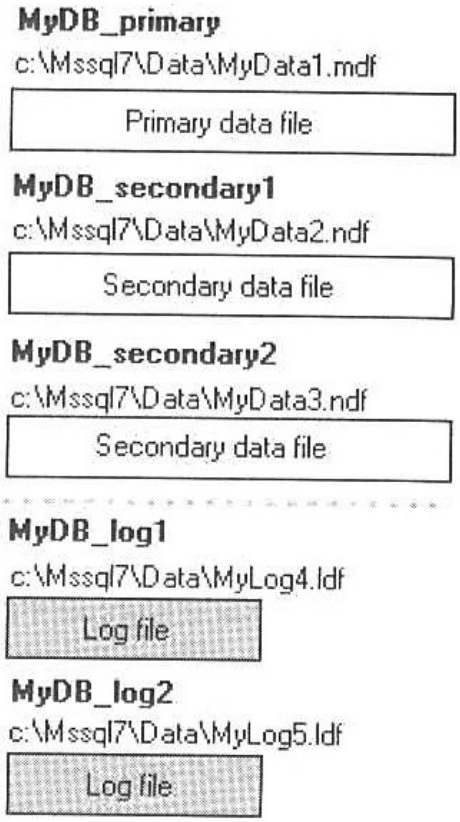
فایل‌ها و گروه‌های فایلی پایگاه داده فیزیکی

در SQL Server 7.0 یک پایگاه داده به مجموعه‌ای از فایل‌های سیستم عامل نگاشته می‌شود. پایگاه داده دارای سه نوع فایل می‌باشد:

- فایل‌های داده‌ای اصلی: این فایل، نقطه شروع پایگاه داده است و به سایر فایل‌ها در پایگاه داده اشاره می‌کند. هر پایگاه داده دارای یک فایل داده‌ای اصلی است. پسوند پیش فرض این فایل، **mdf.** است.
- فایل‌های داده‌ای ثانوی: این فایل‌ها شامل تمام فایل‌های داده‌ای به جز فایل داده‌ای اصلی هستند. بعضی از پایگاه‌های داده ممکن است فایل داده‌ای ثانوی نداشته باشند. پسوند پیش فرض این فایل‌ها، **ndf.** است.
- فایل‌های log: این فایل‌ها تمام اطلاعات log لازم برای ترمیم پایگاه داده را نگهداری می‌کنند. هر پایگاه داده باید حداقل یک فایل log داشته باشد. پسوند پیش فرض این فایل‌ها، **ldf.** است.

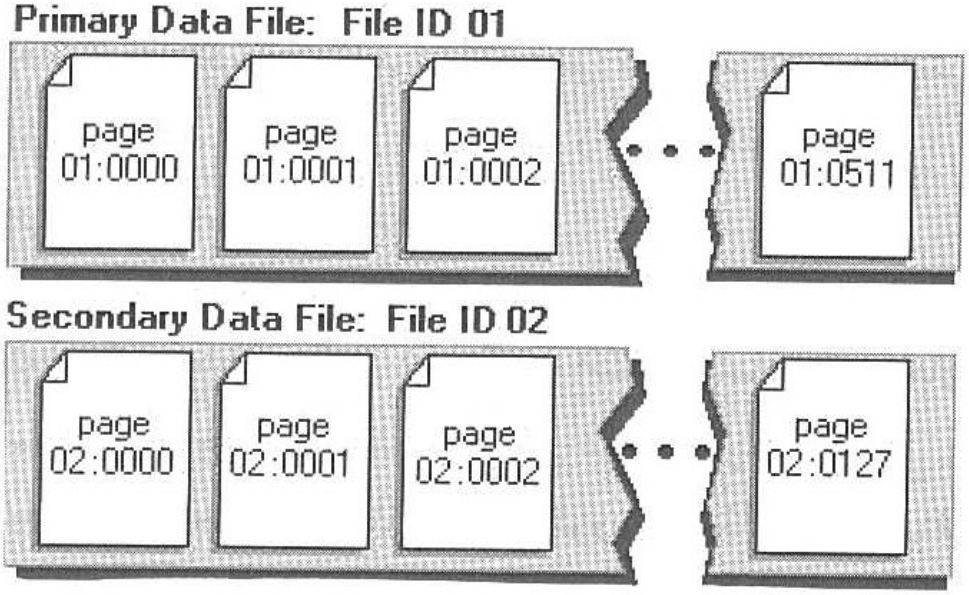
فایل‌های SQL Server دارای دو نام هستند:

- نام منطقی فایل که در دستورات Transact- SQL برای ارجاع به فایل مورد استفاده قرار می‌گیرد.
- نام سیستم عاملی فایل که نام فیزیکی آن می‌باشد.



«شکل ۲۰-۳ : نام فایل‌ها در SQL Server»

صفحه‌ها در یک فایل به طور ترتیبی با شروع از صفر برای اولین صفحه، شماره‌گذاری می‌شوند. هر فایل دارای یک عدد شناسه فایل است. برای شناسائی منحصربفرد یک صفحه در پایگاه داده، هم شناسه فایل و هم شماره صفحه مورد نیاز است. مثال زیر، پایگاه داده‌ای را نشان می‌دهد که دارای فایل داده‌ای اصلی با ۴ مگابایت و فایل داده‌ای ثانوی با ۱ مگابایت است.



«شکل ۲۱-۳: شماره فایل و شماره صفحه»

اولین صفحه در هر فایل شامل اطلاعات در مورد صفحات فایل می‌باشد. نهمین

صفحه در فایل داده‌ای اصلی، صفحه راه اندازی پایگاه داده است که شامل اطلاعاتی در مورد صفات پایگاه داده می‌باشد.

فایل‌ها می‌توانند از اندازه اولیه‌ای که برای آنها مشخص شده است به طور خودکار رشد کنند. وقتی یک فایل را تعریف می‌کنید می‌توانید یک مقدار رشد نیز برای آن مشخص کنید. هر موقع فایل پر می‌شود، اندازه آن به مقدار مشخص شده افزایش می‌یابد.

هر فایل دارای یک حداکثر اندازه نیز می‌باشد. اگر حداکثر اندازه مشخص نشود رشد فایل ادامه می‌یابد تا این که تمام فضای موجود روی دیسک مصرف شود. فایل‌های پایگاه داده به منظور تخصیص و مدیریت آسان می‌توانند در گروه‌های فایلی دسته‌بندی شوند. بعضی از سیستم‌ها می‌توانند با قرار دادن داده‌ها و شاخص‌ها روی دیسک‌های مشخص، کارآیی خود را بهبود بخشند. گروه‌های فایلی می‌توانند به این فرآیند کمک کنند. مدیر سیستم می‌تواند برای هر دیسک گروه فایلی ایجاد کند و سپس جدول‌ها و شاخص‌های ویژه‌ای را به گروه فایلی انتساب کند.

یک فایل فقط می‌تواند عضو یک گروه فایلی باشد. جدول‌ها و شاخص‌ها و داده‌های text، ntext و image می‌توانند با یک گروه فایلی مرتبط باشند، در این صورت تمام صفحه‌هایی که به آن اختصاص داده می‌شود در آن گروه فایلی خواهد بود.

فایل‌های log هرگز در گروه فایلی قرار نمی‌گیرند. فضای log به صورت مجزا از فضای داده‌ای مدیریت می‌شود.

فایل‌های درون یک گروه فایلی به طور خودکار رشد نمی‌کنند، مگر این که در هیچ یک از فایل‌های درون گروه فایلی فضای آزاد وجود نداشته باشد.

سه نوع گروه فایلی وجود دارد:

- اصلی. گروه فایلی اصلی شامل فایل داده‌ای اصلی و فایل‌های دیگری است که در گروه فایلی دیگری قرار نگرفته است. تمام صفحه‌های جدول‌های سیستمی در این گروه فایلی تخصیص داده می‌شوند.
- تعریف شده توسط کاربر. این گروه‌های فایلی با استفاده از کلمه کلیدی FILEGROUP در دستور CREATE DATABASE یا ALTER DATABASE ایجاد می‌شوند.
- پیش‌فرض. این گروه فایلی شامل صفحه‌های تمام جدول‌ها و شاخص‌هایی

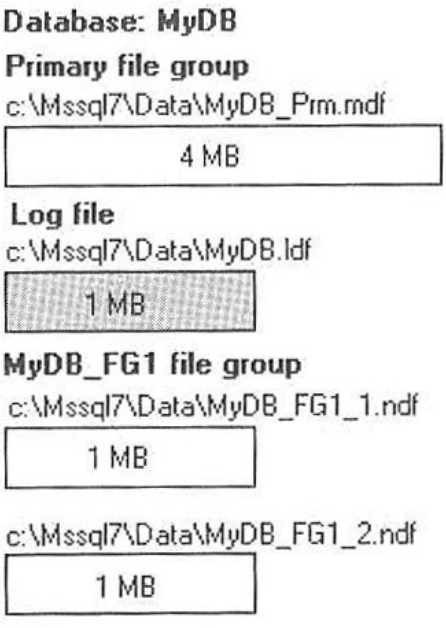
است که هنگام ایجاد شدن، هیچ گروه فایلی برای آنها مشخص نمی‌شود. در هر پایگاه داده، در هر لحظه فقط یک گروه فایلی می‌تواند گروه فایلی پیش فرض باشد. اعضای نقش پایگاه داده **db_owner** می‌توانند گروه فایلی پیش فرض را از یک گروه فایلی به گروه فایلی دیگر منتقل کنند. اگر گروه فایلی پیش فرض مشخص نشود گروه فایلی اصلی، گروه فایلی پیش فرض خواهد بود.

گروه‌های فایلی در واقع، ورودی/ خروجی را بین چند دیسک توزیع می‌کنند.

در مثال زیر، یک پایگاه داده با یک فایل داده‌ای اصلی، یک گروه فایلی تعریف شده توسط کاربر و یک فایل log ایجاد می‌شود. فایل داده‌ای اصلی در گروه فایل اصلی قرار دارد و گروه فایلی تعریف شده توسط کاربر دارای دو فایل داده‌ای ثانوی است.

```
USE master
GO
-- Create the database with the default data
-- filegroup and the log file. Specify the
-- growth increment and the max size for the
-- primary data file.
CREATE DATABASE MyDB
ON PRIMARY
( NAME='MyDB_Primary',
FILENAME='c:\mssql7\data\MyDB_Prm.mdf',
SIZE=4,
MAXSIZE=10,
FILEGROWTH=1),
FILEGROUP MyDB_FG1
( NAME = 'MyDB_FG1_Dat1',
FILENAME = 'c:\mssql7
\data\MyDB_FG1_2.ndf',
SIZE = 1MB,
```

```
MAXSIZE=10,
FILEGROWTH=1)
LOG ON
( NAME='MyDB_log',
FILENAME='c:\mssql7\data\MyDB.ldf',
SIZE=1,
MAXSIZE=10,
FILEGROWTH=1)
GO
ALTER DATABASE MyDB
MODIFY FILEGROUP MyDB_FG1 DEFAULT
GO
-- Create a table in the user-defined filegroup.
USE MyDB
CREATE TABLE MyTable
( cola int PRIMARY KEY,
colb char(8) )
ON MyDB_FG1
GO
```



«شکل ۲۲- ۳: پایگاه داده MyDB»

SQL Server به سرعت صفحه‌ها را برای شئ‌ها تخصیص می‌دهد و فضایی را که بوسیله حذف سطرها آزاد می‌شود دوباره مورد استفاده قرار می‌دهد. این عملیات به طور داخلی در سیستم انجام می‌شود و از ساختارهای داده‌ای استفاده می‌کند که برای کاربران قابل رؤیت نیستند.

اگر در پایگاه داده فضای آزاد زیادی وجود داشته باشد، SQL Server پایگاه داده را به طور خودکار می‌شکند. این در مواقعی انجام می‌شود که گزینه autoshrink در پایگاه داده دارای مقدار true باشد. سرویس‌دهنده، استفاده از فضا در هر پایگاه داده را به طور نوبه‌ای بررسی می‌کند. اگر پایگاه داده دارای فضای آزاد زیادی باشد، SQL Server اندازه فایل‌های پایگاه داده را کاهش می‌دهد. همچنین می‌توان با استفاده از Enterprise Manger یا دستور DBCC SHRINKDATABASE این کار را به طور دستی انجام داد.

ساختارهای داده‌ای SQL Server که فضای آزاد را ردیابی می‌کنند نسبتاً ساده هستند. این امر دو مزیت دارد:

- اطلاعات فضای آزاد بسیار فشرده می‌شود، لذا نسبتاً صفحه‌های کمی شامل این گونه اطلاعات هستند. این امر با کاهش دادن میزان خواندن از دیسک که برای واکشی اطلاعات تخصیص لازم است، سرعت را افزایش می‌دهد. همچنین احتمال این که صفحه‌های تخصیص در حافظه باقی بمانند افزایش می‌یابد که خود باعث کاهش بیشتر خواندن از دیسک می‌شود.
- بیشتر اطلاعات تخصیص به یکدیگر زنجیره نشده‌اند. این امر موجب می‌شود که نگهداری اطلاعات تخصیص تسهیل شود.
- SQL Server برای ثبت تخصیص Extentها از دو نوع نگاشت تخصیص استفاده می‌کند:
- نگاشت تخصیص عمومی (GAM): صفحه‌های GAM، Extentهایی را که تخصیص یافته‌اند ثبت می‌کنند. هر GAM، ۶۴ هزار Extent یا تقریباً ۴ گیگا بایت داده را پوشش می‌دهند. GAM دارای یک بیت برای هر Extent است. اگر بیت برابر ۱ باشد، Extent متناظر آزاد است.

- نگاشت تخصیص عمومی مشترک (SGAM) : صفحه‌های SGAM، Extentهایی را که هم اکنون به عنوان Extent مخلوط استفاده شده‌اند و حداقل دارای یک صفحه استفاده نشده هستند، ثبت می‌کنند. هر SGAM، ۶۴ هزار Extent یا تقریباً ۴ گیگا بایت داده را پوشش می‌دهند. SGAM دارای یک بیت برای هر Extent است. اگر بیت برابر ۱ باشد، Extent از نوع مخلوط است و دارای صفحه‌های آزاد می‌باشد. اگر بیت برابر 0 باشد، Extent به عنوان Extent مخلوط استفاده نشده است و یا Extent مخلوطی است که تمام صفحه‌های آن مورد استفاده قرار گرفته‌اند.

هر Extent دارای الگوهای بیتی به صورت زیر است که بر مبنای استفاده جاری از آنها، در GAM و SGAM ثبت شده‌اند:

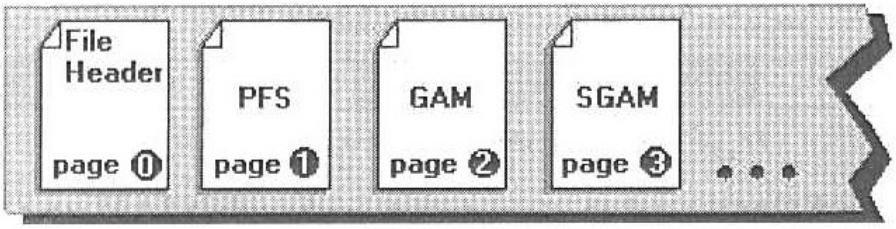
استفاده جاری از Extent	مقدار بیت در GAM	مقدار بیت در SGAM
آزاد	۱	۰
Extent یکنواخت یا Extent مخلوط و کاملاً پر	۰	۰
Extent مخلوط با صفحه‌های آزاد	۰	۱

در نتیجه، مدیریت Extentها الگوریتم ساده‌ای خواهد داشت. برای تخصیص یک Extent یکنواخت، SQL Server در GAM به دنبال بیتی می‌گردد که برابر ۱ باشد و سپس آن را برابر 0 قرار می‌دهد. برای یافتن Extent مخلوط با صفحه‌های آزاد، SGAM برای پیدا کردن بیت ۱ جستجو می‌شود. برای تخصیص یک Extent مخلوط، GAM برای بیت ۱ جستجو می‌شود و مقدار بیت برابر 0 قرار داده می‌شود و سپس بیت متناظر در SGAM نیز برابر ۱ می‌شود. برای آزاد کردن یک Extent، SQL Server تضمین می‌کند که بیت متناظر در GAM برابر ۱ و در SGAM برابر 0 باشد. البته، الگوریتم‌هایی که به طور داخلی در SQL Server اجرا می‌شوند پیچیده‌تر از آن هستند که در اینجا مورد بررسی قرار گرفت.

صفحه‌هایی به نام فضای آزاد صفحه (PFS) وجود دارد که نشان می‌دهند آیا یک صفحه تخصیص داده شده است یا خیر و مقدار فضای آزاد صفحه را نشان می‌دهند. هر PFS، ۸۰۰۰ صفحه را پوشش می‌دهد. برای هر صفحه، PFS دارای یک نقش بیت است که مشخص می‌کند آیا صفحه خالی است، ۵۰-۱ درصد پر، ۸۰-۵۱ درصد پر، ۹۵-۸۱

درصد پر یا ۱۰۰-۹۶ درصد پر است.

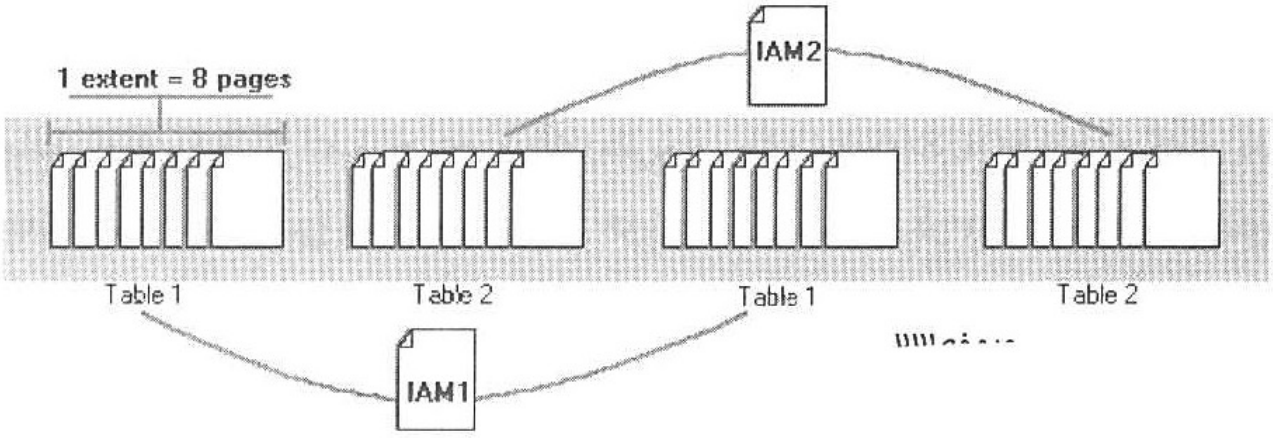
صفحه PFS اولین صفحه بعد از سرصفحه در فایل داده‌ای است (با شماره ۱). بعد از آن GAM (با شماره ۲) و به دنبال آن SGAM (با شماره ۳) قرار می‌گیرد. ۸۰۰۰ صفحه بعد از اولین PFS یک PFS دیگر قرار می‌گیرد. ۶۴ هزار Extent بعد از اولین GAM، یک GAM دیگر قرار می‌گیرد. ۶۴ هزار Extent بعد از اولین SGAM، یک SGAM دیگر قرار می‌گیرد و الی آخر.



«شکل ۲۳-۳: ترتیب قرار گرفتن PFS، GAM، SGAM»

مدیریت فضای مورد استفاده شئ‌ها

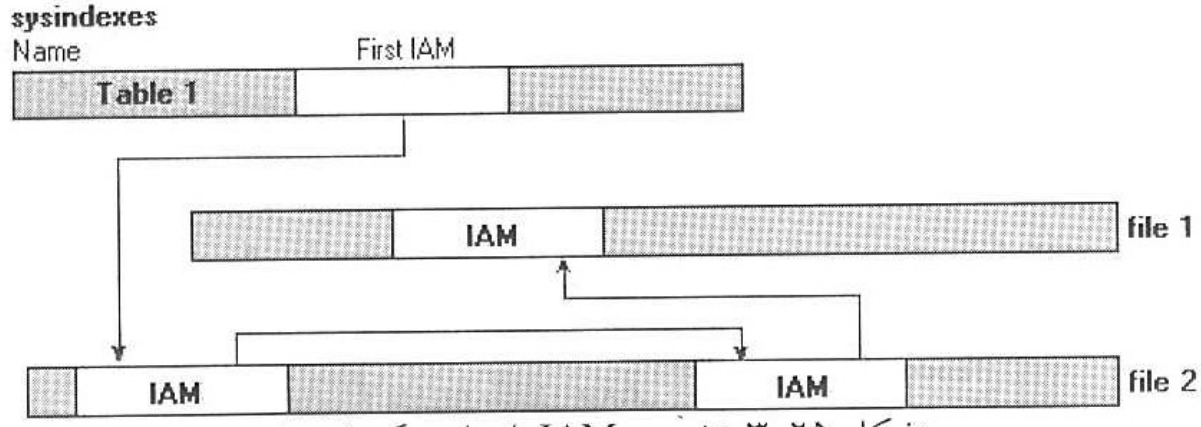
صفحه‌های نگاشت تخصیص شاخص (IAM)، Extentهای یک فایل پایگاه داده را که بوسیله یک heap یا شاخص مورد استفاده قرار گرفته است، معین می‌کند. هر heap یا شاخص دارای یک یا چند صفحه IAM است که تمام Extentهای تخصیص یافته به شئ را ثبت می‌کند. یک heap یا شاخص حداقل دارای یک IAM برای هر فایلی است که در آن Extentهایی دارد. اگر محدوده Extentهای مربوط به heap یا شاخص در فایل فراتر از حدی باشد که IAM بتواند ثبت کند، در این صورت heap یا شاخص ممکن است بیش از یک IAM در یک فایل داشته باشد.



«شکل ۲۴-۳ : صفحه‌های IAM»

صفحه‌های IAM بر حسب نیاز برای شئ تخصیص داده می‌شوند و در فایل به طور

تصادفی قرار می‌گیرند. sysindexes.dbo.FirstIAM به اولین صفحه IAM در شئ اشاره می‌کند و تمام صفحه‌های IAM برای آن شئ در یک زنجیره به هم پیوسته‌اند.



«شکل ۲۵-۳: زنجیره IAMها برای یک شیء»

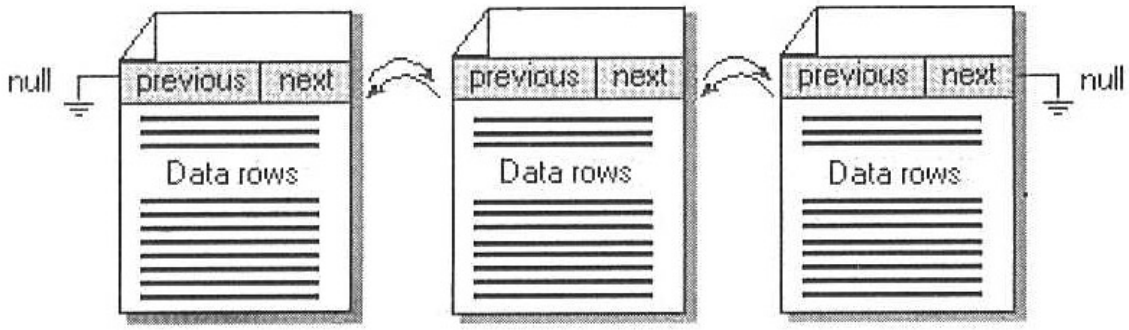
یک صفحه IAM دارای یک سرصفحه است که اولین Extent واقع در محدوده Extentهایی را که توسط IAM نگاشته شده است، معین می‌کند. IAM دارای یک نقش بیت نیز هست که هر بیت آن متناظر با یک Extent است. اولین بیت، متناظر با اولین Extent، دومین بیت، متناظر با دومین Extent و ... می‌باشد. اگر بیت برابر صفر باشد Extent متناظر، به شیءای که مالک صفحه IAM است تخصیص نیافته است.

اگر SQL Server بخواهد سطر جدیدی را درج کند و فضایی در صفحه جاری موجود نباشد، با استفاده از صفحه‌های IAM و PFS صفحه‌ای را پیدا می‌کند که دارای فضای کافی برای نگهداری سطر باشد. از IAM برای یافتن Extentهایی که به شئ اختصاص داده شده، استفاده می‌شود. برای هر Extent، صفحه‌های PFS برای جستجوی صفحه‌ای که دارای فضای آزاد کافی برای نگهداری سطر باشد، مورد استفاده قرار می‌گیرند. هر صفحه IAM و PFS تعداد زیادی از صفحه‌های داده‌ای را پوشش می‌دهند و لذا در یک پایگاه داده تعداد کمی از این صفحه‌ها وجود دارد، به همین دلیل این صفحه‌ها درحافظه اصلی قرار گرفته و جستجوی آنها سریع می‌باشد. SQL Server فقط هنگامی یک Extent جدید به یک شئ تخصیص می‌دهد که نتواند صفحه‌ای در Extentهای موجود پیدا کند که دارای فضای آزاد کافی برای نگهداری سطر باشد. Extentهای جدید از گروه فایلی تخصیص داده می‌شوند. اگر یک گروه فایلی دارای دو فایل باشد که فضای آزاد یکی از آنها دو برابر دیگری است، به ازاء هر صفحه از یک فایل دو صفحه از فایل دیگر تخصیص می‌یابد. این بدان معنی است که در یک گروه فایلی درصد فضای مصرفی برای

تمام فایل‌ها باید یکسان باشد.

ساختار جدول‌ها و شاخص‌ها

داده‌های هر جدول در صفحه‌های ۸ کیلوبایتی ذخیره می‌شوند. هر صفحه داده‌ای دارای ۹۶ بایت به عنوان سرصفحه است که شامل اطلاعات سیستمی مانند شناسه جدولی که مالک صفحه است و اشاره‌گرهایی به صفحه‌های قبلی و بعدی، می‌باشد. یک جدول آفست سطر نیز در انتهای صفحه قرار دارد. سطرهای داده‌ای، باقیمانده صفحه را پر می‌کنند.



«شکل ۲۶-۳: ساختار جدول»

SQL Server از دو روش برای سازماندهی صفحه‌های داده‌ای استفاده می‌کند:

- جدول‌های خوشه‌ای جدول‌هایی هستند که دارای یک شاخص خوشه‌ای می‌باشند. سطرهای داده‌ای به ترتیب کلید شاخص خوشه‌ای ذخیره می‌شوند. صفحه‌های داده‌ای در یک لیست پیوندی دو طرفه به هم پیوسته‌اند. شاخص به صورت ساختار B-tree پیاده‌سازی می‌شود.
- heapها جدول‌هایی هستند که شاخص خوشه‌ای ندارند. سطرهای داده‌ای با هیچ ترتیب خاصی ذخیره نمی‌شوند.

هر جدول می‌تواند ۲۴۹ شاخص غیر خوشه‌ای داشته باشد. شاخص‌های غیر خوشه‌ای نیز ساختار B-tree دارند. این شاخص‌ها هیچ تأثیری بر ترتیب سطرهای داده‌ای ندارند.

صفحه‌هایی که داده‌های text، ntext و image را نگهداری می‌کنند، برای هر جدول به صورت یک واحد منفرد مدیریت می‌شوند. تمام این داده‌ها برای یک جدول در یک مجموعه از صفحه‌ها ذخیره می‌شوند.

به ازاء هر جدول و شاخص، یک سطر در جدول sysindexes وجود دارد که به طور منحصربفرد با ترکیبی از دو ستون شناسه شئ (id) و شناسه شاخص (indid) شناسایی می‌شود. تخصیص صفحه‌ها به جدول و شاخص بوسیله زنجیره‌ای از صفحه‌های IAM مدیریت می‌شود. ستون sysindexes.FirstIAM به اولین صفحه IAM در زنجیره صفحه‌های IAM اشاره می‌کند.

هر جدول دارای مجموعه‌ای از سطرها در sysindexes می‌باشد:

- heap دارای یک سطر در sysindexes است که در آن indid = 0. ستون IAM به زنجیره IAM مربوط به مجموعه صفحه‌های داده‌ای جدول اشاره می‌کند.
- شاخص خوشه‌ای دارای یک سطر در sysindexes است که در آن indid=1. ستون root به بالای شاخص خوشه‌ای B-tree اشاره می‌کند.
- هر شاخص غیر خوشه‌ای که برای جدول ایجاد می‌شود دارای یک سطر در sysindexes است. indid دارای مقادیری بین 2 تا 201 است. ستون root به بالای شاخص غیر خوشه‌ای اشاره می‌کند.
- هر جدول که دارای حداقل یک ستون از نوع text، ntext و image باشد، دارای یک سطر در sysindexes است که در آن indid=255.

ساختار Heap

SQL Server از صفحه‌های IAM برای حرکت در Heap استفاده می‌کند. صفحه‌های داده‌ای و سطرهای درون آنها به ترتیب خاصی قرار ندارند و به یکدیگر پیوسته نیستند. تنها اتصال منطقی بین صفحه‌های داده‌ای، چیزهایی است که در صفحه‌های IAM ثبت می‌شود.

خواندن ترتیبی یک Heap با دنبال کردن IAMها انجام می‌شود تا Extentهایی که صفحه‌های مربوط به Heap در آنها قرار دارند، جستجو شوند. چون IAM، Extentها

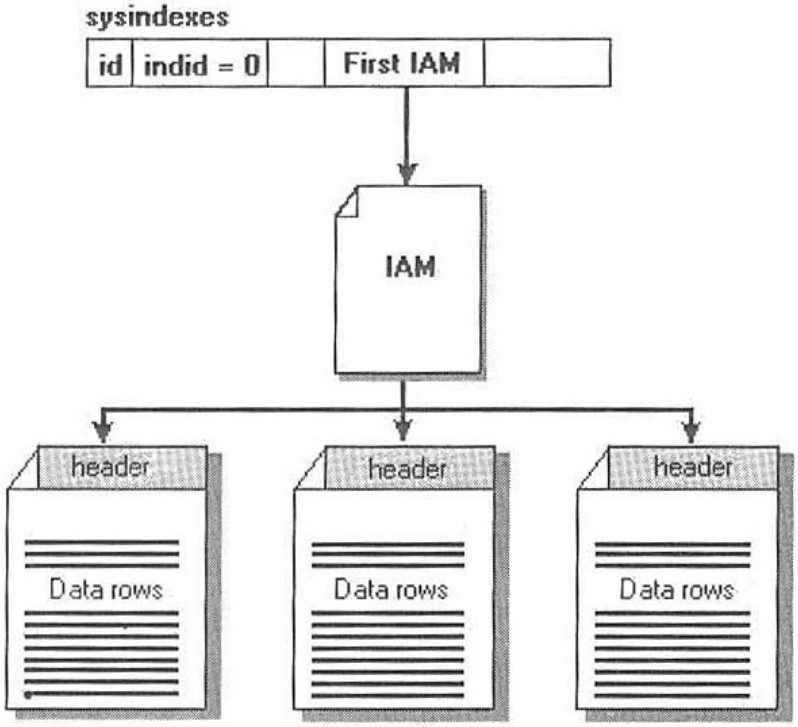
را به ترتیبی که در فایل هستند نشان می‌دهند، لذا خواندن ترتیبی **Heap** به طور یکنواخت تا انتهای فایل ادامه می‌یابد.

خواندن جدول روی **Heap** به صورت زیر انجام می‌شود:

۱- اولین **IAM** در اولین گروه فایلی خوانده می‌شود و تمام **Extent**هـــای موجـود در **IAM** دنبال می‌شوند.

۲- این فرآیند برای هر **IAM** مربوط به **Heap** در فایل انجام می‌شود.

۳- مراحل ۱ و ۲ برای هر فایل در پایگاه داده یا گروه فایلی تکرار می‌شود تا این کـــه آخرین **IAM** در **Heap** پردازش شود.



«شکل ۲۷-۳: ساختار Heap»

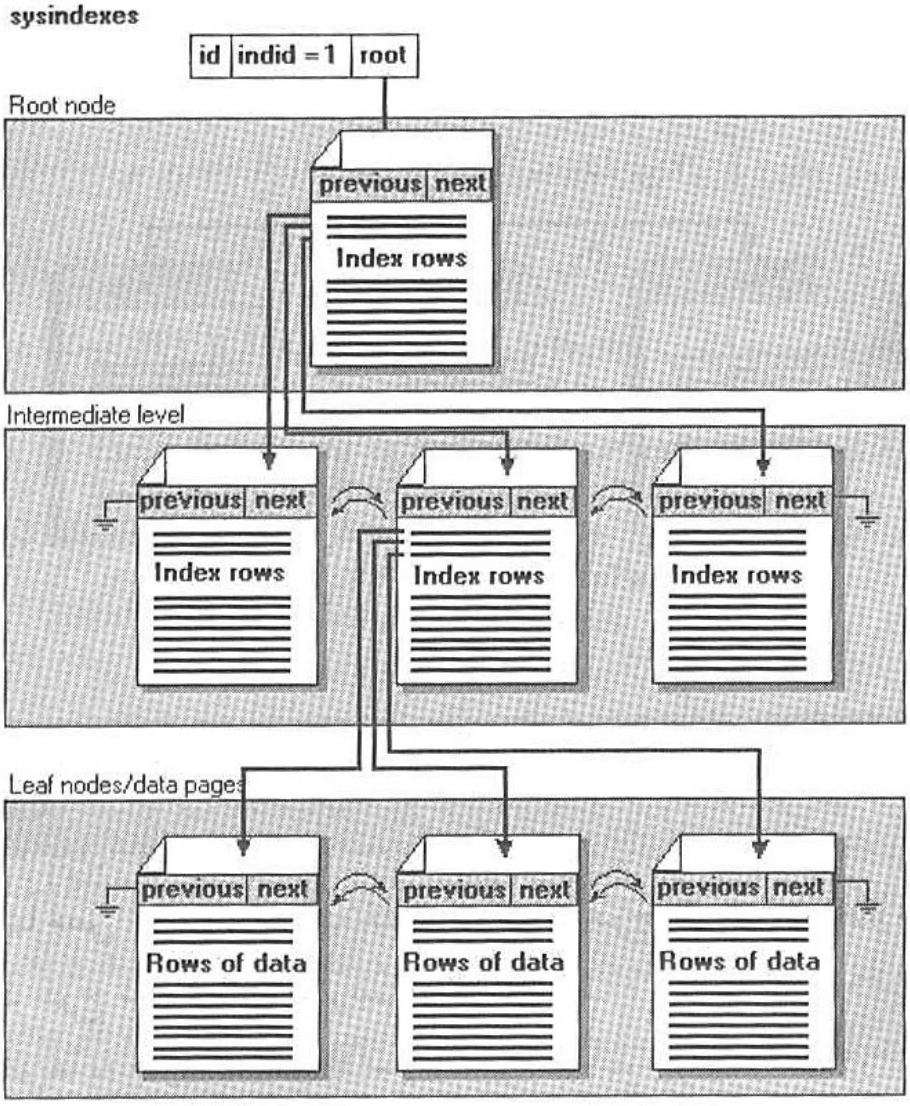
شاخص‌های خوشه‌ای

در شاخص‌های خوشه‌ای، صفحه‌های درون زنجیره داده‌ها و سطرهای درون آنها برحسب کلید شاخص خوشه‌ای مرتب هستند. عملیات درج در محلی انجام می‌شـــود کـه مقدار کلید سطر درج شده مطابق با ترتیب سطرها باشد.

شاخص‌ها به صورت **B-tree** سازماندهی شده‌اند. هر صفحـــه در یـک شـاخص حاوی یک سرصفحه و به دنبال آن سطرهای شاخص می‌باشد. هر سطر شاخص شامل یک مقدار کلید و یک اشاره‌گر است که به یک صفحه یا سطر داده‌ای اشاره می‌کند. هر صفحه

در یک شاخص یک گره شاخص نامیده می‌شود. گره بالایی B-tree گره ریشه نامیده می‌شود. لایه پایینی گره‌ها در شاخص گره‌های برگ نامیده می‌شوند که به صورت یک لیست پیوندی دو طرفه هستند. در شاخص خوشه‌ای صفحه‌های داده‌ای همان گره‌های برگ هستند.

SQL Server برای جستجوی سطری که متناظر با یک کلید شاخص است از بالا تا پایین شاخص حرکت می‌کند. برای جستجوی محدوده‌ای از کلیدها، SQL Server در شاخص حرکت کرده تا اولین مقدار کلید در محدوده را پیدا کند. سپس با استفاده از اشاره‌گرهای next و previous بین صفحه‌های داده‌ای حرکت می‌کند. برای یافتن اولین صفحه در زنجیره صفحه‌های داده‌ای، سمت چپ‌ترین اشاره‌گرها از گره ریشه شاخص دنبال می‌شود.



«شکل ۲۸-۳: ساختار شاخص خوشه‌ای»

ساختار شاخص غیر خوشه‌ای با شاخص خوشه‌ای دو تفاوت عمده دارد:

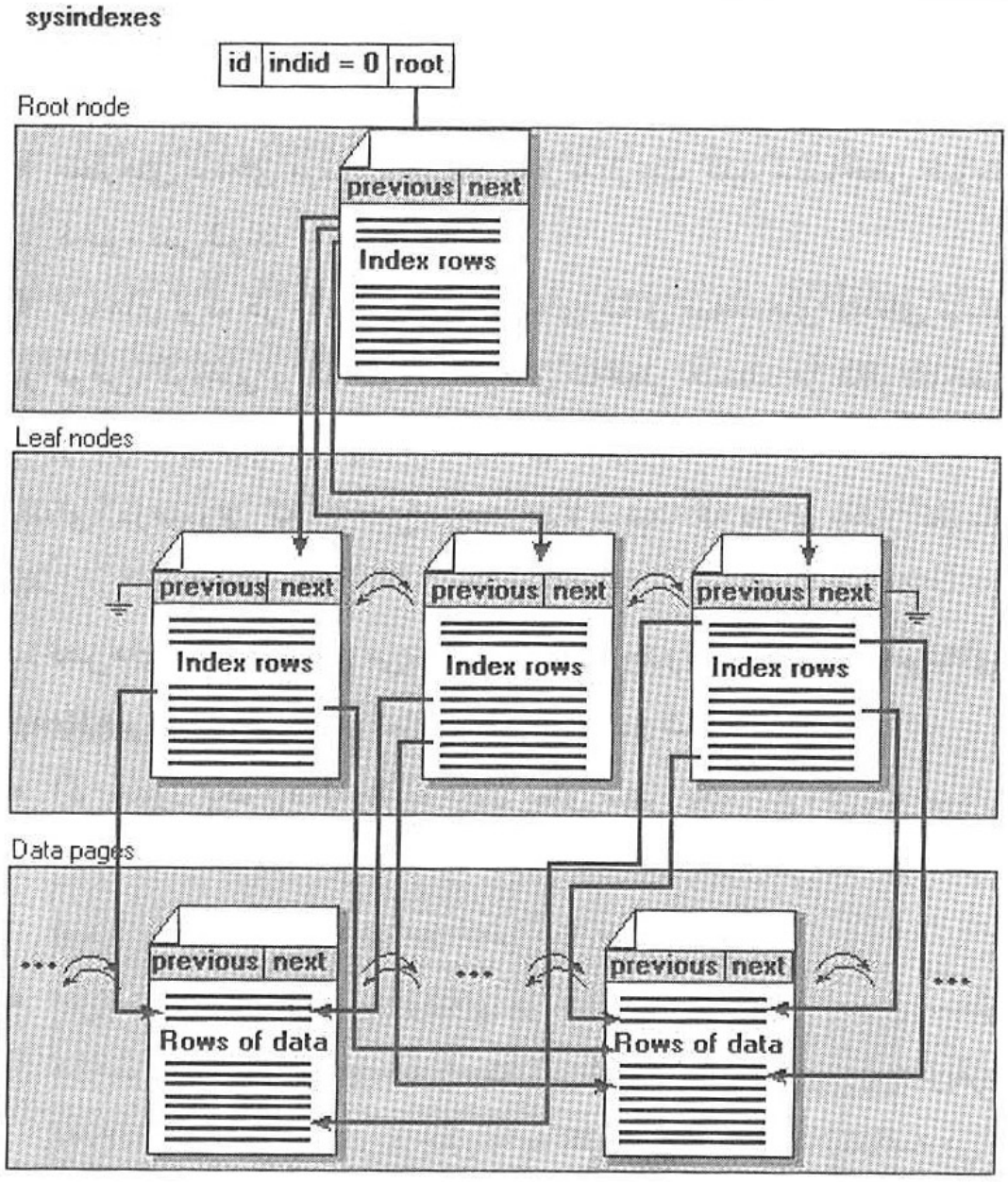
- سطرهای داده‌ای مرتب شده نیستند و به ترتیب کلید شاخص غیرخوشه‌ای ذخیره نمی‌شوند.
- لایه‌های برگ در شاخص غیر خوشه‌ای شامل صفحه‌های داده‌ای نیستند.

در عوض، گره‌های برگ حاوی سطرهای شاخص هستند. هر سطر شاخص شامل مقدار کلید غیر خوشه‌ای و یک یا چند مکان نمای سطر است که به سطر داده‌ای (یا سطرهای داده‌ای، درصورتی که شاخص منحصربفرد نباشد) که دارای مقدار کلید است اشاره می‌کند.

شاخص‌های غیر خوشه‌ای می‌توانند هم روی جدولی با شاخص خوشه‌ای و هم روی یک **Heap** تعریف شوند. مکان نمای سطر دارای دو شکل است:

- اگر جدول یک **Heap** باشد، مکان نمای سطر، یک اشاره‌گر به سطر است. اشاره‌گر از شناسه فایل، شماره صفحه و شماره سطر در صفحه تشکیل شده است. به کل اشاره‌گر، شناسه سطر گفته می‌شود.
- اگر جدول یک شاخص خوشه‌ای داشته باشد، مکان نمای سطر، کلید شاخص خوشه‌ای برای سطر است. اگر شاخص خوشه‌ای، منحصربفرد نباشد، **SQL Server** یک مقدار داخلی را به کلیدهای مضاعف اضافه می‌کند تا منحصربفرد شوند. این مقدار برای کاربر قابل رؤیت نیست. این مقدار برای منحصربفرد کردن کلید برای استفاده در شاخص‌های غیر خوشه‌ای استفاده می‌شود. **SQL Server** برای واکشی سطر داده‌ای، شاخص خوشه‌ای را با استفاده از کلید شاخص خوشه‌ای که در سطر برگ شاخص غیر خوشه‌ای ذخیره شده است، جستجو می‌کند.

چون شاخص‌های غیر خوشه‌ای کلیدهای شاخص خوشه‌ای را به عنوان مکان نمای سطر در خود ذخیره می‌کنند، بهتر است کلیدهای شاخص خوشه‌ای تا حد امکان کوچک باشند. اگر یک جدول دارای شاخص‌های غیر خوشه‌ای نیز هست، از ستون‌های بزرگ به عنوان کلید در شاخص خوشه‌ای استفاده نکنید.



«شکل ۲۹-۳: ساختار شاخص غیر خوشه‌ای»

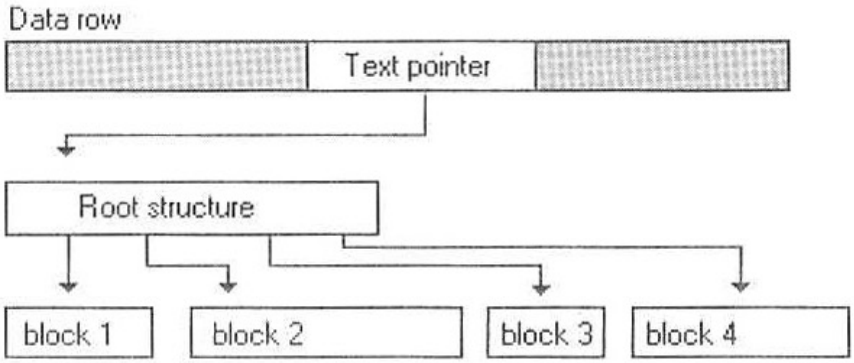
داده‌های text، ntext و image

مقادیر text، ntext و image در سطرهای داده‌ای ذخیره نمی‌شوند، بلکه در مجموعه‌ای از صفحه‌های مجزا ذخیره می‌شوند. برای هر کدام از این مقادیر تنها چیزی که در سطر داده‌ای ذخیره می‌شود یک اشاره‌گر ۱۶ بایتی است. برای هر سطر، این اشاره‌گر به محل این داده‌ها اشاره می‌کند. اگر سطر بیش از یک ستون از نوع text ، ntext و image داشته باشد، برای هر کدام از این ستون‌ها یک اشاره‌گر وجود خواهد داشت.

این نوع داده‌ها در یک مجموعه از صفحه‌های ۸ کیلوبایتی ذخیره می‌شوند. صفحه‌ها به طور منطقی در یک ساختار B-tree ذخیره می‌شوند. برحسب این که مقدار

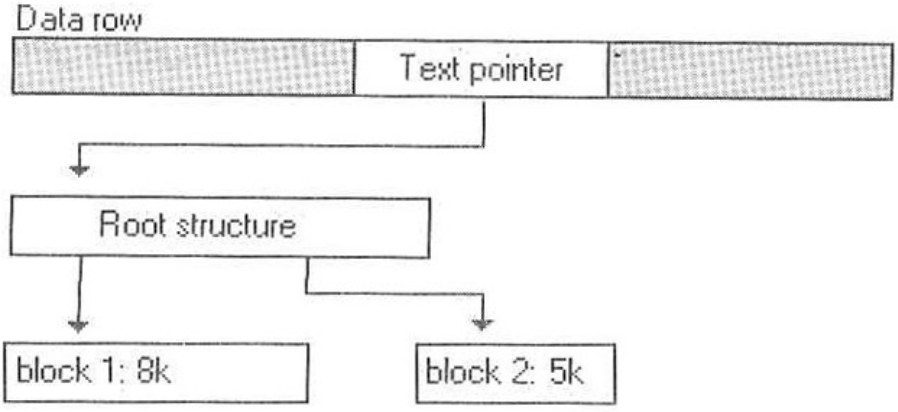
داده کمتر یا بیشتر از ۳۲ کیلو بایت باشد، ساختار **B-tree** قدری متفاوت است.

اگر مقدار داده کمتر از ۳۲ کیلو بایت باشد، اشاره‌گر در سطر داده‌ای به یک ساختار ریشه ۸۴ بایتی اشاره می‌کند که همان گره ریشه ساختار **B-tree** است. گره ریشه به بلاک‌های داده‌ای اشاره می‌کند.



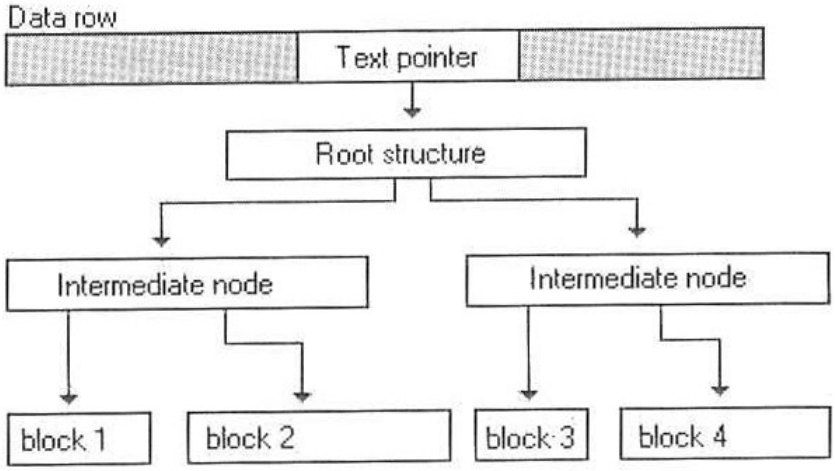
«شکل ۳۰-۳: نحوه ذخیره‌سازی داده‌ها در ساختار B-tree»

اندازهٔ هر بلاک داده‌ای توسط برنامه کاربردی تعیین می‌شود. بلاک‌های داده‌ای کوچک با هم ترکیب شده تا یک صفحه را پر کنند. اگر مقدار داده کمتر از ۶۴ بایت باشد تمام داده‌ها در ساختار ریشه ذخیره می‌شوند. به عنوان مثال، اگر برنامه کاربردی ۱ کیلو بایت داده نوع **image** داشته باشد، به عنوان اولین بلاک ۱ کیلو بایتی برای سطر ذخیره می‌شود. اگر این برنامه مجدداً ۱۲ کیلو بایت داده دیگر را بخواهد ذخیره کند، ۷ کیلو بایت از آن با ۱ کیلو بایت اولی ترکیب شده و بلاک اول پر می‌شود. ۵ کیلو بایت باقیمانده، بلاک دوم را تشکیل خواهد داد.



«شکل ۳۱-۳: نحوه تخصیص بلاک‌ها در ساختار»

اگر مقدار داده در یک ستون نوع **text**، **ntext** و **image** بیشتر از ۳۲ کیلو بایت باشد، **SQL Server** بین بلاک‌های داده‌ای و گره ریشه، گره‌های میانی تولید می‌کند.



«شکل ۳۲-۳ : تأثیر ستون‌های بیش از ۳۲KB در شکل ساختار»

گره‌های میانی در صفحه‌ای مجزا از ساختار ریشه و بلاک‌های داده‌ای ذخیره می‌شوند.

ساختار ثبت تراکنش

هر پایگاه دادهٔ **SQL Server** دارای یک فایل ثبت تراکنش است که تمام تغییرات انجام شده در پایگاه داده توسط تراکنش‌ها را ثبت می‌کند. ثبت تراکنش‌ها و تغییرات آنها سه هدف را تأمین می‌کنند:

- ترمیم انفرادی تراکنش‌ها: اگر یک برنامه کاربردی یک دستور **ROLLBACK** صادر نماید و یا اگر **SQL Server** خطایی را کشف کند (مثلاً فقدان ارتباط با سرویس‌گیرنده)، رکوردهای **log** برای لغو تغییرات انجام شده توسط تراکنش تکمیل نشده مورد استفاده قرار می‌گیرد.
- بازیابی تمام تراکنش‌های تکمیل نشده هنگام شروع به کار **SQL Server**: اگر سرویس‌دهنده‌ای که **SQL Server** را اجرا می‌کند دچار اشکال شود، پایگاه‌های داده ممکن است در حالتی قرار گیرند که بعضی از تغییرات از بافر در فایل‌های داده‌ای نوشته نشده باشند و ممکن است بعضی از تغییرات در فایل‌های داده‌ای ناشی از تراکنش‌های تکمیل نشده باشند. وقتی **SQL Server** اجرا می‌شود شروع به ترمیم تک تک پایگاه‌های داده می‌کند. هر یک از تغییراتی که در فایل **log** ثبت شده‌اند ولی در فایل داده‌ای نوشته نشده‌اند، انجام می‌شوند. سپس تمام تراکنش‌های تکمیل نشده لغو می‌شوند تا جامعیت پایگاه داده حفظ شود.

- بازیابی یک پایگاه داده از نسخه پشتیبان تا نقطه وقوع خطا: وقتی نسخه پشتیبان یک پایگاه داده بازیابی می‌شود، عملیات ترمیم برای لغو تراکنش‌های تکمیل نشده آغاز می‌شود. پس از این که نسخه پشتیبان پایگاه داده بازیابی می‌شود، نسخه‌های پشتیبان ثبت تراکنش، تراکنش‌های تکمیل شده را دوباره اعمال می‌کنند تا پایگاه داده به وضعیت قبل از وقوع خطا بازگردد.

خصوصیات ثبت تراکنش عبارت است از:

- ثبت تراکنش به صورت یک جدول پیاده‌سازی نمی‌شود، بلکه به صورت یک یا مجموعه‌ای از فایل‌ها در پایگاه داده هستند. بافر log جدا از بافر مربوط به صفحه‌های داده‌ای مدیریت می‌شود، در نتیجه کد درون موتور پایگاه داده ساده‌تر، سریع‌تر و قوی‌تر می‌شود.
- قالب رکوردهای log حتی لازم نیست که مانند قالب صفحه‌های داده‌ای باشد.
- ثبت تراکنش می‌تواند روی فایل‌های متعدد پیاده‌سازی شود. فایل‌ها می‌توانند طوری تعریف شوند که در مواقع لازم به طور خودکار رشد کنند. این کار احتمال بالقوه کمبود فضا را کاهش داده و علاوه بر آن سربار مدیریت سیستم را نیز کاهش می‌دهد.
- مکانیزم پاک کردن قسمت‌های استفاده نشده log سریع می‌باشد و تأثیرکمی نیز بر بازدهی تراکنش دارد.

ثبت تراکنش پیشرو

SQL Server همانند بسیاری از پایگاه‌های داده رابطه‌ای از ثبت تراکنش پیشرو استفاده می‌کند. ثبت تراکنش پیشرو تضمین می‌کند که هیچ تغییرات داده‌ای قبل از رکورد log مربوطه روی دیسک نوشته نشود.

SQL Server دارای یک بافر در حافظه می‌باشد که از آن برای خواندن صفحه‌های داده‌ای استفاده می‌کند. تغییرات داده‌ها مستقیماً روی دیسک نوشته نمی‌شوند، بلکه در عوض بر روی کپی صفحه در بافر اعمال می‌شود.

هنگامی که تغییری در صفحه درون بافر اعمال می‌شود، یک رکورد log در بافر

log برای ثبت این تغییر ساخته می‌شود. این رکورد log باید قبل از این که صفحه مربوطه از بافر به فایل منتقل شود، روی دیسک نوشته شود. اگرصفحهٔ مربوطه قبل از رکورد log روی دیسک نوشته شود، در این صورت اگر قبل از این که رکورد log روی دیسک نوشته شود سرویس‌دهنده دچار اشکال شود، تغییری روی دیسک ایجاد شده است که نمی‌توان آن را لغو کرد. چون رکوردهای log پیش از صفحه‌های داده‌ای مربوطه روی دیسک نوشته می‌شوند، ثبت تراکنش پیشرو خوانده می‌شوند.

ساختار منطقی ثبت تراکنش

ثبت تراکنش، رشته‌ای ترتیبی از رکوردهای log است. هر رکورد log توسط یک شماره ترتیبی log (LSN) شناسایی می‌شود. هر رکورد log جدید در انتهای فایل log با یکLSN بزرگتر از LSN رکورد قبلی نوشته می‌شود.

عملیات مختلفی در ثبت تراکنش درج می‌شوند، شامل:

- ابتدا و انتهای تراکنش.
- تغییرات داده‌ای (درج، بهنگام سازی و حذف). این امر شامل تغییراتی که توسط رویه‌های ذخیره شده سیستمی یا دستورات زبان تعریف داده در جدول‌های سیستمی اعمال می‌شود نیز هست.
- تخصیص یا باز پس گرفتن Extentها.
- ایجاد و حذف جدول‌ها و شاخص‌ها.

رکوردهای log به ترتیبی که ایجاد شده‌اند ذخیره می‌شوند. هر رکورد log با شناسه تراکنشی که به آن تعلق دارد شناسایی می‌شود. برای هر تراکنش، رکوردهای log مربوط به آن در یک لیست پیوندی یک طرفه با اشاره‌گرهای عقب گرد قرار دارند که لغو تراکنش را تسریع می‌کنند.

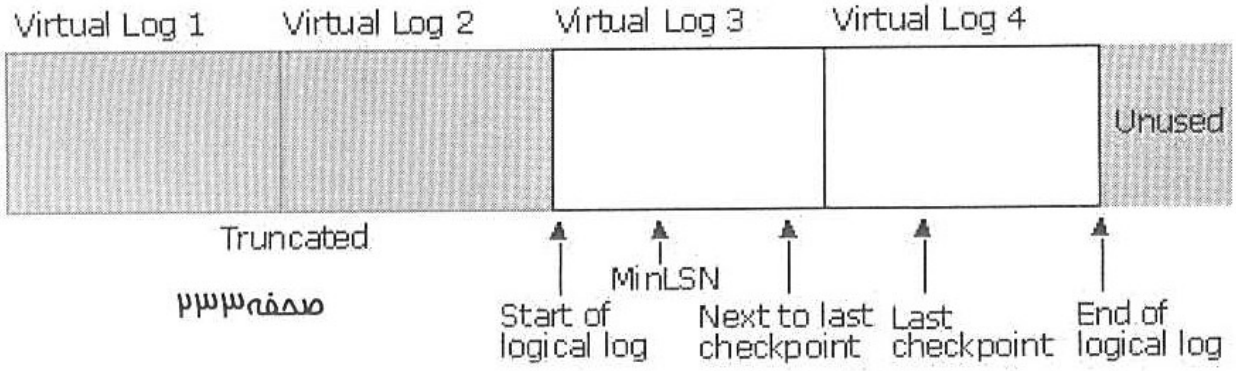
ساختار فیزیکی ثبت تراکنش

در یک پایگاه داده، ثبت تراکنش به یک یا چند فایل فیزیکی نگاشته می‌شود. SQL Server هر فایل log فیزیکی را به چند log مجازی تقسیم می‌کند. فایل‌های log مجازی اندازه ثابتی ندارند و همچنین برای یک فایل log فیزیکی تعداد ثابتی فایل log

مجازی وجود ندارد. SQL Server اندازه فایل‌های log مجازی را به طور پویا هنگام ایجاد این فایل‌ها تعیین می‌کند. SQL Server سعی می‌کند که تعداد این فایل‌های مجازی را کم نماید. تعیین اندازه و تعداد فایل‌های log مجازی خارج از اختیارات مدیر سیستم است و فقط توسط SQL Server تعیین می‌شود.

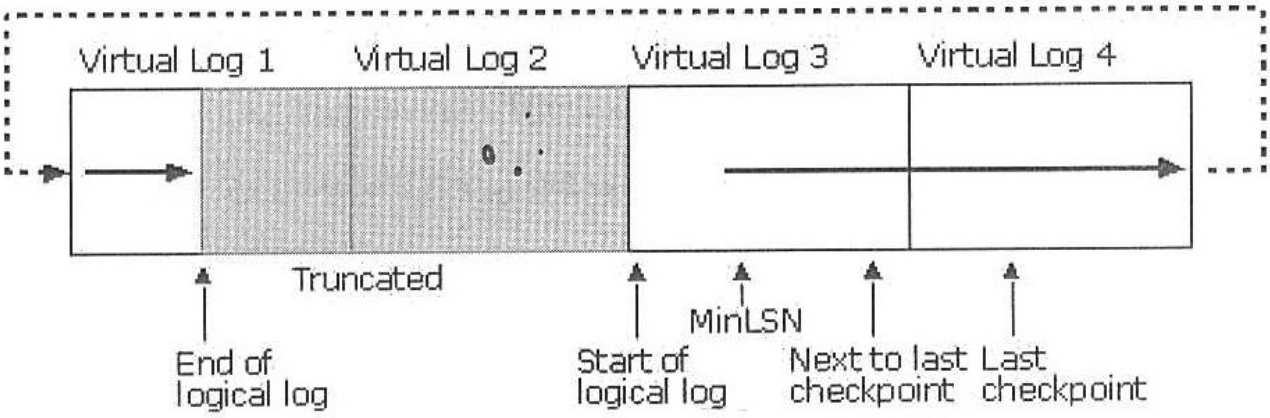
تنها موقعی که فایل‌های log مجازی بر کارآیی تأثیر می‌گذارند زمانی است که این فایل‌ها با اندازه کوچک و میزان رشد کوچک تعریف شوند. اگر این فایل‌های مجازی به مقدار زیاد رشد کنند، فایل‌های log مجازی زیادی تولید خواهد شد که موجب می‌شود عملیات ترمیم کند شود. توصیه می‌شود که فایل‌های log با اندازه‌ای نزدیک اندازه نهایی خود و میزان رشد فایل، بزرگ انتخاب شود.

پایگاه داده‌ای را با یک فایل log فیزیکی در نظر بگیرید که به چهار فایل log مجازی تقسیم شده است. وقتی پایگاه داده ایجاد می‌شود، فایل log منطقی از ابتدای فایل log فیزیکی شروع می‌شود. رکوردهای log جدید به انتهای log منطقی اضافه می‌شوند و به طرف انتهای log فیزیکی رشد می‌کند.



«شکل۳۳-۳: نمونه‌ای از فایل ثبت مجازی»

وقتی انتهای log منطقی به انتهای فایل log فیزیکی می‌رسد، رکوردهای log جدید در ابتدای فایل فیزیکی نوشته می‌شوند.

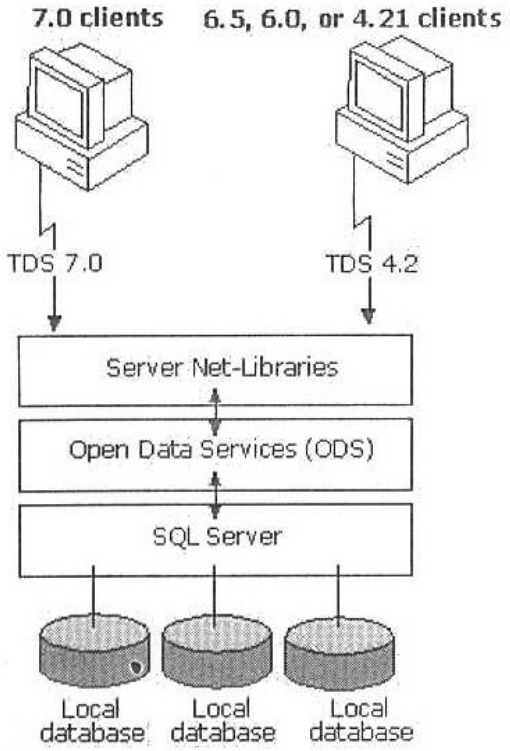


«شکل ۳۴-۳ : خاصیت چرخشی فایل‌های log»

۶-۳- معماری سرویس‌دهنده

سرویس‌دهنده، جزئی از SQL Server است که دستورات SQL را از سرویس‌گیرنده‌ها دریافت کرده و تمام اعمال لازم برای تکمیل این دستورات را انجام می‌دهد.

اجزاء سرویس‌دهنده، دستورات SQL را از سرویس‌گیرنده‌ها دریافت می‌کند و آنها را پردازش کرده و اجرا می‌نماید.



«شکل ۳۵-۳: ساختار سرویس‌دهنده»

دستورات SQL از سرویس‌گیرنده‌ها بوسیله یک پروتکل در سطح برنامه کاربردی که خاص SQL Server است، ارسال می‌شوند. این پروتکل، جریان داده‌های جدولی

(TDS) نامیده می‌شود.

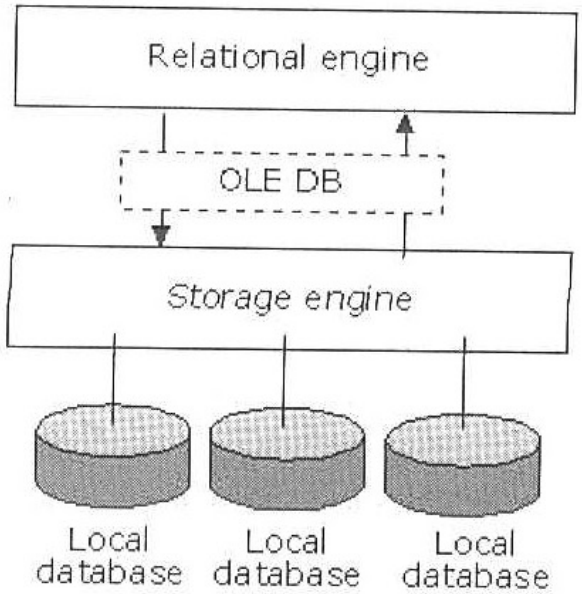
بسته‌های TDS توسط OLE DB، راه‌انداز ODBC یا DB-Library DLL ساخته می‌شود. سپس بسته‌های TDS به کتابخانه شبکه‌ای سرویس گیرنده SQL Server ارسال می‌شود و در آنجا این بسته‌ها به بسته‌های پروتکل شبکه‌ای تبدیل می‌شوند. بر روی سرویس‌دهنده، بسته‌های پروتکل شبکه توسط کتابخانه شبکه‌ای سرویس‌دهنده دریافت می‌شوند و بسته TDS از آن استخراج شده و به سرویس‌های داده باز ارسال می‌شود.

وقتی نتایج به سرویس‌گیرنده باز گردانده می‌شود، عکس این پردازش انجام می‌شود.

سرویس‌های داده باز تمام بسته‌های TDS را که از کتابخانه‌های شبکه‌ای سرویس‌دهنده می‌آیند، مدیریت می‌کند. این سرویس‌ها نوع هر بسته TDS را تعیین می‌کنند و سپس تابع مناسب را در SQL Server فراخوانی می‌کنند. سرویس‌دهنده پایگاه داده تمام درخواست‌های ارسالی از سرویس‌های داده باز را پردازش می‌کند.

سرویس‌دهنده پایگاه داده

سرویس‌دهنده پایگاه داده دارای دو قسمت اصلی می‌باشد: موتور رابطه‌ای (RE) و موتور ذخیره سازی (SE). یکی از مهمترین تغییرات ساختاری که در SQL Server 7.0 بوجود آمده است، جدا کردن اجزاء موتور رابطه‌ای و ذخیره سازی در سرویس‌دهنده است.



«شکل ۳۶-۳: ساختار سرویس‌دهنده پایگاه داده»

پردازش یک دستور SQL که فقط به جدول‌های محلی پایگاه داده ارجاع می‌کند،

به صورت زیر خلاصه می‌شود:

۱- موتور رابطه‌ای، دستور SELECT را به یک طرح اجرای بهینه سازی شده کامپایل می‌کند. طرح اجرا یکسری از عملیات را برای مجموعه‌های سطری حاصل از تک تک جدول‌ها یا شاخص‌هایی که در دستور SELECT ارجاع می‌شوند، تعریف می‌کند.

۲- سپس موتور رابطه‌ای از OLE DB API برای درخواست از موتور ذخیره سازی برای باز کردن مجموعه‌های سطری استفاده می‌کند.

۳- در حین این که موتور رابطه‌ای مراحل مختلف طرح اجرا را اجرا کرده و نیاز به داده پیدا می‌کند، از OLE DB برای واکشی تک تک سطرها از مجموعه‌های سطری استفاده می‌کند. موتور دخیره‌سازی، داده‌ها را از بافرها به موتور رابطه‌ای منتقل می‌کند.

۴- موتور رابطه‌ای داده‌های حاصل از مجموعه‌های سطری موتور ذخیره سازی را با مجموعه نتیجه نهایی ترکیب کرده و برای کاربر باز پس می‌فرستد.

موتور رابطه‌ای

مسئولیت‌های اصلی موتور رابطه‌ای عبارت است از:

- تجزیه دستورات SQL: دستور SQL به واحدهای منطقی مثل کلمه‌های کلیدی، پارامترها، علمگرها و شناسه‌ها شکسته می‌شود.
- بهینه سازی طرح‌های اجرا: راه‌های مختلفی وجود دارد که سرویس‌دهنده می‌تواند از داده‌های جدول‌های منبع، مجموعه نتیجه را تشکیل دهد. بهینه ساز تعیین می‌کند که هر کدام از این راه‌ها چیستند، هزینه‌های آنها را تخمین می‌زند (اساساً بر حسب ورودی/خروجی فایل) و روشی که کمترین هزینه را در بردارد انتخاب می‌کند. سپس این مراحل را با درخت پرس و جو ترکیب کرده تا یک طرح اجرای بهینه شده را تولید کند.
- اجرای عملیات منطقی که در طرح اجرا تعریف شده است.
- پردازش زبان تعریف داده‌ها (DDL) و سایر دستورات.

موتور ذخیره سازی

مسئولیت‌های اصلی موتور ذخیره‌سازی عبارت است از:

- مدیریت فایل‌هایی که پایگاه داده در آن ذخیره شده است و استفاده از فضا در فایل‌ها
- ساختن و خواندن صفحه‌های فیزیکی که برای ذخیره سازی فایل‌ها مورد استفاده قرار می‌گیرد.
- مدیریت بافرهای داده‌ای و تمام ورودی/خروجی فایل‌های فیزیکی.
- کنترل همزمانی. مدیریت تراکنش‌ها و استفاده از قفل‌گذاری برای کنترل همزمانی دستیابی داده‌ها به سطرهای درون پایگاه داده.
- ثبت عملیات و ترمیم.

ساختار پردازنده پرس و جو

دستورات SQL تنها دستوراتی هستند که از برنامه‌های کاربردی به SQL Server ارسال می‌شوند. تمام کارهایی که توسط SQL Server انجام می‌شود نتیجه دریافت، تفسیر و اجرای دستورات SQL است. فرآیندهایی که طی آن دستورات SQL اجرا می‌شوند عبارتند از :

- پردازش یک دستور واحد SQL.
- پردازش دسته‌ای.
- اجرای رویه ذخیره شده و Trigger.
- استفاده دوباره از طرح اجرا.
- پردازش موازی پرس و جو.

پردازش یک دستور واحد SQL

پردازش یک دستور واحد SQL، ساده‌ترین حالت اجرای دستورات توسط SQL Server است. مراحل لازم برای پردازش یک دستور SELECT که فقط به جدول‌های مبنای محلی ارجاع می‌کند، فرآیند اصلی به شمار می‌آیند.

دستور SELECT بیان کنندهٔ مراحل دقیقی که سرویس‌دهنده پایگاه داده باید از آنها برای واکشی داده‌های درخواستی استفاده کند، نیست. این بدان معنی است که سرویس‌دهنده پایگاه داده باید دستور را تجزیه و تحلیل کند تا تعیین نماید که مؤثرترین راه استخراج داده‌ها کدام است. این کار، بهینه سازی دستور SELECT نامیده می‌شود و جزئی از SQL Server که این کار را انجام می‌دهد بهینه‌ساز پرس و جو نامیده می‌شود.

یک طرح اجرای پرس و جو، تعریفی است از:

- ترتیبی که جدول‌های منبع دستیابی می‌شوند. نوعاً ترتیب‌های بسیار مختلفی وجود دارد که سرویس‌دهنده پایگاه داده می‌تواند برای ساختن مجموعه نتیجه، به جدول‌های مبنا دسترسی یابد. به عنوان مثال، اگر دستور SELECT به سه جدول ارجاع نماید، سرویس‌دهنده پایگاه داده می‌تواند ابتدا به TableA دسترسی یابد، از داده‌های TableA برای استخراج سطرهای متناظر از TableB استفاده کند و سپس از داده‌های TableB برای واکشی داده‌ها از TableC استفاده کند و یا سرویس‌دهنده پایگاه داده می‌تواند به جدول‌ها به ترتیب معکوس دسترسی یابد و یا به صورت TableC، TableA و TableB.
- روش مورد استفاده برای استخراج داده از هر یک از جدول‌ها. روش‌های بسیار مختلفی برای دستیابی به داده‌ها در هر یک از جدول‌ها وجود دارد. اگر فقط به سطرهای کمی با مقادیر کلید ویژه نیاز است، سرویس‌دهنده پایگاه داده می‌تواند از یک شاخص استفاده کند. اگر تمام سطرهای جدول مورد نیاز است، سرویس‌دهنده پایگاه داده می‌تواند شاخص‌ها را نادیده گرفته و کل جدول را در نظر بگیرد. اگر تمام سطرهای یک جدول مورد نیاز است ولی شاخصی وجود دارد که ستون‌های کلید آن در عبارت ORDER BY به کار رفته‌اند، می‌توان کل شاخص را در نظر گرفت.

فرآیند انتخاب یک طرح اجرا از میان چند طرح اجرای ممکن، بهینه سازی نامیده می‌شود. بهینه‌ساز پرس و جو یکی از مهمترین اجزاء یک سیستم SQL Server است. در حالی که مقداری سربار توسط بهیه ساز برای تجزیه و تحلیل پرس و جو و انتخاب یک طرح به سیستم تحمیل می‌شود، با این حال این سربار پس از انتخاب کارآترین طرح، جبران می‌شود. به عنوان مثال، دو شرکت ساختمانی را در نظر بگیرید که نقشه یکسانی برای یک ساختمان دارند. اگر یک شرکت در ابتدای کار چند روز را برای طرح‌ریزی

چگونگی ساخت ساختمان صرف کند و شرکت دیگر بدون طراحی بلافاصله شروع به کار نماید، به احتمال قریب به یقین شرکتی که کار خود را ابتدا با طرح ریزی شروع کرده است، زودتر طرح را به اتمام می‌رساند.

بهینه ساز پرس و جو در SQL Server یک بهینه ساز بر اساس هزینه می‌باشد. هر طرح اجرای ممکن دارای یک هزینه برحسب میزان منابع محاسباتی مصرف شده دارد. بهینه ساز باید طرح‌های ممکن را بررسی کرده و یکی را با کمترین هزینه تخمینی انتخاب کند. بعضی از دستورات SELECT پیچیده، هزاران طرح اجرای ممکن دارند. در این حالات، بهینه ساز تمام ترکیب‌های ممکن را تجزیه و تحلیل نمی‌کند، بلکه از الگوریتم‌های پیچیده‌ای برای جستجوی سریع یک طرح اجرا که دارای هزینه معقولی باشد استفاده می‌کند.

بهینه ساز پرس و جو دقیقاً طرح اجرایی را که دارای کمترین هزینه باشد، انتخاب نمی‌کند، بلکه طرحی را انتخاب می‌کند که با توجه به هزینه‌ای معقول، نتایج لازم را سریع‌تر به کاربر بازگرداند. به عنوان مثال، پردازش یک پرس و جو به صورت موازی نسبت به پردازش ترتیبی، منابع بیشتری را مصرف می‌کند اما پرس و جو را سریع‌تر اجرا می‌کند. اگر استفاده از طرح اجرای موازی بر بار کاری سرویس دهنده تأثیر نداشته باشد، SQL Server از این روش استفاده می‌کند.

مراحل اصلی که SQL Server برای پردازش یک دستور SELECT استفاده می‌کند عبارتند از:

۱- دستور SQL تجزیه شده و به واحدهای منطقی مانند کلمات کلیدی، عبارات، عملگرها و شناسه‌ها شکسته می‌شود.

۲- یک درخت پرس و جو تولید می‌شود که مراحل منطقی لازم برای تبدیل داده‌های منبع به قالب مورد نیاز در مجموعه نتیجه را مشخص می‌کند.

۳- بهینه ساز، تمام روش‌های دستیابی به جدول‌ها را بررسی کرده و مراحلی را که در کمترین زمان و با صرف کمترین منابع مجموعه نتیجه را باز می‌گرداند، انتخاب می‌کند. درخت پرس و جو بهنگام سازی می‌شود تا نشان دهندهٔ این مراحل باشد. شکل نهایی درخت پرس و جو را طرح اجرا می‌گویند.

۴- موتور رابطه‌ای شروع به اجرای طرح اجرا می‌کند. در مراحلی که به داده‌های جدول‌های منبع نیاز است، موتور ذخیره‌سازی، آنها را تأمین می‌کند.

۵- موتور رابطه‌ای داده‌های بازگشتی از موتور ذخیره سازی را به قالب تعریف شده برای مجموعه نتیجه پردازش می‌کند و مجموعه نتیجه را به سرویس‌گیرنده باز می‌گرداند.

مراحلی که برای پردازش دستور **SELECT** ذکر شد، به دستورات دیگر **SQL** مانند **UPDATE**، **DELETE** و **INSERT** نیز اعمال می‌شود.

یک دستور **SQL** به یک طرح اجرا تبدیل می‌شود حتی اگر به یک دید ارجاع کند. برای دیدها، طرح‌های اجرا ذخیره نمی‌شوند بلکه کد منبع دید ذخیره می‌شود. وقتی یک دستور **SQL** به یک دید ارجاع می‌کند، بهینه ساز هم کد منبع دستور **SQL** و هم کد منبع دید را تجزیه و تحلیل کرده و آن را به یک طرح اجرای واحد تبدیل می‌کند. طرح اجرای دستور **SQL** و طرح اجرای دید به طور مجزا تولید نمی‌شود.

بهینه‌سازی دسته‌ای

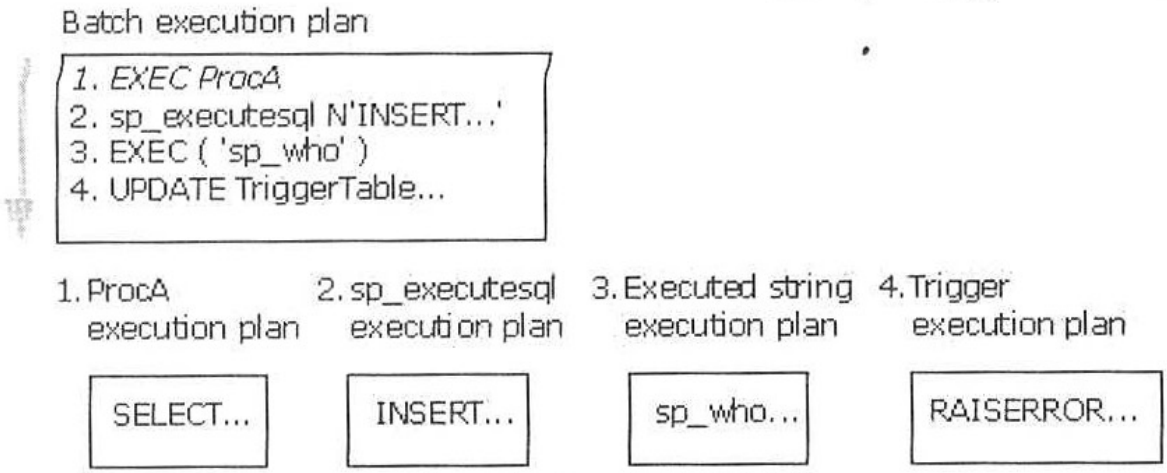
بعضی اوقات سرویس‌گیرنده ممکن است چند دستور **SQL** را به عنوان یک واحد به سرویس‌دهنده ارسال کند. به این مجموعه دستورات یک دسته گویند. هر دسته به یک طرح اجرا تبدیل می‌شود. برای مشخص کردن یک دسته، چند روش وجود دارد:

- تمام دستورات موجود در یک رویه ذخیره شده یا **Trigger** یک دسته تلقی می‌شود.
- رشته‌ای که توسط دستور **EXECUTE** اجرا می‌شود یک دسته است.
- رشته‌ای که توسط رویه ذخیره شده سیستمی **sp_executesql** اجرا می‌شود یک دسته است.

به عنوان مثال، دسته‌ای که شامل چهار دستورالعمل زیر می‌باشد، از پنج طرح اجرا استفاده می‌کند:

- یک دستور **EXECUTE** که یک رویه ذخیره شده را اجرا می‌کند.
- یک فراخوانی **sp_executesql** که یک رشته را اجرا می‌کند.
- یک دستور **EXECUTE** که یک رشته را اجرا می‌کند.

- یک دستور **UPDATE** که به جدولی ارجاع می‌کند که دارای یک **Trigger** از نوع بهنگام‌سازی است.



«شکل ۳۷-۳: طرح اجرای مثال فوق»

اجرای رویه ذخیره شده و Trigger

برای رویه‌های ذخیره شده و **Trigger**، **SQL Server** فقط کد منبع آنها را ذخیره می‌کند. وقتی برای اولین بار رویه ذخیره شده یا **Trigger** اجرا می‌شود این دستورات به طرح اجرا تبدیل می‌شوند. تا وقتی که طرح اجرا در حافظه از بین نرفته باشد، اگر رویه ذخیره شده یا **Trigger** دوباره اجرا شود، موتور رابطه‌ای طرح اجرای موجود را کشف کرده و دوباره مورد استفاده قرار می‌دهد. اگر طرح از حافظه پاک شده باشد، یک طرح جدید ساخته می‌شود.

استفاده دوباره از طرح اجرا

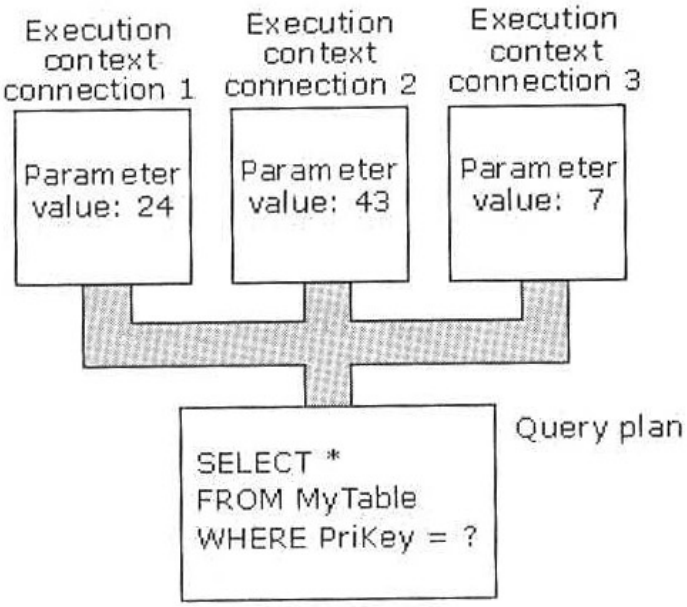
SQL Server دارای حافظه‌ای است که طرح‌های اجرا و بافرهای داده‌ای در آن ذخیره می‌شوند. مقدار حافظه تخصیص داده شده به این امر، بسته به وضعیت سیستم به طور پویا تغییر می‌کند.

طرح اجرا دارای دو جزء اصلی است:

- طرح پرس و جو: ساختار داده‌ای است که می‌تواند توسط چند کاربر مورد

استفاده قرار گیرد. فقط یک یا دو کپی از طرح پرس و جو در حافظه قرار می‌گیرد. یک کپی از آن برای تمام اجراهای ترتیبی و یک کپی از آن برای تمام اجراهای موازی مورد استفاده قرار می‌گیرد.

- زمینه اجرا: هر کاربری که در حال اجرای پرس و جو است دارای ساختمان داده‌ای است که داده‌های مربوط به اجرا مثل مقادیر پارامترها را نگهداری می‌کند. این ساختار داده، زمینه اجرا نامیده می‌شود.



«شکل ۳۸-۳: زمینه اجرای یک طرح اجرا»

وقتی یک دستور SQL اجرا می‌شود، موتور رابطه‌ای ابتدا حافظه را جستجو کرده تا یک طرح اجرا برای دستور بیابد. اگر طرح اجرایی مطابق با دستور SQL موجود باشد آن را مورد استفاده قرار می‌دهد، در نتیجه سربار کامپایلِ دوباره دستور SQL از بین می‌رود. اگر طرح اجرایی موجود نباشد، یک طرح اجرای جدید برای پرس و جو ساخته می‌شود.

استفاده دوباره از طرح‌های اجرای موجود لازمه‌اش این است که تمام شیءهایی که در پرس و جو به آن ارجاع می‌شود به طور کامل مقید باشند. به عنوان مثال، از دو دستور زیر، دستور اول با یک طرح اجرا مطابقت نخواهد کرد.

```
SELECT * FROM Employees
SELECT * FROM Northwind.dbo.Employees
```

پردازش پرس و جوی گیرنده موازی

در کامپیوترهایی که بیش از یک پردازنده دارند، برای بهینه کردن اجرای پرس و جو می‌توان از پرس و جوهای موازی استفاده کرد. اجرای موازی پرس و جو باعث می‌شود پرس و جوهای پیچیده که با مقادیر زیاد داده سروکار دارند با سرعت و به طور کارآ اجرا شوند.

SQL Server درحین بهینه سازی پرس و جو، پرس و جوهایی را که اجرای موازی آنها باعث بهبود کارآیی می‌شود شناسایی می‌کند. برای این پرس و جوها، SQL Server عملگرهای خاصی را (عملگرهای مبادله) در طرح اجرای پرس و جو درج می‌کند تا پرس و جو را برای اجرای موازی آماده نماید. این عملگرها در طرح اجرا، مدیریت فرآیند، توزیع دوباره داده و کنترل جریان را تأمین می‌کنند. پس از درج این عملگرها، نتیجه حاصل یک طرح اجرای پرس و جوی موازی خواهد بود. یک طرح اجرای پرس و جوی موازی می‌تواند بیش از یک ریسمان داشته باشد در صورتی که یک طرح اجرای ترتیبی که بوسیله یک پرس و جوی گیرنده غیر موازی مورد استفاده قرار می‌گیرد، فقط از یک ریسمان برای اجرای خود استفاده می‌کند. تعداد واقعی ریسمان‌های مورد استفاده توسط یک پرس و جوی موازی، در زمان شروع اجرای طرح اجرا تعیین شده و درجه توازی نامیده می‌شود.

SQL Server بهترین درجه توازی برای یک اجرای پرس و جوی موازی را به طور خودکار با در نظر گرفتن عوامل زیر تعیین می‌کند:

- وجود بیش از یک پردازنده در کامپیوتر. فقط کامپیوترهایی که بیش از یک پردازنده دارند می‌توانند از مزایای پرس و جوهای موازی بهره‌مند شوند.
- تعداد کاربرانی که به طور همزمان در حال حاضر فعال هستند. SQL Server بر میزان استفاده از CPU نظارت کرده و درجه توازی را هنگام شروع پرس و جو تعدیل می‌کند.
- وجود حافظه لازم برای اجرای موازی. اجرای یک پرس و جوی موازی نسبت به یک پرس و جوی غیر موازی به حافظه بیشتری نیاز دارد. مقدار حافظه مورد نیاز برای اجرای یک پرس و جوی موازی با افزایش درجه توازی افزایش می‌یابد. اگر مقدار حافظه لازم یک طرح موازی برای یک درجه توازی مشخص تأمین نشود، SQL Server درجه توازی را به طور خودکار کاهش می‌دهد و یا به طور کلی طرح موازی را لغو کرده و طرح

ترتیبی آن را اجرا می‌کند.

- نوع پرس و جوی در حال اجرا. پرس و جوهایی که سیکل‌های CPU را بسیار مصرف می‌کنند، داوطلب خوبی برای یک پرس و جوی موازی هستند، مانند پیوند جدول‌های بزرگ و مرتب کردن مجموعه‌های نتیجه بزرگ.
- وجود سطرهای کافی برای پردازش.

دستورات INSERT، UPDATE و DELETE به طور ترتیبی اجرا می‌شوند. عبارت WHERE در دستور UPDATE یا DELETE و قسمت SELECT در دستور INSERT ممکن است به صورت موازی اجرا شود. سپس تغییرات واقعی داده‌ها به طور ترتیبی در پایگاه داده اعمال می‌شود.

به عنوان مثال، پرس و جوی زیر را در نظر بگیرید:

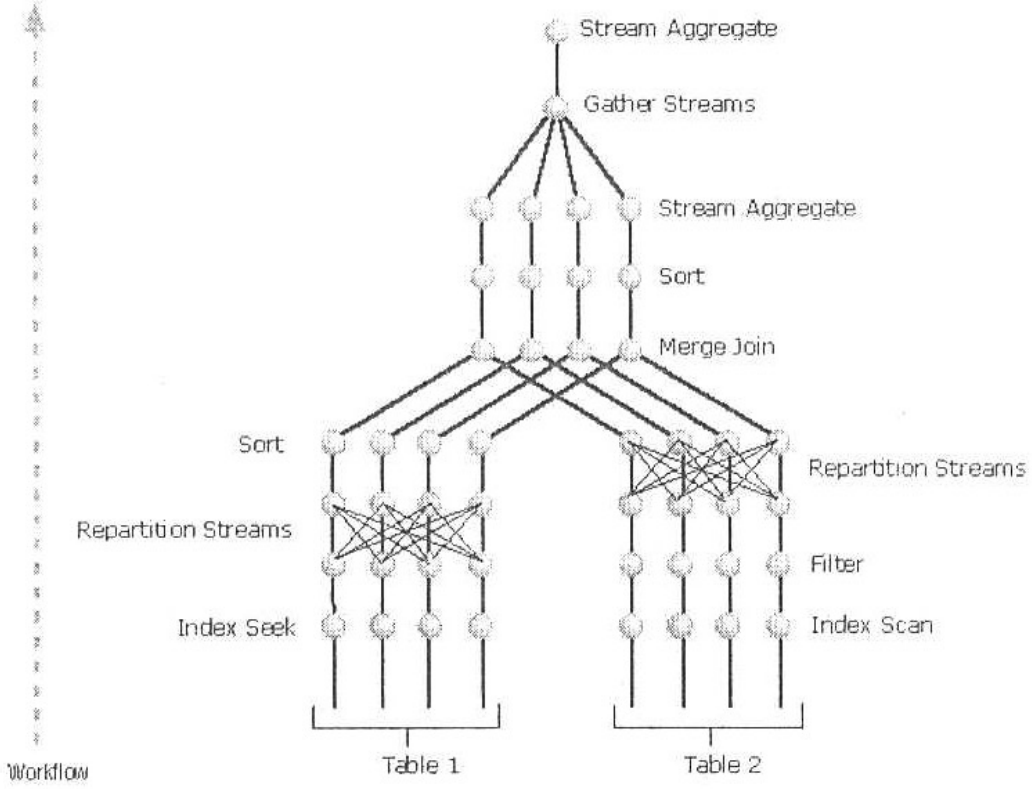
```
SELECT o_orderpriority, COUNT(*) AS Order_Count
FROM orders
WHERE o_orderdate >= '1997/04/01'
AND o_orderdate < DATEADD (mm, 3, '1997/04/01')
AND EXISTS
(
SELECT *
FROM lineitem
WHERE l_orderkey = o_orderkey
AND l_commitdate < l_receiptdate
)
GROUP BY o_orderpriority
ORDER BY o_orderpriority
CREATE INDEX l_order_dates_idx
ON lineitem
(l_orderkey, l_receiptdate, l_commitdate, l_shipdate)
CREATE UNIQUE INDEX o_datkeyopr_idx
ON ORDERS
(o_orderdate, o_orderkey, o_custkey, o_orderpriority)
```

یک طرح موازی ممکن برای این پرس و جو می‌تواند به صورت زیر باشد:

```
-- Specify the logical backup device.
```

```
BACKUP DATABASE accounting
TO Accounting_Backup
-- Or, specify the physical backup device.
BACKUP DATABASE accounting
TO DISK = 'C:\Backups\Accounting\Full.Bak'
```

شکل زیر، طرح اجرای موازی را با درجه توازی ۴ که شامل یک پیوند دو جدولی است نشان می‌دهد:



«شکل ۳۹-۳: شکل اجرای موازی مثال فوق»

ساختار حافظه

SQL Server 7.0 به طور پویا و در مواقع لازم حافظه را بدست آورده و آزاد می‌کند. مدیر سیستم دیگر مجبور نیست مشخص کند چه مقدار حافظه باید برای SQL Server تخصیص داده شود، اگر چه گزینه‌های مربوطه همچنان وجود دارند.

سیستم‌های عامل جدید مانند ویندوز NT دارای حافظه مجازی هستند. حافظه مجازی در واقع فایل‌های روی دیسک می‌باشد که صفحه‌های حافظه را درخود جای می‌دهند. حافظه به واحدهایی به نام صفحه تقسیم می‌شود. صفحه‌هایی که اخیراً به آنها ارجاع شده است در حافظه فیزیکی یا RAM قرار می‌گیرند. اگر یک صفحه برای مدتی ارجاع نشود در فایل فیزیکی یا همان حافظه مجازی نوشته می‌شود. اگر آن قسمت از

حافظه توسط یک برنامه کاربردی مجدداً ارجاع شود، سیستم عامل صفحه حافظه مربوطه را خوانده و در حافظه قرار می‌دهد. میزان حافظه کلی که در دسترس برنامه‌های کاربردی قرار دارد، حافظه فیزیکی در کامپیوتر بعلاوه اندازه حافظه مجازی است.

یکی از اهداف اصلی طراحی نرم افزارهای پایگاه داده، به حداقل رساندن ورودی/خروجی دیسک است زیرا خواندن و نوشتن روی دیسک در کامپیوتر در شمار عملیاتی است که منابع زیادی را مصرف می‌کند. SQL Server دارای بافری در حافظه است که صفحه‌های خوانده شده از پایگاه داده را نگهداری می‌کند. هدف اصلی این است که تعداد خواندن و نوشتن فیزیکی بین دیسک و این بافر به حداقل برسد. هرچه این بافر بزرگ‌تر باشد، میزان ورودی/خروجی کاهش می‌یابد. هر چند اگر این بافر باعث شود که میزان حافظه مورد نیاز SQL Server بیش از حد حافظه فیزیک سرویس‌دهنده شود، سیستم عامل شروع به استفاده از حافظه مجازی می‌کند.

ورودی/خروجی فیزیکی زیاد در فایل‌های پایگاه داده جزء لاینفک نرم افزار پایگاه داده است. به طور پیش فرض، SQL Server سعی می‌کند بین دو هدف زیر، تعادل برقرار کند:

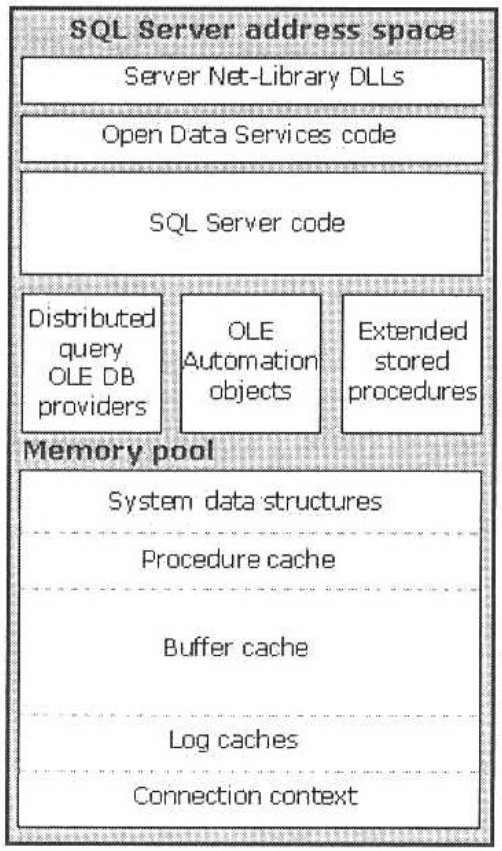
- به حداقل رساندن یا حذف کامل ورودی/خروجی حافظه مجازی تا منابع ورودی/خروجی برای خواندن و نوشتن فایل‌های پایگاه داده متمرکز شود.
- به حداقل رساندن ورودی/خروجی فیزیکی فایل‌های پایگاه داده با به حداکثر رساندن اندازه بافر در حافظه.

تفاوت‌هایی در روش استفاده از حافظه مجازی بین ویندوز NT و ویندوز ۹۵/۹۸ وجود دارد. به این دلیل، SQL Server از الگوریتم‌های مختلفی برای مدیریت حافظه در دو سیستم عامل استفاده می‌کند.

استخر حافظه SQL Server

در ویندوز NT و ویندوز ۹۵/۹۸، میزان کلی حافظه مجازی که در دسترس برنامه‌های کاربردی قرار دارد، مجموعه مجاز آدرس‌های حافظه را برای برنامه کاربردی تشکیل می‌دهد. حافظه مجازی که برای یک برنامه کاربردی اختصاص یافته است فضای آدرس‌دهی نامیده می‌شود.

فضای آدرس‌دهی SQL Server دارای دو جزء اساسی است: کد اجرایی و استخر حافظه.



«شکل ۴۰-۳: فضای آدرس‌دهی SQL Server»

اندازه استخر حافظه بسیار متغیر می‌باشد، بویژه اگر کامپیوتر، برنامه‌های کاربردی دیگری را نیز اجرا نماید. به طور پیش فرض، SQL Server سعی می‌کند که میزان تخصیص حافظه مجازی روی کامپیوتر را ۵ مگابایت کمتر از حافظه فیزیکی نگهدارد. تنها راهی که SQL Server می‌تواند این کار را انجام دهد این است که اندازه فضای آدرس‌دهی متغیر باشد. تنها جزء متغیر در فضای آدرس‌دهی SQL Server که توسط SQL Server کنترل می‌شود، استخر حافظه است. اجزاء متغیر دیگر در فضای آدرس‌دهی مثل اندازه و تعداد رویه‌های ذخیره شده توسعه یافته، بوسیله درخواست برنامه کاربردی کنترل می‌شود.

ساختار ورودی/خروجی

هدف اصلی پایگاه داده، ذخیره‌سازی و واکشی داده است، بنابراین انجام مقدار زیاد خواندن و نوشتن روی دیسک یکی از صفات لاینفک یک سرویس‌دهنده پایگاه داده است. عملیات ورودی/خروجی دیسک منابع زیادی را مصرف می‌کند و نسبتاً زمان زیادی را می‌گیرد.

SQL Server بیشتر حافظه مجازی خود را به بافرها تخصیص می‌دهد و از بافر برای کاهش ورودی/خروجی فیزیکی استفاده می‌کند. داده از فایل‌های پایگاه داده روی دیسک خوانده شده و در بافر قرار می‌گیرد. برای مدتی خواندن داده‌ها به طور منطقی از این بافر انجام می‌شود بدون این که نیاز باشد مجدداً داده به صورت فیزیکی خوانده شود.

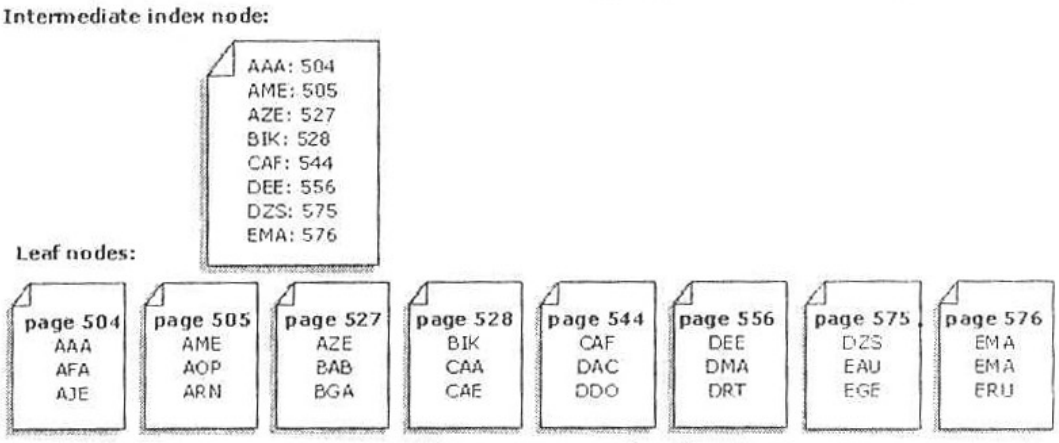
این داده‌ها در بافر باقی می‌مانند تا وقتی که برای مدتی به آنها ارجاع نشود و پایگـــاه داده بافر را برای خواندن داده‌های بیشتر نیاز داشته باشد. اگر داده‌ها تغییر یافته باشند، بـــر روی دیسک نوشته می‌شوند. قبل از این که داده‌های جدید به طـــور فـــیزیکی بـــر روی دیســـک نوشته شوند داده‌ها را چندین بار می‌توان بطور منطقی نوشت.

ورودی/ خروجی SQL Server به ورودی/ خروجی منطقی و فیزیکی تقسیم می‌شـود. خواندن منطقی در مواقعی است که سرویس‌دهنده پایگـــاه داده صفحـــه‌ای را از بـــافر درخواســـت می‌کند. اگر صفحه در بافر وجود نداشته باشد یک خواندن فیزیکی انجام می‌شود تا صفحه از روی دیسک خوانده شده و در بافر قرار گیرد. اگر صفحه در بافر قرار داشته باشد هیچ خواندن فـــیزیکی انجام نمی‌شود. نوشتن منطقی وقتی اتفاق می‌افتد که داده‌های درون یک صفحـــه در حافظـــه تغییـــر یافته باشند. نوشتن فیزیکی وقتی اتفاق می‌افتد که صفحه بر روی دیسک نوشته می‌شود.

درخواست‌های خواندن، که توسط سیستم تولید می‌شوند بوسیلهٔ موتور رابطـــه‌ای کنترل می‌شود و پس از آن توسط موتور ذخیره سازی بهینه می‌شود. روش دستیابی مـــورد استفاده درخواندن صفحه‌ها از یک جدول، الگوی کلی عملیات خواندن را تعیین می‌کنـــد. موتور رابطه‌ای مؤثرترین روش دستیابی را تعیین می‌کند.

در SQL Server 7.0 صفحه‌های IAM، Extentهای مورد استفاده بوســـیله یـــک جدول یا شاخص را مشخص می‌کنند. موتور ذخیره ســـازیIAM را خوانـــده و لیســـتی از آدرس‌های دیسک را که باید خوانده شوند تولید می‌کند.

SQL Server 7.0 صفحه‌های شاخص را نیز به طور ترتیبی می‌خواند کـــه بـــاعث بهبود کارآیی دنبال کردن شاخص‌ها می‌شود. به عنوان مثال، شکل زیر نمایش ســـاده‌ای از مجموعه‌ای از صفحه‌های برگ می‌باشد که شامل مجموعه‌ای از کلیدها است و گره میـــانی شاخص، صفحه‌های برگ را مشخص می‌کنند.



«شکل ۴۱-۳: چگونگی خواندن صفحه‌های شامل کلید»

SQL Server 7.0 از اطلاعات درون صفحه‌های شاخص میانی بالای سطح برگ برای زمان‌بندی ورودی/ خروجی‌ها برای صفحه‌هایی که شامل کلیدها است استفاده می‌کند.

ایجاد و نگهداری پایگاه‌های داده

- اهداف
- ملاحظات مربوط به طراحی پایگاه داده
- ایجاد و مدیریت یک پایگاه داده
- طراحی جدول‌ها
- ایجاد و مدیریت جدول
- دیاگرام‌های پایگاه داده
- ایجاد و مدیریت شاخص
- ایجاد و مدیریت ویو
- ایجاد و مدیریت رویه‌های ذخیره شده
- برنامه‌نویسی رویه‌های ذخیره شده
- ایجاد و مدیریت Triggerها
- برنامه‌نویسی Triggerها

۱-۴- اهداف

یک سیستم پایگاه داد، سرویس‌دهنده/سرویس‌گیرنده از دو جزء تشکیل شده است: برنامه‌هایی که برای دستیابی به داده‌ها، یک واسط کاربر برای کاربران سرویس‌گیرنده فراهم می‌کند و ساختار پایگاه داده که داده‌های سرویس‌دهنده را مدیریت و ذخیره سازی می‌کند. به عنوان مثال، اگر برای ایجاد یک برنامه کاربردی مربوط به حساب پس‌انداز از SQL Server استفاده می‌کنید، باید یک ساختار پایگاه داده برای مدیریت داده‌های تراکنشی حساب و یک برنامه کاربردی که نقش واسط کاربر برای پایگاه داده را ایفا می‌کند، برای دستیابی کاربران به اطلاعات حساب پس‌انداز ایجاد نمایید.

لازمه ایجاد پایگاه داده‌ای که نیازهای تجاری شما را مرتفع سازد این است که نحوه طراحی، ایجاد و نگهداری هر یک از این اجزاء را بدانید تا مطمئن شوید که پایگاه داده شما به بهترین وجه کار می‌کند. در این فصل به بررسی هر یک از این اجزاء پرداخته می‌شود.

۲-۴- ملاحظات مربوط به طراحی پایگاه داده

لازمه طراحی یک پایگاه داده این است که شما عملکردهای تجاری را که می‌خواهید مدل‌سازی کنید و مفاهیم پایگاه داده و ویژگی‌های مورد استفاده در نمایش آن عملکردهای تجاری را درک کرده باشید.

طراحی دقیق پایگاه داده برای مدل‌سازی این عملکرد تجاری از اهمیت خاصی برخوردار است زیرا پس از پیاده سازی، تغییر طراحی پایگاه داده زمان و هزینه قابل ملاحظه‌ای را تلف می‌کند.

هنگام طراحی یک پایگاه داده، توجه به نکات زیر ضروری است:

- هدف از پایگاه داده و تأثیر آن بر روی طراحی: برای تأمین این هدف، طرحی از پایگاه داده ایجاد کنید.
- قواعد نرمال سازی پایگاه داده برای اجتناب از اشتباهات ممکن در طراحی پایگاه داده.
- حفاظت از جامعیت داده‌ها.
- ملزومات ایمنی پایگاه داده و جوازهای کاربران.

- ملزومات کارآیی برنامه کاربردی. شما باید مطمئن باشید در طراحی پایگاه داده از ویژگی‌های سودمند SQL Server برای بهبود کارآیی استفاده شده است. بدست آوردن یک توازن بین اندازه پایگاه داده و پیکربندی سخت افزاری برای کارآیی مهم است.
- نگهداری.
- تخمین اندازه پایگاه داده.

ایجاد یک طرح پایگاه داده

اولین قدم در ایجاد یک پایگاه داده، بوجود آوردن طرحی است که هم در مرحله پیاده‌سازی به عنوان راهنمای شما عمل کند و هم پس از پیاده‌سازی به عنوان یک شاخص عملیاتی برای پایگاه داده باشد. پیچیدگی و جزئیات طراحی یک پایگاه داده بستگی به پیچیدگی و اندازه برنامه کاربردی پایگاه داده و تعداد کاربران آن دارد.

برای ایجاد طرح پایگاه داده، بدون توجه به اندازه و پیچیدگی آن، باید مراحل ابتدایی زیر را دنبال کنید:

- جمع آوری اطلاعات: قبل از ایجاد پایگاه داده، می‌بایست اعمالی را که از پایگاه داده انتظار انجام دادنشان را دارید به خوبی درک کرده باشید. مصاحبه با افرادی که در سیستم وجود دارند برای یافتن این که آنها چه کاری انجام داده و چه *نیازهایی از پایگاه داده دارند مهم است. همچنین شناخت این که* آنها چه انتظاری از سیستم جدید دارند و نیز شناخت مشکلات، محدودیت‌ها و گلوگاه‌های سیستم موجود ضروری است. مجموعه‌ای از نیازهای مشتریان، لیست‌های کالا، گزارش‌های مدیریت و هر مدرکی که بخشی از سیستم موجود است را جمع‌آوری کنید، زیرا برای طراحی پایگاه داده و واسطهای کاربری آن مفید واقع می‌شوند.
- شناسایی شئ‌ها: در حین فرآیند جمع آوری اطلاعات، لازم است شئ‌های کلیدی و موجودیت‌هایی که پایگاه داده باید آنها را مدیریت کند، شناسایی کنید. شئ می‌تواند یک چیز ملموس مانند یک شخص یا یک محصول و یا یک چیز غیر ملموس مانند یک تراکنش تجاری یا یک لیست حقوق نوبه‌ای

باشد. شئ‌های اصلی معمولاً کم هستند و پس از شناسائی آنها، اقلام دیگری که با آنها مرتبط هستند ظاهر می‌شوند. هر قلم مجزا در پایگاه داده شما باید یک جدول متناظر داشته باشد.

- مدل سازی شئ‌ها: پس از شناسائی شئ‌های موجود در سیستم، نحوه ثبت آنها به طریقی که شکل ظاهری سیستم را نمایش دهد، اهمیت دارد. می‌توانید از مدل پایگاه داده به عنوان مرجعی در طی فرآیند پیاده سازی پایگاه داده استفاده کنید.
- شناسایی انواع اطلاعات برای هر یک از شئ‌ها: پس از شناخت شئ‌های اصلی در پایگاه داده به عنوان کاندیدهایی برای جدول‌ها، قدم بعدی، شناخت انواع اطلاعاتی است که باید برای هر یک از شئ‌ها ذخیره شود. این اطلاعات، ستون‌های جدول مربوط به آن شئ خواهد بود.
- شناخت ارتباطات بین شئ‌ها: یکی از قابلیت‌های پایگاه داده رابطه‌ای، قدرت ایجاد ارتباط اطلاعاتی بین شئ‌های مختلف می‌باشد. شناخت ارتباطات بین این شئ‌ها در فرآیند طراحی مستلزم بازنگری جدول‌ها، تعیین ارتباطات منطقی بین آنها و افزودن ستون‌های ارتباطی برای برقراری ارتباط بین جدول‌های مختلف می‌باشد.

نرمال سازی

طراحی منطقی پایگاه داده، هسته اصلی یک پایگاه داده رابطه‌ای بهینه شده است. یک طراحی منطقی خوب می‌تواند باعث ایجاد یک پایگاه داده بهینه گردد در حالی که یک طراحی منطقی ضعیف می‌تواند باعث افت کارآیی کل سیستم شود.

نرمال سازی یک طراحی منطقی پایگاه داده شامل استفاده از روش‌های صوری برای جدا کردن داده‌ها و قرار دادن آنها در چند جدول مربوط به هم است. نرمال سازی معقول اغلب باعث افزایش کارآیی می‌گردد. برخی از مزایای نرمال سازی عبارت است از:

- مرتب سازی و شاخص گذاری سریع‌تر.
- تعداد زیاد شاخص‌های خوشه‌ای.
- شاخص‌های فشرده‌تر.

- تعداد شاخص‌های کمتر به ازاء هر جدول که کارآیی دستورات INSERT، UPDATE و DELETE را افزایش می‌دهد.

با افزایش نرمال سازی، تعداد و پیچیدگی پیوندها برای واکشی داده‌ها افزایش می‌یابد. پیوندهای رابطه‌ای زیاد بین جدول‌ها بر کارآیی تأثیر منفی خواهد داشت.

نرمال سازی دارای قواعد متعددی است که بحث کامل آن خارج از حوزه این کتاب است. برای مطالعه بیشتر می‌توانید به کتاب‌های مربوط به تئوری پایگاه داده مراجعه نمایید.

جامعیت داده‌ها

اعمال کردن جامعیت داده‌ها، کیفیت داده‌های درون پایگاه داده را تضمین می‌کند. مثلاً اگر یک کارمند با شناسه ۱۲۳ وارد شده باشد، پایگاه داده نباید اجازه دهد کارمند دیگری همان شناسه را داشته باشد. و یا اگر حوزه مقادیر یک ستون بین ۱ و ۰ باشد، پایگاه داده نباید مقدار ۶ را برای این ستون بپذیرد. دو گام مهم در طراحی جدول‌ها عبارتند از: تشخیص مقادیر مجاز برای یک ستون و نحوه به کارگیری جامعیت داده برای آن ستون. جامعیت داده به چهار منطقه تقسیم می‌شود:

- جامعیت موجودیتی: جامعیت موجودیتی، یک سطر را به عنوان یک موجودیت یکتا برای یک جدول مشخص می‌کند. جامعیت موجودیتی، جامعیت ستون(های) شناسه یا کلید اصلی یک جدول را اعمال می‌کند (از طریق شاخص‌ها، محدودیت‌های UNIQUE، محدودیت‌های PRIMARY KEY یا خواص IDENTITY).
- جامعیت میدانی: جامعیت میدانی، صحت ورودی‌ها را برای یک ستون خاص تضمین می‌کند. شما می‌توانید جامعیت میدانی را با محدود کردن نوع (از طریق نوع‌های داده‌ای)، قالب (از طریق محدودیت‌های CHECK و قاعده‌ها) یا حوزهٔ مقادیر ممکن (از طریق محدودیت‌های FORE IGN KEY، محدودیت‌های CHECK، تعریف DEFAULT، تعریف NOT NULL و قاعده‌ها) اعمال کنید.
- جامعیت ارجاعی: جامعیت ارجاعی، ارتباطات تعریف شده بین جدول‌ها را هنگام

وارد کردن رکوردها یا حذف آنها حفظ می‌کند. در SQL Server جامعیت ارجاعی بر اساس روابط بین کلیدهای خارجی و کلیدهای اصلی یا روابط بین کلیدهای خارجی و کلیدهای یکتا است. جامعیت ارجاعی تضمین می‌کند که مقادیر کلید در تمام جدول‌ها سازگار باشند. لازمه این سازگاری این است که هیچ ارجاعی به مقادیر غیر موجود وجود نداشته باشد. همچنین اگر یک مقدار کلید تغییر کند تمام ارجاع‌های به آن به طور سازگار در سراسر پایگاه داده تغییر کند.

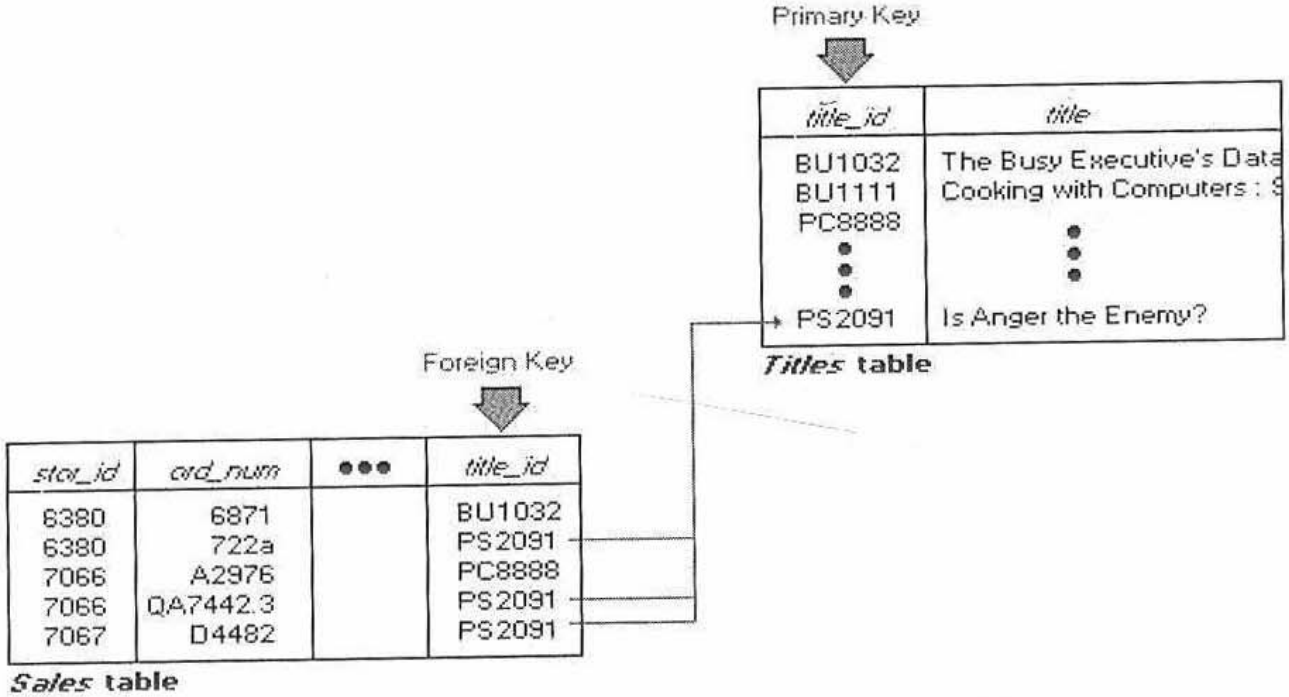
هنگام اعمال جامعیت ارجاعی، SQL Server از انجام موارد زیر توسط کاربران جلوگیری می‌کند:

الف - افزودن رکوردها به یک جدول مرتبط در صورتی که رکورد متناظر در جدول اصلی وجود نداشته باشد.

ب- تغییر مقادیر در جدول اصلی که منجر به ایجاد رکوردهای یتیم (رکوردهایی در جدول مرتبط که دارای رکورد متناظر در جدول اصلی نیستند) در جدول مرتبط شود.

ج- حذف رکوردها از جدول اصلی در صورتی که رکوردهای مرتبط متناظر وجود داشته باشد.

به عنوان مثال در شکل زیر، جامعیت ارجاعی بر اساس رابطه بین کلید خارجی (title_id) در جدول sales و کلید اصلی (title_id) در جدول titles است.



«شکل ۱-۴ : جامعیت ارجاعی بر اساس رابطه دو جدول»

- جامعیت تعریف شده توسط کاربر: این نوع جامعیت اجازه می‌دهد کاربر هر نوع قاعده مشخصی را که در هیچ کدام از گروه‌های جامعیت فوق قرار نمی‌گیرند تعریف کند.

امنیت داده

یکی از وظایف پایگاه داده این است که از داده‌ها در برابر مشاهده یا تغییر داده‌های خیلی حساس توسط کاربران محافظت کند. سیستم امنیتی در SQL Server کنترل می‌کند که کدام یک از کاربران با چه داده‌هایی می‌توانند کار کنند و کدام یک از کاربران چه اعمالی را در پایگاه داده می‌توانند انجام دهند.

کارآیی پایگاه داده

وقتی یک پایگاه داده را طراحی می‌کنید باید مطمئن باشید که پایگاه داده تمام وظایف مهم را به درستی و با سرعت انجام می‌دهد. برخی موارد مربوط به کارآیی پس از این که پایگاه داده در محیط عملیاتی قرار گرفت قابل حل هستند، اما برخی از مسائل مربوط به طراحی ضعیف پایگاه داده بوده و با تغییر ساختار و طراحی پایگاه داده قابل حل هستند.

وقتی یک پایگاه داده را طراحی و پیاده سازی می‌کنید، باید جدول‌های بزرگ و پردازش‌های پیچیده‌تر را تشخیص داده و در هنگام طراحی آنها ملاحظاتی را برای افزایش کارآیی در نظر بگیرید. همچنین تأثیر تعداد کاربران برکارآیی را باید در نظر داشت.

نگهداری

پس از این که یک پایگاه داده ایجاد شد و تمام شئ‌ها و داده‌ها به آن اضافه شده و شروع به کار نمود، مواقعی پیش می‌آید که عملیات نگهداری پایگاه داده باید انجام شود. مثلاً تهیه نسخه پشتیبان به طور مرتب از پایگاه داده ضروری است. همچنین ممکن است برای بهبود کارآیی، چند شاخص جدید ایجاد کنید. هنگام طراحی پایگاه داده این موارد باید مد نظر قرار گیرند تا تأثیر آن بر کاربران و زمان صرف شده برای انجام آن به

حداقل برسد.

تخمین اندازه یک پایگاه داده

هنگام طراحی پایگاه داده، ممکن است نیاز داشته باشید بزرگی پایگاه داده را وقتی که از داده‌ها انباشته می‌شود، تخمین بزنید. تخمین اندازه پایگاه داده به شما کمک می‌کند پیکربندی سخت‌افزاری را که برای هر یک از موارد زیر به آن نیاز خواهید داشت، تخمین بزنید:

- رسیدن به بازده و کارآیی مورد نیاز برنامه کاربردی شما.
- اطمینان از فضای فیزیکی مناسب برای ذخیره سازی داده‌ها و شاخص‌ها.

تخمین اندازه پایگاه داده ممکن است موجب شود شما در طراحی پایگاه داده تجدید نظر نمایید. به عنوان مثال، ممکن است اندازه یک پایگاه داده را بسیار بزرگ تخمین بزنید و لذا نیاز به نرمال سازی بیشتر داشته باشد. برعکس، ممکن است اندازه تخمینی بسیار کمتر از حد انتظار بوده و شما برای بهبود کارآیی پرس و جوها، پایگاه داده را از حالت نرمال خارج کنید.

برای تخمین اندازه یک پایگاه داده، اندازه هر یک از جدول‌ها را تک تک تخمین بزنید و سپس آنها را با هم جمع کنید. اندازه یک جدول بستگی به این دارد که جدول دارای شاخص است یا خیر و از چه نوع شاخصی استفاده می‌کند.

۳-۴- ایجاد و مدیریت یک پایگاه داده

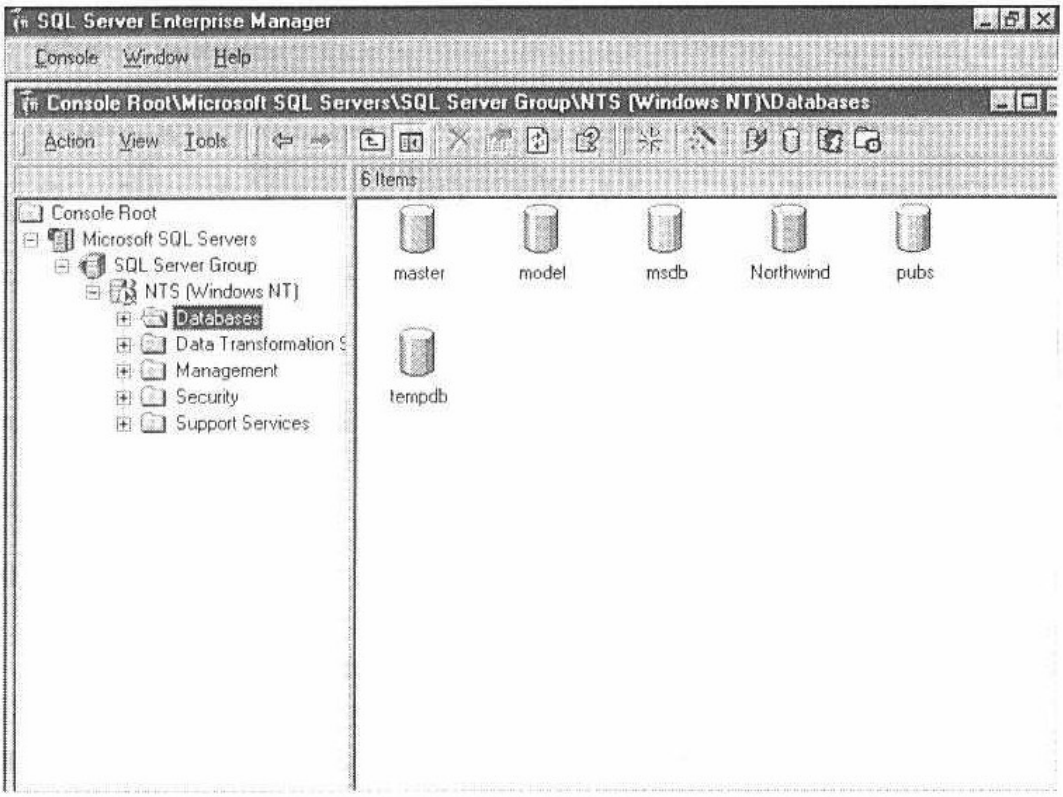
برای ایجاد یک پایگاه داده باید نام پایگاه داده، مالک آن (کاربری که پایگاه داده را ایجاد می‌کند)، اندازه آن و فایل‌ها و گروه‌های فایلی که برای ذخیره آن مورد استفاده قرار می‌گیرند، تعیین شود.

توصیه می‌شود که برای فایل‌ها حداکثر میزان رشد را مشخص کنید. این کار باعث می‌شود که هنگام افزودن سطر، از رشد فایل جلوگیری شود و لذا عمل درج تسریع می‌گردد.

پس از ایجاد یک پایگاه داده، پیشنهاد می‌شود یک نسخه پشتیبان از پایگاه داده master ایجاد شود.

ایجاد یک پایگاه داده

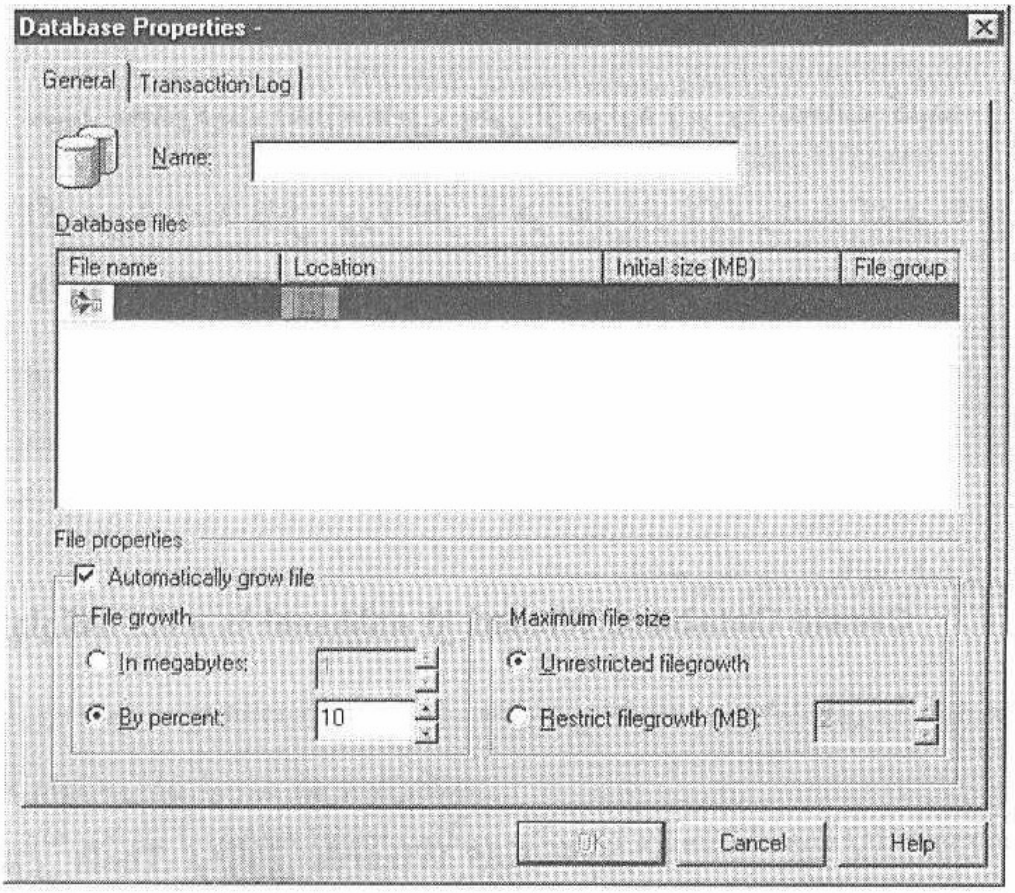
۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.



«شکل ۲-۴ : باز کردن یک سرویس‌دهنده در Enterprise Manager»

۲- روی **Databases** رایت کلیک کنید. سپس روی **New Database...** کلیک کنید.

۳- نام پایگاه داده جدید را وارد کنید.



«شکل ۳-۴: ورود مشخصات پایگاه داده»

نام پایگاه داده به عنوان پیشوند نام فایل‌های اصلی پایگاه داده و ثبت تراکنش در نظر گرفته می‌شود. مثلاً اگر نام پایگاه داده **newdb** باشد، نام فایل اصلی **newdb_Data.mdf** و نام فایل ثبت تراکنش **newdb_Log.ldf** می‌شود. اندازهٔ اولیه پایگاه داده و ثبت تراکنش برابر مقدار پیش فرض برای پایگاه داده **model** است.

۴- برای تغییر مقادیر پیش فرض فایل اصلی پایگاه دادهٔ جدید، دکمهٔ **General** و برای تغییر مقادیر پیش فرض فایل ثبت تراکنش، دکمهٔ **Transaction Log** را انتخاب کنید.

۵- برای تغییر مقادیر پیش فرض ستون‌های **File name**، **Location**، **Initial size (MB)** و **File group** بر روی سلول مورد نظر کلیک کنید و مقدار آن را تغییر دهید.

۶- برای تعیین چگونگی رشد فایل، یکی از موارد زیر را انتخاب کنید:

- برای این که فایل مورد نظر در مواقع لازم رشد کند، **Automatically grow file** را انتخاب کنید.
- اگر می‌خواهید فایل مورد نظر هر بار به میزان ثابتی رشد کند، **In megabytes** را انتخاب و مقداری برای آن مشخص کنید.
- اگر می‌خواهید فایل مورد نظر براساس درصدی از اندازه جاری خود رشد کند، **By percent** را انتخاب و مقداری برای آن مشخص کنید.

۷- برای تعیین محدودیت اندازه فایل، یکی از موارد زیر را انتخاب کنید:

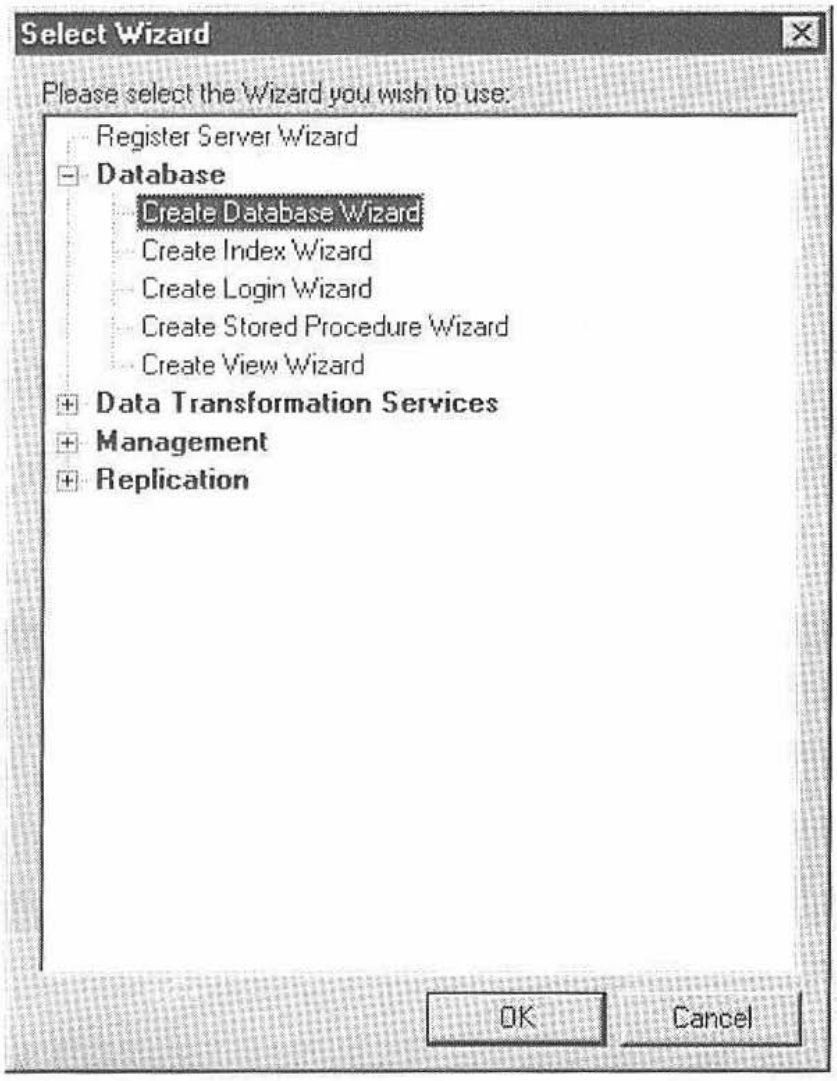
- اگر می‌خواهید فایل مورد نظر به هر مقدار لازم رشد کند، **Unrestricted filegrowth** را انتخاب کنید.
- برای تعیین حداکثر اندازه فایل، **Restricted filegrowth (MB)** را انتخاب و مقداری برای آن مشخص کنید.

ایجاد یک پایگاه داده با استفاده از Create Database Wizard

۱- یک گروه سرویس‌دهنده را باز کنید. سپس سرویس‌دهنده‌ای را که می‌خواهید پایگاه داده در آن ایجاد شود باز کنید.

۲- در منوی **Tools**، بر روی **Wizards** کلیک کنید.

۳- در پنجره‌ای که ظاهر می‌شود، **Database** را باز کنید.



«شکل ۴-۴ : ایجاد پایگاه داده در Create Database Wizard»

۴- بر روی **Create Database Wizard** دابل کلیک کنید.

۵- مراحل مختلف آن را دنبال کنید.

تغییر یک پایگاه داده

پس از ایجاد یک پایگاه داده، می‌توان در تعریف اولیه آن تغییراتی اعمال کرد. این تغییرات شامل موارد زیر می‌باشد.

توسعه پایگاه داده

SQL Server به طور خودکار یک پایگاه داده را مطابق با پارامترهای رشد که هنگام ایجاد پایگاه داده مشخص شده است توسعه می‌دهد. همچنین می‌توانید به طور

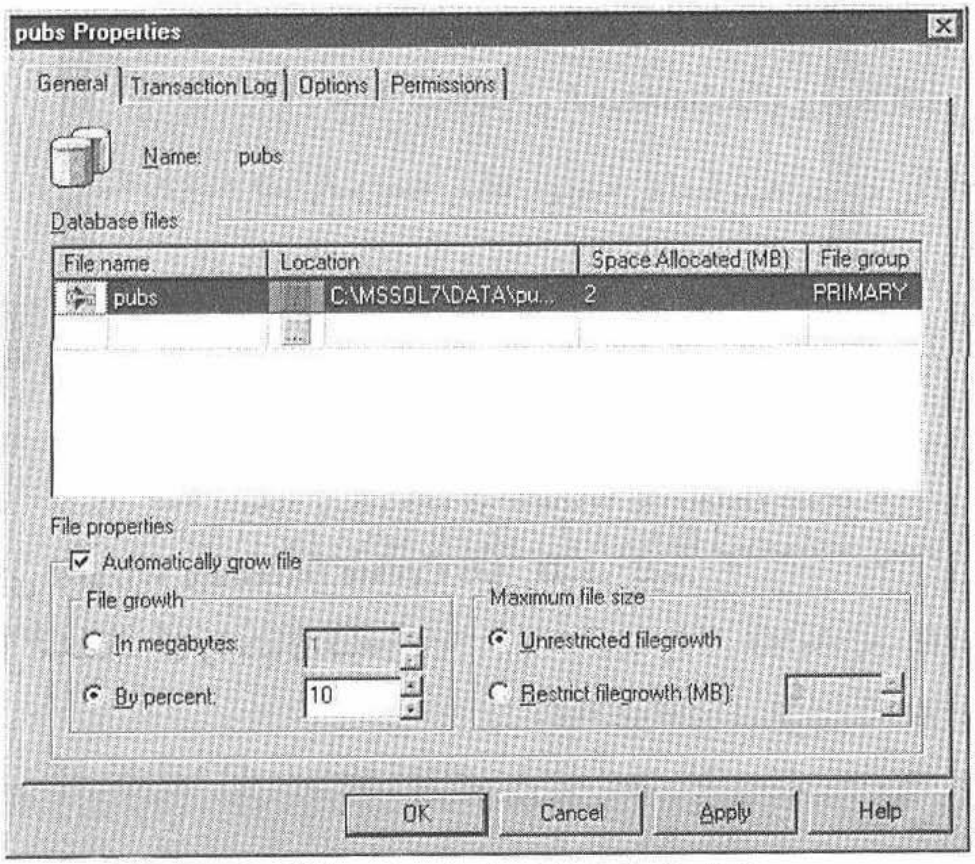
دستی فضای بیشتری به یکی از فایل‌های پایگاه داده اختصاص دهید. هنگام توسعه پایگاه داده، باید اندازه آن را حداقل به مقدار یک مگا بایت افزایش دهید.

افزایش اندازه یک پایگاه داده

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- **Databases** را باز کنید. بر روی پایگاه داده‌ای که می‌خواهید اندازه آن را افزایش دهید، رایت کلیک کنید. سپس روی **properties** کلیک کنید.

۳- برای افزایش فضای داده‌ای، دکمهٔ **General** را انتخاب کنید. برای افزایش فضای ثبت تراکنش، دکمه **Transaction Log** را انتخاب کنید.



«شکل ۵-۴: افزایش اندازه یک پایگاه داده»

۴- برای افزودن یک فایل جدید بر روی سطر خالی بعدی کلیک کنید و در ستون **File name** نام فایلی را که شامل فضای اضافی است وارد کنید.

مکان فایل به طور خودکار تولید می‌شود و پسوند آن برحسب این که فایل پایگاه داده یا فایل ثبت تراکنش است به ترتیب **ndf.** یا **ldf.** می‌شود.

۵- برای تغییر مقادیر پیش فرض ستون‌های **File name**، **Location**، **Space allocation (MB)** و **Filegroup**، بر روی سلول مورد نظر کلیک کنید و مقدار آن

را تغییر دهید.

برای فایل‌های موجود فقط مقدار **Space allocated (MB)** می‌تواند تغییر کند. مقدار جدید باید بزرگتر از مقدار موجود باشد.

۶- برای تعیین چگونگی رشد فایل، یکی از موارد زیر را انتخاب کنید:

- برای اینکه فایل مورد نظر در مواقع لازم رشد کند، **Automatically grow file** را انتخاب کنید.
- اگر می‌خواهید فایل مورد نظر هر بار به میزان ثابتی رشد کند، **In megabytes** را انتخاب و مقداری برای آن مشخص کنید.
- اگر می‌خواهید فایل مورد نظر بر اساس درصدی از اندازه جاری خود رشد کند، **By percent** را انتخاب و مقداری برای آن مشخص کنید.
- اگر می‌خواهید فایل مورد نظر به هر مقدار لازم رشد کند، **Unrestricted filegrowth** را انتخاب کنید.
- برای تعیین حداکثر اندازه فایل، **Restricted filegrowth(MB)** را انتخاب و مقداری برای آن مشخص کنید.

انقباض پایگاه داده

SQL Server به منظور حذف صفحه‌های بدون استفاده، به فایل‌های درون پایگاه داده اجازه می‌دهد کوچک شوند. فایل‌های پایگاه داده را به طور دستی می‌توان منقبض کرد. همچنین می‌توان پایگاه داده را طوری تنظیم کرد که به طور خودکار در فواصل زمانی معین منقبض شود.

اگر پایگاه داده طوری تنظیم شده باشد که به طور خودکار منقبض شود، عمل انقباض وقتی انجام می‌شود که فضای آزاد به مقدار زیاد در پایگاه داده موجود باشد. با این حال نمی‌توان درصد فضای آزاد را که باید حذف شود مشخص نمود.

پایگاه داده را نمی‌توان طوری منقبض کرد که از اندازهٔ اولیه‌اش کوچکتر شود. بنابراین اگر پایگاه داده‌ای با اندازه اولیهٔ ۱۰MB ایجاد شده باشد و تا ۱۰۰MB رشد کند، کمترین حجم پایگاه داده پس از انقباض با فرض این که تمام داده‌ها در پایگاه داده حذف شود، برابر ۱۰MB است. لیکن می‌توانید یک فایل منفرد در پایگاه داده را با استفاده از

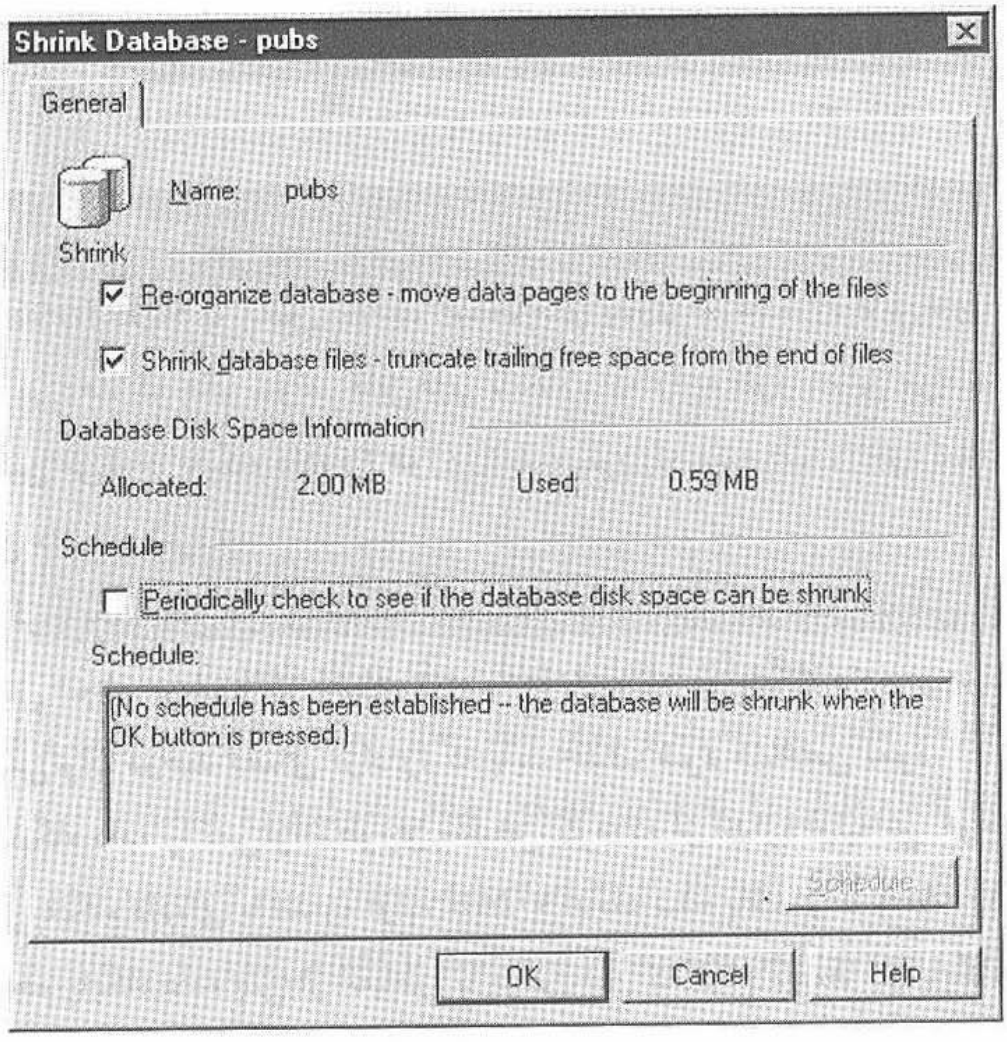
دستور DBCC SHRINKFILE به کمتر از اندازه اولیه‌اش منقبض کنید.

انقباض یک فایل ثبت تراکنش منجر به انقباض سریع آن نمی‌شود بلکه باعث می‌شود فایل برای انقباض بعدی علامتگذاری شود. هر بار که ثبت تراکنش پاک شده و یا از آن نسخه پشتیبان تهیه می‌شود، SQL Server سعی می‌کند تا حد امکان فایل ثبت تراکنش را منقبض کند تا این که به اندازه مطلوب که توسط کاربر تعیین شده است برسد. اگر بخش فعال ثبت تراکنش در انتهای فایل ثبت تراکنش باشد، فایل نمی‌تواند منقبض شود، اما به محض این که بخش فعال به ابتدای فایل منتقل شود، فایل ثبت تراکنش می‌تواند منقبض شود.

انقباض یک پایگاه داده

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- Databases را باز کنید. روی پایگاه داده‌ای که می‌خواهید منقبض شود، رایت کلیک کنید. سپس All Task و به دنبال آن ... Shink Database را انتخاب کنید.

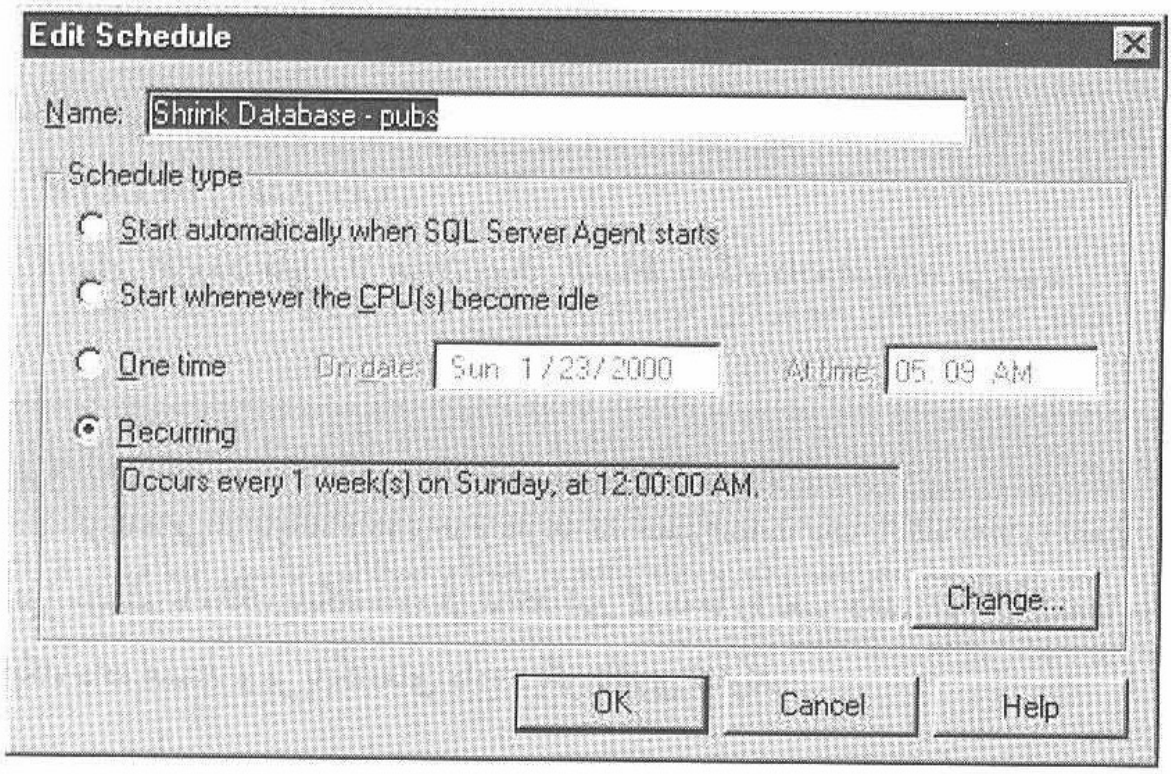


«شکل ۶-۴: انقباض یک پایگاه داده»

۳- برای این که مشخص کنید پایگاه داده چقدر منقبض شود، یکی از گزینه‌های زیر را انتخاب کنید:

- برای این که درصد مشخصی از اندازه جاری پایگاه داده منقبض شود، **Shrink database By(%)** را انتخاب و مقداری برای آن مشخص کنید.
- برای این که پایگاه داده تا حد ممکن منقبض شود، **Remove as much Free space as possible** را انتخاب کنید.
- می‌توانید به دلخواه **peroidically check to see if the database disk space can be shrunk** را انتخاب کنید تا پایگاه داده به طور خودکار و نوبه‌ای منقبض شود.

۴- برای این که تعیین کنید چه زمانی یا چند وقت به چند وقت پایگاه داده به طور خودکار منقبض شود، روی **... Schedule** کلیک کنید.



«شکل ۷-۴: زمانبندی انقباض پایگاه داده»

افزوده و حذف کردن فایل‌های داده‌ای و ثبت تراکنش

وقتی فایلی به پایگاه داده افزوده می‌شود، آن فایل بلافاصله جهت استفاده توسط پایگاه داده در دسترس قرار می‌گیرد. SQL Server فایل‌های داده‌ای را به طور یکسان پر می‌کند، اما فایل‌های ثبت تراکنش در گروه فایلی قرار نمی‌گیرند و از یکدیگر مجزا

می‌باشند. لذا وقتی ثبت تراکنش رشد می‌کند ابتدا اولین فایل log سپس دومین فایل و به همین ترتیب بقیه فایل‌ها پر می‌شوند. بنابراین وقتی یک فایل log اضافه می‌شود ممکن است بوسیله ثبت تراکنش استفاده نشود مگر این که بقیه فایل‌ها قبلاً پر شده باشند.

اگر بخواهید یک فایل را از پایگاه داده حذف کنید، آن فایل باید قبلاً از داده یا اطلاعات ثبت تراکنش خالی شده باشد.

حذف فایل‌های داده‌ای یا log از پایگاه داده

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- Databases را بازکنید. روی پایگاه داده‌ای که می‌خواهید از آن فایلی را حذف کنید، رایت کلیک کنید. سپس properties را انتخاب کنید.

۳- برای حذف فایل‌های داده‌ای دکمه General و برای حذف فایل‌های log، دکمه Transaction log را انتخاب کنید.

۴- در ستون Filename، بر روی پیکان کنار نام فایل کلیک کنید و سپس کلید DELETE را فشار دهید.

برای افزودن فایل، از همان روش انبساط پایگاه داده استفاده می‌شود.

تنظیم گزینه‌های پایگاه داده

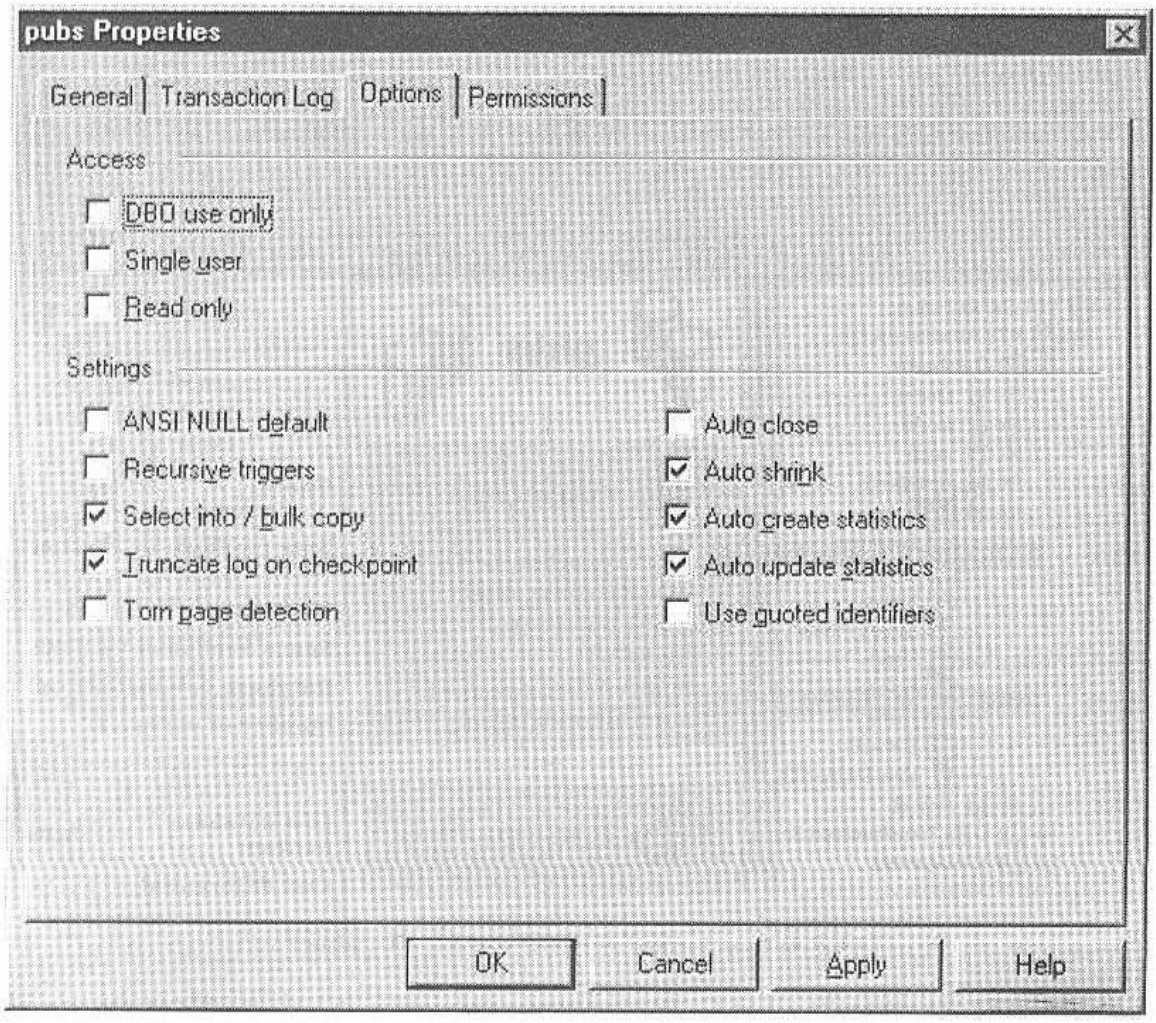
گزینه‌هایی از پایگاه داده وجود دارند که خصوصیات یک پایگاه داده را تنظیم می‌کند. فقط مدیر سیستم یا مالک پایگاه داده می‌توانند این گزینه‌ها را تغییر دهند. این گزینه‌ها مختص یک پایگاه داده هستند و بر پایگاه‌های داده دیگر تأثیری ندارند.

تغییر تنظیمات پیکربندی برای یک پایگاه داده

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- Databases را باز کنید، روی پایگاه داده‌ای که می‌خواهید تغییر دهید، رایت کلیک کنید و سپس properties را انتخاب کنید.

۳- دکمه options و سپس گزینه‌های دلخواه را انتخاب کنید.



«شکل ۸-۴: تعیین گزینه‌های مختلف پایگاه داده»

برای آشنایی با هر یک از این گزینه‌ها به مراجع SQL Server مراجعه کنید.

مشاهده یک پایگاه داده

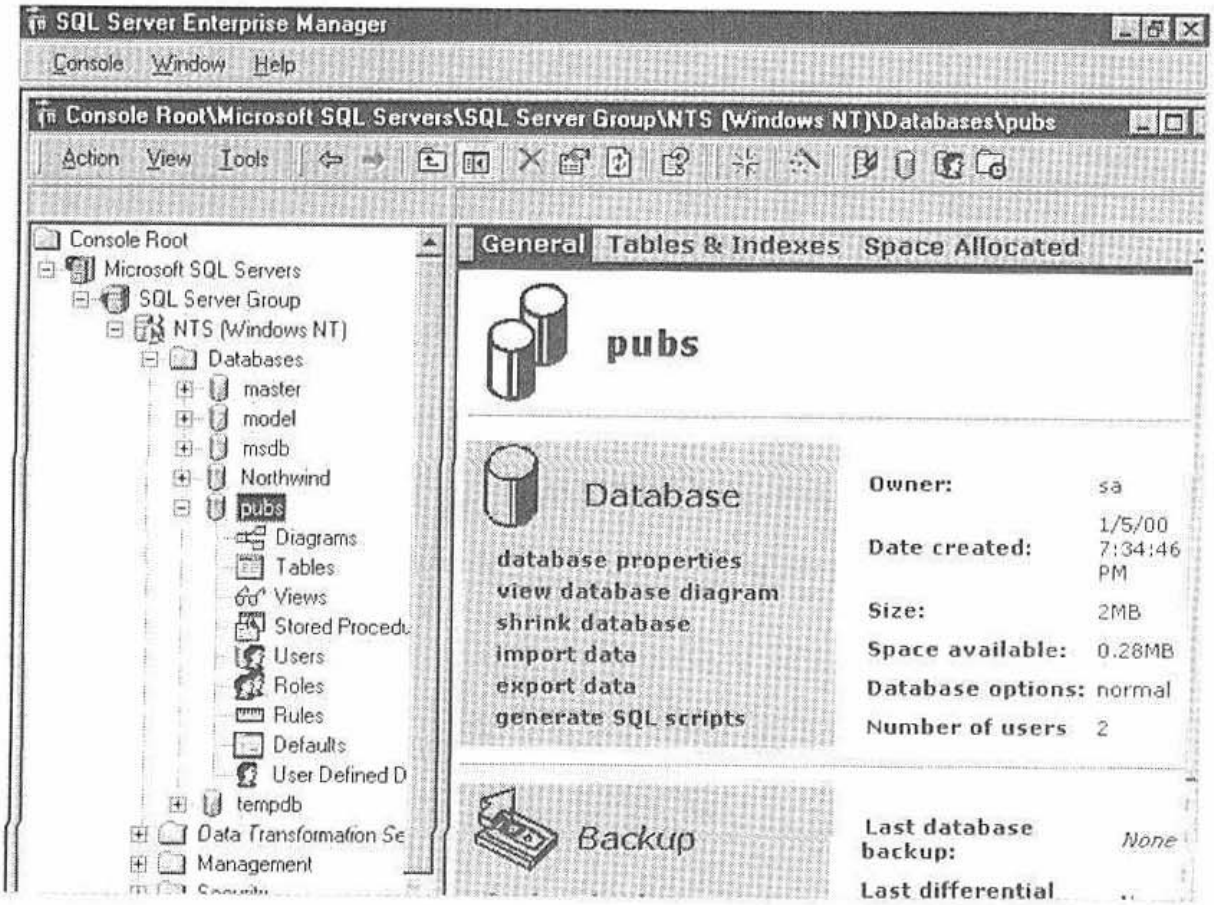
شما می‌توانید تعریف یک پایگاه داده را هنگام اشکال زدایی یا قبل از انجام تغییرات در پایگاه داده، مشاهده کنید.

مشاهده یک پایگاه داده

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- **Databases** را باز کنید، روی پایگاه داده‌ای که می‌خواهید مشاهده کنید، کلیک کنید.

۳- روی **General**، **Tables and Indexes** یا **Space Allocated** کلیک کنید تا اطلاعات بیشتری در مورد پایگاه داده مشاهده کنید.



«شکل ۹-۴: مشاهده اطلاعات پایگاه داده»

نمایش فضای پایگاه داده و ثبت تراکنش

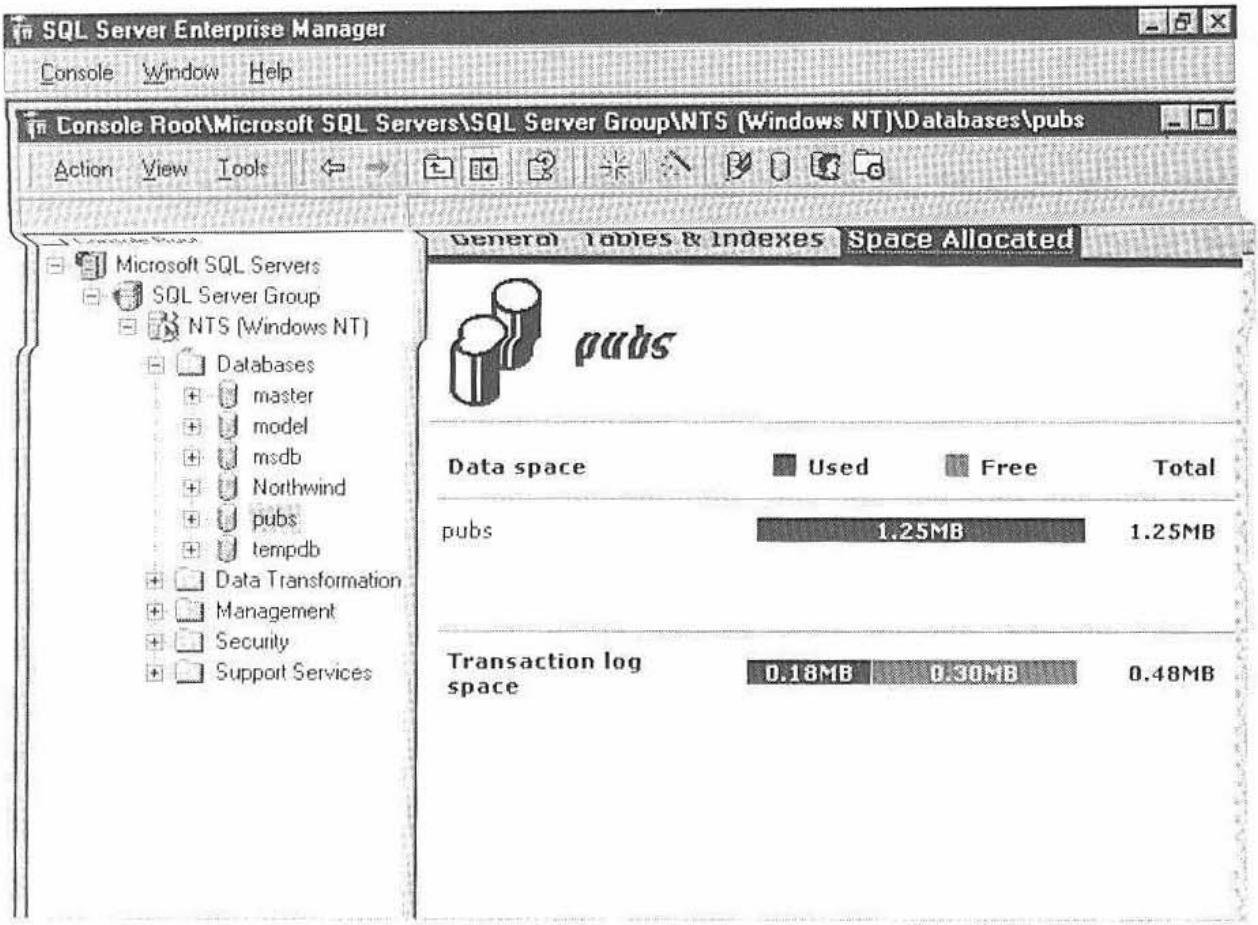
SQL Server می‌تواند تعداد سطرها، فضای رزرو شده دیسک و فضای مورد استفاده بوسیله یک جدول پایگاه داده را نمایش دهد.

نمایش اطلاعات مربوط به فضای داده‌ای و Log برای یک پایگاه داده

۱- یک گروه سرویس دهنده را باز کنید. یک سرویس‌دهنده را باز کنید.

۲- Databases را باز کنید، روی پایگاه داده‌ای که می‌خواهید مشاهده کنید، کلیک کنید.

۳- در پانل details، روی Space Allocated کلیک کنید تا اطلاعات فضای پایگاه داده را مشاهده کنید.



«شکل ۱۰-۴: مشاهده اطلاعات مربوط به فضای پایگاه داده»

حذف یک پایگاه داده

وقتی یک پایگاه داده حذف می‌شود، فایل‌ها و داده‌های آن از دیسک مربوط به سرویس‌دهنده حذف می‌گردد و دیگر آن پایگاه داده قابل بازیابی نخواهد بود.

حذف یک پایگاه داده

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- **Databases** را باز کنید. روی پایگاه داده‌ای که می‌خواهید حذف کنید، رایت کلیک کنید و سپس **Delete** را انتخاب کنید.

۳- عمل حذف را تأیید کنید.

۴-۴- طراحی جدول ها

هنگامی که یک پایگاه داده را طراحی می‌کنید، در مورد این که به چه

جدول‌هایی نیاز دارید، چه نوع داده‌هایی در هر جدول قرار می‌گیرد، چه کسانی به هر جدول می‌توانند دستیابی داشته باشند و غیره تصمیم‌گیری خواهید کرد.

مؤثرترین روش برای ایجاد یک جدول این است که هر چه را که نیاز دارید در یک لحظه در جدول تعریف کنید که این امر شامل محدویت‌های داده‌ای و اجزاء اضافی دیگر می‌باشد. البته شما می‌توانید یک جدول اولیه نیز ایجاد کنید، داده‌هایی به آن اضافه کنید و برای مدتی با آن کار کنید. از این راه خواهید فهمید که چه نوع تراکنش‌هایی بیشتر مورد استفاده قرار می‌گیرند و چه نوع داده‌هایی اغلب وارد می‌شوند و لذا قبل از طراحی نهایی خود، محدودیت‌ها، شاخص‌ها، پیش فرض‌ها، قاعده‌ها و شئ‌های دیگر را به درستی مشخص خواهید کرد.

قبل از ایجاد جدول بهتر است در موارد زیر تصمیم‌گیری نمایید:

- نوع‌های داده‌ای که در جدول قرار خواهند گرفت.
- ستون‌های جدول و نوع داده‌ای آنها.
- چه ستون‌هایی داده‌های **Null** قبول می‌کنند.
- چرا و در کجا از محدویت‌ها یا پیش فرض‌ها و قاعده‌ها استفاده شود.
- انواع شاخص‌های مورد نیاز، محل مورد نیاز آنها، ستون‌های مورد استفاده در کلیدهای اصلی و کلیدهای خارجی.

اولین قدم در طراحی یک جدول، انتساب نوع داده‌ای به هر ستون است. نوع‌های داده‌ای مقدار داده مجاز هر ستون را تعریف می‌کنند. برای انتساب یک نوع داده‌ای به یک ستون باید از نوع‌های داده‌ای SQL Server و یا نوع‌های داده‌ای تعریف شده توسط کاربر بر اساس این نوع‌های داده‌ای، استفاده کنید. نوع‌های داده‌ای برای اعمال جامعیت داده‌ها به کار می‌روند.

نوع‌های داده‌ای تعریف شده توسط کاربر بر مبنای نوع‌های داده‌ای سیستمی SQL Server ایجاد می‌شوند. هنگامی که چند جدول باید نوع یکسانی از داده‌ها را در یک ستون ذخیره کنند و باید اطمینان حاصل کرد که این ستون‌ها دقیقاً دارای نوع داده‌ای، طول و NULL پذیری هستند، می‌توان از نوع داده‌ای تعریف شده توسط کاربر استفاده کرد.

هنگام ایجاد نوع داده‌ای تعریف شده توسط کاربر، سه پارامتر زیر باید مشخص شوند:

- نام.
- نوع داده‌ای سیستمی که نوع داده‌ای جدید بر مبنای آن می‌باشد.
- NULL پذیری.

اگر یک نوع داده‌ای تعریف شده توسط کاربر در پایگاه داده **model** ایجاد شود، در تمام پایگاه‌های داده‌ای که کاربر تعریف می‌کند وجود خواهد داشت. اگر نوع داده‌ای در یک پایگاه داده تعریف شده توسط کاربر ایجاد شود، نوع داده‌ای فقط در آن پایگاه داده وجود خواهد داشت.

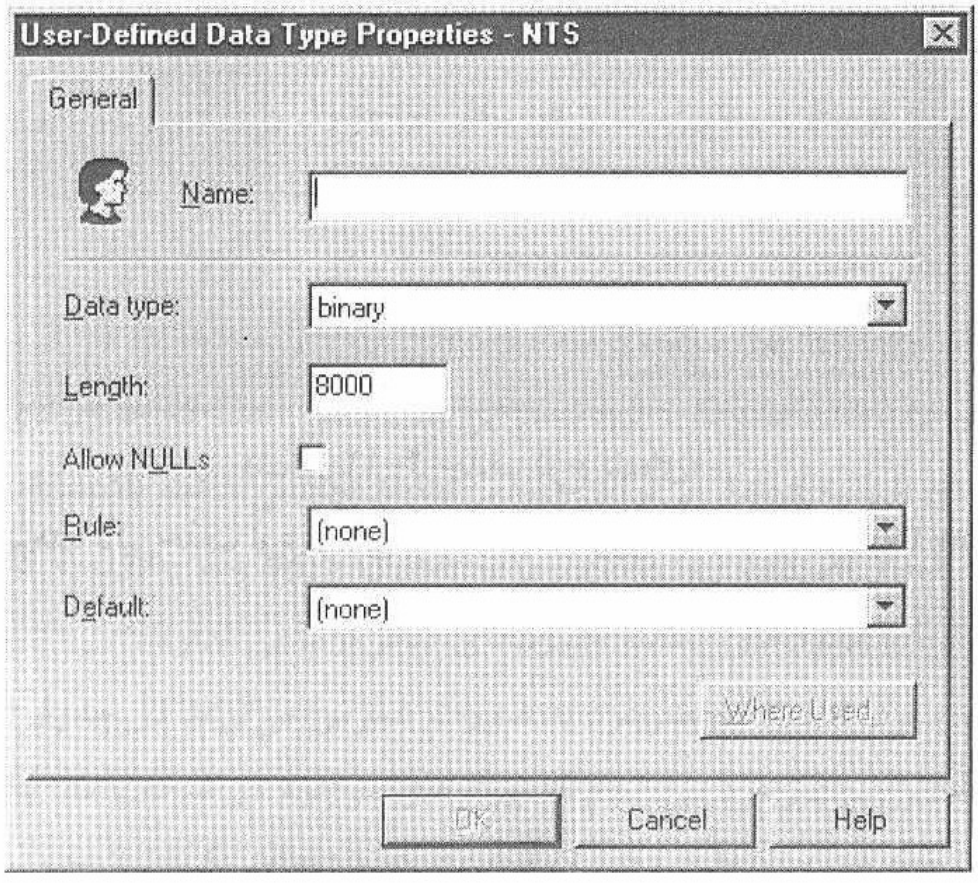
ایجاد نوع داده‌ای تعریف شده توسط کاربر

۱- یک گروه سرویس‌دهنده را باز کنید. یک سرویس‌دهنده را باز کنید.

۲- **Databases** را باز کنید. سپس پایگاه داده‌ای را که می‌خواهید در آن نوع داده‌ای را تعریف کنید، باز کنید.

۳- روی **User Defined Data Types** راست کلیک کنید. سپس روی **New User Defined Data Type ...** کلیک کنید.

۴- نام نوع داده‌ای جدید را وارد کنید.



«شکل ۱۱-۴: تعریف نوع داده‌ای جدید»

۵- در لیست **Data type**، نوع داده‌ای مبنا را انتخاب کنید.

۶- اگر **Length** فعال بود، مقداری به عنوان حداکثر طول برای آن مشخص کنید.

۷- اگر می‌خواهید نوع داده‌ای مقدار **NULL** را نیز قبول کند، **Allow NULLS** را انتخاب کنید.

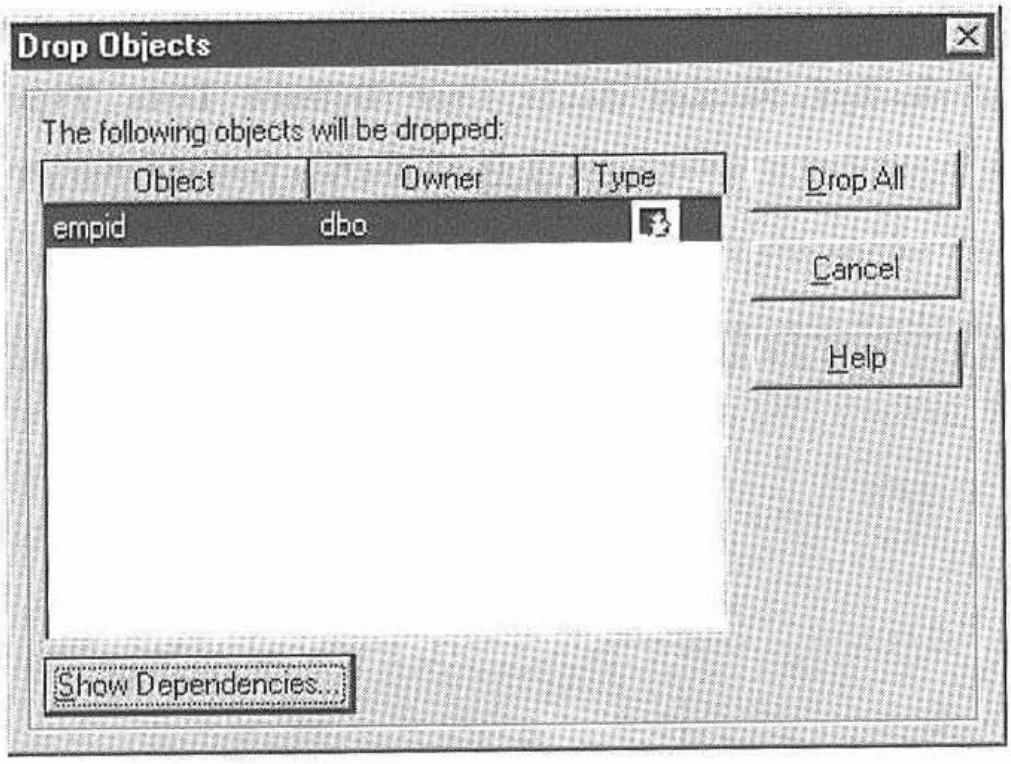
۸- به دلخواه، در لیست‌های **Rule** و **Default** یک قاعده یا پیش فرض انتخاب کنید.

حذف یک نوع داده‌ای تعریف شده توسط کاربر

۱- یک گروه سرویس دهنده را باز کنید سپس یک سرویس‌دهنده را باز کنید.

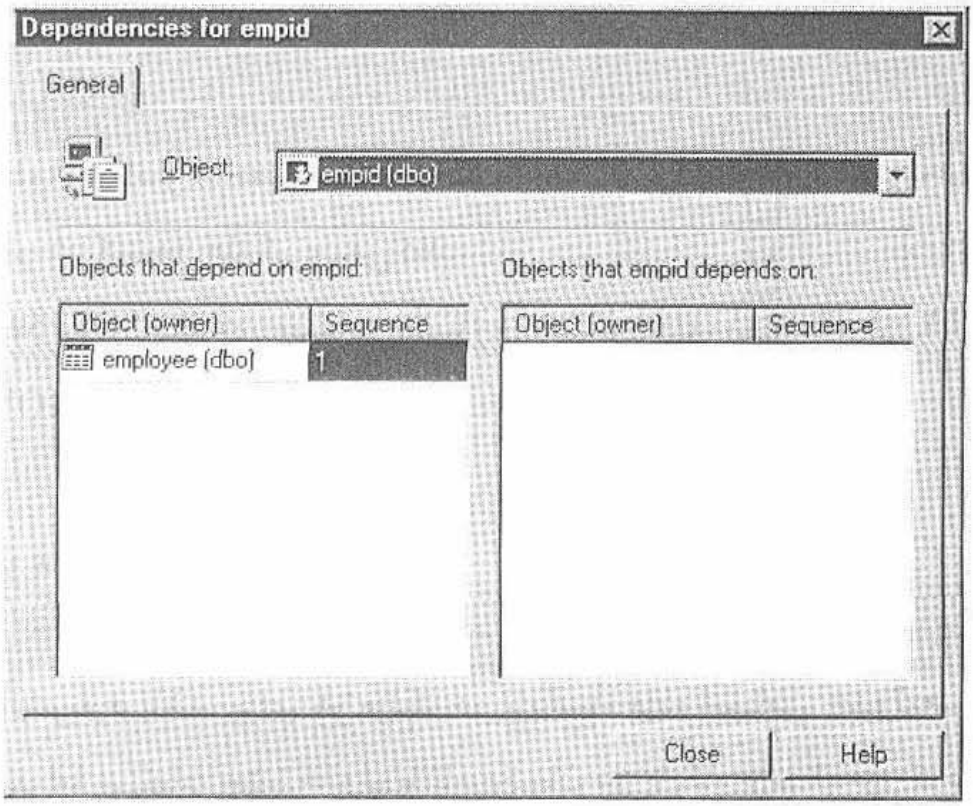
۲- **Databases** را باز کنید. پایگاه داده را باز کنید و سپس روی **User Defined Data Types** کلیک کنید.

۳- در پانل **details**، روی نوع داده‌ای رایت کلیک کنید و سپس روی **Delete** کلیک کنید.



«شکل ۱۲-۴: حذف نوع داده‌ای»

۴- برای مشاهده تأثیر حذف نوع داده‌ای بر پایگاه داده، روی **... Show Dependencies** کلیک کنید.



«شکل ۱۳-۴: وابستگی نوع داده‌ای به پایگاه داده»

۵- در جعبه مکالمه **Drop Objects** روی **Drop All** کلیک کنید.

شماره گذاری خودکار و ستون‌های شناسه

برای هر جدول می‌توان یک ستون شناسهٔ منفرد ایجاد کرد که محتوی مقادیر ترتیبی تولید شده توسط سیستم است و هر سطر در جدول را به طور منحصربفرد شناسایی می‌کند. به عنوان مثال، یک ستون شناسه می‌تواند هنگام درج یک سطر در جدول، شماره‌های پذیرش مشتری را به صورت منحصربفرد برای یک برنامه کاربردی به طور خودکار تولید کند. ستون‌های شناسه همواره شامل مقادیری هستند که درون جدولی که در آن تعریف شده‌اند منحصربفرد می‌باشند. این بدان معنی است که جدول‌های دیگری که شامل ستون‌های شناسه هستند می‌توانند مقادیر شناسه یکسانی داشته باشند. البته این امر یک مشکل به شمار نمی‌رود زیرا مقادیر شناسه همواره درون یک جدول به کار می‌روند. ستون‌های شناسه با ستون‌های شناسه جدول‌های دیگر ارتباطی ندارند.

یک ستون شناسه منحصربفرد به صورت عمومی می‌تواند در جدول ایجاد شود که محتوی مقادیر منحصربفرد بین تمام کامپیوترهای موجود در شبکه باشد. یک ستون با مقادیر منحصربفرد عمومی اغلب وقتی مفید است که داده‌های یکسان از چند سیستم پایگاه

داده باید با هم ادغام شوند. مثلاً یک سیستم صورت حساب مشتری موجود در شرکت‌های مختلف را در نظر بگیرید. وقتی داده‌ها در سایت مرکزی جهت گزارش‌گیری با هم ادغام می‌شوند، استفاده از مقادیر منحصربفرد عمومی از یکسان بودن شمارهٔ صورتحساب برای مشتریان در سایر شرکت‌ها جلوگیری می‌کند.

SQL Server برای پیاده سازی ستون‌های شناسه از IDENTITY و ROWGUIDCOL استفاده می‌کند.

۵-۴- ایجاد و مدیریت جدول

در هر جدول می‌توان حداکثر۱۰۲۴ ستون تعریف کرد. اسامی ستون‌ها در یک جدول باید منحصربفرد باشد. برای هر مالک در پایگاه داده اسامی جدول‌های آن باید منحصربفرد باشد اما می‌توان چند جدول با نام یکسان در پایگاه داده ایجاد کرد به شرطی که هر کدام از آنها مربوط به یک مالک مجزا باشند.

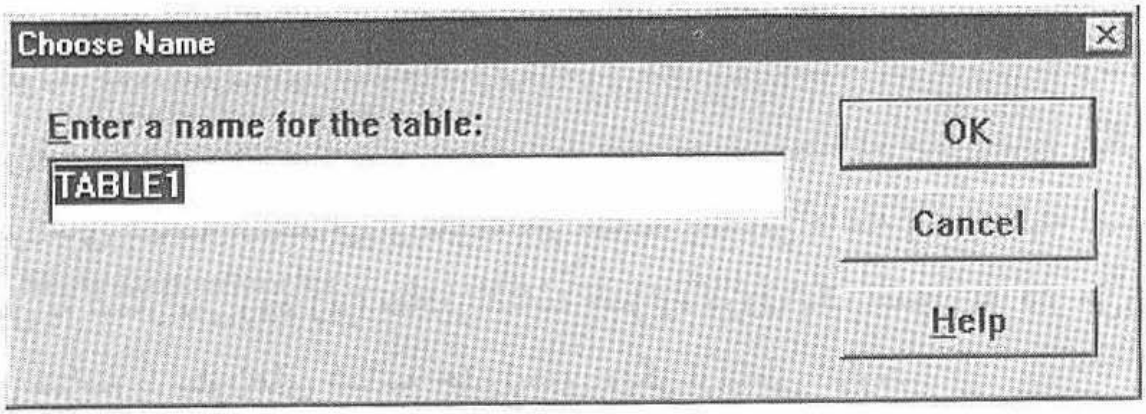
ایجاد یک جدول

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- Databases را باز کنید. سپس پایگاه داده‌ای را که می‌خواهید جدول در آن ایجاد شود، باز کنید.

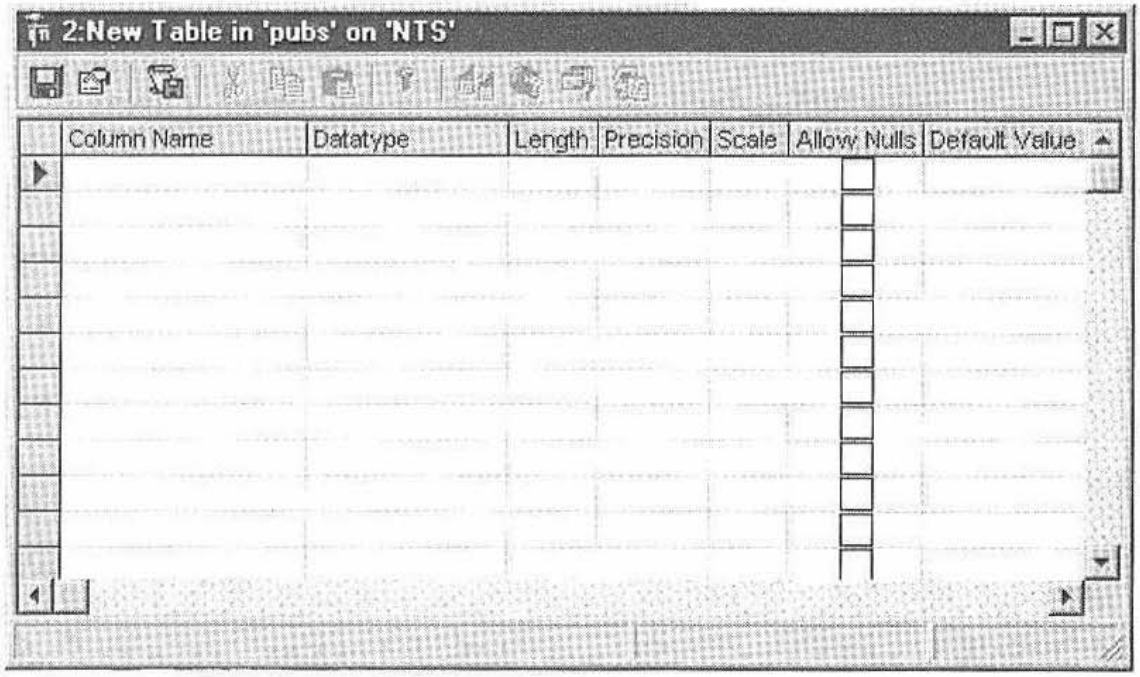
۳- روی Tables رایت کلیک کنید و سپس ... New Table را انتخاب کنید.

۴- در جعبه مکالمه Choose Name یک نام برای جدول وارد کنید.



«شکل ۱۴-۴: تعیین نام جدول»

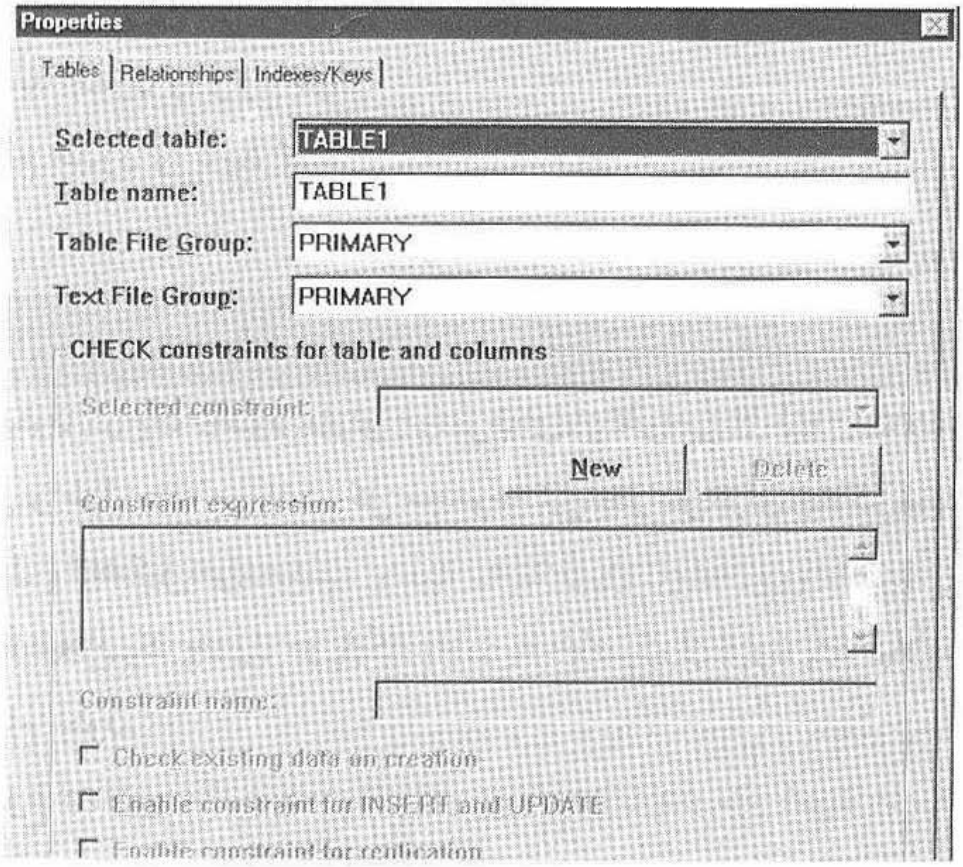
۵- ستون‌ها را به طور مناسب پر کنید. هر سطر نشان دهنده یک ستون از جدول می‌باشد.



«شل ۱۵-۴: تعریف ستون‌های جدول»

۶- روی هر یک از سطرها رایت کلیک کنید و سپس **properties** را انتخاب کنید.

۷- روی دکمه **Tables** کلیک کنید.



«شکل ۱۶-۴: تعیین خواص دیگری از جدول»

۸- به دلخواه، گروه فایلی را که می‌خواهید جدول در آن ایجاد شود، انتخاب کنید:

- در لیست **Table file group**، گروه فایلی را که جدول باید در آن ایجاد شود انتخاب کنید.

- به دلخواه، در لیست **Text file group**، گروه فایلی را که ستون‌های نوع **text**، **ntext** و **image** باید در آن قرار گیرند، انتخاب کنید.

۹- به دلخواه، بر روی **New** کلیک کنید تا محدودیت **CHECK** مورد نظر خود را ایجاد کنید:

- در قسمت **Constraint expression**، متن محدودیت را وارد کنید.
- به دلخواه، در قسمت **Constraint name** نام محدودیت را مشخص کنید.

تغییر نام یک جدول

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- **Databases** را باز کنید. پایگاه داده‌ای را که جدول در آن قرار دارد باز کنید و سپس بر روی **Tables** کلیک کنید.

۳- در پانل **details** روی جدول مورد نظر راست کلیک کنید و سپس **Rename** را انتخاب کنید.

۴- نام جدید جدول را وارد کنید.

۵- تغییر نام جدول را تأیید کنید.

تغییر خواص ستون

هر ستون در یک جدول دارای مجموعه‌ای از خواص از قبیل نام، نوع داده‌ای، **NULL** پذیری و طول داده می‌باشد. کل مجموعه این خواص برای یک ستون، تعریف ستون در جدول را تشکیل می‌دهند.

قبل از ایجاد یک جدول در پایگاه داده، سه خاصیت از ستون باید قبلاً مشخص شود:

- نام ستون.
- نوع داده‌ای.
- طول داده.

تعیین خواص ستون

۱- یک گروه سرویس دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- Databases را باز کنید. پایگاه داده‌ای که جدول در آن قرار دارد را باز کنید. سپس روی Tables کلیک کنید.
۳- در پانل details، روی جدول رایت کلیک کنید و سپس Design Table را انتخاب کنید.
۴- درون سلولی که مقدار خاصیت را نشان می‌دهد، کلیک کنید.
۵- مقدار جدید را تایپ کرده یا انتخاب کنید.

اضافه کردن و حذف ستون ها

SQL Server اجازه می‌دهد که ستون‌هایی به جدول‌های موجود اضافه کرد، با این شرط که NULL پذیری یا محدودیت DEFAULT بر روی ستون‌ها اعمال گردد. وقتی یک ستون جدید به جدول اضافه می‌شود، برای هر سطر موجود در جدول، SQL Server یک مقدار در ستون درج می‌کند.

همچنین می‌توان ستون‌هایی را از جدول‌های موجود حذف کرد. اگر ستون در یک شاخص به کار رفته باشد و یا بر روی آن محدودیت‌هایی تعریف شده باشد، حذف آن امکان پذیر نیست.

اضافه یا حذف یک ستون

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.
۲- Databases را باز کنید. پایگاه داده‌ای که جدول در آن قرار دارد را باز کنید. سپس روی Tables کلیک کنید.
۳- در پانل details، روی جدول رایت کلیک کنید و سپس Design Table را انتخاب کنید.
۴- برای حذف یک ستون، روی نام ستون رایت کلیک کنید و سپس Delete Column را انتخاب کنید.
۵- برای اضافه کردن یک ستون جدید، بر روی نام ستونی که بعد از آن قرار می‌گیرد رایت کلیک کنید و سپس Insert Column را انتخاب کنید.
۶- برای ذخیره تعریف جدید جدول، بر روی نام یکی از ستون‌ها رایت کلیک کنید و سپس Save Selection را انتخاب کنید.

وقتی یک محدودیت **PRIMARY KEY** به ستون یا ستون‌های موجود در یک جدول اضافه می‌شود، **SQL Server** داده‌های موجود در ستون‌ها را چک می‌کند تا تضمین کند که داده‌های موجود از قوانین مربوط به کلیدهای اصلی تبعیت می‌کنند:

- مقادیر **NULL** وجود نداشته باشد.
- مقادیر تکراری وجود نداشته باشد.

اگر محدودیت **PRIMARY KEY** به ستونی اضافه شود که دارای مقادیر **NULL** یا تکراری باشد، **SQL Server** خطایی باز می‌گرداند و محدودیت را اضافه نمی‌کند.

ایجاد یک محدودیت PRIMARY KEY در یک جدول

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- **Databases** را باز کنید. پایگاه داده‌ای که جدول در آن قرار دارد را باز کنید و سپس روی **Tables** کلیک کنید.

۳- در پانل **details**، روی جدول رایت کلیک کنید و سپس **Design Table** را انتخاب کنید.

۴- کلید **CTRL** را پایین نگهداشته و روی هر یک از ستون‌هایی که در کلید اصلی قرار دارند، کلیک کنید.

۵- روی نام یکی از ستون‌های انتخاب شده رایت کلیک کنید و سپس **Set Primary KEY** را انتخاب کنید.

اگر جدول از قبل وجود نداشته باشد، برای ایجاد محدودیت، عملیات فوق هنگام ایجاد جدول انجام می‌شود.

تغییر یک محدویت PRIMARY KEY

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

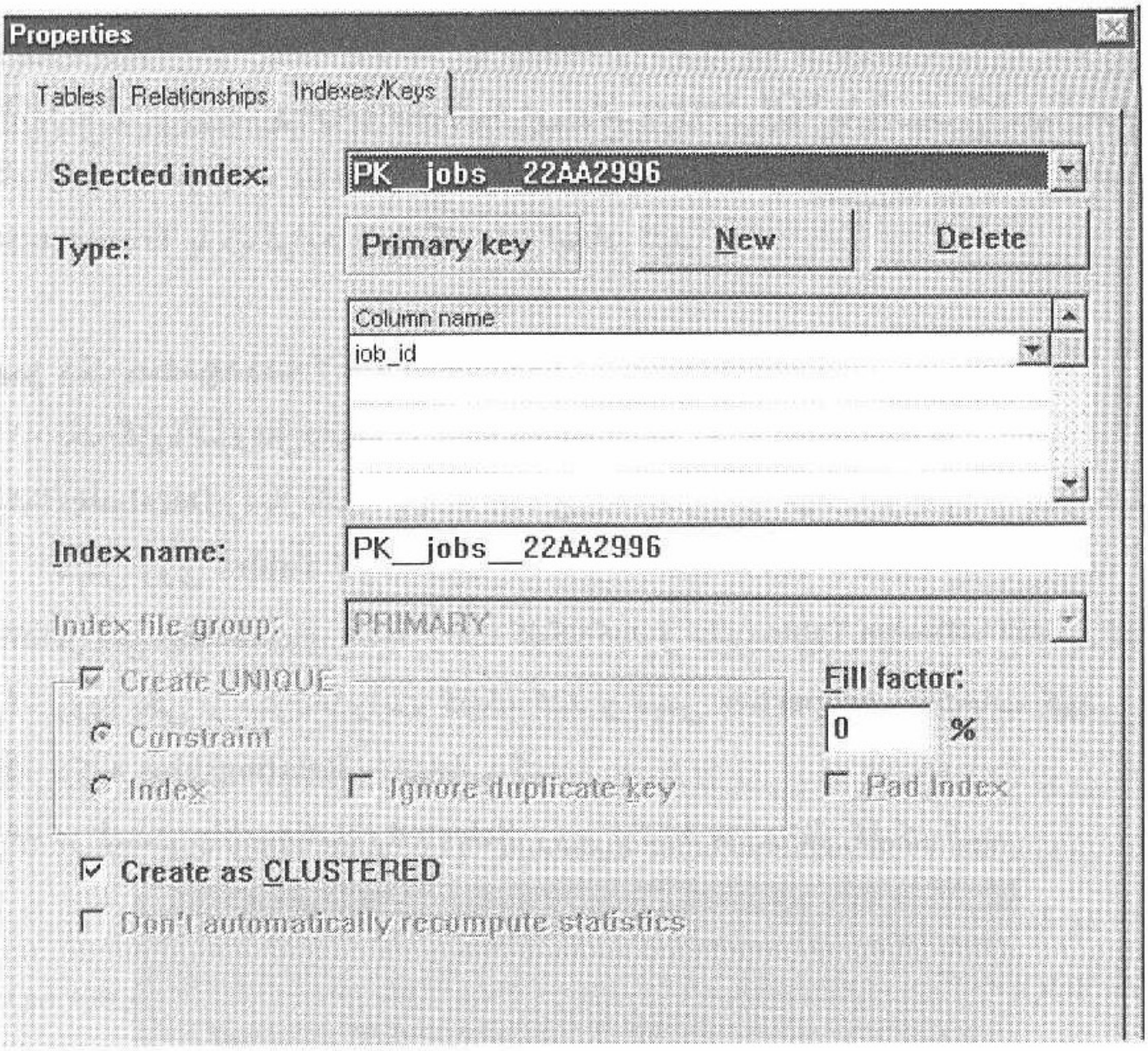
۲- **Databases** را باز کنید. پایگاه داده‌ای که جدول در آن قرار دارد را باز کنید و سپس روی **Tables** کلیک کنید.

۳- در پانل **details**، روی جدول رایت کلیک کنید و سپس **Design Table** را انتخاب کنید.

۴- روی یکی از ستون‌ها رایت کلیک کنید و سپس **properties** را انتخاب کنید.

۵- روی دکمه **Indexes/keys** کلیک کنید.

۶- بر روی شاخص کلید اصلی در لیست **Selected index** کلیک کنید.



«شکل ۱۷-۴: تغییر محدودیت PRIMARY KEY»

۷- در اینجا موارد زیر را می‌توانید انجام دهید:

- تغییر ترتیب ستون‌ها: در قسمت **Column name**، لیست ستون‌ها را باز کنید و ستون‌ها را به ترتیب دلخواه انتخاب کنید.
- تغییر نام کلید اصلی: در جعبه **Index name** یک نام جدید وارد کنید.
- گزینه خوشه‌ای بودن: **Create as CLUSTERED** را انتخاب کنید.
- تعریف عامل سرریز: در قسمت **Fill factor** یک عدد صحیح بین ۰ تا ۱۰۰ وارد کنید. اگر **0** وارد شود، از عامل سرریز پیش فرض استفاده خواهد شد.

وقتی یک محدودیت **FOREIGN KEY** به ستون یا ستون‌های جدول اضافه می‌شود، **SQL Server** به طور پیش فرض داده‌های موجود در ستون‌ها را چک می‌کند تا تضمین کند که تمام مقادیر به جز **NULL**، در محدودیت **PRIMARY KEY** یا **UNIQUE** ارجاع شده وجود دارند. البته می‌توان **SQL Server** را طوری تنظیم کرد که هنگام ایجاد محدودیت، داده‌های موجود را چک نکند. این محدودیت را می‌توانید با **Transact-SQL** یا دیاگرام‌های پایگاه داده ایجاد کنید.

تغییر یک محدودیت FOREIGN KEY

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

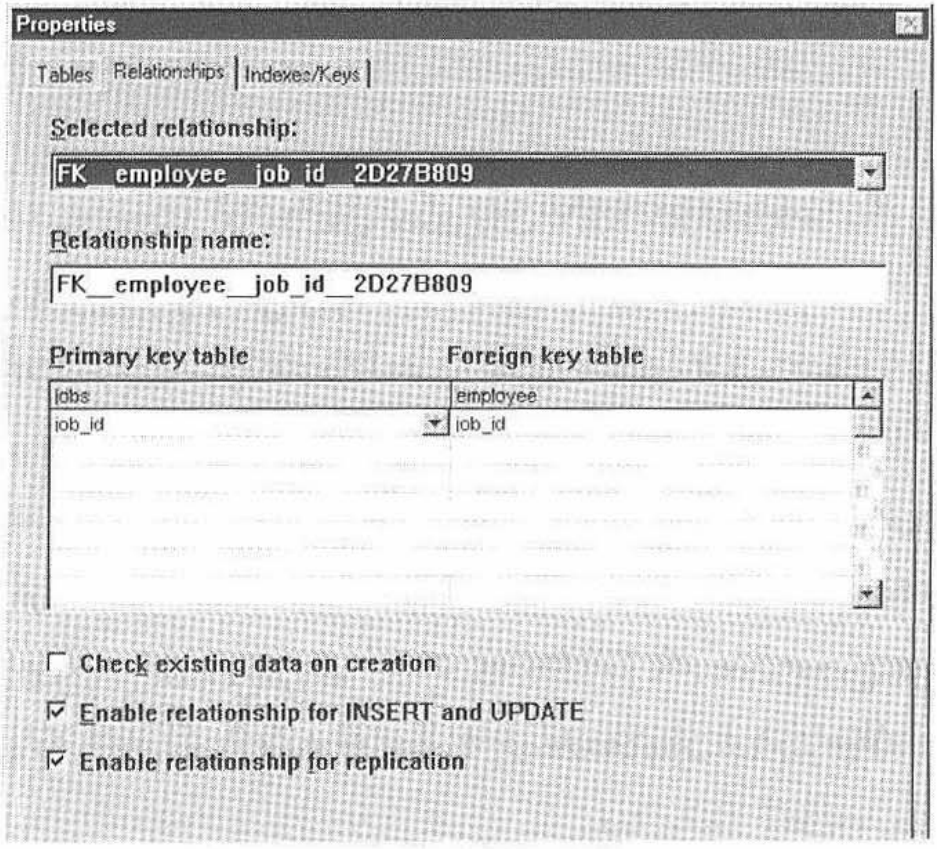
۲- **Databases** را باز کنید، پایگاه داده‌ای که جدول در آن قرار دارد را باز کنید و سپس روی **Tables** کلیک کنید.

۳- در پانل **details**، روی جدول رایت کلیک کنید و سپس **Design Table** را انتخاب کنید.

۴- روی یکی از ستون‌ها رایت کلیک کنید و سپس **properties** را انتخاب کنید.

۵- دکمه **Relationships** را انتخاب کنید.

۶- در لیست **Selected relationship**، روی ارتباط مورد نظر کلیک کنید.



«شکل ۱۸-۴: تغییر محدویت FOREIGN KEY»

ایجاد و مدیریت محدودیت‌های UNIQUE

وقتی یک محدودیت UNIQUE به ستون یا ستون‌های موجود در جدول اضافه می‌شود، SQL Server به طور پیش فرض داده‌های موجود در ستون‌ها را چک می‌کند تا تضمین کند که تمام مقادیر به جز NULL، منحصربفرد هستند. اگر این محدودیت به ستونی اضافه شود که دارای مقادیر تکراری است، SQL Server یک خطا باز می‌گرداند و محدودیت را اضافه نمی‌کند.

ایجاد محدودیت UNIQUE در یک جدول

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- Databases را باز کنید. پایگاه داده‌ای را که جدول در آن قرار دارد باز کنید و سپس روی Tables کلیک کنید.

۳- در پانل details، روی جدول رایت کلیک کنید و سپس Design Table را انتخاب کنید.

۴- روی یکی از سطرها رایت کلیک کنید و سپس properties را انتخاب کنید.

۵- روی دکمه Indexes/keys کلیک کنید. سپس روی New کلیک کنید.

۶- در لیست Column name، ستون‌های مورد نظر را انتخاب کنید.

۷- Create UNIQUE را انتخاب کنید تا این شاخص، یک محدودیت منحصربفرد شود.

برای تغییر یک محدودیت UNIQUE به همان صورت محدودیت PRIMARY KEY رفتار می‌شود با این تفاوت که در بند ۶، در لیست Selected index، محدودیت UNIQUE را انتخاب می‌کنیم.

برای حذف محدودیت UNIQUE، در صفحه Indexes/keys، در لیست Selected index، نام شاخصی که محدودیت UNIQUE را اعمال می‌کند انتخاب کرده و سپس روی Delete کلیک کنید.

ایجاد و مدیریت محدودیت‌های CHECK

وقتی یک محدودیت CHECK به یک جدول اضافه می‌شود، می‌توان آن را فقط به داده‌های جدید و یا علاوه بر آن به داده‌های موجود در جدول نیز اعمال کرد. به طور

پیش فرض، محدودیت CHECK هم بر داده‌های موجود در جدول و هم بر داده‌های جدید اعمال می‌شود.

ایجاد محدودیت CHECK بر روی جدول

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- Databases را باز کنید. پایگاه داده‌ای که جدول در آن قرار دارد باز کنید و سپس بر روی Tables کلیک کنید.

۳- در پانل details، روی جدول رایت کلیک کنید و سپس Design Table را انتخاب کنید.

۴- بر روی نام ستونی که می‌خواهید برای آن محدودیت CHECK ایجاد کنید، رایت کلیک کنید و سپس Properties را انتخاب کنید.

۵- بر روی دکمه Tables کلیک کنید. سپس بر روی New کلیک کنید.

۶- در قسمت Constraint expression، متن محدودیت را وارد کنید.

۷- هر یک از موارد زیر را به دلخواه انتخاب کنید:

- در قسمت Constraint name، یک نام برای محدودیت وارد کنید.
- برای این که محدودیت بر روی داده‌های موجود نیز اعمال شود، Check existing data on creation را انتخاب کنید.
- اگر بخواهید هنگام اضافه کردن داده‌های جدید یا بهنگام‌سازی داده‌های موجود، محدودیت غیر فعال باشد، Enable constraint for INSERT and UPDATE را از حالت انتخاب خارج کنید.

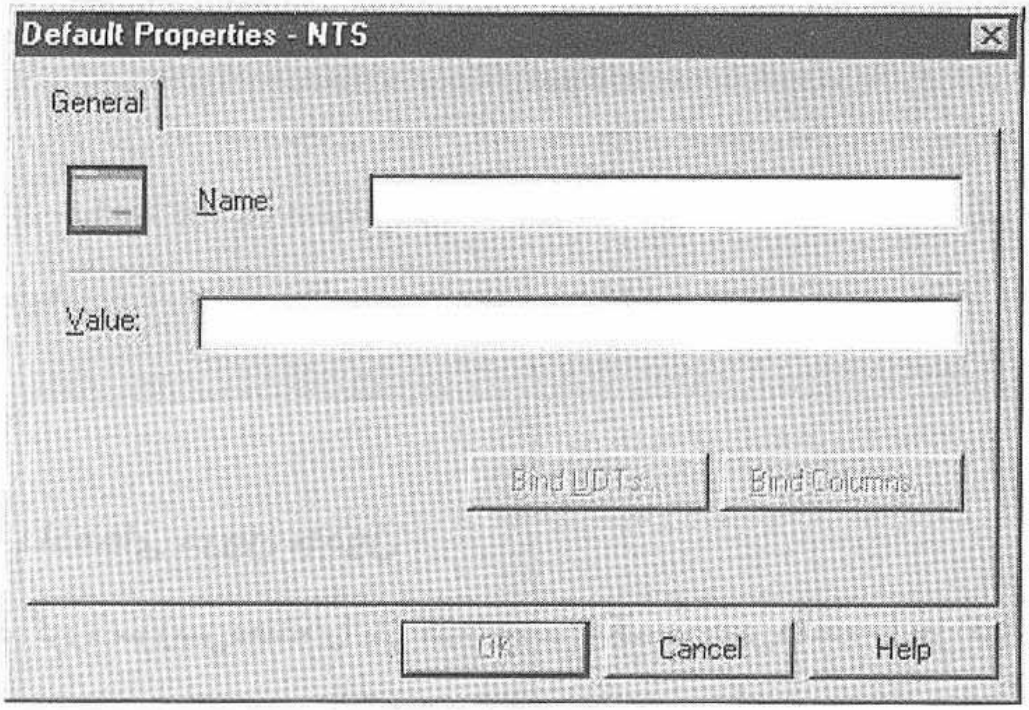
ایجاد یک شئ DEFAULT

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- Databases را باز کنید، پایگاه داده‌ای را که می‌خواهید شئ DEFAULT در آن ایجاد شود، باز کنید.

۳- بر روی Defaults رایت کلیک کنید. سپس ... New Default را انتخاب کنید.

۴- در قسمت Name یک نام برای شئ انتخاب کنید.



«شکل ۱۹-۴: ایجاد یک شئ DEFAULT»

۵- در قسمت **Value**، یک مقدار برای شئ وارد کنید. مقدار می‌تواند یک ثابت، عبارت یا متغیر باشد.

۶- به دلخواه، موارد زیر را انتخاب کنید:

- برای مقید کردن شئ به یک نوع داده‌ای تعریف شده توسط کاربر، **Bind UDTs ...** را انتخاب کنید.
- برای مقید کردن شئ به یک ستون از یک جدول، **Bind Columns** را انتخاب کنید.

حذف یک شئ DEFAULT

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- **Databases** را باز کنید. پایگاه داده‌ای را که شئ **DEFAULT** در آن قرار دارد باز کنید و سپس روی **Defaults** کلیک کنید.

۳- در پانل **details** روی شئ **DEFAULT** مورد نظر رایت کلیک کنید و سپس روی **Delete** کلیک کنید.

۴- برای مشاهده این که حذف شئ چه تأثیری بر پایگاه داده دارد، روی **Show Dependencies** کلیک کنید.

۵- روی **Drop All** کلیک کنید.

پس از این که جدول‌های پایگاه داده را ایجاد کردید، نیاز به مشاهده اطلاعات درون جدول خواهید داشت. همچنین می‌توانید وابستگی‌های جدول را نمایش دهید تا مشخص نمایید چه شیءها، دیدها، رویه‌های ذخیره شده و Triggerهایی به جدول وابسته هستند.

مشاهده داده‌های درون جدول

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- Databases را باز کنید، پایگاه داده‌ای که جدول در آن قرار دارد را باز کنید و سپس روی Tables کلیک کنید.

۳- در پانل details، روی جدول رایت کلیک کنید. Open Table را انتخاب کنید و سپس روی Return all rows کلیک کنید تا تمام اطلاعات جدول را مشاهده کنید.

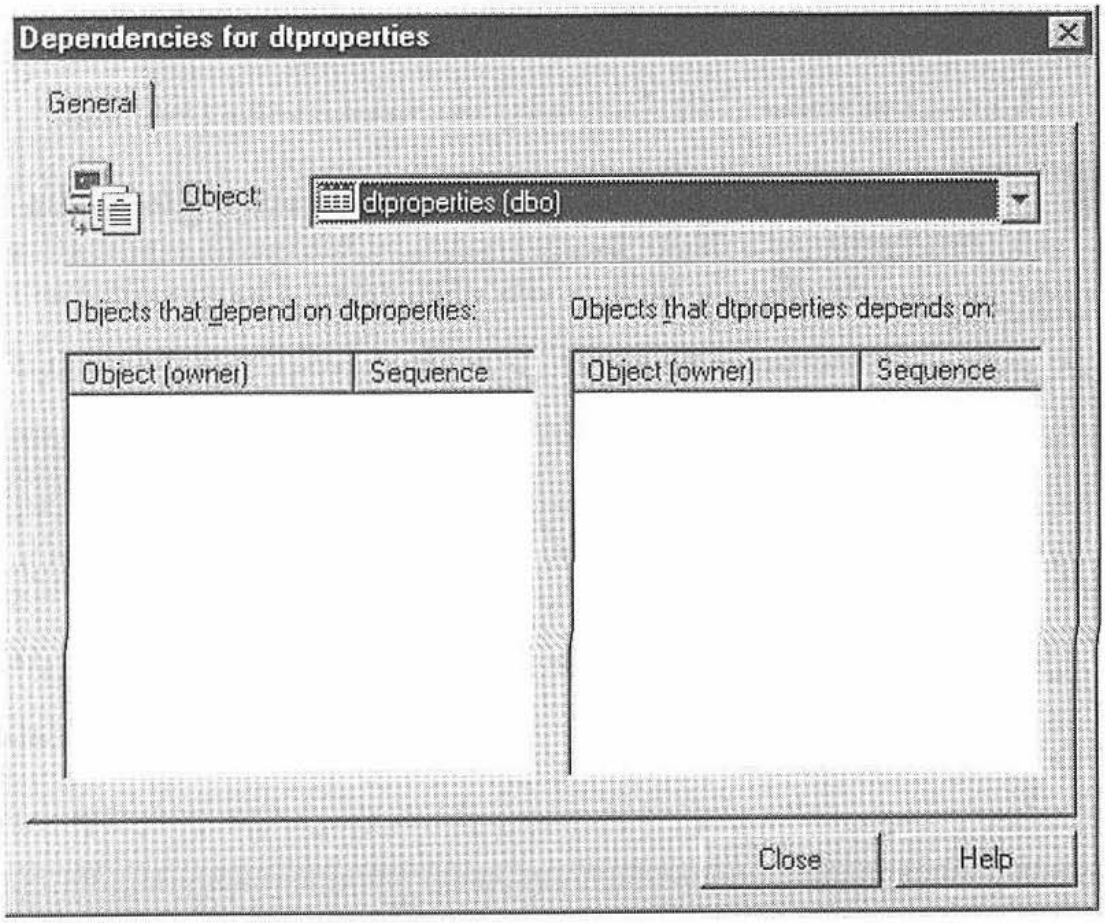
۴- به طور دلخواه، در پانل details روی جدول رایت کلیک کنید، Open Table را انتخاب کرده و بر روی ... Return Top کلیک کرده و تعداد سطرهایی را که می‌خواهید نمایش داده شود، وارد کنید.

مشاهده وابستگی‌های یک جدول

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- Databases را باز کنید، پایگاه داده‌ای که جدول در آن قرار دارد را باز کنید و سپس روی Tables کلیک کنید.

۳- در پانل Details روی جدول مورد نظر رایت کلیک کنید. All Task را انتخاب کرده و سپس روی ... Display Dependencies کلیک کنید.



«شکل ۲۰-۴ : وابستگی‌های یک جدول»

حذف جدول

وقتی جدولی را حذف می‌کنید، تعاریف ساختاری، داده‌ها، محدودیت‌ها و شاخص‌ها برای همیشه از پایگاه داده حذف می‌شوند و فضای خالی جدیدی برای سایر جدول‌ها مهیا می‌گردد. اگر می‌خواهید جدولی را حذف کنید که از طریق محدودیت‌های **FOREIGN KEY** و **PRIMARY KEY** یا **UNIQUE** با جداول دیگر ارتباط دارد، باید ابتدا جداول با محدودیت‌های **FOREIGN KEY** را حذف کنید.

حذف یک جدول

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- **Databases** را باز کنید. پایگاه داده‌ای که جدول در آن قرار دارد را باز کرده و روی **Tables** کلیک کنید.

۳- در پانل **details** روی جدول رایت کلیک کرده و سپس روی **Delete** کلیک کنید.

۴- برای مشاهده تأثیر حذف جدول بر پایگاه داده، روی **Show Dependencies** کلیک کنید.

۵- روی Drop All کلیک کنید.

۶-۴- دیاگرام‌های پایگاه داده

با استفاده از دیاگرام‌های پایگاه داده می‌توانید با پایگاه داده سرویس‌دهنده ارتباط برقرار کنید. دیاگرام‌های پایگاه داده از طریق Enterprise Manager دستیابی می‌شوند. دیاگرام‌های پایگاه داده جدول‌های درون پایگاه داده را به صورت گرافیکی نمایش می‌دهند. این دیاگرام‌ها ستون‌های درون جدول‌ها، ارتباط‌های بین جدول‌ها و شاخص‌ها و محدودیت‌های مربوط به جدول‌ها را نشان می‌دهند. به طور کلی می‌توان گفت که دیاگرام‌های پایگاه داده در ایجاد، مدیریت و مشاهده شئ‌های پایگاه داده به شکل گرافیکی مورد استفاده قرار می‌گیرند.

ایجاد و مدیریت دیاگرام‌های پایگاه داده

دیاگرام‌های پایگاه داده در موارد زیر مورد استفاده قرار می‌گیرند:

- کار با شئ‌های پایگاه داده بدون استفاده از Transact - SQL.
- نمایش ساختار جدول‌های پایگاه داده و ارتباطات بین آنها.
- تهیه نمودارهای متنوع از پایگاه‌های داده پیچیده.
- ایجاد تغییرات آزمایشی در پایگاه داده بدون این که به طور دائم اعمال شود.
- ایجاد جدول‌ها، شاخص‌ها، ارتباطات و سایر محدودیت‌های جدید.
- تغییر ساختار پایگاه داده.

شما می‌توانید بدون تأثیر بر تعریف‌های موجود در پایگاه داده، اندازه، شکل و مکان شئ‌ها را در دیاگرام تغییر دهید. وقتی دیاگرام پایگاه داده را ذخیره می‌کنید، هم طرح کلی دیاگرام و هم تغییراتی که در تعریف‌های پایگاه داده انجام شده است ابقاء می‌شوند.

دیاگرام‌های پایگاه داده در جدول Dtproperties ذخیره می‌شوند. شئ‌هایی از پایگاه داده که می‌توان آنها را توسط دیاگرام‌های پایگاه داده ایجاد یا اصلاح کرد عبارتند از: جدول‌ها، ستون‌ها، ارتباطات بین جدول‌ها، شاخص‌ها، محدودیت‌ها، رویه‌های ذخیره شده و Triggerها.

ایجاد دیاگرام پایگاه داده

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- **Databases** را باز کنید، پایگاه داده‌ای را می‌خواهید در آن دیاگرام ایجاد کنید، باز نمایید.

۳- بر روی **Diagrams** رایت کلیک کنیـــد و سـپس **New Database Diagram...** را انتخاب کنید.

اضافه کردن جدول‌ها به دیاگرام پایگاه داده

جدول‌ها از اجزاء اصلی دیاگرام‌های پایگاه داده به شمار می‌آیند. هر دیاگرام از یک یا چند جدول و شئ‌های مختلف مرتبط با آن جدول‌ها تشکیل شده‌اند.

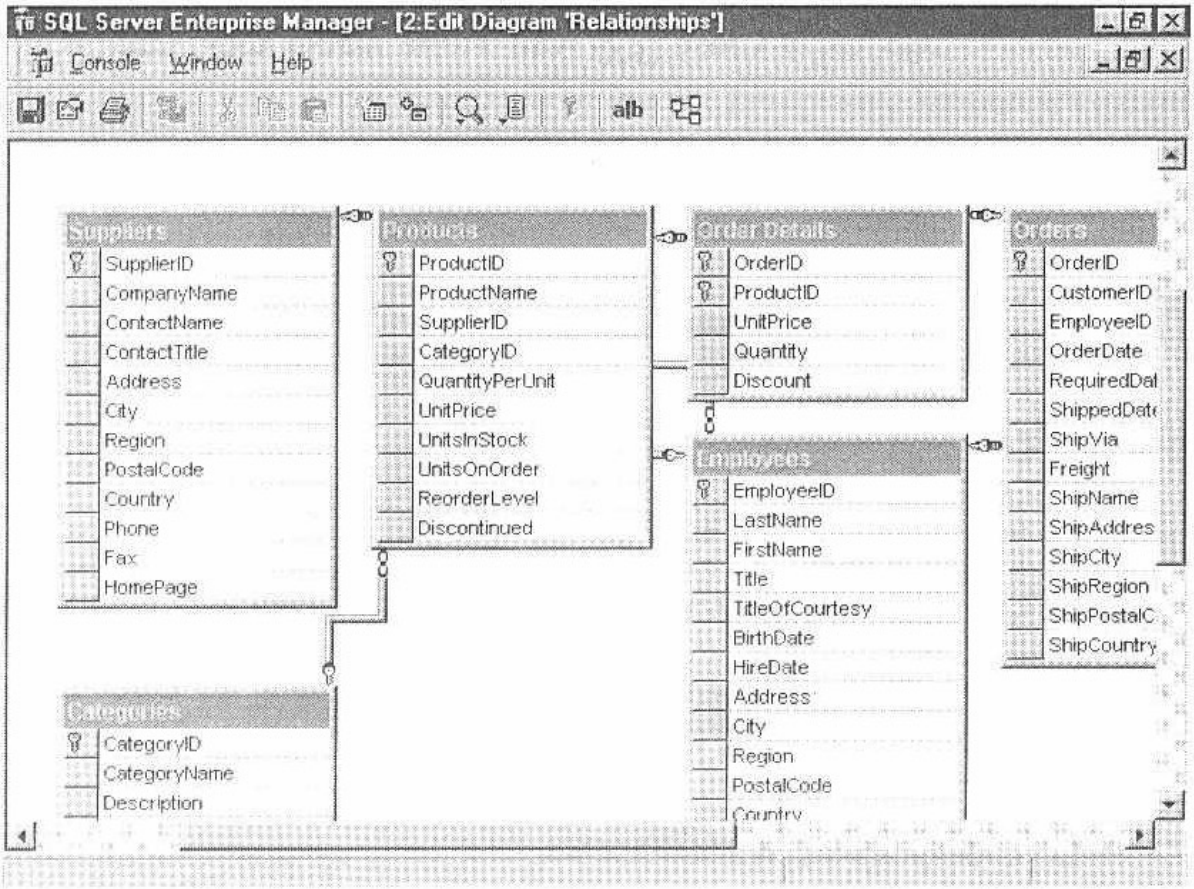
باز کردن یک دیاگرام پایگاه داده

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- **Databases** را باز کنید، پایگاه داده‌ای را که دیاگرام در آن قرار دارد باز کنید.

۳- روی **Diagrams** کلیک کنید.

۴- روی یک نمودار پایگاه داده رایـــت کلیـک کنیـد و سـپس **Design Diagram** را انتخاب کنید.



«شکل ۲۱-۴ : دیاگرام پایگاه داده»

درج یک جدول جدید در دیاگرام پایگاه داده

۱- یک دیاگرام پایگاه داده را باز کنید.

۲- درون دیاگرام رایت کلیک کنید و سپس ... **New Table** را انتخاب کنید.

۳- در جعبه مکالمه **Choose Name** نام جدول جدید را تایپ کنید. یک جدول در دیاگرام ظاهر خواهد شد.

۴- در اولین سلول از جدول جدید، نام ستون را وارد کنید. کلید **TAB** را فشار دهید تا به سلول بعدی بروید.

۵- تحت **Datatype** یک نوع داده‌ای برای ستون انتخاب کنید.

۶- مراحل ۳ تا ۵ را برای هر ستون تکرار کنید.

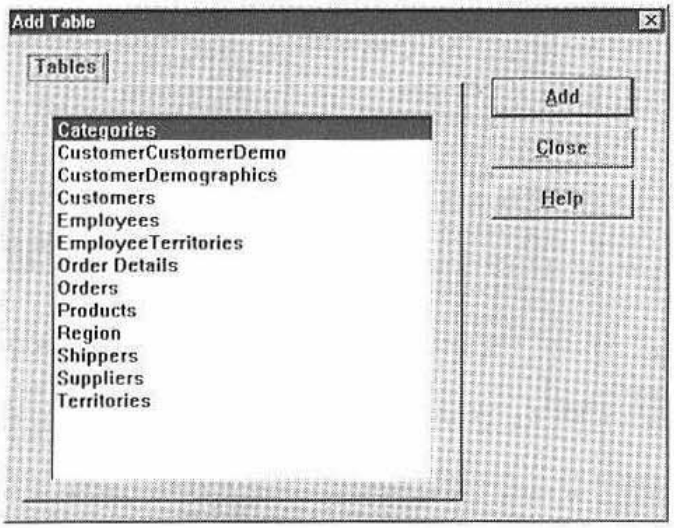
اضافه کردن یک جدول موجود به دیاگرام پایگاه داده

۱- یک دیاگرام را باز کنید.

۲- درون دیاگرام رایت کلیک کنید و سپس ... **Add Existing Table** را انتخاب کنید.

۳- در جعبه مکالمه **Add Table** روی جدول مورد نظر کلیک کرده و سپس روی **Add**کلیک کنید.

۴- برای بستن جعبه مکالمه **Add Table** روی **Close** کلیک کنید.



«شکل ۲۲-۴: اضافه کردن یک جدول موجود به دیاگرام»

اضافه کردن جدول‌های مرتبط به دیاگرام پایگاه داده

۱- یک دیاگرام را باز کنید.

۲- یک یا چند جدول با محدودیت‌های کلید خارجی را در پایگاه داده انتخاب کنید.

۳- روی یکی از جدول‌های انتخاب شده رایت کلیک کنید و سپس **Add Related Tables** را انتخاب کنید.

ذخیره سازی دیاگرام پایگاه داده

برای این که تغییرات انجام شده در طرح دیاگرام و تعریف شئ‌های دیاگرام در پایگاه داده بهنگام سازی شوند، دیاگرام پایگاه داده بایستی ذخیره شود.

ذخیره دیاگرام پایگاه داده

۱- درون دیاگرام پایگاه داده رایت کلیک کرده و **Save** را انتخاب کنید.

۲- در جعبه مکالمه **Save As** یک نام برای دیاگرام وارد کنید.

۳- روی **Yes** کلیک کنید تا پایگاه داده بهنگام سازی شده و با دیاگرام هماهنگ شود.

چاپ و مشاهده قبل از چاپ دیاگرام پایگاه داده

برای این که تصویر مفیدی از ساختار پایگاه داده داشته باشید، می‌توانید آن را چاپ کنید. قبل از چاپ، شئ‌ها را در دیاگرام به نحو مطلوب مرتب کنید.

چاپ یک دیاگرام پایگاه داده

۱- یک دیاگرام پایگاه داده را باز کنید.

۲- درون دیاگرام رایت کلیک کنید و سپس **Print** را انتخاب کنید.

۳- در جعبه مکالمه Print، انتخاب‌های لازم را انجام داده و روی **ok** کلیک کنید.

مشاهده قبل از چاپ دیاگرام پایگاه داده

۱- یک دیاگرام را باز کنید.

۲- درون دیاگرام رایت کلیک کنید و سپس View page Breaks را انتخاب کنید.

حذف جدول از دیاگرام پایگاه داده

حذف یک جدول از دیاگرام پایگاه داده، پایگاه داده سرویس‌دهنده را تغییر نخواهد داد.

حذف یک جدول از دیاگرام پایگاه داده

۱- یک دیاگرام پایگاه داده را باز کنید.

۲- روی جدولی که می‌خواهید حذف کنید رایت کلیک کرده و سپس Remove Table from Diagram را انتخاب کنید.

اگر جدول در نتیجه ویرایش‌های صورت گرفته در دیاگرام تغییر کرده باشد، قبل از حذف جدول از دیاگرام در مورد ذخیره جدول از شما سؤال خواهد شد.

۷-۴- ایجاد و مدیریت شاخص

شاخص‌ها می‌توانند خوشه‌ای، غیر خوشه‌ای، منحصربفرد، غیر منحصربفرد و یا ترکیبی از یک یا چند ستون باشند. شاخص‌های خوشه‌ای در موارد زیر به کار می‌روند:

- ستون‌هایی که شامل تعداد محدودی مقادیر مجزا می‌باشند، مثل یک ستون وضعیت که فقط شامل ۵۰ کد وضعیتی می‌باشد. هر چند اگر تعداد مقادیر متمایز خیلی کم باشد، مثلاً فقط ۰ و ۱ باشد، هیچ شاخصی نباید ایجاد شود.
- پرس و جوهایی که محدوده‌ای از مقادیر را با استفاده از عملگرهای BETWEEN، <، =<، > و => بر می‌گردانند.
- ستون‌هایی که به صورت ترتیبی دستیابی می‌شوند.
- پرس و جوهایی که مجموعه‌های نتیجه بزرگ را باز می‌گردانند.
- ستون‌هایی که به طور مکرر توسط پرس و جوهایی دستیابی می‌شوند که شامل پیوند یا عبارت Group By می‌باشد. نوعاً، این ستون‌ها کلید خارجی می‌باشند.

ستون‌هایی که به طور مکرر دستخوش تغییرات می‌شوند، برای شاخص‌های خوشه‌ای مفید نیستند.

شاخص‌های غیر خوشه‌ای در موارد زیر به کار می‌روند:

- ستون‌هایی که حاوی تعداد زیادی مقادیر متمایز می‌باشند.
- پرس و جوهایی که مجموعه‌های نتیجه بزرگ برنمی‌گردانند.
- ستون‌هایی که اغلب تحت شرایط جستجوی پرس و جویی که تطابق دقیق باز می‌گردانند، قرار دارند.

ایجاد یک شاخص

پس از طراحی شاخص‌ها می‌توان آنها را بر روی جدول‌های پایگاه داده ایجاد کرد.

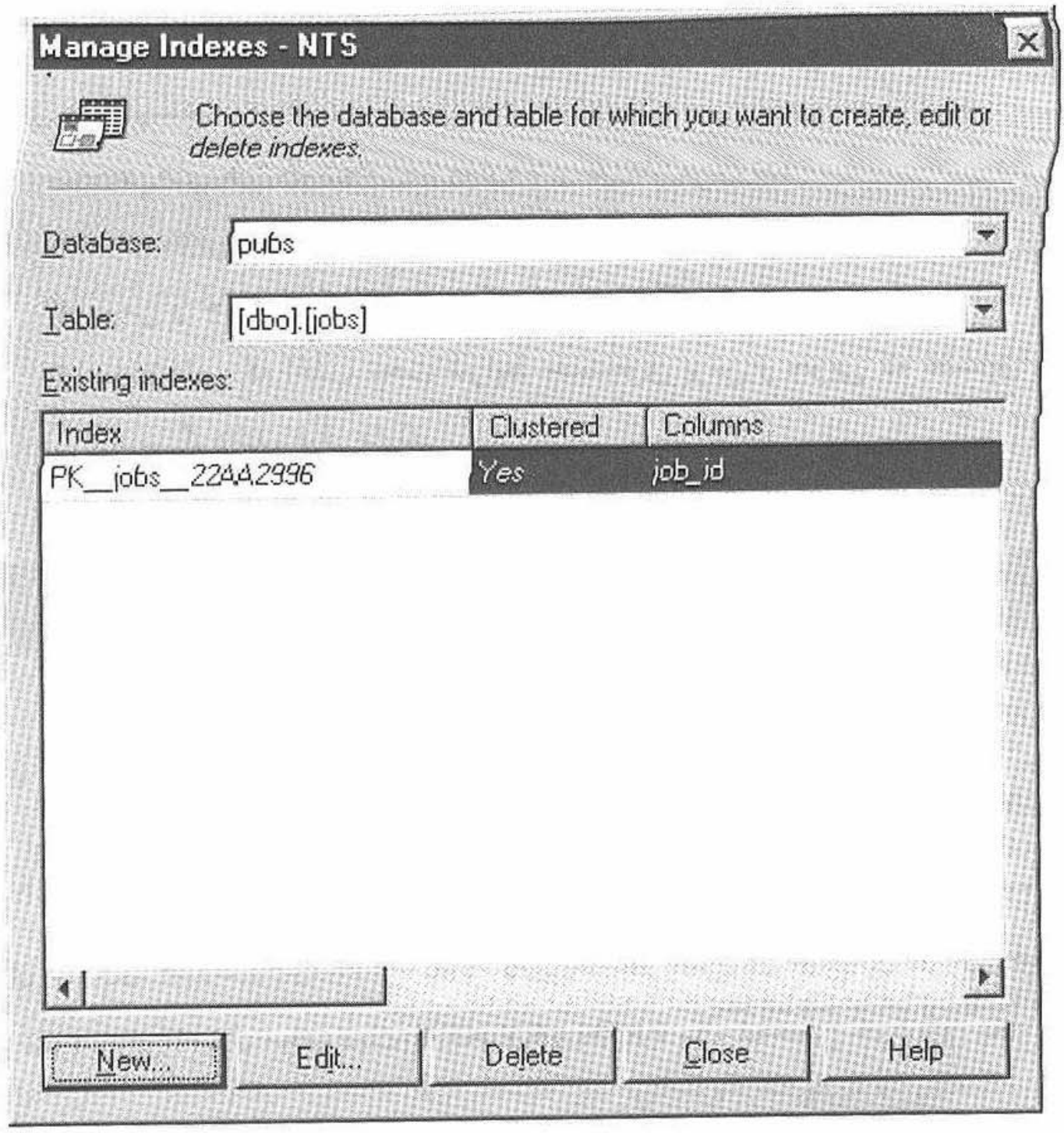
SQL Server برای اعمال یکتایی مقدار محدودیت‌های PRIMARY KEY و UNIQUE به طور خودکار شاخص‌های یکتا ایجاد می‌کند. اگر از قبل یک شاخص خوشه‌ای بر روی جدول وجود نداشته باشد و یا یک شاخص غیرخوشه‌ای به طور صریح تعریف نشود، برای اعمال محدودیت PRIMARY KEY یک شاخص خوشه‌ای یکتا ایجاد می‌گردد.

ایجاد شاخص بر روی یک جدول

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- Databases را باز کنید. پایگاه داده‌ای که جدول در آن قرار دارد را باز کنید و سپس روی Tables کلیک کنید.

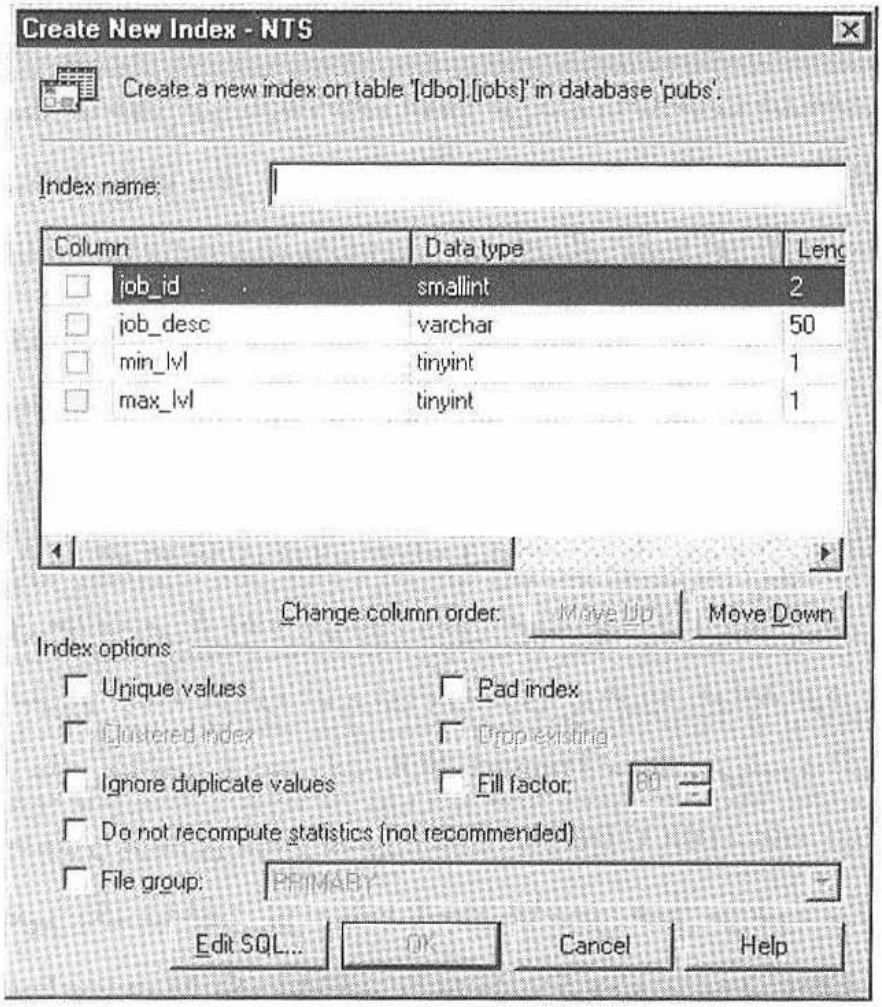
۳- در پانل details روی جدول رایت کلیک کنید و سپس All Tasks را انتخاب کنید. روی ...Manage Indexes کلیک کنید.



«شکل ۲۳-۴: شاخص‌های موجود»

۴- روی ... New کلیک کنید.

۵- در قسمت **Index name**، نام شاخص را وارد کنید.



«شکل۲۴-۴: ایجاد شاخص جدید»

۶- در قسمت **Column**، ستونی را که باید در شاخص ظاهر شود انتخاب کنید. شاخص‌های ترکیبی با انتخاب چند ستون ایجاد می‌شوند.

۷- برای تغییر ترتیب ستون‌ها در شاخص از **Move Up** یا **Move Down** استفاده کنید.

۸- به دلخواه، در قسمت **Index options** موارد زیر را انتخاب کنید:

- برای ایجاد شاخص یکتا، **UNIQUE Values** را انتخاب کنید.
- برای ایجاد شاخص خوشه‌ای، **Clustered index** را انتخاب کنید. اگر قبلاً شاخص خوشه‌ای ایجاد شده باشد این گزینه غیر فعال است.
- برای نادیده گرفتن مقادیر غیر یکتا **Ignore duplicate values** را انتخاب کنید.
- برای اینکه مشخص کنید هنگام بهنگام سازی شاخص، اطلاعات آماری به طور خودکار دوباره محاسبه نشوند، **Do not recompute statistics (not recommended)** را انتخاب کنید.
- برای این که مشخص کنید شاخص در کدام گروه فایلی ایجاد شود، در قسمت **Filegroup** گروه فایلی مورد نظر را انتخاب کنید.

بازسازی شاخص

وقتی یک شاخص در پایگاه داده ایجاد می‌کنید، اطلاعات شاخص که توسط پرس و جوها استفاده می‌شود در صفحه‌های شاخص ذخیره می‌شوند. صفحه‌های شاخص متوالی بوسیله اشاره‌گرها مانند حلقه‌های زنجیر به یکدیگر متصل می‌گردند.

بازسازی یک شاخص در SQL Server وقتی مؤثر است که به جای حذف شاخص قبلی و ساختن مجدد آن، شاخص اولیه را در یک مرحله مجدداً ایجاد نمایید.

اگر یک شاخص خوشه‌ای را حذف و دوباره آن را ایجاد کنید موجب می‌شود که تمام شاخص‌های غیرخوشه‌ای دوبار حذف و ایجاد شوند. بار اول وقتی است که شاخص خوشه‌ای حذف می‌گردد و بار دوم وقتی است که آن را ایجاد می‌کنید. بنابراین حذف شاخص قبلی و ایجاد دوباره آن برای بازسازی، پرهزینه خواهد بود.

برای بازسازی شاخص از دستورات Transact -SQL استفاده می‌شود. اگر بخواهید تمام شاخص‌های یک جدول را بازسازی کنید از دستور DBCC DBREINDEX استفاده کنید.

برای تغییر یک شاخص نیز همانند ایجاد شاخص عمل می‌شود با این تفاوت که در صفحه Manage Indexes یک شاخص را انتخاب کرده و بر روی Edit کلیک می‌کنیم. سایر موارد همانند ایجاد یک شاخص می‌باشد.

تغییر نام شاخص

تغییر نام یک شاخص آن را بازسازی نمی‌کند بلکه صرفاً نام آن را عوض می‌کند. دو جدول می‌توانند شاخصی با نام یکسان داشته باشند اما یک جدول نمی‌تواند دو شاخص با یک نام داشته باشد.

مشاهده شاخص ها

پس از ایجاد شاخص‌ها بر روی جدول‌ها ممکن است به اطلاعاتی در مورد شاخص‌ها احتیاج داشته باشید از قبیل کل فضای استفاده شده از پایگاه داده توسط یک شاخص یا نوع شاخص‌های یک جدول.

مشاهده تمام شاخص‌های پایگاه داده

۱- یک گروه سرویس‌دهنده را باز کنید سپس یک سرویس‌دهنده را باز کنید.
۲- **Databases** را باز کنید و بر روی پایگاه داده مورد نظر کلیک کنید.
۳- برای مشاهده اطلاعات مربوط به شاخص‌های پایگاه داده، روی **Tables & Indexes** کلیک کنید.

حذف شاخص

حذف شاخص خوشه‌ای به زمان نیاز دارد زیرا تمام شاخص‌های غیرخوشه‌ای جدول مربوطه باید بازسازی شود. برای حذف شاخصی که توسط محدودیت **PRIMARY KEY** یا **UNIQUE** ایجاد شده است باید محدودیت مربوطه را حذف کرد. با حذف یک جدول، شاخص‌های ایجاد شده بر روی آن (اعم از دائم یا موقت) به طور خودکار حذف خواهند شد.

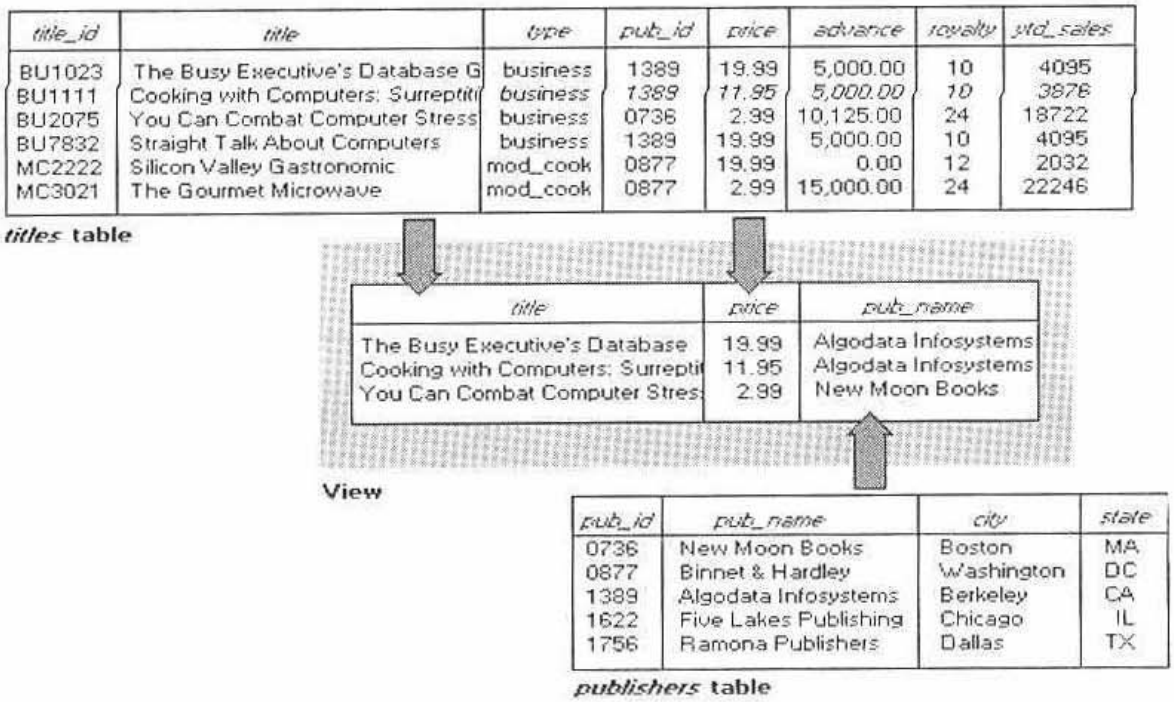
حذف شاخص

۱- یک گروه سرویس‌دهنده را باز کنید سپس یک سرویس‌دهنده را باز کنید.
۲- **Databases** را باز کنید. پایگاه داده‌ای که جدول در آن قرار دارد را باز کنید و سپس بر روی **Tables** کلیک کنید.
۳- در پانل **details** بر روی جدول رایت کلیک کرده و **All Tasks** را انتخاب کنید. سپس بر روی **Manage Indexes** کلیک کنید.
۴- در قسمت **Existing indexes** شاخص مورد نظر را انتخاب کرده و بر روی **Delete** کلیک کنید.
۵- عمل حذف را تأیید کنید.

۸-۴- ایجاد و مدیریت دید

دید، یک جدول مجازی است که محتوای آن توسط یک پرس و جو تعریف می‌شود. همانند جدول، دید نیز شامل سطرها و ستون‌ها است. شکل زیر، یک دید را که بر

اساس دو جدول به وجود آمده است نشان می‌دهد:



title_id	title	type	pub_id	price	advance	royalty	ytd_sales
BU1023	The Busy Executive's Database G	business	1389	19.99	5,000.00	10	4095
BU1111	Cooking with Computers: Surreptiti	business	1389	11.95	5,000.00	10	3876
BU2075	You Can Combat Computer Stress	business	0736	2.99	10,125.00	24	18722
BU7832	Straight Talk About Computers	business	1389	19.99	5,000.00	10	4095
MC2222	Silicon Valley Gastronomic	mod_cook	0877	19.99	0.00	12	2032
MC3021	The Gourmet Microwave	mod_cook	0877	2.99	15,000.00	24	22246

titles table

title	price	pub_name
The Busy Executive's Database	19.99	Algodata Infosystems
Cooking with Computers: Surrepti	11.95	Algodata Infosystems
You Can Combat Computer Stres	2.99	New Moon Books

View

pub_id	pub_name	city	state
0736	New Moon Books	Boston	MA
0877	Binnet & Hardley	Washington	DC
1389	Algodata Infosystems	Berkeley	CA
1622	Five Lakes Publishing	Chicago	IL
1756	Ramona Publishers	Dallas	TX

publishers table

«شکل ۲۵-۴: یک دید بر پایه دو جدول»

موارد کاربرد دیدها عبارتند از:

- تمرکز بر روی داده‌های خاص: دیدها به کاربران اجازه می‌دهند بر روی داده‌هایی که به آن نیاز دارند، متمرکز گردند. بنابراین داده‌های غیر ضروری می‌توانند از دید خارج گردند. دیدها امنیت داده‌ها را نیز افزایش می‌دهند زیرا کاربران فقط می‌توانند داده‌هایی را رؤیت نمایند که در دید تعریف شده است.
- سادگی کار با داده‌ها: می‌توان پیوندها، پرتوها، عملیات انتخاب و اجتماع را به عنوان یک دید تعریف کرد تا کاربران مجبور نباشند هر بار تمام شرایط لازم را روی داده‌ها تعریف کنند.
- سفارشی کردن داده‌ها: دیدها به کاربران اجازه می‌دهند تا داده‌ها را به روش‌های مختلف رؤیت نمایند.
- **Import** و **Export** کردن داده‌ها: دیدها می‌توانند برای مبادله داده‌ها با سایر برنامه‌های کاربردی مورد استفاده قرار گیرند.
- ترکیب داده‌های مجزا.

وقتی یک دید ایجاد می‌کنید، نام آن باید در بین نام جدول‌ها و دیدهای دیگری که کاربر مورد نظر مالک آنها می‌باشد، منحصربفرد باشد. دیدها را می‌توان بر روی دیدهای دیگر یا رویه‌هایی که به دیدها ارجاع می‌کنند، ایجاد کرد. دیدها حداکثر تا ۳۲ سطح می‌توانند تودرتو باشند. قاعده‌ها، پیش فرض‌ها و **Trigger**ها را نمی‌توان بر روی دید ایجاد کرد. پرس و جویی که دید را تعریف می‌کند نمی‌تواند شامل عبارات **ORDER BY**، **COMPUTE** یا **COMPUTE BY** یا کلمهٔ کلیدی **INTO** باشد. همچنین نمی‌توان بر روی دید، شاخص ایجاد کرد.

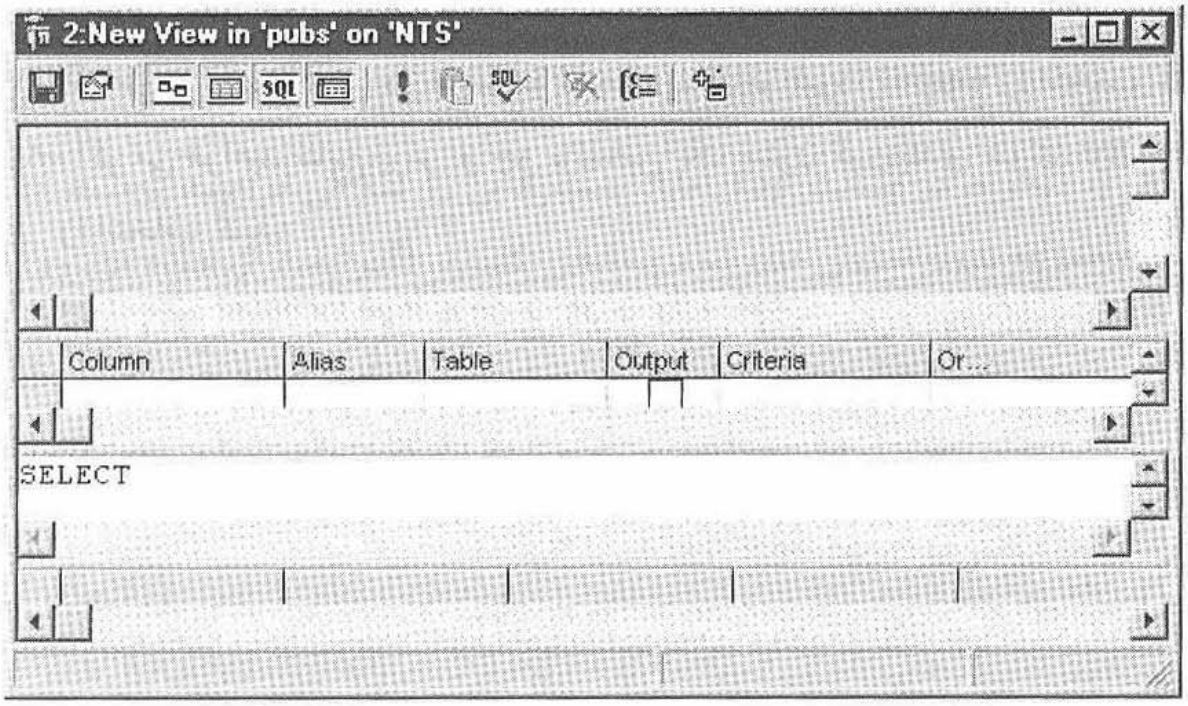
ایجاد یک دید

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- **Databases** را باز کنید. پایگاه داده‌ای را که می‌خواهید دید را در آن ایجاد نماید، باز کنید.

۳- روی **Views** رایت کلیک کنید و سپس ... **New View** را انتخاب کنید.

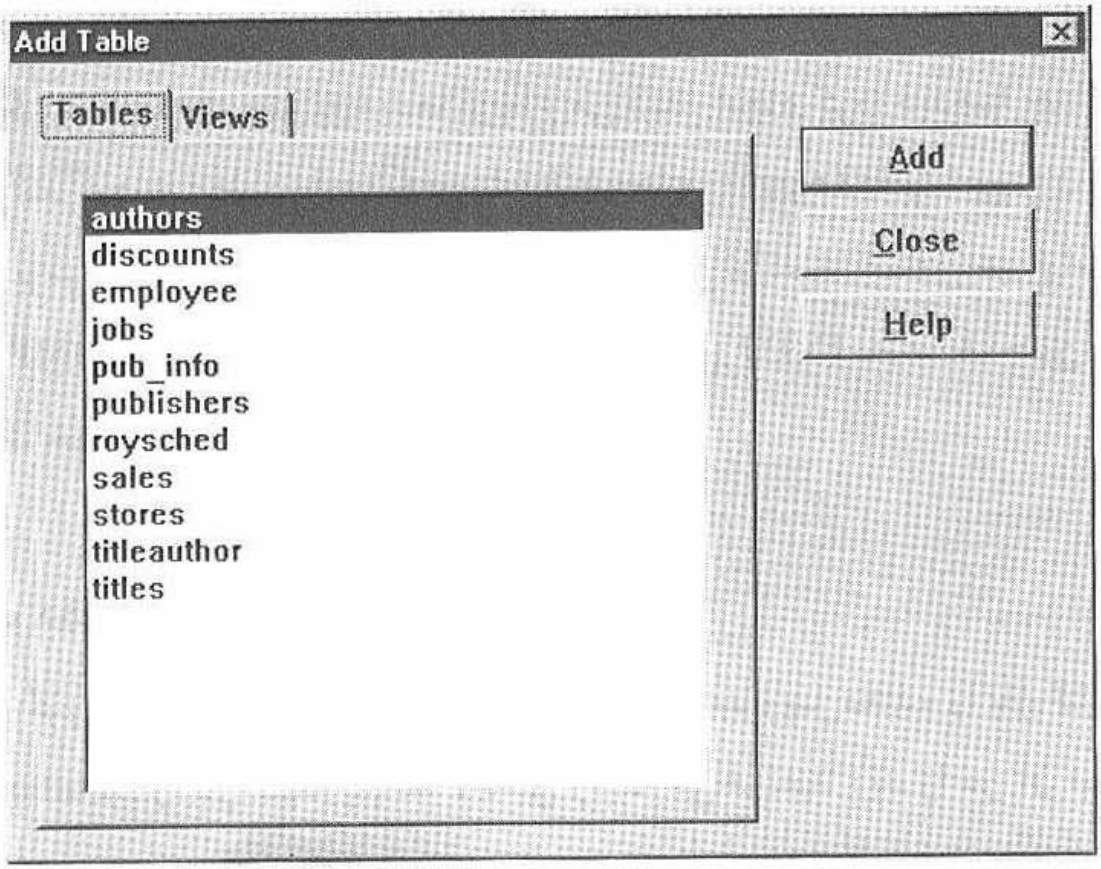
۴- در پانل دیاگرام رایت کلیک کنید و سپس ... **Add Table** را انتخاب کنید.



«شکل ۲۶-۴: ایجاد یک دید»

۵- در دکمه‌های **Tables** یا **Views**، جدول یا دید مورد نظر را انتخاب کرده و سپس

روی Add کلیک کنید. این کار را برای هر جدول یا دیدی که می‌خواهید به دید جدید اضافه شود تکرار کنید.



«شکل ۲۷-۴: افزودن جدول یا دید به دید جدید»

۶- در قسمت Column از پانل مشبک، ستون‌هایی را که می‌خواهید در دید به آن ارجاع نمایید انتخاب کنید.

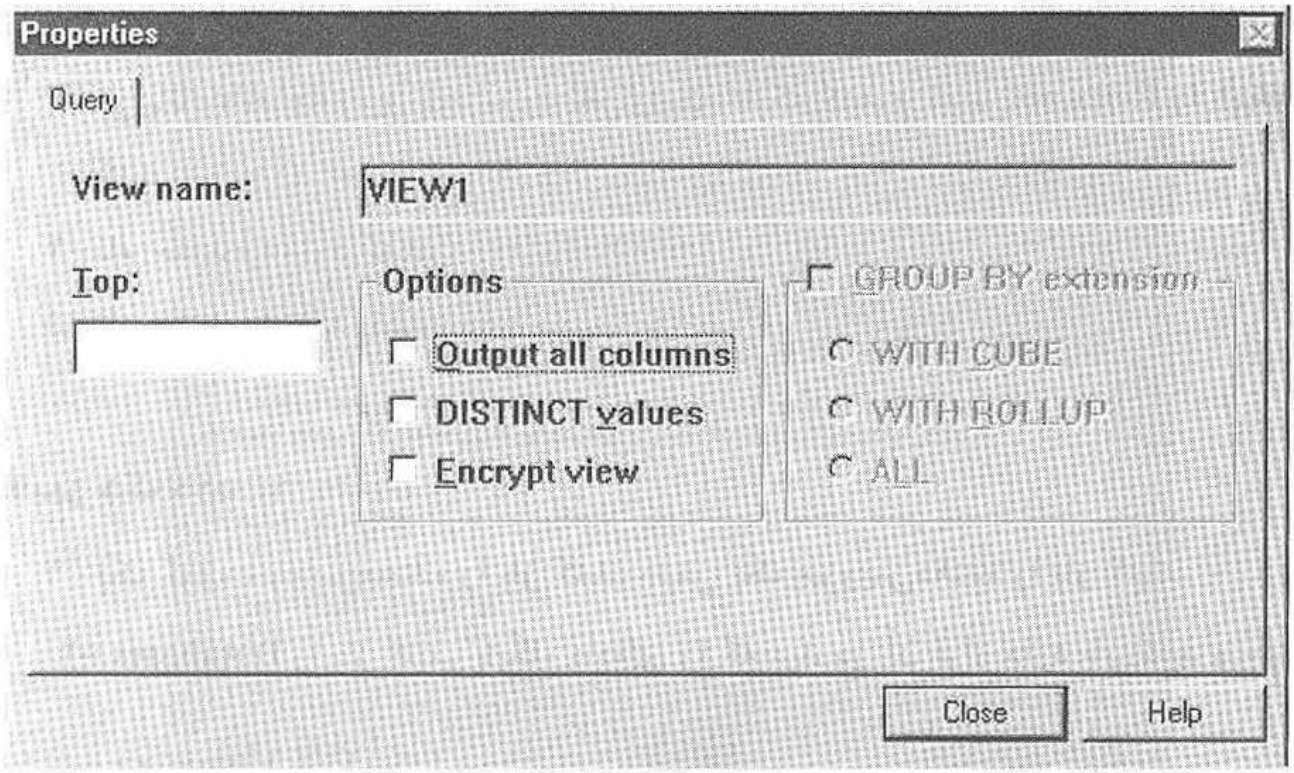
۷- اگر می‌خواهید ستونی در مجموعه نتیجه دید ظاهر شود، *output* را انتخاب کنید.

۸- اگر می‌خواهید نتایج پرس و جو براساس یک ستون گروه‌بندی شوند، Group By را انتخاب کنید.

۹- در قسمت Criteria شرط پرس و جو را وارد کنید.

۱۰- در قسمت or، شرایط دیگر پرس و جو را وارد کنید.

۱۱- در پانل مشبک رایت کلیک کنید و سپس properties را انتخاب کنید.



«شکل ۲۸-۴: انتخاب گزینه‌های اختیاری»

۱۲- به دلخواه موارد زیر را انتخاب کنید:

- **output all columns** را انتخاب کنید تا تمام ستون‌های موجود در دید در مجموعه نتیجه ظاهر شوند.
- **DISTINCT Values** را انتخاب کنید تا سطرهای تکراری از مجموعه نتیجه حذف شود.
- **Encrypt View** را انتخاب کنید تا تعریف دید رمزگذاری شود.

۱۳- به دلخواه، در قسمت **Top** تعداد سطرهایی را که می‌خواهید توسط دید در مجموعه نتیجه نمایش داده شود مشخص کنید. اگر بعد از عدد مزبور کلمه **PERCENT** را تایپ کنید درصدی از تعداد سطرها در مجموعه نتیجه ظاهر می‌شود.

۱۴- در پانل دیاگرام رایت کلیک کنید. برای ذخیره دید، **Save** را انتخاب کنید. برای دیدن مجموعه نتیجه، **Run** را انتخاب کنید.

تغییر نام یک دید

۱- یک گروه سرویس‌دهنده را باز کنید سپس یک سرویس‌دهنده را باز کنید.

۲- **Databases** را باز کنید. پایگاه داده‌ای را که دید در آن قرار دارد باز کنید و سپس

روی Views کلیک کنید.

۳- در پانل details روی دید مورد نظر رایت کلیک کنید و سپس Rename را انتخاب کنید.

۴- نام جدید را وارد کنید.

۵- نام جدید را تأیید کنید.

تغییر یک دید

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- Databases را باز کنید. پایگاه داده‌ای را که دید در آن قرار دارد باز کنید و سپس روی Views کلیک کنید.

۳- در پانل details روی دید مورد نظر رایت کلیک کنید و سپس Design view را انتخاب کنید.

۴- مراحلی را که در ایجاد دید بررسی شد دنبال نمایید.

مشاهده اطلاعات در مورد یک دید

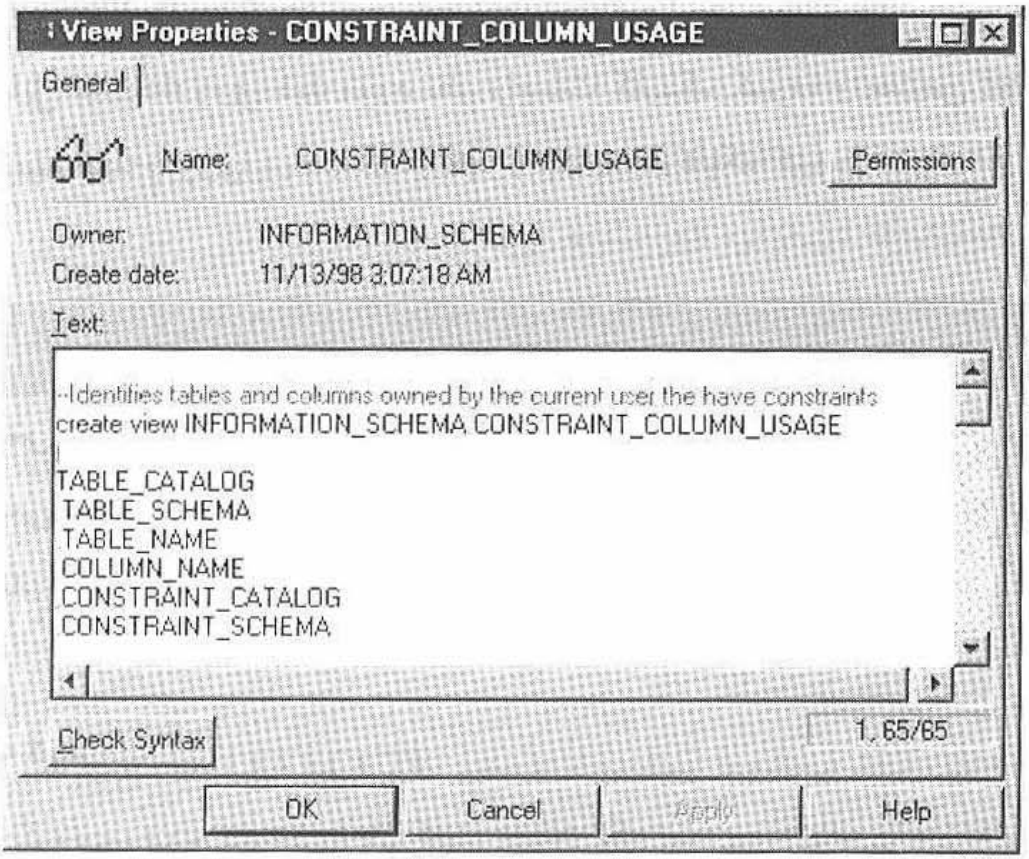
اگر دیدی رمزگذاری نشده باشد، می‌توانید اطلاعاتی در مورد تعریف دید بدست آورید:

بدست آوردن اطلاعات در مورد یک دید

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- Databases را باز کنید، پایگاه داده‌ای که دید در آن قرار دارد باز کنید و سپس روی Views کلیک کنید.

۳- در پانل details روی دید مورد نظر رایت کلیک کرده و سپس properties را انتخاب کنید.



«شکل ۲۹-۴: خصوصیات یک دید»

مشاهده داده‌های یک دید

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- **Databases** را باز کنید، پایگاه داده‌ای را که دید در آن قرار دارد باز کنید و سپس روی **Views** کلیک کنید.

۳- در پانل **details** روی دید رایت کلیک کنید. **Open View** را انتخاب کرده و سپس روی **Return All Rows** کلیک کنید تا تمام داده‌هایی که توسط دید تعریف شده است باز گردانده شود.

۴- برای این که مشخص کنید چه تعداد از سطرها باز گردانده شود، **Return top...** را انتخاب کنید.

مشاهده وابستگی‌های یک دید

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- **Databases** را باز کنید، پایگاه داده‌ای را که دید در آن قرار دارد باز کنید و سپس

روی Views کلیک کنید.

۳- در پانل details روی دید مورد نظر رایت کلیک کرده و سپس All Tasks را انتخاب کنید و روی ...Display Dependencies کلیک کنید.

حذف دید

وقتی یک دید حذف می‌شود، جدول‌ها و داده‌هایی که دید بر مبنای آنها قرار دارد متأثر نمی‌شوند. پرس وجوهایی که از شئ‌هایی استفاده می‌کنند که به دید حذف شده وابسته هستند، قابل اجرا نخواهند بود.

حذف یک دید

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- Databases را باز کنید، پایگاه داده‌ای را که دید در آن قرار دارد باز کنید و سپس روی Views کلیک کنید.

۳- در پانل details روی دید مورد نظر رایت کلیک کرده و سپس Delete را انتخاب کنید.

۴- برای مشاهده این که حذف دید چه تأثیری بر پایگاه داده خواهد داشت، روی Show Dependencies کلیک کنید.

۵- روی Drop All کلیک کنید.

۹-۴- ایجاد و مدیریت رویه‌های ذخیره شده

وقتی با SQL Server یک برنامه کاربردی تولید می‌کنید، زبان برنامه‌نویسی Transact-SQL رابط اصلی برنامه‌نویسی بین برنامه کاربردی شما و پایگاه داده خواهد بود. وقتی از برنامه‌های Transact- SQL استفاده می‌کنید، دو روش برای ذخیره و اجرای برنامه‌ها وجود دارد. می‌توانید برنامه‌ها را به طور محلی ذخیره کرده و برنامه‌های کاربردی ایجاد کنید تا دستورات را به SQL Server ارسال کنند و نتایج را پردازش کنند. و یا این که می‌توانید برنامه‌ها را به صورت رویه‌های ذخیره شده در SQL Server ذخیره کنید و

برنامه‌های کاربردی ایجاد کنید که رویه‌های ذخیره شده را اجرا کرده و نتایج را پردازش می‌کنند.

رویه‌های ذخیره شده در SQL Server همانند رویه‌ها در سایر زبان‌های برنامه‌نویسی هستند که در موارد زیر مشابه می‌باشند:

- پارامترهای ورودی را دریافت کرده و مقادیری را به شکل پارامترهای خروجی به رویه فراخواننده باز می‌گردانند.
- دارای دستورات برنامه‌نویسی است که عملیاتی را در پایگاه داده انجام می‌دهد، از جمله فراخوانی سایر رویه‌ها.
- یک مقدار وضعیتی را به رویهٔ فراخواننده باز می‌گردانند که نشانه موفقیت یا عدم موفقیت است.

رویه‌های ذخیره شده با استفاده از دستور EXECUTE اجرا می‌شوند.

رویه‌های ذخیره شده در SQL Server نسبت به برنامه‌های Transact-SQL که به طور محلی در کامپیوتر سرویس‌گیرنده ذخیره می‌شوند دارای مزیت‌هایی می‌باشند که عبارتند از:

- امکان برنامه‌نویسی پیمانه‌ای: می‌توانید یک بار رویه را ایجاد کنید و آن را در پایگاه داده ذخیره کنید و در برنامه چندین بار آن را فراخوانی کنید.
- رویه‌های ذخیره شده توسط برنامه نویسان پایگاه داده ایجاد می‌شوند.
- امکان اجرای سریع‌تر: در مواردی که یک عمل به مقدار زیادی کد Transact-SQL نیاز دارد و یا مکرراً اجرا می‌شود، رویه‌های ذخیره شده سریع‌تر عمل می‌کنند. وقتی این رویه‌ها ایجاد می‌شوند، تجزیه و بهینه می‌گردند و پس از این که برای اولین بار فراخوانی شوند، در حافظه باقی مانده و فراخوانی‌های بعدی از نسخه درون حافظه استفاده خواهد کرد. دستورات Transact-SQL که به طور مکرر از سرویس‌گیرنده‌ها ارسال می‌شوند هر بار قبل از اجرا، کامپایل شده و بهینه می‌گردند و لذا زمان بیشتری را مصرف می‌کنند.
- کاهش ترافیک شبکه: عملی که نیازمند صدها خط کد Transact-SQL است می‌تواند از طریق یک دستور منفرد انجام می‌شود و دیگر لازم به ارسال صدها خط برنامه از طریق شبکه نیست.

۵- در قسمت Path، مسیر کتابخانه پیوند پویایی (DLL) را که حاوی رویه ذخیره شده توسعه یافته است وارد کنید.

ایجاد رویه ذخیره شده

بسیاری از فعالیت‌های مدیریتی شما در SQL Server به کمک نوع خاصی از رویه‌ها به نام رویهٔ ذخیره شده سیستمی انجام می‌گیرد. رویه‌های ذخیره شده سیستمی در پایگاه داده master ایجاد و ذخیره می‌شوند و دارای پیشوند _SP هستند. اکیداً توصیه می‌شود که نام رویه‌های ذخیره شده را با پیشوند _SP همراه نکنید زیرا SQL Server همواره برای یافتن رویه‌های ذخیره شده‌ای که نام آنها با _SP شروع می‌شود ابتدا پایگاه داده master را جستجو می‌کند. اگر یک رویه ذخیره شده که توسط کاربر ایجاد شده است با یک رویه ذخیره شدهٔ سیستمی همنام باشد، رویه‌ای که توسط کاربر ایجاد شده است هرگز اجرا نخواهد شد.

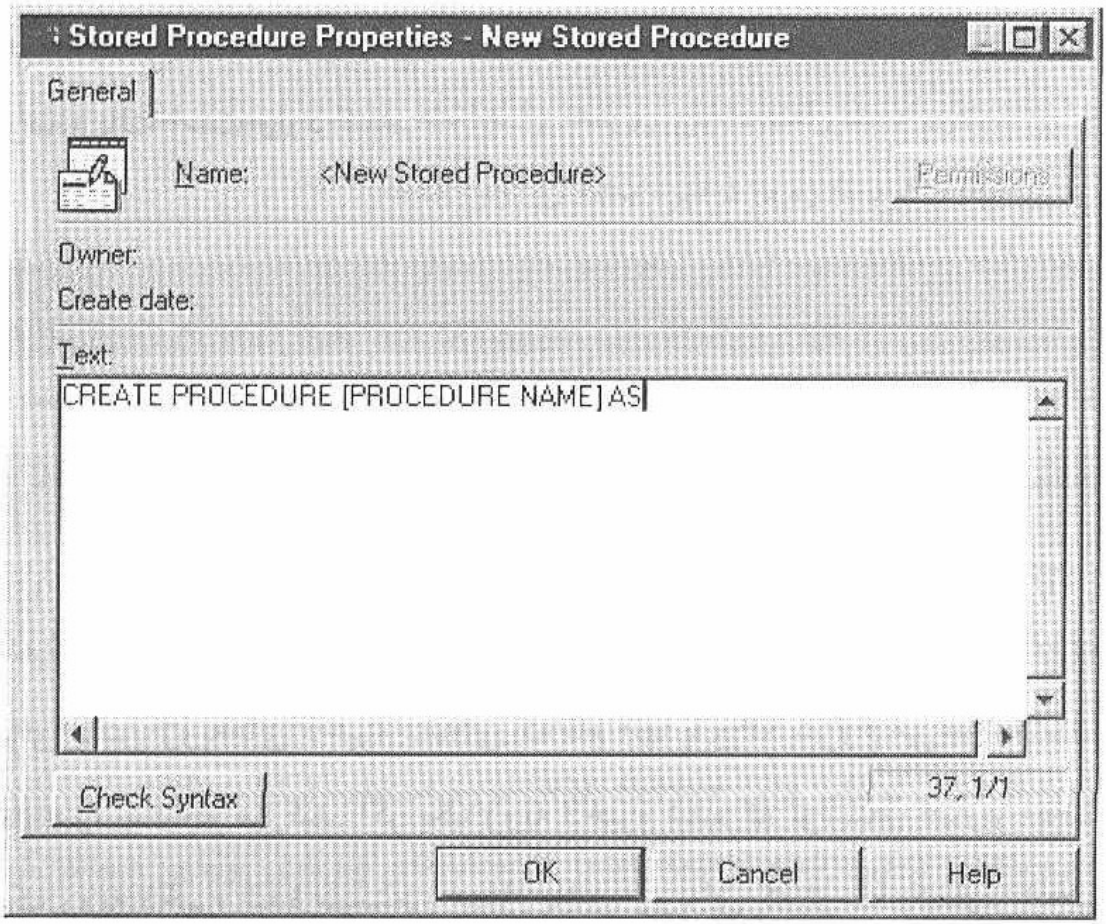
ایجاد یک رویه ذخیره شده

۱- یگ گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- Databases را باز کنید. پایگاه داده‌ای را که می‌خواهید رویه ذخیره شده در آن ایجاد نمایید باز کنید.

۳- روی Stored Procedure رایت کلیک کنید. سپس روی ... Stored procedure کلیک کنید.

۴- متن رویه ذخیره شده را وارد کنید. برای دندانه‌ای کردن، CTRL+TAB را فشار دهید.



«شکل ۳۱-۴: ایجاد رویه ذخیره شده»

۵- برای آزمایش درستی شکل نحوی، روی **Check Syntax** کلیک کنید.

۶- برای تنظیم حقوق ویژه، روی **permissions** کلیک کنید.

اجرای خودکار رویه‌های ذخیره شده

هنگامی که رویه‌های ذخیره شده‌ای را برای اجرای خودکار مشخص می‌کنید، هر بار که **SQL Server** اجرا می‌شود، این رویه‌ها نیز اجرا می‌گردند. رویه‌های ذخیره شده عمدتاً در موارد زیر به کار می‌روند:

- انجام منظم یک سری از عملیات.
- اجرای رویه ذخیره شده به عنوان یک فرآیند پس زمینه به طور دائمی.
- انجام وظایف سیستمی یا نگهداری توسط رویهٔ ذخیره شده در **tempdb**، مثل ایجاد یک جدول موقت عمومی. در نتیجه در این حالت جدول موقت همیشه وجود خواهد داشت و بازسازی **tempdb** هر بار که **SQL Server** آغاز می‌شود تغییری در آن به وجود نخواهد آورد.

اگرچه رویه‌های ذخیره شده برای اجرای خودکار تک تک تنظیم می‌شوند،

اما گزینه پیکربندی **Scan for startup procs** را می‌توان به گونه‌ای تنظیم کـرد کـه از اجرای خودکار تمام رویه‌های ذخیره شده هنگام آغاز **SQL Server** جلوگیری کرد.

اصلاح و تغییر نام رویه‌های ذخیره شده

اگر می‌خواهید دستورات یا پارامترهای یک رویه ذخـیره شـده را تغیـیر دهیـد، می‌توانید آن رویه‌ها را حذف و مجدداً ایجاد کنید و یا این که آن را به یکباره تغییر دهیـد. در حالت اول تمام حقوق ویژه مرتبط با رویه از بین خواهـد رفـت. در حـالت دوم فقـط تعریف رویه یا پارامترها تغییر می‌کند ولی حقـوق ویژه مرتبـط بـا آن بـدون تغیـیر بـاقی می‌ماند.

یک رویه ذخیره شده را می‌توان به گونه‌ای اصـلاح کـرد کـه تعریـف آن رمـز گذاری شود و یا در هر بار اجرا مجدداً کامپایل شود.

همچنین می‌توانید نام یک رویه ذخیره شده را تغییر دهید. تغییر نام یا تعریف یک رویه ذخیره شده باعث می‌شود که شئ‌های وابسته اجرا نشوند. پس از تغییر رویه، شئ‌های وابسته به آن نیز باید بهنگام سازی شوند تا تغییرات اعمال شده در رویه ذخـیره شـده در آنها نیز منعکس گردد.

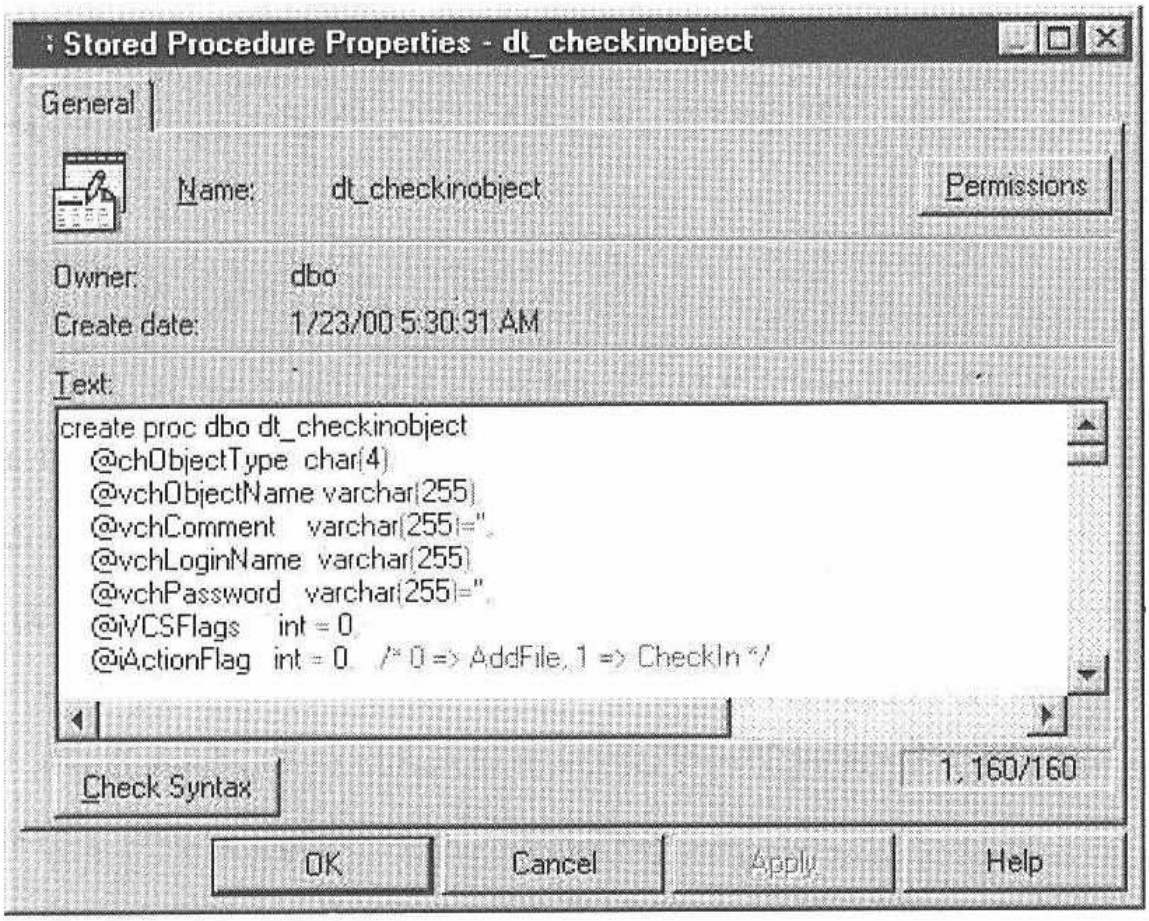
تغییر یک رویه ذخیره شده

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- **Databases** را باز کنید. پایگاه داده‌ای را که رویـه در آن قـرار دارد بـاز کـرده و سپس روی **Stored Procedures** کلیک کنید.

۳- در پانل **details** روی رویه ذخیره شده رایت کلیک کنیـد و سـپس **properties** را انتخاب کنید.

۴- در قسمت **Text** متن رویه را تغییر دهید. برای دندانه‌ای کردن متـن، **CTRL+TAB** را فشار دهید.



«شکل ۳۲-۴: تغییر رویه ذخیره شده»

۵- برای آزمایش درستی شکل نحوی، روی **Check Syntax** کلیک کنید.

۶- برای تغییر حقوق ویژه روی **Permissions** کلیک کنید.

تغییر نام یک رویه ذخیره شده

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- **Databases** را باز کنید. پایگاه داده‌ای را که رویه در آن قرار دارد باز کنید و سپس روی **Stored Procedures** کلیک کنید.

۳- در پانل **details** روی رویهٔ ذخیره شده رایت کلیک کنید و سپس **Rename** را انتخاب کنید.

۴- نام جدید رویه را وارد کنید.

۵- نام جدید را تأیید کنید.

مشاهده رویه‌های ذخیره شده

رویه‌های ذخیره شده سیستمی متعددی وجود دارند که با استفاده از داده‌های

جدول‌های سیستمی اطلاعاتی درباره رویه‌های ذخیره شده فراهم می‌کنند. شما می‌توانید:

- دستورات Transact-SQL را که برای ایجاد یک رویه ذخیره شده مورد استفاده قرار گرفته‌اند ببینید.
- اطلاعاتی درباره یک رویه ذخیره شده از قبیل مالک آن، زمان ایجاد آن و پارامترهای مربوط را بدست آورید.
- شئ‌های مورد استفاده بوسیله یک رویه ذخیره شده را مشاهده کنید.

مشاهده وابستگی‌های یک رویه ذخیره شده

۱- یگ گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- Databases را باز کنید. پایگاه داده‌ای که رویهٔ ذخیره شده در آن قرار دارد باز کنید و سپس روی Stored procedures کلیک کنید.

۳- در پانل details روی رویه ذخیره شده رایت کلیک کنید و All Tasks را انتخاب کنید. سپس روی ... Display Dependencies کلیک کنید.

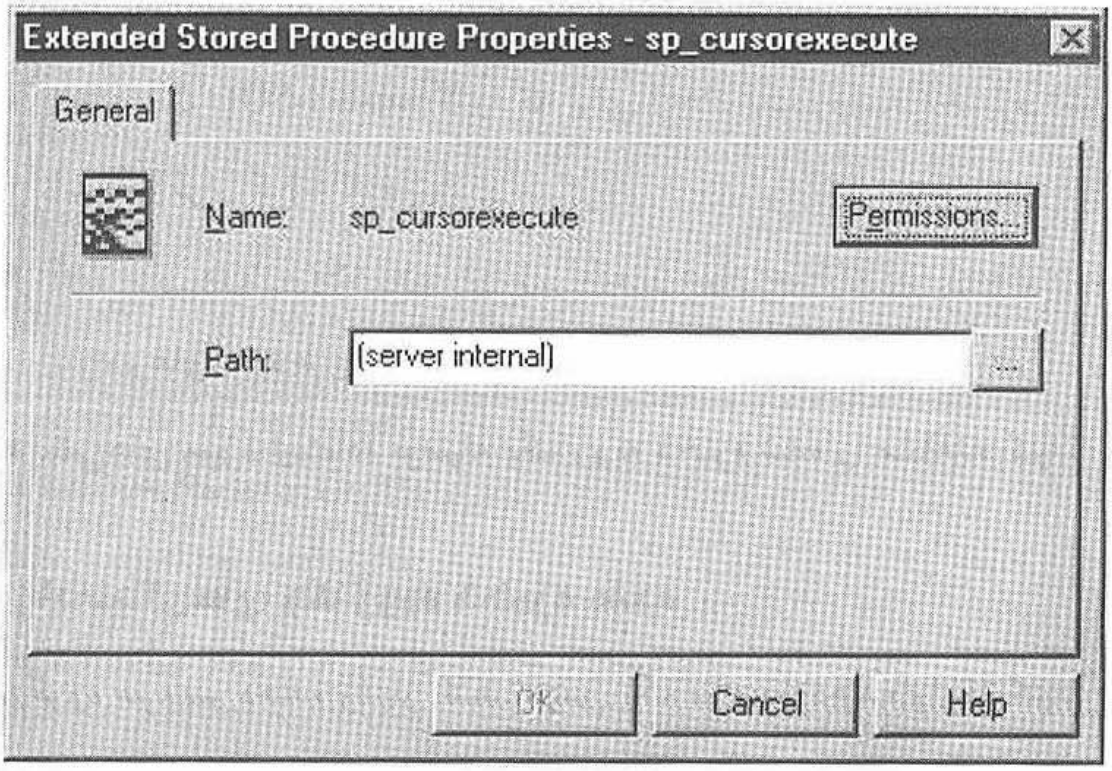
مشاهده اطلاعات مربوط به یک رویه ذخیره شده توسعه یافته

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- Database را باز کنید. پایگاه داده master را باز کنید و سپس روی Extended Stored Procedures کلیک کنید.

۳- در پانل details روی رویهٔ ذخیره شده توسعه یافته رایت کلیک کنید و سپس properties را انتخاب کنید.

۴- به دلخواه، روی permissions کلیک کنید تا حقوق ویژه رویهٔ ذخیره شده توسعه یافته *را مشاهده یا تنظیم کنید.*



«شکل ۳۳-۴: مشاهده اطلاعات رویه ذخیره شده توسعه یافته»

حذف رویهٔ ذخیره شده

اگر یک رویه ذخیره شده که توسط رویهٔ ذخیره شده دیگری فراخوانی می‌شود حذف گردد، SQL Server هنگام اجرای رویه فراخواننده، یک خطا باز می‌گرداند.

حذف یک رویه ذخیره شده

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- Databases را باز کنید. پایگاه داده‌ای که رویهٔ ذخیره شده در آن قرار دارد باز کنید و سپس روی Stored Procedures کلیک کنید.

۳- در پانل details روی رویه ذخیره شده رایت کلیک کنید و سپس روی Delete کلیک کنید.

۴- برای مشاهده این که حذف رویه ذخیره شده چه تأثیری بر پایگاه داده دارد، روی Show Dependencies کلیک کنید.

۵- روی Drop All کلیک کنید.

۱۰-۴- برنامه نویسی رویه‌های ذخیره شده

تقریباً تمام کدهای Transact-SQL که در یک دسته قرار می‌گیرند می‌توانند برای ایجاد یک رویه ذخیره شده مورد استفاده قرار گیرند.

قواعد برنامه‌نویسی رویه‌های ذخیره شده به صورت زیر است:

- تعریف CREATE PROCEDURE به خودی خود می‌تواند شامل هر تعداد و هر نوع از دستورات SQL باشد به جز دستورات زیر که نمی‌توانند در هیچ جای یک رویه ذخیره شده به کار روند:

CREATE DEFAULT **CREATE TRIGGER**
CREATE PROCEDURE **CREATE VIEW**
CREATE RULE

- سایر شئ‌های پایگاه داده می‌توانند در داخل یک رویه ذخیره شده ایجاد شوند.
- می‌توانید در داخل یک رویه ذخیره شده به جدول‌های موقت ارجاع کنید.
- اگر یک جدول موقت خصوصی در داخل یک رویه ذخیره شده ایجاد شود، آن جدول فقط برای همان رویه قابل استفاده است و هنگام خروج از رویه از بین می‌رود.
- اگر یک رویه دخیره شده رویهٔ ذخیره شده دیگری را فراخوانی کند، رویه ذخیره شده فراخوانده شده می‌تواند به تمام شئ‌هایی که بوسیله رویهٔ ذخیره شده اولیه ایجاد شده‌اند از جمله جدول‌های موقت دسترسی داشته باشد.
- حداکثر تعداد پارامترها در یک رویه ذخیره شده، ۱۰۲۴ است.
- حداکثر تعداد متغیرهای محلی در یک رویهٔ ذخیره شده فقط به حافظه موجود بستگی دارد.
- حداکثر اندازه یک رویه ذخیره شده، MB ۱۲۸ است به شرطی که حافظه کافی وجود داشته باشد.

مقید کردن نام‌ها درون رویهٔ ذخیره شده

در یک رویه ذخیره شده اگر اسامی شئ‌هایی که در دستوراتی نظیر SELECT یا INSERT استفاده می‌شوند به نام کاربری مقید نشده باشند، آن مالک رویهٔ ذخیره شده به

طور پیش فرض مورد استفاده قرار می‌گیرد. اگر یک کاربر که رویه ذخیره شده‌ای را ایجاد کرده است اسامی جدول‌هایی را که در دستورات SELECT، INSERT، UPDATE یا DELETE درون رویه ذخیره شده به آنها ارجاع شده است مقید نکرده باشد، دستیابی به آن جدول‌ها از طریق رویه ذخیره شده به طور پیش فرض منحصر به ایجاد کننده رویه است. اسامی شیءهایی که در دستورات ALTER TABLE، CREATE TABLE، DROP TABLE، TAUNCATE TABLE، CREATE INDEX، DROP INDEX، UPDATE STATISTICS و DBCC به کار می‌روند باید با نام مالک آن شیء مقید شوند. به عنوان مثال، اگر Mary مالک جدول marytab باشد و بخواهد که کاربران دیگر نیز قادر باشند رویه ذخیره شده‌ای را که در آن از جدول او استفاده شده است اجرا کنند باید نام جدول خود را مقید نماید.

این قاعده از اهمیت بالایی برخوردار است، زیرا اسامی شیءها هنگام اجرای رویهٔ ذخیره شده تجزیه و تحلیل می‌شوند. اگر marytab مقید نشده باشد و کاربر دیگری به نام john بخواهد رویه را اجرا کند، SQL Server به دنبال جدولی به نام marytab می‌گردد که مالک آن john باشد.

رمزگذاری تعریف رویه

اگر رویه ذخیره شده‌ای ایجاد می‌کنید و می‌خواهید مطمئن باشید که هیچ کاربری نمی‌تواند تعریف رویه را مشاهده کند، از عبارت WITH ENCRYPTION استفاده کنید. در این صورت تعریف رویه به شکل غیر قابل خواندن ذخیره می‌شود. پس از رمزگذاری، تعریف رویه ذخیره شده توسط هیچ کس حتی مالک رویه ذخیره شده و مدیر سیستم قابل رمزگشایی یا مشاهده نخواهد بود.

مثال ۱- ایجاد یک رویه ذخیره شده که از پارامترها استفاده می‌کند.

در این رویه با دادن نام خانوادگی و نام یک مؤلف، عنوان و ناشر تمام کتاب‌های مربوط به آن مؤلف نمایش داده می‌شود.

```
CREATE PROC au_info @lastname varchar(40), @firstname varchar(20)
AS
SELECT au_lname, au_fname, title, pub_name
FROM authors INNER JOIN titleauthor ON authors.au_id = titleauthor.au_id
    JOIN titles ON titleauthor.title_id = titles.title_id
    JOIN publishers ON titles.pub_id = publishers.pub_id
WHERE au_fname = @firstname
     AND au_lname = @lastname
GO
```

مثال ۲- ایجاد یک رویه ذخیره شده که از مقادیر پیش فرض برای پارامترها استفاده می‌کند.

این رویه نام تمام مؤلفانی را نمایش می‌دهد که ناشر کتاب آنها به عنوان پارامتر ارسال می‌شود. اگر نام ناشر برای رویه مشخص نشود، رویه ذخیره شده نام مؤلفان مرتبط با ناشر **Algodata Infosystems** را نمایش می‌دهد.

```
CREATE PROC pub_info2 @pubname varchar(40) = 'Algodata Infosystems'
AS
SELECT au_lname, au_fname, pub_name
FROM authors a INNER JOIN titleauthor ta ON a.au_id = ta.au_id
    JOIN titles t ON ta.title_id = t.title_id
    JOIN publishers p ON t.pub_id = p.pub_id
WHERE @pubname = p.pub_name
```

مثال ۳- ایجاد یک رویه ذخیره شده که مقدار پیش فرض پارامتر را با یک مقدار صریح لغو می‌کند.

```
CREATE PROC showind2 @table varchar(30) = 'titles'
AS
SELECT TABLE_NAME = sysobjects.name,
     INDEX_NAME = sysindexes.name, INDEX_ID = indid
FROM sysindexes INNER JOIN sysobjects ON sysobjects.id = sysindexes.id
WHERE sysobjects.name = @table
```

مثال ۴- ایجاد یک رویه ذخیره شده که از مقدار **NULL** به عنوان مقدار پیش فرض پارامتر استفاده می‌کند.

مقدار پیش فرض یک پارامتر می‌تواند **NULL** باشد. در این حالت اگر کاربر مقداری را برای پارامتر تعیین نکند **SQL Server** رویه ذخیره شده را برحسب دستورات دیگر آن اجرا می‌کند.

```
CREATE PROC showind3 @table varchar(30) = NULL
AS IF @table IS NULL
  PRINT 'Give a table name'
ELSE
  SELECT TABLE_NAME = sysobjects.name,
      INDEX_NAME = sysindexes.name, INDEX_ID = indid
  FROM sysindexes INNER JOIN sysobjects
     ON sysobjects.id = sysindexes.id
  WHERE sysobjects.name = @table
```

مثال ۵- ایجاد یک رویه ذخیره شده که مقدار پیش فرض پارامتر آن شامل کاراکترهای ویژه است.

```
CREATE PROC showind4 @table varchar(30) = 'sys%'
AS SELECT TABLE_NAME = sysobjects.name,
      INDEX_NAME = sysindexes.name, INDEX_ID = indid
  FROM sysindexes INNER JOIN sysobjects
     ON sysobjects.id = sysindexes.id
  WHERE sysobjects.name LIKE @table
```

رویه‌های ذخیره شده تودرتو

رویه‌های ذخیره شده را تودرتو گویند اگر یک رویه ذخیره شده، رویه ذخیره

شده دیگری را فراخوانی نماید. رویه‌های ذخیره شده می‌توانند تا ۳۲ سطح تودرتو باشند. اگر تعداد سطوح لانه‌ای شدن از ۳۲ تجاوز کند، کل زنجیره رویه‌های ذخیره شده با شکست مواجه می‌شود.

مشخص کردن پارامترها

رویه‌های ذخیره شده از طریق پارامترهایش با برنامه فراخواننده ارتباط برقرار می‌کند. هر پارامتر دارای نام، نوع داده‌ای، مسیر و مقدار پیش فرض است.

نام هر پارامتر در رویهٔ ذخیره شده باید منحصربفرد باشد. نام پارامترها با کاراکتر @ آغاز می‌شود.

ارسال مقادیر به رویه ذخیره شده به دو شکل می‌تواند انجام شود: ذکر نام پارامتر به طور صریح و انتساب یک مقدار مناسب به آن و یا بدون ذکر نام و به همان ترتیبی که در دستور **CREATE PROCEDURE** تعریف شده‌اند. مثلاً اگر رویه ذخیره شده **my_proc** دارای سه پارامتر به نام‌های **@first**، **@second** و **@third** باشد، مقادیر ارسالی به رویه ذخیره شده به یکی از دو صورت زیر می‌تواند مشخص شود:

```
EXECUTE my_proc @Second=2 , @first =1, @third=3
EXECUTE my_proc 1,2,3
```

پارامتر رویهٔ ذخیره شده می‌تواند هر نوع داده‌ای در **SQL Server** را اختیار کند، اما نوع داده‌ای **cursor** فقط می‌تواند با پارامترهای نوع **Output** به کار روند.

هنگامی که رویه ذخیره شده توسط یک برنامه فراخوانی می‌شود، تمام پارامترهای ورودی را دریافت می‌کند.

```
CREATE PROCEDURE get_sales_for_title
@title varchar(80) -- This is the input parameter.
AS
-- Get the sales for the specified title.
SELECT "YTD_SALES" = ytd_sales
FROM titles
WHERE title = @title
RETURN
GO
```

بازگشت داده‌ها از یک رویه ذخیره شده

در SQL Server رویه‌های ذخیره شده به چهار شکل داده‌ها را باز می‌گردانند:

- پارامترهای خروجی که یا داده و یا یک متغیر کرزر را باز می‌گردانند.
- کدهای بازگشتی که همیشه یک مقدار صحیح هستند.
- مجموعه نتیجه هر دستور SELECT موجود در رویه ذخیره شده یا هر رویه ذخیره شدهٔ دیگری که بوسیله آن فراخوانی شده است.
- کرزر عمومی که می‌توان خارج از رویه به آن ارجاع کرد.

پارامترهای Output

این نوع پارامترها، مقدار جاری خود را در رویه ذخیره شده به برنامه فراخواننده نیز منتقل می‌کنند. برای ذخیره کردن مقدار پارامتر در متغیری که در برنامه فراخواننده نیز قابل استفاده باشد، باید از پارامتر نوع Output استفاده شود.

به عنوان مثال، رویهٔ زیر از یک پارامتر خروجی استفاده می‌کند.

```
CREATE PROCEDURE get_sales_for_title
@title varchar(80), -- This is the input parameter.
@ytd_sales int OUTPUT -- This is the output parameter.
AS
-- Get the sales for the specified title and
-- assign it to the output parameter.
SELECT @ytd_sales = ytd_sales
FROM titles
WHERE title = @title
RETURN
GO
```

کدهای بازگشتی

رویه ذخیره شده می‌تواند یک مقدار صحیح موسوم به کد بازگشت را که

نشان‌دهنده وضعیت اجرای رویه است، بازگرداند. کد بازگشتی در رویه ذخیره شده بوسیله دستور **RETURN** مشخص می‌شود. برای استفاده از مقدار کد بازگشتی در برنامه فراخواننده، هنگام اجرای رویه ذخیره شده این کد باید در یک متغیر ذخیره شود. به مثال زیر توجه کنید:

```
DECLARE @result int
EXECUTE @result=my_proc
```

مثال ۱- بازگرداندن کدهای بازگشتی مختلف برحسب نوع خطا.

رویه ذخیره شده زیر برای خطاهای مختلف کدهای بازگشتی ویژه‌ای را اختصاص می‌دهد.

0	اجرای موفقیت آمیز
1	مقدار پارامتر مشخص نشده است.
2	مقدار پارامتر مشخص شده نامعتبر است.
3	خطا هنگام گرفتن مقدار فروش.
4	برای عنوان، مقدار فروش **NULL** مجاز نیست.

```
CREATE PROCEDURE get_sales_for_title
-- This is the input parameter, with a default.
@title varchar(80) = NULL,
-- This is the output parameter.
@ytd_sales int OUTPUT
AS
-- Validate the @title parameter.
IF @title IS NULL
BEGIN
  PRINT "ERROR: You must specify a title value."
  RETURN(1)
END
ELSE
BEGIN
```

```
-- Make sure the title is valid.
  IF (SELECT COUNT(*) FROM titles
      WHERE title = @title) = 0
    RETURN(2)
END
-- Get the sales for the specified title and
-- assign it to the output parameter.
SELECT @ytd_sales = ytd_sales
FROM titles
WHERE title = @title
-- Check for SQL Server errors.
IF @@ERROR <> 0
BEGIN
  RETURN(3)
END
ELSE
BEGIN
-- Check to see if the ytd_sales value is NULL.
  IF @ytd_sales IS NULL
    RETURN(4)
ELSE
-- SUCCESS!!
  RETURN(0)
END
GO
```

مثال ۲- پردازش کدهای بازگشتی مختلف که از یک رویه ذخیره شده بازگردانده می‌شود.

```
-- Declare the variables to receive the output  value and return code -- of the procedure.
DECLARE @ytd_sales_for_title int, @ret_code INT
-- Execute the procedure with a title_id value
```

```
-- and save the output value and return code in variables.
EXECUTE @ret_code = get_sales_for_title
"Sushi, Anyone?",
@ytd_sales = @ytd_sales_for_title OUTPUT
-- Check the return codes.
IF @ret_code = 0
BEGIN
  PRINT "Procedure executed successfully"
-- Display the value returned by the procedure.
                PRINT 'Sales for "Sushi, Anyone?": ' +
CONVERT(varchar(6),@ytd_sales_for_title)
END
ELSE IF @ret_code = 1
   PRINT "ERROR: No title_id was specified."
ELSE IF @ret_code = 2
   PRINT "ERROR: An invalid title_id was specified."
ELSE IF @ret_code = 3
   PRINT "ERROR: An error occurred getting the ytd_sales."
GO
```

۱۱-۴- ایجاد و مدیریت Triggerها

Trigger نوع خاصی از رویه ذخیره شده است که هر وقت داده‌ها در جدول تغییر می‌کنند، به طور خودکار فراخوانی می‌شود. **Trigger**ها در پاسخ به دستورات **INSERT**، **UPDATE** یا **DELETE** فراخوانی می‌شوند. یک **Trigger** می‌تواند پرس و جویی روی جدول‌های دیگر انجام دهد و شامل دستورات پیچیده **Transact - SQL** باشد. **Trigger** و دستور العملی که باعث اجرای آن می‌شود به عنوان یک تراکنش منفرد در نظر گرفته می‌شود که می‌توان آن را از درون **Trigger** لغو کرد.

Triggerها در موارد زیر مفید می‌باشند:

- **Trigger**ها می‌توانند تغییرات را از طریق جدول‌های مرتبط در پایگاه داده به شکل آبشاری در آورند. به عنوان مثال، یک **Trigger** نوع **DELETE** روی

ستون title_id از جدول titles باعث می‌شود که سطرهای متناظر در جدول‌های دیگر نیز که دارای مقدار یکسان در ستون title_id هستند، حذف شوند.

- Triggerها می‌توانند تغییراتی را که جامعیت ارجاعی را خدشه‌دار می‌کند، لغو کرده و یا غیر مجاز کنند.
- Triggerها می‌توانند محدودیت‌هایی را اعمال کنند که از محدودیت‌هایی که توسط CHECK تعریف می‌شوند پیچیده‌تر باشند. برخلاف محدودیت‌های CHECK، Triggerها می‌توانند به ستون‌های جدول‌های دیگر نیز ارجاع کنند.
- چندین Trigger از یک نوع بر روی یک جدول امکان می‌دهد چندین عمل مختلف در پاسخ به یک دستورالعمل انجام شود.
- SQL Server از دو مکانیزم اصلی برای اعمال قواعد و جامعیت داده‌ها استفاده می‌کند: محدودیت‌ها و Triggerها، هر کدام از این دو دارای مزایای خاص خود هستند که در وضعیت‌های ویژه مفید می‌باشند. مزیت اصلی Triggerها این است که می‌توانند شامل منطق پردازشی پیچیده‌ای باشند که از دستورات Transact-SQL استفاده می‌کنند. بنابراین Triggerها می‌توانند تمام کارکردهای محدودیت‌ها را پشتیبانی کنند، هر چند که کاربرد Triggerها همیشه بهترین روش نیست.

جامعیت موجودیتی همیشه باید در پایین‌ترین سطح بوسیله شاخص‌هایی که قسمتی از محدودیت PRIMARY KEY و UNIQUE هستند و یا مستقل از محدودیت‌ها ایجاد شده‌اند، اعمال شوند. جامعیت میدانی باید از طریق محدودیت CHECK اعمال شود. جامعیت ارجاعی باید از طریق محدودیت FOREIGN KEY اعمال گردد.

اگر در مواردی ویژگی‌هایی که توسط محدودیت‌ها پشتیبانی می‌شوند نتوانند نیازهای عملیاتی برنامه‌های کاربردی را برآورده کنند، از Triggerها استفاده می‌شود.

نیازهای عملیاتی برنامه‌های کاربردی را برآورده کنند، از Triggerها استفاده می‌شود.

ایجاد Trigger

قبل از ایجاد Trigger موارد زیر را در نظر بگیرید:

- دستور CREATE TRIGGER باید اولین دستور در دسته باشد.
- جواز ایجاد Triggerها به طور پیش فرض به مالک جدول تعلق دارد و قابل انتقال به سایر کاربران نیست.
- Triggerها را نمی‌توان بر روی دید، جدول موقت یا جدول سیستم ایجاد کرد، اگرچه Triggerها می‌توانند به جدول‌های موقت یا دیدها ارجاع داشته باشند.

یک جدول می‌تواند شامل چند Trigger از یک نوع با نام‌های متفاوت باشد و هر یک از آنها می‌تواند اعمال مختلفی را انجام دهد. هر Trigger را می‌توان فقط به یک جدول اختصاص داد.

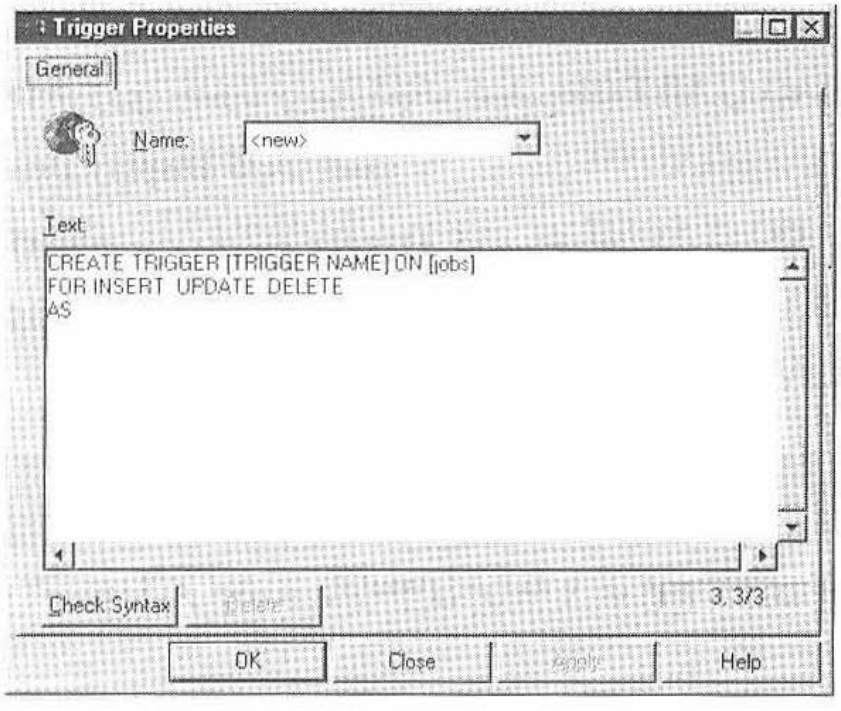
ایجاد یک Trigger

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس دهنده را باز کنید.

۲- Databases را باز کنید. پایگاه داده‌ای را که جدول در آن قرار دارد باز کنید و سپس روی Tables کلیک کنید.

۳- در پانل details روی جدول رایت کلیک کنید. All Tasks را انتخاب کرده و سپس روی ... Manage Triggers کلیک کنید.

۴- در قسمت Name روی new کلیک کنید.



۵- در قسمت Text، متن Trigger را وارد کنید. برای دندانه‌ای کردن متن Trigger، کلیدهای CTRL + TAB را فشار دهید.

۶- برای تست شکل نحوی، روی Check Syntax کلیک کنید.

اصلاح و تغییر نام Trigger

اگر می‌خواهید تعریف یک Trigger را تغییر دهید، Trigger را حذف کرده و دوباره ایجاد کنید و یا می‌توانید در یک مرحله Trigger موجود را دوباره تعریف کنید.

اگر نام شیء‌ای را که توسط Trigger ارجاع می‌شود تغییر دهید باید Trigger را طوری تغییر دهید که از نام جدید شئ استفاده کند. بنابراین قبل از تغییر نام یک شیء، ابتدا وابستگی‌های شئ را نمایش دهید تا مشخص شود آیا تغییر نام بر Triggerهای دیگر تأثیر می‌گذارد یا خیر.

برای اصلاح یک Trigger به همان شکل تعریف Trigger رفتار می‌شود، با این تفاوت که در ... Manage Trigger به جای new، نام Trigger مورد نظر را انتخاب می‌کنیم.

مشاهده وابستگی‌های یک Trigger

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.

۲- Databases را باز کنید. پایگاه داده‌ای را که جدول مورد نظر در آن قرار دارد باز کرده و روی Tables کلیک کنید.

۳- در پانل details روی جدول رایت کلیک کنید. All Tasks را انتخاب کرده و روی Display Dependencies کلیک کنید.

۴- در قسمت object، بر روی نام Trigger که می‌خواهید وابستگی‌های آن نمایش داده شود، کلیک کنید.

حذف Trigger

حذف یک Trigger بر جدول و داده‌های مرتبط با آن هیچ تأثیری ندارد. اما حذف

یک جدول به طور خودکار باعث حذف تمام Triggerهای آن جدول می‌شود.

حذف یک Trigger

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس‌دهنده را باز کنید.
۲- **Databases** را باز کنید. پایگاه داده‌ای که جدول در آن قرار دارد را باز کرده و روی **Tables** کلیک کنیا.
۳- در پانل **details** بر روی جدول رایت کلیک کنید. **All Tasks** را انتخاب کرده و روی **... Manage Triggers** کلیک کنید.
۴- در قسمت **Name** روی **Trigger** ای که می‌خواهید حذف کنید، کلیک کنید.
۵- روی **Delete** کلیک کنید.
۶- عمل حذف را تأیید کنید.

۱۲-۴- برنامه‌نویسی Triggerها

تقریباً تمام دستورات **Transact-SQL** که می‌توانند در یک دسته قرار گیرند، می‌توانند برای ایجاد یک **Trigger** مورد استفاده قرار گیرند، به جز دستورات زیر:

ALTER DATABASE	ALTER PROCEDURE	ALTER TABLE
ALTER TRIGGER	ALTER VIEW	CREATE DATABASE
CREATE DEFAULT	CREATE INDEX	CREATE PROCEDURE
CREATE RULE	CREATE SCHEMA	CREATE TABLE
CREATE TRIGGER	CREATE VIEW	DENY
DISK INIT	DISK RESIZE	DROP DATABASE
DROP DEFAULT	DROP INDEX	DROP PROCEDURE
DROP RULE	DROP TABLE	DROP TRIGGER
DROP VIEW	GRANT	LOAD DATABASE
LOAD LOG	RESTORE DATABASE	RESTORE LOG
REVOKE	RECONFIGURE	
TRUNCATE TABLE	UPDATE STATISTICS	

رمزگذاری تعریف Trigger ها

اگر می‌خواهید پس از ایجاد یک **Trigger** مطمئن شوید که تعریف **Trigger** نمی‌تواند توسط کاربران دیگر مشاهده شود، می‌توانید از عبارت **WITH ENCRYPTION** استفاده کنید. در این صورت تعریف **Trigger** به شکل غیر قابل خواندن ذخیره می‌شود و پس از آن توسط هیچ کس حتی مالک **Trigger** یا مدیر سیستم قابل رمزگشایی نخواهد بود.

بررسی تغییرات انجام شده روی ستون‌های مشخص

در تعریف یک **Trigger** می‌توان از عبارت **IF UPDATE (column_name)** برای تعیین این که آیا یک دستور **INSERT** یا **UPDATE** بر روی ستون خاصی از جدول تأثیر داشته است یا خیر، استفاده کرد. اگر در نتیجه اجرای این دستورات، مقداری در ستون قرار گرفته باشد، حاصل عبارت، **TRUE** خواهد بود.

از آنجایی که نمی‌توان مقدار یک ستون را با دستور **DELETE** حذف کرد، این عبارت برای دستور **DELETE** به کار نمی‌رود.

همچنین برای بررسی این که کدام یک از ستون‌ها در جدول بوسیله دستورات **INSERT** یا **UPDATE** بهنگام سازی شده‌اند، می‌توان از عبارت **IF COLUMNS_UPDATED ()** استفاده کرد.

به مثال زیر توجه کنید.

```
CREATE TABLE my_table
(a int NULL, b int NULL)
GO
CREATE TRIGGER my_trig
ON my_table
FOR INSERT
AS
IF UPDATE(b)
  PRINT 'Column b Modified'
GO
```

مثال فوق را می‌توان با استفاده از عبارت ( ) COLUMNS_UPDATED به صورت زیر انجام داد:

```
CREATE TRIGGER my_trig2
ON my_table
FOR INSERT
AS
IF ( COLUMNS_UPDATED() & 2 = 2 )
  PRINT 'Column b Modified'
GO
```

بازگرداندن نتایج

توصیه می‌شود که Triggerها نتیجه‌ای را باز نگردانند. برای این منظور، در تعریف Trigger نباید از دستور SELECT و یا انتساب متغیر استفاده کرد. اگر در یک Trigger باید از انتساب متغیر استفاده شود، در ابتدای Trigger از دستور SET NOCOUNT برای حذف مجموعه‌های نتیجه استفاده کنید.

Triggerها و دستور ROLLBACK TRANSACTION

هنگامی که Triggerهایی که شامل دستور ROLL BACK TRANSACTION هستند از یک دسته اجرا می‌شوند، کل دسته لغو می‌شود. در مثال زیر، اگر دستور INSERT باعث فعال شدن Triggerای شود که شامل دستور ROLLBACK TRANSACTION است، دستور DELETE اجرا نخواهد شد، زیرا دسته لغو می‌شود.

```
/* Start of Batch */
INSERT employee VALUES ('XYZ12345M', 'New', 'M', 'Employee', 1, 1,
'9952', '6/1/95') -- Causes trigger to fire and ROLLBACK TRANSACTION
DELETE employee WHERE emp_id = 'PMA42628M'
GO
```

اگر Triggerهایی که شامل دستور ROLLBACK TRANSACTION است از

درون یک تراکنش تعریف شده توسط کاربر فعال شوند، کل تراکنش لغو می‌شود. در مثال زیر، اگر دستور INSERT سبب فعال شدن Triggerای شود که شامل دستور ROLLBACK TRANSACTION است، دستور UPDATE نیز لغو می‌شود.

```
/* Start of Transaction */
BEGIN TRANSACTION
UPDATE employee SET hire_date = '7/1/94' WHERE emp_id = 'VPA30890F'
INSERT employee VALUES ('XYZ12345M', 'New', 'M', 'Employee', 1, 1,
'9952', '6/1/95') -- Causes trigger to fire and ROLLBACK TRANSACTION
```

استفاده از جدول‌های ویژه inserted و deleted

دو جدول ویژه در دستورات موجود در Trigger مورد استفاده قرار می‌گیرند: جدول deleted و جدول inserted. از این جدول‌های موقت می‌توانید برای بررسی تأثیر تغییر داده‌ها استفاده کنید. نمی‌توانید مستقیماً داده‌های درون این دو جدول را تغییر دهید اما می‌توانید از این جدول‌ها در دستور SELECT برای تعیین این که آیا Trigger توسط یک دستور INSERT، UPDATE یا DELETE فعال شده است استفاده کنید.

جدول deleted کپی‌هایی از سطرهای متأثر را هنگام اجرای دستورات DELETE و UPDATE ذخیره می‌کند. در حین اجرای یک دستور DELETE یا UPDATE، سطرها از جدول حذف شده و به جدول deleted منتقل می‌شود. جدول شامل Trigger و جدول deleted به طور معمول سطرهای مشترک ندارند.

جدول inserted کپی‌هایی از سطرهای متأثر را هنگام اجرای دستورات INSERT و UPDATE ذخیره می‌کند. در حین اجرای دستورات INSERT یا UPDATE سطرهای جدید به طور همزمان به جدول شامل Trigger و جدول inserted اضافه می‌شوند.

دستور UPDATE مشابه یک حذف و یک درج است. ابتدا سطرهای قدیمی به جدول deleted کپی می‌شود و سپس سطرهای جدید به جدول Trigger و جدول inserted کپی می‌شود.

در مثال زیر، Trigger برای ذخیره مجموع یک ستون طراحی شده است. برای

درج یک سطر منفرد، این **Trigger** به خوبی کار می‌کند. **Trigger** بوسیله یک دستور **INSERT** فعال می‌شود و سطر جدید در حین اجرای **Trigger** در جدول **inserted** درج می‌شود. دستور **UPDATE** مقدار ستون **qty** را برای سطر خوانده و آن را به مقدار موجود در ستون **ytd_sales** و از جدول **titles** اضافه می‌کند.

```
-- Trigger valid for single row INSERTS.
CREATE TRIGGER intrig
ON sales
FOR INSERT AS
   UPDATE titles
   SET ytd_sales = ytd_sales + qty
   FROM inserted
   WHERE titles.title_id = inserted.title_id
```

در حالت یک **INSERT** چند سطری این **Trigger** ممکن است به درستی عمل نکند. عبارت سمت راست دستور انتساب در دستور **UPDATE** فقط می‌تواند یک مقدار منفرد باشد نه لیستی از مقادیر. برای بهنگام کردن صحیح جدول **titles**، **Trigger** باید امکان درج چند سطر در جدول **inserted** را داشته باشد. این امر می‌تواند به کمک تابع **SUM** انجام شود.

```
-- Trigger valid for multirow and single row inserts.
CREATE TRIGGER intrig
ON sales
FOR INSERT AS
   UPDATE titles
   SET ytd_sales = ytd_sales +
      (SELECT SUM(qty) -- Correlated subquery.
       FROM inserted
       WHERE titles.title_id = inserted.title_id)
   WHERE titles.title_id IN
       (SELECT title_id FROM inserted)
```

این **Trigger** برای درج تک سطر نیز درست کار می‌کند اما با این **Trigger**

پرس و جوی فرعی وابسته و عملگر IN به کار رفته در عبارت WHERE نیازمند پردازش اضافی در SQL Server است که برای درج یک سطر، غیر ضروری است. بنابراین می‌توانید Trigger را طوری تغییر دهید که بر اساس تعداد سطرها به طور بهینه کار کند. مثلاً تابع @@ROWCOUNT در منطق Trigger می‌تواند جهت تشخیص یک درج تک سطری از یک درج چند سطری به کار رود.

```
-- Trigger valid for multirow and single row inserts
-- and optimal for single row inserts.
CREATE TRIGGER intrig
ON sales
FOR INSERT AS
IF @@ROWCOUNT = 1
BEGIN
   UPDATE titles
   SET ytd_sales = ytd_sales + qty
   FROM inserted
   WHERE titles.title_id = inserted.title_id
END
ELSE
BEGIN
   UPDATE titles
   SET ytd_sales = ytd_sales +
      (SELECT SUM(qty)
       FROM inserted
       WHERE titles.title_id = inserted.title_id)
   WHERE titles.title_id IN
       (SELECT title_id FROM inserted)
END
```

Trigger درج شرطی

یک **Tigger** می‌تواند یک تراکنش را به کلی لغو کرده یا تأیید نماید. هر چند با استفاده از یک پرس و جوی فرعی وابسته در یک **Trigger** می‌توان **Trigger** را وادار کرد که سطرهای تغییر یافته را یکی یکی بررسی نماید.

به عنوان مثال، اگر بخواهید هر رکورد را که در جدول درج می‌شود بررسی کنید، **Trigger** زیر با نام **conditionalinsert** درج را سطر به سطر تجزیه و تحلیل کرده و سطرهایی را که یک **title_id** در جدول **titles** ندارند حذف می‌کند:

```
CREATE TRIGGER conditionalinsert
ON sales
FOR INSERT AS
IF
(SELECT COUNT(*) FROM titles, inserted
WHERE titles.title_id = inserted.title_id) <> @@ROWCOUNT
BEGIN
   DELETE sales FROM sales, inserted
   WHERE sales.title_id = inserted.title_id AND
      inserted.title_id NOT IN
      (SELECT title_id
       FROM titles)
   PRINT 'Only sales records with matching title_ids added.'
END
```

Triggerهای بازگشتی

یک **Trigger** نمی‌تواند به طور بازگشتی خود را فراخوانی کند مگر این که گزینه پایگاه داده **Recursive Triggers** انتخاب شده باشد. دو نوع مختلف بازگشت وجود دارد:

- بازگشت مستقیم وقتی اتفاق می‌افتد که یک **Trigger** فعال شود و عملی را انجام دهد که باعث فعال شدن مجدد همان **Trigger** شود.

- بازگشت غیر مستقیم وقتی اتفاق می‌افتد که یک **Trigger** فعال شود و عملی را انجام دهد که باعث فعال شدن یک **Trigger** دیگر بر روی جدول دیگری شود. **Trigger** دوم باعث می‌شود تا یک عمل بهنگام سازی بر روی جدول اول اتفاق بیفتد که منجر به فعال شدن **Trigger** اولیه می‌شود. یک مثال از کاربرد **Trigger** بازگشتی به صورت زیر است:

```
USE pubs
GO
-- Turn recursive triggers ON in the database.
EXECUTE sp_dboption 'pubs', 'recursive triggers', TRUE
GO
CREATE TABLE emp_mgr (
emp char(30) PRIMARY KEY,
mgr char(30) NULL FOREIGN KEY REFERENCES emp_mgr(emp),
NoOfReports int DEFAULT 0
)
GO
CREATE TRIGGER emp_mgrins ON emp_mgr
FOR INSERT
AS
DECLARE @e char(30), @m char(30)
DECLARE c1 CURSOR FOR
SELECT emp_mgr.emp
FROM emp_mgr, inserted
WHERE emp_mgr.emp = inserted.mgr
OPEN c1
FETCH NEXT FROM c1 INTO @e
WHILE @@fetch_status = 0
BEGIN
   UPDATE emp_mgr
```

```
   SET emp_mgr.NoOfReports = emp_mgr.NoOfReports + 1 -- add 1 for newly
   WHERE emp_mgr.emp = @e -- added employee
   FETCH NEXT FROM c1 INTO @e
END
CLOSE c1
DEALLOCATE c1
GO
-- This recursive UPDATE trigger works assuming:
-- 1. Only singleton updates on emp_mgr.
-- 2. No inserts in the middle of the org tree.
CREATE TRIGGER emp_mgrupd ON emp_mgr FOR UPDATE
AS
IF UPDATE (mgr)
BEGIN
   UPDATE emp_mgr
   SET emp_mgr.NoOfReports = emp_mgr.NoOfReports + 1 -- Increment mgr's
   FROM inserted -- (no. of reports) by
   WHERE emp_mgr.emp = inserted.mgr -- 1 for the new report.
   UPDATE emp_mgr
   SET emp_mgr.NoOfReports = emp_mgr.NoOfReports - 1 -- Decrement mgr's
   FROM deleted -- (no. of reports) by 1
   WHERE emp_mgr.emp = deleted.mgr -- for the new report
END
GO
```

افزودن و تغییر داده‌ها

- اهداف
- اضافه کردن داده‌ها
- تغییر داده‌ها
- تراکنش‌ها
- کرزرها
- قفل‌گذاری

۱-۵- اهداف

این فصل شامل اطلاعاتی درباره چگونگی استفاده از دستورات Transact-SQL به همراه SQL Server است. پس از ایجاد جدول‌ها و ساختارهای دیگر، لازم است با روش‌های درج سطرهای جدید، بهنگام سازی و حذف سطرها آشنا شوید. این فصل همچنین به بررسی تراکنش‌ها، کرزرها و چگونگی قفل‌گذاری SQL Server بر روی داده‌ها خواهد پرداخت.

۲-۵- اضافه کردن داده‌ها

SQL Server از روش‌های زیر برای افزودن داده‌ها به جدول استفاده می‌کند:

۱- دستور INSERT با یکی از گزینه‌های زیر:

- عبارت VALUES جهت درج یک سطر با مجموعه‌ای مشخص از مقادیر.
- یک پرس و جوی فرعی جهت درج داده‌های انتخابی از یک جدول یا دید.

۲- دستور SELECT INTO

جهت ایجاد جدول جدیدی حاوی تمام سطرهای مجموعه جواب SELECT INTO بکار می‌رود.

دستورات INSERT علاوه بر جدول‌ها با اندکی محدودیت بر روی دیدها نیز قابل اجرا هستند.

افزودن سطرها با دستور INSERT

دستور INSERT یک یا چند سطر جدید را به یک جدول اضافه می‌کند. فرم ساده‌ای از دستور INSERT به صورت زیر است:

```
INSERT [INTO] table-or-view [(column-list)] data-values
```

دستور فوق باعث می‌گردد تا **data-values** به عنوان یک یا چند سطر در یک جدول یا دید درج شود. **column-list** شامل لیستی از اسامی ستون‌ها است که با علامت (,) از هم جدا شده‌اند. این لیست جهت تعیین ستون‌هایی که داده‌ها باید در آنها درج شوند استفاده می‌شود. اگر **column-list** تعیین نشود، تمام ستون‌های جدول یا دید، داده‌ها

را دریافت می‌کنند.

اگر **column-list** حاوی نام تمام ستون‌های جدول یا دید نباشد، یک مقدار **NULL** (یا یک مقدار پیش فرض در صورتی که برای آن ستون تعریف شده باشد) در هر ستونی که در لیست قرار ندارد درج خواهد شد. مقادیر داده‌ها باید با لیست ستون‌ها منطبق باشد. تعداد مقادیر داده باید با تعداد ستون‌ها برابر باشد و همچنین نوع داده، دقت و اندازه هر مقدار داده باید با ستون‌های متناظر منطبق باشد. دو راه برای تعیین مقادیر داده وجود دارد:

- استفاده از یک عبارت **VALUES** جهت تعیین مقادیر داده‌ها برای یک سطر.

```
INSERT INTO MyTable (PriKey, Description)
    VALUES (123, 'A description of part 123.')
```

- استفاده از یک پرس و جوی فرعی **SELECT** جهت تعیین مقادیر داده برای یک یا چند سطر.

```
INSERT INTO MyTable (PriKey, Description)
    SELECT ForeignKey, Description
    FROM SomeView
```

درج یک سطر با استفاده از INSERT ... VALUES

کلمه کلیدی **VALUES**، مقادیری را برای یک سطر از جدول مشخص می‌کند. این مقادیر به صورت لیستی از عبارت‌های اسکالر که با علامت (,) از هم جدا شده‌اند مشخص می‌شوند. اگر یک لیست ستون‌ها مشخص نشود، مقادیر باید به همان ترتیبی که ستون‌ها در جدول یا دید قرار دارند مشخص گردند.

در مثال زیر، یک مسافر جدید با استفاده از عبارت **VALUES** در جدول **Shippers** درج می‌گردد:

```
INSERT INTO Northwind.dbo.Shippers (CompanyName, Phone)
  VALUES (N'Snowflake Shipping', N'(503)555-7233'
```

درج چند سطر با استفاده از INSERT ... SELECT

پرس و جوی فرعی **SELECT** در دستور **INSERT** می‌تواند به منظور افزودن داده‌ها به یک جدول از یک یا چند جدول یا دید دیگر استفاده شود. همچنین پرس و جوی فرعی **SELECT** اجازه می‌دهد تا در یک لحظه، بیش از یک سطر درج شود.

در مثال زیر، داده‌هایی از تمام سطرهای جدول **titles** که فیلد **type** آنها برابر **modern cooking** است در یک جدول مجزا درج می‌گردد:

```
USE pubs
INSERT INTO MyBooks
   SELECT title_id, title, type
   FROM titles
   WHERE type = 'mod_cook'
```

درج چند سطر با استفاده از SELECT INTO

دستور **SELECT INTO** یک جدول جدید که حاوی مُجموعه جواب **SELECT** است ایجاد می‌کند. ساختار جدول جدید به کمک صفت‌های موجود در لیست انتخابی تعیین می‌گردد. به عنوان مثال:

```
SELECT Shippers.*, Link.Address, Link.City,
            Link.Region, Link.PostalCode
INTO NewShippers
FROM Shippers
    JOIN LinkServer.DB.dbo.Shippers AS Link
     ON (Shippers.ShipperID = Link.ShipperID)
```

این دستور می‌تواند جهت ترکیب داده‌ها از چند جدول یا دید در یک جدول استفاده شود. همچنین از این دستور می‌توان جهت ایجاد یک جدول جدید که حاوی داده‌های انتخابی از یک سرویس‌دهنده متصل شده است، استفاده نمود.

۳-۵- تغییر داده‌ها

پس از این که جدول‌ها ایجاد شده و داده‌ها به آن اضافه شدند، تغییر و بهنگام‌سازی داده‌های درون جدول‌ها، یکی از پردازش‌های روزانه در نگهداری یک پایگاه داده است. اعمال بهنگام‌سازی علاوه بر جدول‌ها با اندکی محدودیت بر روی دیدها نیز قابل اجرا هستند.

تغییر داده‌ها با استفاده از UPDATE

دستور **UPDATE** می‌تواند مقادیر داده را در سطرهای منفرد، گروهی یا تمام سطرهای موجود در یک جدول یا دید تغییر دهد. یک دستور **UPDATE** که به یک جدول یا دید ارجاع می‌کند، در هر لحظه می‌تواند فقط داده‌های یک جدول مبنا را تغییر دهد. دستور **UPDATE** دارای عبارت‌های زیر است:

- **SET**: حاوی لیستی از ستون‌ها است که باید بهنگام‌سازی شوند به همراه مقدار جدید برای هر ستون به شکل **column-name=expression**. مقادیر عبارت‌ها می‌تواند شامل ثابت‌ها، مقادیر انتخابی از ستون یک جدول یا دید دیگر و یا مقادیر محاسبه شده توسط یک عبارت پیچیده باشد.
- **FROM**: جدول‌ها یا دیدهایی که مقادیری را برای عبارت‌های موجود در **SET** فراهم می‌کنند تعیین می‌کند.
- **WHERE**: شرط جستجوی سطرهای جدول‌ها یا دیدهای منبع را مشخص می‌سازد.

به عنوان مثال، دستور بهنگام‌سازی زیر، ده درصد به قیمت تمام محصولات **Northwind** در گروه 2 می‌افزاید.

```
UPDATE Northwind.dbo.Products
SET UnitPrice = UnitPrice * 1.1
WHERE CategoryID = 2
```

تغییر داده‌ها با استفاده از عبارت SET

SET، ستون‌هایی را که باید تغییر یابند و همچنین مقادیر جدید ستون‌ها را تعیین

می‌کند. ستون‌های مشخص شده با مقادیر موجود در عبارت **SET** در تمام سطرهایی که در شرط **WHERE** صادق هستند بهنگام سازی می‌شوند و در صورت مشخص نشدن عبارت **WHERE**، تمام سطرها بهنگام سازی می‌شوند. مثلاً اگر دفترهای انتشار در جدول **publishers** به شهر **Atlanta** از ایالت **Georgia** منتقل شوند، دستور **UPDATE** به صورت زیر خواهد بود:

```
UPDATE publishers SET city = 'Atlanta', state = 'Georgia'
```

عبارت‌های به کار رفته در **SET** می‌توانند پرس و جوهای فرعی که فقط یک مقدار بر می‌گردانند، نیز باشند. مثلاً اگر پایگاه داده **Northwind**، یک جدول **OrderSummary** داشته باشد آنگاه:

```
UPDATE OrderSummary
SET Last30Days =
  (SELECT SUM(OrdDet.UnitPrice * OrdDet.Quantity)
   FROM [Order Details] AS OrdDet
      JOIN Orders AS Ord
      ON (OrdDet.OrderID = Ord.OrderID
         AND Ord.OrderDate > DATEADD(dd,-30,GETDATE()) )
  )
```

تغییر داده‌ها با استفاده از عبارت WHERE

عبارت **WHERE** دو کار را انجام می‌دهد:

- تعیین سطرهایی که باید بهنگام‌سازی شوند.
- در صورت وجود عبارت **FROM**، نشان‌دهنده سطرهایی از جدول منبع است که مقادیر را برای بهنگام سازی فراهم می‌کنند.

دستور **UPDATE** زیر یک تغییر نام را برای مسافران اعمال می‌کند.

```
UPDATE Northwind.dbo.Shippers
SET CompanyName = 'United Shippers'
WHERE CompanyName = 'United Packages'
```

تغییر داده‌ها با استفاده از عبارت FROM

از عبارت **FROM** برای گرفتن داده‌ها از یک یا چند جدول یا دید به داخل جدولی که باید بهنگام‌سازی شود استفاده می‌گردد. مثال زیر سطر **Drik Stringer** از جدول **titleauthor** را بهنگام سازی می‌کند تا یک شماره شناسایی عنوان برای آخرین کتاب او تحت عنوان **"The Psychology of Computer Cooking"** اضافه کند:

```
UPDATE titleauthor
  SET title_id = titles.title_id
  FROM titles INNER JOIN titleauthor
    ON titles.title_id = titleauthor.title_id
    INNER JOIN authors
    ON titleauthor.au_id = authors.au_id
  WHERE titles.title = 'Net Etiquette'
    AND au_lname = 'Locksley'
```

۴-۵- حذف داده‌ها

SQL Server از روش‌های زیر جهت حذف داده‌ها از جدول استفاده می‌کند:

۱- دستور **DELETE**.

۲- دستور **TRUNCATE TABLE**.

دستورات تغییر داده علاوه بر جدول‌ها با اندکی محدودیت بر روی دیدها نیز قابل اجرا هستند.

حذف سطرها با استفاده از DELETE

دستور **DELETE** یک یا چند سطر از یک جدول یا دید را حذف می‌کند. یک شکل ساده از دستور **DELETE** به صورت زیر است:

```
DELETE table_or_view FROM table_sources WHERE search_condition
```

table-or- view نام جدول یا دیدی است که سطرهایی از آن حذف می‌گردند.

تمام سطرهای table-or-view که شرط جستجوی WHERE را بر آورده کنند حذف می‌شوند و در صورتی که عبارت WHERE وجود نداشته باشد، تمام سطرها در table-or-view حذف می‌گردند. عبارت FROM نیز جدول‌ها یا دیدهای اضافی و شرایط پیوندی را تعریف می‌کند که می‌توانند در شرط جستجوی WHERE جهت تعیین سطرهای حذفی table-or-view به کار روند.

دستور DELETE فقط سطرهای جدول را حذف می‌کند اما خود جدول فقط با دستور DROP TABLE از پایگاه داده حذف می‌گردد. مثال‌های زیر، سه دستور DELETE مورد نیاز جهت حذف سطرهای مرتبط با محصولات تولیدی توسط شرکت Lyngbysild در پایگاه داده Northwind را نشان می‌دهند.

```
USE Northwind
GO

DELETE [Order Details]
FROM Suppliers, Products
WHERE Products.SupplierID = Suppliers.SupplierID
  AND Suppliers.CompanyName = 'Lyngbysild'
  AND [Order Details].ProductID = Products.ProductID
GO

DELETE Products
FROM Suppliers
WHERE Products.SupplierID = Suppliers.SupplierID
  AND Suppliers.CompanyName = 'Lyngbysild'
GO

DELETE Suppliers
WHERE CompanyName = 'Lyngbysild'
GO
```

حذف تمام سطرها با استفاده از TRUNCATE TABLE

دستور **TRUNCATE TABLE** یک روش سریع جهت حذف تمام سطرهای یک جدول است. این دستور تقریباً همیشه از دستور **DELETE** بدون شرط، سریع‌تر است. همانند دستور **DELETE**، تعریف یک جدول که توسط دستور **TRUNCATE TABLE** پاک شده در پایگاه داده باقی می‌ماند.

۵-۵- تراکنش‌ها

تراکنش، دنباله‌ای از اعمال است که به عنوان یک واحد منطقی کار انجام می‌شود. یک واحد منطقی کار باید چهار خاصیت تجزیه‌ناپذیری، سازگاری، جداسازی و پایداری را داشته باشد تا به عنوان یک تراکنش شناخته شود.

- تجزیه ناپذیری: یک تراکنش باید جزء تجزیه ناپذیری از کار باشد؛ یعنی یا تمام تغییرات داده آن انجام شود یا هیچ کدام از آنها انجام نشود.
- سازگاری: یک تراکنش باید پس از اتمام، کلیه داده‌ها را در یک حالت سازگار باقی بگذارد. تمام ساختارهای داخلی داده نظیر شاخص‌های **B-tree** یا لیست‌های پیوندی دو طرفه باید در پایان تراکنش به طور صحیح وجود داشته باشد.
- جداسازی: تغییرات انجام گرفته توسط تراکنش‌های همزمان باید از تغییرات انجام شده توسط سایر تراکنش‌های همزمان مجزا باشد.
- پایداری: پس از کامل شدن یک تراکنش، تأثیرات آن به طور دائم در سیستم باقی می‌ماند. این تغییرات در مواقع بروز اشکال در سیستم نیز پایدار می‌ماند.

کنترل تراکنش‌ها

برنامه‌های کاربردی، عمدتاً تراکنش‌ها را با تعیین زمان شروع و پایان تراکنش کنترل می‌کنند. سیستم نیز باید قادر باشد خطاهایی که یک تراکنش را قبل از کامل شدن قطع می‌کنند، پردازش نماید. مدیریت تراکنش‌ها در سطح اتصال انجام می‌شود. وقتی تراکنش در یک اتصال آغاز می‌شود، تمام دستورات **Transact-SQL** که در اتصال اجرا

می‌شوند قسمتی از تراکنش به شمار می‌آیند تا این که تراکنش خاتمه یابد. در SQL Server می‌توانید تراکنش‌ها را به صورت صریح، تأیید خودکار یا ضمنی شروع کنید. علاوه بر این می‌توانید با دستور COMMIT یا ROLLBACK، تراکنش‌ها را خاتمه دهید.

اگر تراکنشی موفقیت‌آمیز باشد، آن را تأیید کنید. دستور COMMIT تضمین می‌کند که تمام تغییرات حاصل از تراکنش یک بخش دائمی از پایگاه داده خواهند بود. اگر خطایی در یک تراکنش پدید آید یا کاربر تصمیم به لغو تراکنش بگیرد، باید آن تراکنش لغو گردد. دستور ROLLBACK با برگرداندن داده‌ها به وضعیتی که در زمان شروع تراکنش داشته‌اند، تمام تغییرات انجام شده در تراکنش را به حالت اول بر می‌گرداند.

اگر یک خطا از اتمام موفقیت آمیز تراکنشی جلوگیری کند، SQL Server به طور خودکار تراکنش را به حالت اول برمی‌گرداند (لغو می‌کند) و تمام منابعی را که توسط تراکنش نگه داشته شده آزاد می‌نماید.

تراکنش‌های صریح

در یک تراکنش صریح می‌توانید شروع و پایان تراکنش را به طور صریح تعریف کنید. تراکنش‌های صریح در نسخه‌های اولیه SQL Server به تراکنش‌های تعریف شده توسط کاربر موسوم بودند. برنامه‌های کاربردی برای تعریف تراکنش‌های صریح از دستورات BEGIN TRANSACTLON، COMMIT TRANSACTION، ROLLBACK TRANSACTION یا ROLLBACK WORK استفاده می‌کنند.

BEGIN TRANSACTLON نقطه شروع یک تراکنش را برای یک اتصال مشخص می‌کند.

COMMIT WORK یا COMMIT TRANSACTOIN در صورتی که تراکنش با خطایی مواجه نشود، برای پایان دادن موفقیت آمیز یک تراکنش به کار می‌رود.

ROLLBACK WORK یا ROLLBACK TRANSACTLON برای لغو تراکنش در مواقع بروز خطا به کار می‌رود.

تراکنش‌های تأیید خودکار

وضعیت مدیریت تراکنش پیش فرض در SQL Server، حالت تأیید خودکار

است. دستورات Transact-SQL پس از کامل شدن، تأیید (در صورت اتمام موفقیت آمیز) و یا لغو (در صورت مواجه شدن با خطا) می‌گردند.

یک اتصال SQL Server در حالت تأیید خودکار عمل می‌کند مگر آن که یک دستور BEGIN TRANSACTION یک تراکنش صریح را آغاز نماید و یا این که وضعیت تراکنش ضمنی فعال باشد. پس از تأیید یا لغو تراکنش صریح و یا غیر فعال شدن تراکنش ضمنی، SQL Server به حالت تأیید خودکار برمی‌گردد.

تراکنش‌های ضمنی

هنگامی که یک اتصال در حالت تراکنش ضمنی عمل می‌کند، SQL Server به طور خودکار پس از تأیید یا لغو تراکنش جاری، یک تراکنش جدید را آغاز می‌کند. بنابراین وضعیت تراکنش ضمنی باعث ایجاد یک زنجیره پیوسته از تراکنش‌ها می‌شود.

پس از فعال شدن وضعیت تراکنش ضمنی، وقتی یکی از دستورات زیر اجرا شود، SQL Server بلافاصله به طور خودکار یک تراکنش را آغاز می‌کند:

ALTER TABLE	FETCH	REVOKE
CREATE	GRANT	SELECT
DELETE	INSERT	TRUNCATE TABLE
DROP	OPEN	UPDATE

تا وقتی که دستور COMMIT یا ROLLBACK اجرا نشده است، تراکنش فعال باقی می‌ماند. پس از تأیید یا لغو اولین تراکنش، SQL Server به طور خودکار بعد از اجرای یکی از دستورات فوق، یک تراکنش جدید را آغاز می‌کند.

برنامه‌های کاربردی از دستور SET IMPLICIT_TRANSACTIONS ON برای آغاز وضعیت تراکنش ضمنی استفاده می‌کنند. از دستور SET IMPLICIT_TRANSACTIONS OFF نیز جهت پایان دادن به وضعیت تراکنش ضمنی استفاده می‌شود. برای پایان دادن به تراکنش‌ها از دستورات COMMIT TRANSACTION، COMMIT WORK، ROLLBACK TRANSACTION یا ROLLBACK WORK استفاده کنید.

```
SET QUOTED_IDENTIFIER OFF
GO

SET NOCOUNT OFF
GO

USE pubs
GO

CREATE TABLE ImplicitTran (Cola int PRIMARY KEY,
        Colb char(3) NOT NULL)
GO

SET IMPLICIT_TRANSACTIONS ON
GO

/* First implicit transaction started by an INSERT statement */
INSERT INTO ImplicitTran VALUES (1, 'aaa')
GO
INSERT INTO ImplicitTran VALUES (2, 'bbb')
GO
/* Commit first transaction */
COMMIT TRANSACTION
GO

/* Second implicit transaction started by a SELECT statement */
SELECT COUNT(*) FROM ImplicitTran
GO
INSERT INTO ImplicitTran VALUES (3, 'ccc')
```

```
GO
SELECT * FROM ImplicitTran
GO
/* Commit second transaction */
COMMIT TRANSACTION
GO
SET IMPLICIT_TRANSACTIONS OFF
GO
```

تراکنش‌های تو در تو (آشیانه‌ای)

تراکنش‌های صریح می‌توانند تو در تو باشند. این امر اساساً به منظور پشتیبانی از تراکنش‌ها در رویه‌های ذخیره شده‌ای است که یا می‌توانند از فرآیندی که در حال حاضر در یک تراکنش است و یا از فرآیندی که هیچ تراکنش فعالی ندارد فراخوانی شوند.

مثال زیرکاربرد تراکنش‌های تودرتو را نشان می‌دهد. رویه **TransProc**، تراکنش خود را بدون توجه به وضعیت تراکنش فرآیندی که آن را اجرا می‌کند، اعمال می‌کند. اگر هنگامی که یک تراکنش فعال است **TransProc** فراخوانی شود، تراکنش تودرتو در **TransProc** عمدتاً نادیده گرفته می‌شود و دستورات **INSERT** بر حسب آخرین عملی که از تراکنش خارجی‌تر انجام شده لغو یا تأیید می‌گردد. اگر **TransProc** توسط فرآیندی اجرا شود که یک تراکنش معوق ندارد، دستور **COMMIT TRANSACTION** در انتهای رویه، دستورات **INSERT** را تأیید می‌کند.

```
SET QUOTED_IDENTIFIER OFF
GO

SET NOCOUNT OFF
GO

USE pubs
```

```
GO

CREATE TABLE TestTrans(Cola INT PRIMARY KEY,
            Colb CHAR(3) NOT NULL)
GO
CREATE PROCEDURE TransProc @PriKey INT, @CharCol CHAR(3) AS
BEGIN TRANSACTION InProc
INSERT INTO TestTrans VALUES (@PriKey, @CharCol)
INSERT INTO TestTrans VALUES (@PriKey + 1, @CharCol)
COMMIT TRANSACTION InProc
GO

/* Start a transaction and execute TransProc */
BEGIN TRANSACTION OutOfProc
GO
EXEC TransProc 1, 'aaa'
GO
/* Roll back the outer transaction, this will
 roll back TransProc?s nested transaction */

ROLLBACK TRANSACTION OutOfProc
GO
EXECUTE TransProc 3,'bbb'
GO
/* The following SELECT statement shows only rows 3 and 4 are
 still in the table. This indicates that the commit
 of the inner transaction from the first EXECUTE s.  ement of
 TransProc was overridden by the subsequent rollback. */
SELECT * FROM TestTrans
GO
```

تأیید کردن تراکنش‌های داخلی توسط SQL Server نادیده گرفته می‌شود. تراکنش بر اساس عملی که در انتهای خارجی‌ترین تراکنش صورت گرفته تأیید یا لغو می‌شود. اگر تراکنش خارجی تأیید نشود، تراکنش‌های تو در توی داخلی نیز تأیید می‌شوند و اگر تراکنش خارجی لغو شود آنگاه تمام تراکنش‌های داخلی نیز لغو خواهند شد.

نقاط ذخیره تراکنش

نقاط ذخیره مکانیزمی برای لغو کردن قسمت‌هایی از تراکنش‌ها ارائه می‌دهند. با استفاده از دستور SAVE TRANSACTION savepoint-name می‌توانید یک نقطه ذخیره ایجاد کرده و سپس با دستور ROLLBACK TRANSACTION savepoint-name تراکنش را تا نقطه ذخیره لغو کنید.

نقاط ذخیره در حالتی که خطاهای غیر قابل پیش بینی به وجود می‌آیند مفید هستند. به کار بردن یک نقطه ذخیره برای لغو کردن بخشی از یک تراکنش در حالتی که یک خطای غیر معمول رخ داده بسیار کارآتر از تست کردن تراکنش است. نقاط ذخیره فقط در مواقعی که احتمال مواجه شدن با خطا کم و هزینه تست صحت یک عمل بهنگام سازی نسبتاً بالا باشد، مؤثر هستند.

مثال زیر، کاربرد نقطه ذخیره را در یک سیستم سفارش نشان می‌دهد که در آن احتمال اجرای خارج از موجودی انبار به دلیل وجود تهیه کنندگان و نقاط سفارش مجدد مؤثر کم است. معمولاً یک برنامه کاربردی قبل از انجام بهنگام سازی‌هایی که سفارش را ثبت می‌کنند موجودی را بررسی می‌کند. برای این منظور، در این مثال فرض بر این است که بررسی اولیه مقدار موجودی قابل دسترس نسبتاً گران است. برنامه کاربردی می‌تواند به گونه‌ای نوشته شود که فقط بهنگام سازی را انجام دهد و در صورت دریافت خطایی مبنی بر عدم موجودی کافی، عمل بهنگام سازی را لغو کند. در این حالت، یک تست سریع از تابع @@ ERROR بعد از عمل درج بسیار سریع‌تر از بررسی مقدار قبل از بهنگام سازی خواهد بود.

جدول InvCtrl یک محدودیت CHECK دارد که یک خطای 547 را در صورتی که ستون QtyInStk کمتر از صفر شود، بوجود می‌آورد. رویه OrderStock یک نقطه ذخیره ایجاد می‌کند. اگر یک خطای 547 به وقوع بپیوندد تا نقطه ذخیره لغو می‌شود

و تعداد اقلام در دسترس را به رویه فراخواننده بر می‌گرداند. اگر **OrderStock** یک مقدار صفر برگرداند، رویه فراخواننده می‌فهمد که موجودی در دسترس برای سفارش کافی است.

```
SET NOCOUNT OFF
GO

USE pubs
GO

CREATE TABLE InvCtrl
      (WhrhousID    int,
       PartNmbr     int,
       QtyInStk     int,
       ReordrPt     int,
       CONSTRAINT InvPK PRIMARY KEY
       (WhrhousID, PartNmbr),
       CONSTRAINT QtyStkCheck CHECK (QtyInStk > 0) )
GO
CREATE PROCEDURE OrderStock @WhrhousID int, @PartNmbr int,
            @OrderQty int
AS
DECLARE @ErrorVar int
SAVE TRANSACTION StkOrdTrn
UPDATE InvCtrl SET QtyInStk = QtyInStk - @OrderQty
WHERE WhrhousID = 1
   AND PartNmbr = 1
SELECT @ErrorVar = @@error
IF (@ErrorVar = 547)
BEGIN
```

```
    ROLLBACK TRANSACTION StkOrdTrn
    RETURN (SELECT QtyInStk
        FROM InvCtrl
        WHERE WhrhousID = @WhrhousID
          AND PartNmbr = @PartNmbr)
END
ELSE
    RETURN 0
GO
```

ROLLBACKها در رویه‌های ذخیره شده و Triggerها

اگر مقدار @@TRANCOUNT قبل و بعد از اجرای رویه ذخیره شده یکسان نباشد، خطای شماره 266 تولید می‌شود. در شرایط یکسان، این خطا در Triggerها تولید نمی‌شود.

اگر مقدار @@TRANCOUNT برابر 1 یا بزرگتر باشد و یک رویه ذخیره شده فراخوانی شود و این رویه یک دستور ROLLBACK TRANSACTION یا ROLLBACK WORK را اجرا نماید، خطای شماره 266 بوجود می‌آید، زیرا ROLLBACK تمام تراکنش‌های فوق را لغو کرده و مقدار @@TRANCOUNT را برابر صفر قرار می‌دهد.

اگر یک ROLLBACK TRANSACTION در یک Trigger اتفاق بیفتد:

- تمام تغییرات داده‌ها تا آن نقطه در تراکنش جاری لغو می‌شود، از جمله هر آنچه که توسط Trigger انجام شده است.
- Trigger به اجرای دستورات باقیمانده بعد از دستور ROLLBACK ادامه می‌دهد. اگر هر کدام از این دستورات، داده را تغییر دهند آن تغییرات لغو نخواهند شد. هیچ Trigger تودرتویی با اجرای دستورات باقیمانده تحریک نمی‌شود.
- هیچیک از دستورات موجود در یک دسته بعد از دستوری که Trigger را تحریک می‌کند اجرا نمی‌شوند.

- یک **ROLLBACK** در یک **Trigger** تمام کرزرهای تعریف شـــده و بـاز در دسته که حاوی دستور تحریک کننده **Trigger** است را می‌بندد.

از دستور **SAVE TRANSACTION** باید برای انجام یک لغـــو جزئـی در یـک **Trigger** استفاده کنید، حتی اگر همواره در حالت تأیید خودکار فراخوانی شود. ایـن امـر در مثال زیر نشان داده شده است:

```
CREATE TRIGGER TestTrig ON TestTab FOR UPDATE AS
SAVE TRANSACTION MyName
INSERT INTO TestAudit
  SELECT * FROM inserted
IF (@@error <> 0)
BEGIN
  ROLLBACK TRANSACTION MyName
END
```

اگر یک دستور **ROLLBACK TRANSACTION** بعـد از **COMMIT** اجـرا شود، **ROLLBACK** هر گونه تغییری را تا خارجی‌ترین **BEGIN TRANSACTION** لغو می‌کند. این امر در مثال زیر نشان داده شده است:

```
CREATE TRIGGER TestTrig ON TestTab FOR UPDATE AS
BEGIN TRANSACTION
INSERT INTO TrigTarget
  SELECT * FROM inserted
COMMIT TRANSACTION
ROLLBACK TRANSACTION
```

دستورات Transact - SQL مجاز در تراکنش‌ها

شما می‌توانید تمام دستورات **Transact - SQL** را به غیر از دســـتورات زیـر در یک تراکنش به کار ببرید:

ALTER DATABASE	DATABASE	RECONFIGURE
BACKUP LOG	TRANSACTION	DATABASE
CREATE DATABASE	DATABASE	LOG
DISK INIT	TRANSACTION	STATISTICS

شما نمی‌توانید از **sp_dboption** برای تنظیم گزینه‌های پایگاه داده و رویه‌های سیستمی که پایگاه داده **master** را در داخل تراکنش‌های تعریف شده توسط کاربر تغییر می‌دهند استفاده کنید.

ایجاد تراکنش‌های کارآ

کوچک نگهداشتن تراکنش‌ها از اهمیت خاصی برخوردار است. وقتی یک تراکنش آغاز می‌شود، **DBMS** باید بسیاری از منابع را تا خاتمه تراکنش نگهدارد. اگر داده‌ای تغییر یابد، سطرهای تغییر یافته باید با قفل‌گذاری انحصاری محافظت گردند تا از خواندن سطرها توسط سایر تراکنش‌ها جلوگیری شود و این قفل‌ها تا وقتی که تراکنش تأیید یا لغو نشده باید باقی بمانند. به ویژه در سیستم‌های با تعداد زیاد کاربران، تراکنش‌ها باید حتی الامکان کوچک باشند تا قفل‌گذاری منابع بین اتصالات همزمان کاهش یابد. تراکنش‌های بزرگ ممکن است برای تعداد کم کاربران مسأله ساز نباشد اما برای سیستمی با هزاران کاربر، غیر قابل تحمل خواهند بود.

نکات زیر در ایجاد تراکنش‌ها باید رعایت شود:

۱- در طول اجرای تراکنش نیازی به گرفتن ورودی از کاربران نباشد. تمام ورودی‌های مورد نیاز را قبل از آغاز تراکنش از کاربران دریافت کنید. اگر به ورودی جدیدی در طول اجرای تراکنش احتیاج باشد تراکنش جاری را لغو کرده و پس از اخذ ورودی از کاربر، تراکنش را مجدداً شروع کنید.

۲- تراکنش را در حین مشاهده داده‌ها باز نکنید. تراکنش‌ها نباید آغاز شوند مگر آن که تمام آنالیزهای اولیه داده به اتمام برسد.

۳- تراکنش را تا حد امکان کوچک نگه دارید.

پس از آن که تغییرات لازم را بررسی کردید، تراکنش را آغاز کرده، دستورات را اجرا کنید و سپس بلافاصله تراکنش را تأیید یا لغو کنید. تراکنشی را که به آن نیاز ندارید

باز نکنید.

۴- از گزینه‌های همزمانی پایین‌تر کرزر نظیر گزینه‌های همزمانی خوش بینانه استفاده کنید.

در یک سیستم با احتمال پایین بهنگام سازی همزمان، سربار پرداختن به خطای «Somebody else changed your data after you read it» بسیار پایین‌تر از سربار قفل‌گذاری دائمی سطرها به هنگام خواندن آنها است.

۵- دستیابی به حداقل مقدار داده در تراکنش.

این امر، تعداد سطرهای قفل‌گذاری شده را کاهش می‌دهد و لذا باعث کاهش رقابت بین تراکنش‌ها می‌گردد.

اجتناب از مسائل همزمانی

برای اجتناب از مسائل همزمانی، تراکنش‌های ضمنی را با دقت مدیریت کنید. هنگام استفاده از تراکنش‌های ضمنی، دستور Transact-SQL بعدی به طور خودکار یک تراکنش جدید را بعد از COMMIT یا ROLLBACK آغاز می‌کند. این امر می‌تواند باعث باز شدن یک تراکنش جدید در حین مشاهده داده‌ها یا اخذ ورودی جدید از کاربر شود. پس از اینکه آخرین تراکنش مورد نیاز جهت محافظت از تغییرات داده‌ها کامل شد، تراکنش‌های ضمنی را غیر فعال کنید تا این که مجدداً یک تراکنش دیگر جهت حفاظت از تغییرات داده‌ها نیاز باشد. این فرآیند به SQL-Server اجازه می‌دهد تا هنگام مشاهده داده‌ها یا اخذ ورودی از کاربر، از حالت تأیید خودکار استفاده کند.

۶-۵- کرزرها

مجموعه سطرهایی که توسط دستور SELECT بر می‌گردد حاوی تمام سطرهایی است که در شرط عبارت WHERE این دستور صدق می‌کنند. این مجموعه کامل از سطرها که توسط دستور فوق بر می‌گردد، مجموعه نتیجه نامیده می‌شود. برنامه‌های کاربردی به ویژه آنهایی که بر خط و محاوره‌ای هستند نمی‌توانند همیشه با کل مجموعه نتیجه به عنوان یک واحد به طور مؤثر کار کنند. این برنامه‌ها به مکانیزمی جهت کار با یک سطر یا یک بلاک کوچک از سطرها نیاز دارند. کرزرها، بسط یافته مجموعه‌های نتیجه هستند که این مکانیزم را فراهم می‌کنند.

کرزرها پردازش نتیجه را به روش‌های زیر توسعه می‌دهند:

- اجازه قرار گرفتن در سطرهای ویژه‌ای از مجموعه نتیجه.
- بازیابی یک سطر یا بلاکی از سطرها از موقعیت جاری در مجموعه نتیجه.
- پشتیبانی از تغییر داده‌ها برای سطرها در موقعیت جاری از مجموعه نتیجه.
- پشتیبانی از سطوح مختلف دید جهت تغییرات انجام شده توسط سایر کاربران پایگاه داده که در مجموعه نتیجه نشان داده شده است.
- ارائه دستورات Transact-SQL در رویه‌های ذخیره شده و Triggerها که به داده‌ها در مجموعه نتیجه دسترسی دارند.

درخواست یک کرزر

SQL Server دو روش را برای درخواست یک کرزر پشتیبانی می‌کند:

۱- Transact-SQL.

۲- توابع کرزر API پایگاه داده.

برنامه‌های کاربردی هرگز نباید این دو روش را جهت درخواست یک کرزر با هم ترکیب کنند. اگر هیچ کرزر Transact-SQL یا API درخواست نگردد، SQL Server به طور پیش فرض، یک مجموعه نتیجه کامل را که مجموعه نتیجه پیش فرض برای برنامه کاربردی نامیده می‌شود، برمی‌گرداند.

پردازش کرزر

کرزرهای Transact-SQL و API، اشکال نحوی متفاوتی دارند اما پردازش کلی زیر برای تمام کرزرهای SQL Server به کار می‌رود:

۱- ارتباط یک کرزر با مجموعه نتیجه یک دستور Transact SQL و تعریف ویژگی‌های کرزر از قبیل این که آیا سطرهای کرزر می‌توانند بهنگام سازی شوند یا خیر.

۲- اجرای دستور Transact SQL جهت پر کردن کرزر.

۳- بازیابی سطرهایی از کرزر که می‌خواهید آنها را ببینید. بازیابی یک سطر یا بلاکی از سطرها از یک کرزر، یک واکشی نامیده می‌شود. انجام یک سری از اعمال واکشی در مسیرهای جلو یا عقب، scroll نام دارد.

۴- انجام عملیات تغییر داده بصورت اختیاری (حذف یا بهنگام سازی) بر روی سطر واقع در موقعیت جاری کرزر.

۵- بستن کرزر.

مجموعه‌های نتیجه پیش فرض

SQL Server مجموعه‌های نتیجه را به روش زیر برای سرویس‌گیرنده‌ها بر می‌گرداند:

۱- SQL Server یک بسته شبکه حاوی دستور Transact-SQL یا دسته حاوی دستورات Transact-SQL را جهت اجرا دریافت می‌کند.

۲- SQL Server، دستور یا دسته را ترجمه و اجرا می‌کند.

۳- SQL Server، سطرهای مجموعه نتیجه را در بسته شبکه قرار داده و شروع به ارسال آن به سرویس‌گیرنده می‌کند. SQL Server سطرهای مجموعه نتیجه را تا حد امکان در هر بسته قرار می‌دهد.

۴- بسته‌های حاوی سطرهای مجموعه نتیجه در بافرهای شبکه سرویس‌گیرنده قرار می‌گیرند. هنگامی که برنامه کاربردی سرویس‌گیرنده سطرها را واکشی می‌کند، راه‌انداز ODBC، سطرها را از بافرهای شبکه خارج و به برنامه کاربردی سرویس‌گیرنده انتقال می‌دهد.

مجموعه‌های نتیجه پیش فرض، کارآترین روش برای انتقال نتایج به سرویس‌گیرنده هستند. مجموعه‌های نتیجه پیش فرض فقط می‌توانند جهت برگرداندن نتیجه به یک برنامه کاربردی سرویس‌گیرنده به کار روند. داده‌های موجود در یک مجموعهٔ نتیجه پیش فرض توسط سایر دستورات Transact-SQL یا متغیرهای موجود در یک دسته، رویه‌های ذخیره شده یا Triggerها قابل دسترسی نیستند.

کرزرهای Transact-SQL

کرزرهای Transact-SQL، عمدتاً در رویه‌های ذخیره شده و Triggerها به کار می‌روند. فرآیند نمونه برای استفاده از یک کرزر Transact-SQL در یک رویه ذخیره شده یا Trigger به صورت زیر است:

۱- تعریف متغیرهای Transact-SQL که محتوی داده‌های بازگشتی توسط کرزر است. هر متغیر برای یک ستون از مجموعه نتیجه به کار می‌رود.

۲- ارتباط دادن یک کرزر Transact-SQL با یک دستور SELECT به کمک دستور DECLARE CURSOR. دستور DECLARE CURSOR همچنین مشخصات کرزر از قبیل نام، ویژگی فقط خواندنی یا پیشرو کرزر را تعریف می‌کند.

۳- استفاده از دستور OPEN جهت اجرای دستور SELECT و پر کردن کرزر.

۴- استفاده از دستور FETCH INTO جهت واکشی سطرهای منفرد و انتقال داده هر ستون به یک متغیر مشخص. سایر دستورات Transact-SQL می‌توانند به این متغیرها جهت دستیابی به مقادیر دادهٔ واکشی شده ارجاع کنند. کرزرهای Transact-SQL، واکشی بلاک‌هایی از سطرها را پشتیبانی نمی‌کنند.

۵- استفاده از دستور CLOSE برای بستن کرزر. بستن یک کرزر باعث آزاد شدن برخی از منابع مانند مجموعهٔ نتیجهٔ کرزر و همچنین قفل‌های آن بر روی سطر جاری می‌شود اما ساختار کرزر همچنان جهت پردازش، در دسترس باقی می‌ماند. دستور DEALLOCATE باعث آزاد شدن کامل تمام منابع تخصیص یافته به کرزر از جمله نام کرزر می‌شود.

استفاده از متغیر نوع cursor

SQL Server متغیرهای با نوع داده‌ای cursor را پشتیبانی می‌کند. یک کرزر می‌تواند به یکی از دو روش زیر با یک متغیر از نوع cursor در ارتباط باشد:

```
/* Use DECLARE @local_variable, DECLARE CURSOR and SET. */
DECLARE @MyVariable CURSOR

DECLARE MyCursor CURSOR FOR
SELECT LastName FROM Northwind.dbo.Employees
SET @MyVariable = MyCursor
/* Use DECLARE @local_variable and SET */
DECLARE @MyVariable CURSOR
```

```
SET @MyVariable = CURSOR SCROLL KEYSET FOR
SELECT LastName FROM Northwind.dbo.Employees
```

پس از آن که یک کرزر با متغیری از نوع **cursor** مرتبط شد، به جای نام کرزر می‌توان از متغیر **cursor** در دستورات **Transact-SQL** استفاده کرد. پارامترهای خروجی رویه ذخیره شده نیز می‌تواند به یک نوع داده‌ای **cursor** منتسب گردد.

واکشی و اداره کردن

عمل بازیابی یک سطر از یک کرزر را واکشی می‌گویند. گزینه‌های واکشی به قرار زیر هستند:

- **FECH FIRST**: اولین سطر کرزر را واکشی می‌کند.
- **FETCH NEXT**: سطر بعد از آخرین سطر واکشی شده را واکشی می‌کند.
- **FETCH PRIOR**: سطر قبل از آخرین سطر واکشی شده را واکشی می‌کند.
- **FETCH LAST**: آخرین سطر کرزر را واکشی می‌کند.
- **FETCH ABSOLUTE n**: اگر **n** عدد صحیح مثبتی باشد، **n** امین سطر نسبت به سطر اول کرزر را واکشی می‌کند و اگر **n** یک عدد صحیح منفی باشد، **n** امین سطر قبل از انتهای کرزر واکشی می‌شود. اگر **n** برابر صفر باشد، هیچ سطری واکشی نمی‌شود.
- **FETCH RELATIVE n**: **n** امین سطر نسبت به آخرین سطر واکشی شده را واکشی می‌کند. اگر **n** مثبت باشد، **n** امین سطر بعد از آخرین سطر واکشی شده را واکشی می‌کند و اگر **n** منفی باشد، **n** امین سطر قبل از آخرین سطر واکشی شده را واکشی می‌کند. اگر **n** برابر صفر باشد، همان سطر دوباره واکشی می‌گردد.

وقتی یک کرزر باز می‌شود، موقعیت سطر جاری در کرزر به طور منطقی قبل از اولین سطر است. این امر باعث می‌شود تا در صورتی که گزینه‌های واکشی بعد از باز شدن کرزر، به عنوان اولین واکشی عمل کنند، رفتارهای متفاوتی داشته باشد:

- **FETCH FIRST**: اولین سطر کرزر را واکشی می‌کند.
- **FETCH NEXT**: اولین سطر کرزر را واکشی می‌کند.

کرزرهای پیشرو

یک کرزر پیشرو از scroll پشتیبانی نمی‌کند، بلکه فقط واکشی سطرهای ویژه‌ای از ابتدا تا انتهای کرزر را پشتیبانی می‌کند. تأثیر تمام دستورات INERT، UPDATE و DELETE که توسط کاربر جاری یا سایر کاربران بر روی مجموعه نتیجه انجام می‌شود، هنگام واکشی سطرها از کرزر قابل رؤیت است.

کرزرهای ایستا

وقتی یک کرزر ایستا باز می‌شود، مجموعه نتیجه آن در tempdb ذخیره می‌شود. کرزر ایستا سطرهای جدید درج شده در پایگاه داده را پس از باز شدن نشان نمی‌دهد حتی اگر آن سطرها با شرایط جستجوی دستور SELECT کرزر مطابقت داشته باشند. اگر سطرهای تشکیل دهنده مجموعه نتیجه توسط سایر کاربران بهنگام سازی گردد، مقادیر داده جدید در کرزر ایستا نمایش داده نمی‌شود. کرزر ایستا سطرهای حذف شده از پایگاه داده را پس از باز شدن نشان می‌دهد. کرزرهای ایستای SQL Server همواره فقط خواندنی هستند. نتایج حاصل از دستورات تغییر داده در کرزر ایستا منعکس نمی‌شود مگر آن که کرزر را بسته و دوباره آن را باز کنیم. از آن جایی که مجموعه نتیجه یک کرزر ایستا در یک جدول کاری در پایگاه داده tempdb ذخیره می‌گردد، اندازه سطرها در مجموعه نتیجه نمی‌تواند از حداکثر اندازه سطر یک جدول SQL Server بیشتر شود.

کرزرهای پویا

کرزرهای پویا، برعکس کرزرهای ایستا عمل می‌کنند. این کرزرها، تمام تغییرات سطرها را در مجموعه نتیجه منعکس می‌کنند. تمام دستورات UPDATE، INSERT و DELETE انجام شده توسط تمام کاربران از طریق این نوع کرزرها قابل رؤیت است.

کرزرهای keyset-driven

عضویت و ترتیب سطرها در یک کرزر keyset-driven بهنگام باز شدن کرزر، ثابت است. کرزرهای keyset-driven بوسیله یک مجموعه از شناسه‌های یکتا

(کلیدها) موسوم به keyset کنترل می‌شوند. کلیدها، متشکل از یک مجموعه از ستون‌ها هستند که منحصراً سطرهای مجموعه نتیجه را تعیین می‌کند. وقتی یک کرزر keyset-driven باز می‌شود، keyset مربوط به آن در tempdb ذخیره می‌شود.

۷-۵- قفل‌گذاری

SQL Server به منظور تأمین جامعیت تراکنشی و سازگاری پایگاه داده از قفل‌گذاری استفاده می‌کند. قفل‌گذاری از خواندن داده‌هایی که توسط کاربران دیگر در حال تغییر است و همچنین تغییر همزمان داده‌ها توسط چند کاربر جلوگیری می‌کند. اگرچه SQL Server به طور خودکار، قفل‌گذاری را اعمال می‌کند، شما نیز می‌توانید برنامه‌های کاربردی کارآتری را با شناسایی، و سفارشی کردن قفل‌گذاری طراحی کنید.

اگر قفل‌گذاری وجود نداشته باشد و چند کاربر به طور همزمان به یک پایگاه داده دسترسی داشته باشند، در صورتی که تراکنش‌های آنها در یک لحظه از داده خاصی استفاده کند، چهار مسأله به قرار زیر ممکن است اتفاق بیفتد:

- مسأله بهنگام سازی گمشده یا مخفی: وقتی اتفاق می‌افتد که دو یا چند تراکنش یک سطر را انتخاب کرده و آن سطر را بر اساس مقداری که در ابتدا انتخاب شده بهنگام سازی نمایند. چون هر تراکنش از سایر تراکنش‌های دیگر بی‌اطلاع است، آخرین بهنگام سازی بر روی بهنگام سازی انجام شده توسط سایر تراکنش‌ها رونویسی شده و در نتیجه، داده گم می‌شود.
- مسأله وابستگی تأیید نشده: به عنوان مثال، یک ویرایشگر در حال تغییر دادن یک متن را در نظر بگیرید. در حین تغییر یک ویرایشگر دیگر، یک کپی از متن را که حاوی آخرین تغییرات اعمال شده است خوانده و آن را چاپ و توزیع می‌کند. ویرایشگر اول تصمیم می‌گیرد که تغییرات را نادیده گرفته و متن را به شکل ابتدایی ذخیره کند. متن‌های چاپ شده شامل تغییراتی است که دیگر وجود ندارد، گو این که هرگز وجود نداشته است. بنابراین بهتر است تا وقتی که ویرایشگر تغییرات را تأیید نکرده است، هیچ کاربری نتواند متن را بخواند.
- مسأله تجزیه و تحلیل ناسازگار (خواندن غیر قابل تکرار): به عنوان مثال، یک

ویرایشگر، متنی را دو بار می‌خواند اما در بین هر بار خواندن، نویسنده آن متن را بازنویسی می‌کند. وقتی که ویرایشگر برای بار دوم متن را می‌خواند، آن متن کاملاً تغییر یافته است. خواندن اولیه قابل تکرار نیست و منجر به اشتباه می‌شود. بنابراین بهتر است ویرایشگر هنگامی متن را بخواند که نویسنده، نوشتن متن را کاملاً به پایان رسانیده باشد.

- مسأله خواندن ظاهری: به عنوان مثال، یک ویرایشگر، متنی را خوانده و ویرایش می‌کند اما وقتی که می‌خواهد آن را با نسخه اصلی متن ترکیب کند، متوجه می‌شود که یک متن جدید و ویرایش نشده توسط نویسنده به متن اصلی اضافه شده است.

متن شامل اقلام جدیدی است که قبلاً وجود نداشته و همین امر موجب تداخل می‌شود. بنابراین بهتر است تا وقتی که ویرایشگر کار خود را با متن تمام نکرده است، هیچ کس نتواند چیزی به آن اضافه کند.

قفل‌گذاری در SQL Server

در SQL Server انواع مختلفی از منابع می‌توانند در یک تراکنش قفل‌گذاری شوند. برای کاهش هزینه قفل‌گذاری، SQL Server به طور خودکار منابع را در سطحی مناسب با کار مورد نظر قفل‌گذاری می‌کند. قفل‌گذاری در سطحی پایین‌تر مثل سطرها، همزمانی را افزایش می‌دهد اما سربار بیشتری دارد؛ زیرا اگر سطرهای زیادی قفل‌گذاری شوند، قفل‌های زیادی باید نگهداری شود. قفل‌گذاری در سطحی بالاتر مثل جدول‌ها، از نظر همزمانی از هزینه بیشتری برخوردار است زیرا قفل‌گذاری کامل جدول دستیابی به قسمت‌های جدول بوسیله تراکنش‌ها را محدود می‌کند اما سربار آن پایین‌تر است زیرا قفل‌های کمتری نگهداری می‌شود.

SQL Server با استفاده از حالت‌های مختلف قفل‌گذاری که چگونگی دستیابی تراکنش‌های همزمان به منابع را تعیین می‌کنند، منابع را قفل‌گذاری می‌کند. حالت‌های قفل‌گذاری منابع عبارتند از:

حالت قفل‌گذاری	شرح
Shared (S)	برای عملیاتی که داده‌ها را تغییر نداده یا بهنگام سازی نمی‌کنند (نظیر یک دستور SELECT) بکار می‌رود. (عملیات فقط خواندنی)
Update (U)	برای منابع بهنگام سازی به کار می‌رود.
Exclusive (X)	برای عملیات تغییر داده نظیر INSERT،UPDATE یا DELETE به کار می‌رود و از انجام چند عمل بهنگام سازی بر روی یک منبع در یک لحظه جلوگیری می‌کند.
Intent	برای ایجاد یک سلسله مراتب قفل‌گذاری به کار می‌رود.
Schema	وقتی عملیاتی که وابسته به شمایی از یک جدول است اجرا می‌شود، به کار می‌رود. دو نوع قفل شما وجود دارد: Sch-S یا Schema Stability و Sch-M یا Schema Modification.

سازگاری قفل ها

فقط قفل‌های سازگار با قفلی که قبلاً بر روی یک منبع قرار گرفته می‌توانند بر روی آن منبع قرار بگیرند. مثلاً اگر یک قفل از نوع X بر روی منبعی وجود داشته باشد، تراکنش‌های دیگر نمی‌توانند هیچ نوع قفلی (اعم از Shared، Update یا Exclusive) را بر روی آن منبع اعمال کنند مگر آن که قفل Exclusive در پایان اولین تراکنش آزاد شود.

جدول زیر که بر اساس افزایش قدرت قفل‌ها تنظیم شده است، نشان می‌دهد که کدام قفل با سایر قفل‌های موجود بر روی یک منبع سازگار است:

	Existing granted mode					
Requested mode	IS	S	U	IX	SIX	X
---	---	---	---	---	---	---
Intent shared (IS)	Yes	Yes	Yes	Yes	Yes	No
Shared (S)	Yes	Yes	Yes	No	No	No
Update (U)	Yes	Yes	No	No	No	No

Intent exclusive (IX)	Yes	No	No	Yes	No	No
Shared with intent exclusive (SIX)	Yes	No	No	No	No	No
Exclusive (X)	No	No	No	No	No	No

- قفل از نوع **Sch-S** با تمام حالت‌های قفل‌گذاری بجز **Sch-M** سازگار است.
- قفل از نوع **Sch-M** با هیچیک از حالت‌های قفل‌گذاری سازگار نیست.

قفل‌گذاری پویا

اگرچه قفل‌های در سطح سطر باعث افزایش همزمانی می‌گردد، لیکن هزینه سربار سیستم را نیز افزایش می‌دهند. قفل‌های در سطح جدول یا صفحه، سربار کمتری برای سیستم دارند اما باعث کاهش هزینه همزمانی می‌گردند.

SQL Server از یک استراتژی قفل‌گذاری پویا جهت تعیین کارآترین قفل‌ها استفاده می‌کند. **SQL Server** به طور خودکار تعیین می‌کند که هنگام اجرای یک پرس و جو، کدام یک از قفل‌ها بر اساس مشخصه‌های شما و پرس و جو مناسب‌تر است. مثلاً برای کاهش سربار قفل‌گذاری، بهینه‌ساز ممکن است قفل‌های در سطح صفحه را در یک شاخص انتخاب کند.

قفل‌گذاری پویا مزایای زیر را در بردارد:

۱- ساده کردن مدیریت پایگاه داده.

۲- افزایش کارآیی ؛ زیرا **SQL Server** با انتخاب قفل‌های مناسب سربار سیستم را به حداقل می‌رساند.

۳- توسعه دهندگان برنامه‌های کاربردی می‌توانند تمرکز بیشتری بر روی توسعه داشته باشند، زیرا **SQL Server** به طور خودکار قفل‌گذاری را تنظیم می‌کند.

نمایش اطلاعات قفل‌گذاری

SQL Server در صورت اجرای رویه ذخیره شده سیستمی **sp-lock** گزارشی از قفل‌های فعال ارائه می‌دهد. در زیر، یک مجموعه نتیجه نشان داده شده است:

spid	dbid	ObjId	IndId	Type	Resource	Mode	Status
1	1	0	0	DB		S	GRANT
6	1	0	0	DB		S	GRANT
7	1	0	0	DB		S	GRANT
8	1	0	0	DB		S	GRANT
8	1	1396200024	0	RID	1:1225:2	X	GRANT
8	1	1396200024	0	PAG	1:1225	IX	GRANT
8	1	1396200024	2	PAG	1:1240	IX	GRANT
8	1	21575115	0	TAB		IS	GRANT
8	1	1396200024	2	KEY	(03000100cb04)	X	GRANT
8	1	1396200024	0	TAB		IX	GRANT

در ادامه به شرح مختصری از ستون‌های شکل فوق می‌پردازیم:

ستون Type

این ستون، نوع منبعی را که هم اکنون قفل‌گذاری شده است نشان می‌دهد. مقادیر این ستون می‌تواند شامل موارد زیر باشد:

نوع منبع	شرح
RID	منبع از نوع شناسه سطر
KEY	منبع از نوع کلید
PAG	منبع از نوع صفحه داده یا شاخص
EXT	منبع از نوع Extent (گروه بهم پیوسته از ۸ صفحه داده یا شاخص)
TAB	منبع از نوع جدول شامل تمام داده‌ها و شاخص‌ها
DB	منبع از نوع پایگاه داده

ستون Resource

این ستون، اطلاعاتی درباره منبعی که قفل‌گذاری شده ارائه می‌دهد. مقادیر این ستون به ازاء نوع منبع قفل‌گذاری شده طبق جدول زیر می‌باشد:

نوع منبع	شرح
RID	شناسه سطر قفل‌گذاری شده در داخل جدول. سطر به صورت ترکیبی از Fileid:page:rid نشان داده می‌شود که در آن rid عبارت است از شناسه سطر در صفحه.
KEY	یک عدد در مبنای ۱۶ مورد استفاده داخلی SQL Server. PAG شماره صفحه. صفحه به صورت ترکیبی از fileid:page نشان داده می‌شود که fileid عبارت است از شناسه فایل در جدول sysfiles و page نیز عبارت است از شماره صفحه منطقی در داخل آن فایل.
EXT	شماره اولین صفحه در Extent قفل‌گذاری شده. این صفحه به صورت ترکیبی از fileid:page نشان داده می‌شود.
TAB	حاوی هیچگونه اطلاعاتی نیست زیرا ستون ObjId محتوی ID جدول می‌باشد.
DB	حاوی هیچگونه اطلاعاتی نیست زیرا ستون dbid محتوی ID پایگاه داده می‌باشد.

ستون Mode

این ستون بیانگر نوع قفلی است که برای منبع به کار رفته است.

مدیریت امنیت

- اهداف
- معماری امنیت
- تنظیم حساب‌های امنیتی
- مدیریت حساب‌های امنیتی
- مدیریت جوازها

۱-۶- اهداف

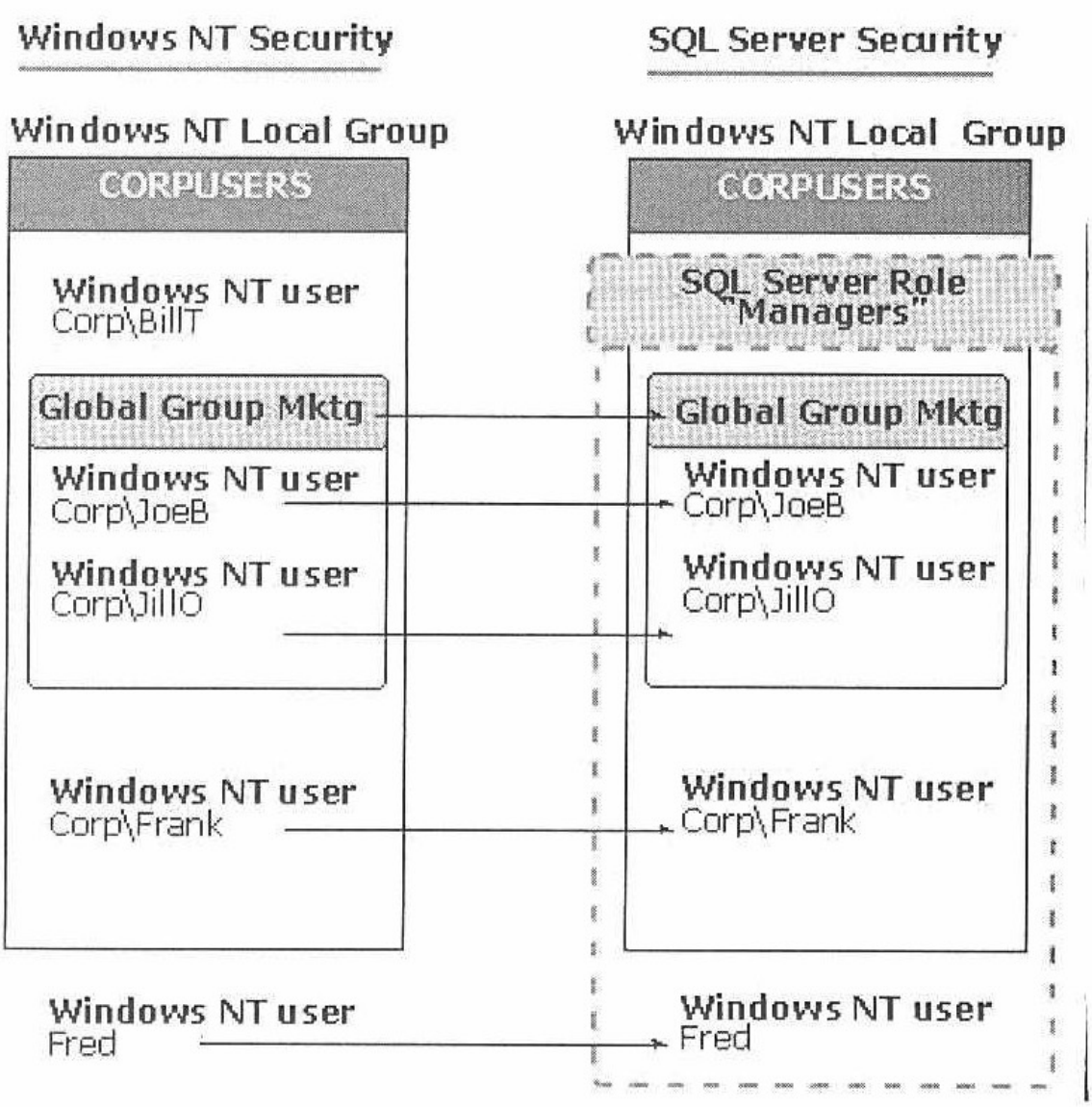
عملیات موجود در مشاغل، سیستمی هماهنگ از افراد، تجهیزات، لوازم، تسهیلات و فعالیت‌ها است. علاوه بر تأمین احتیاجات عملیاتی که به کارمندان اجازه می‌دهد وظایف خود را انجام دهند، سیستم نیز دارای کنترل‌هایی است تا اطمینان حاصل شود که فقط افراد خاصی با جواز و تجربه مناسب برخی از فعالیت‌ها را انجام می‌دهند.

این قاعده در مورد اطلاعات شرکت نیز صادق است. بعضی از اطلاعات می‌توانند برای تمام اشخاص در شرکت به اشتراک گذاشته شود. اطلاعاتی که حساس‌تر می‌باشند در پرونده‌های امن‌تری نگهداری می‌شوند. یک فرد برای دستیابی به این اطلاعات باید دارای جوازهای ویژه‌ای باشد.

با توجه به موارد فوق، مکانیزم‌های امنیتی ویژه‌ای نیز در SQL Server پیش‌بینی شده است. بدین ترتیب حفاظت از داده‌ها بدون توجه به این که کاربران چگونه به پایگاه داده دسترسی پیدا می‌کنند و چه عملی در آن انجام می‌دهند، تضمین می‌شود. در این فصل به بررسی امنیت در SQL Server که شامل معماری، مدیریت و طراحی امنیت می‌باشد، پرداخته خواهد شد.

۲-۶- معماری امنیت

معماری یک سیستم امنیتی، بر اساس کاربران و گروه‌های کاربران موسوم به قوانین امنیت می‌باشد. شکل زیر نشان می‌دهد که چگونه کاربران و گروه‌های محلی و عمومی در ویندوز NT می‌توانند به حساب‌های امنیتی در SQL Server، نگاشته شوند و چگونه SQL Server می‌تواند با حساب‌های امنیتی، مستقل از حساب‌های ویندوز کار کند.



«شکل ۱-۶: امنیت در SQL Server و ویندوز NT»

گروه محلی **CORPUSERS** دارای دو کاربر و یک گروه عمومی بـه نـام **Mktg** است که آن نیز دارای دو کاربر است. **SQL Server** به گروه‌های محلی و عمومی ویـنـدوز **NT** اجازه می‌دهد تا برای سازماندهی حساب‌های کاربران خود مستقیماً مورد استفاده قرار گیرند. علاوه بر این، کاربران ویندوز **NT** به نام‌های **Fred** و **Jerry** می‌توانند مستقیماً یا به عنوان یک کاربر ویندوز **NT** (مثلاً **Fred**) و یا به عنوان یـک کـاربر **SQL Server** (مثـلاً **Jerry**) به **SQL Server** اضافه شوند.

SQL Server، مـدل فـوق را بـا اسـتفاده از نقش‌هـا توسـعه می‌دهـد. نقش‌هـا گروه‌هایی از کاربران هستند که همـانند گروه‌هـای ویـنـدوز **NT** بـرای مقـاصد مدیریتـی سازماندهی شده‌اند. نقش‌ها می‌توانند در مواردی که یک گروه ویندوز **NT** معـادل وجـود ندارد برای سازماندهی کاربران به کار روند. مثلاً نقش **Managers** حاوی گـروه عمومـی **Mktg** و کاربران با نام‌های **Frank** و **Fred** از ویندوز **NT** است.

چگونگی پیاده سازی امنیت توسط SQL Server

یک کاربر هنگام کار با SQL Server از دو مرحلهٔ امنیتی عبور می‌کند: مرحله تأیید صلاحیت و مرحله صحت جوازها. مرحله تأیید صلاحیت، کاربر را با استفاده از یک حساب login شناسایی می‌کند و فقط توانایی اتصال آن را به SQL Server بررسی می‌نماید. از این پس، کاربر برای دستیابی به پایگاه‌های دادهٔ سرویس دهنده به جوازهایی نیاز دارد که با استفاده از یک حساب در هر پایگاه داده که به login کاربر نگاشته شده است، انجام می‌شود. مرحلهٔ صحت جوازها، فعالیت‌هایی را که کاربر مجاز به انجام آنها در پایگاه داده است کنترل می‌کند.

تأیید صلاحیت

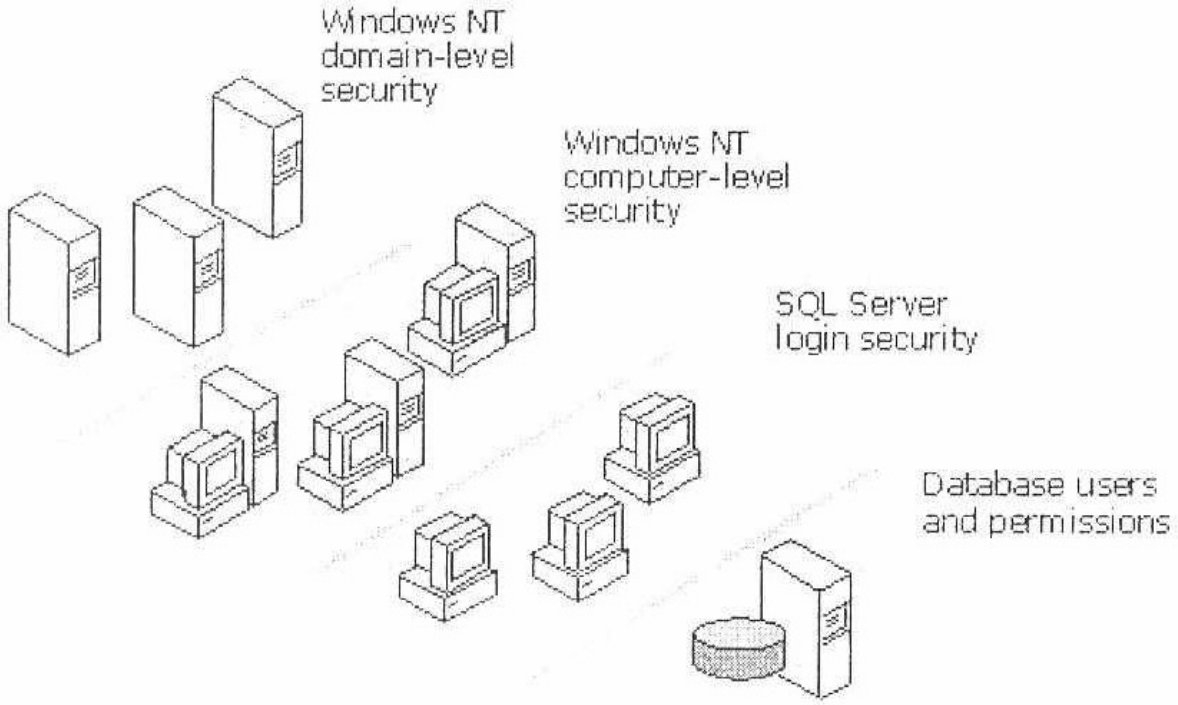
SQL Server می‌تواند در یکی از دو حالت امنیت عمل کند:

۱- حالت تأیید صلاحیت توسط ویندوز NT.

۲- حالت مخلوط (تأیید صلاحیت توسط ویندوز NT و تأیید صلاحیت توسط SQL Server).

حالت مخلوط به کاربران اجازه می‌دهد تا بوسیله تأیید صلاحیت توسط SQL Server یا ویندوز NT متصل شوند. بعد از اتصال موفقیت‌آمیز به SQL Server، مکانیزم امنیت برای هر دو حالت، یکسان است. وقتی کاربری از طریق یک حساب کاربر ویندوز NT متصل می‌شود، SQL Server هنگام Logon شدن کاربر به ویندوز NT یا ویندوز ۹۵/۹۸، صحت نام حساب و کلمه عبور را بررسی می‌کند. SQL Server امنیت Login با ویندوز NT را با استفاده از خواص امنیتی یک کاربر شبکه برقرار می‌کند تا دستیابی به Login را کنترل کند. خواص امنیتی شبکه کاربر در لحظه Login شبکه ایجاد شده و از طریق یک مکانیزم پیچیده رمزگذاری کلمه عبور، تأیید صلاحیت می‌شود. وقتی کاربری سعی در برقراری ارتباط می‌کند، SQL Server برای تعیین اعتبار نام کاربر شبکه، از امکانات ویندوز NT استفاده می‌کند، سپس SQL Server بر اساس این نام کاربر شبکه، اجازه دستیابی به Login را می‌دهد.

توجه: اگر کاربر هنگام اتصال به SQL Server، نام Login را مشخص نکند، SQL Server به طور خودکار از تأیید صلاحیت توسط ویندوز NT استفاده می‌کند. اگر SQL Server برای حالت تأیید صلاحیت توسط ویندوز NT پیکربندی شده باشد و کاربر سعی کند تا با استفاده از یک Login به SQL Server متصل شود، Login نادیده گرفته می‌شود و از حالت تأیید صلاحیت توسط ویندوز NT استفاده می‌گردد.



«شکل ۲-۶: تأیید صلاحیت توسط ویندوز NT»

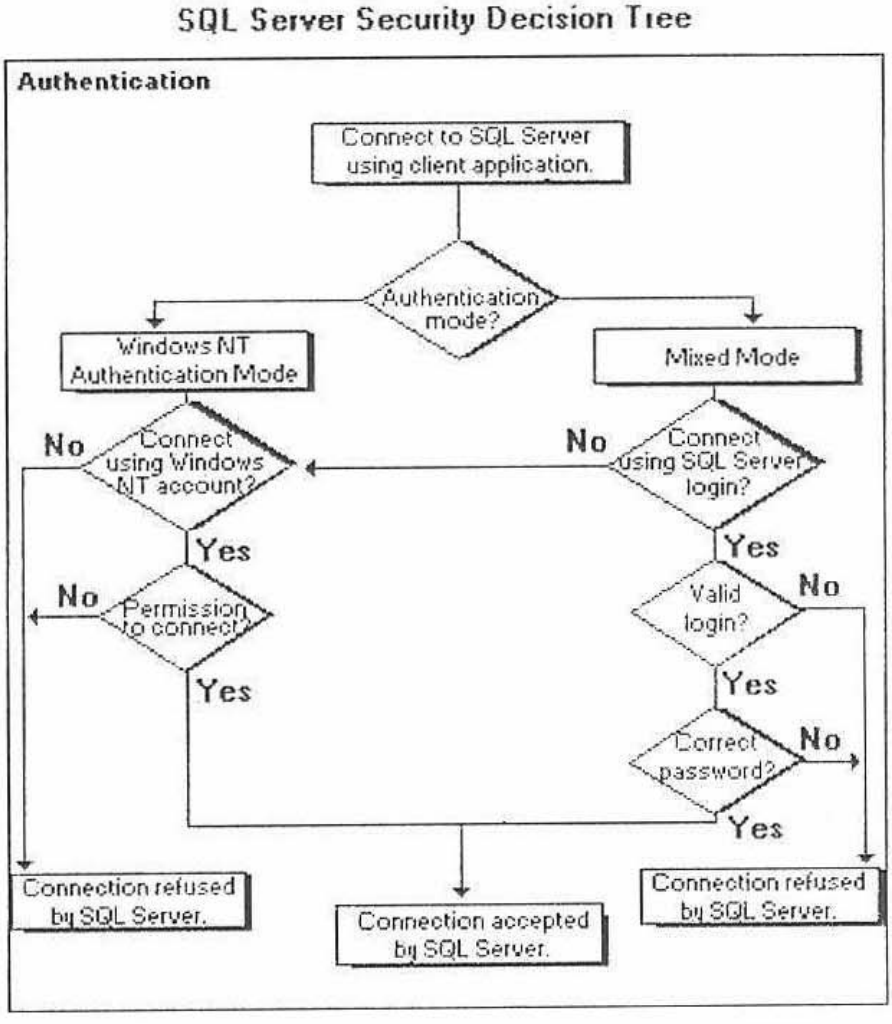
تأیید صلاحیت توسط ویندوز NT نسبت به تأیید صلاحیت توسط SQL Server در جامعیت آن با سیستم امنیتی ویندوز NT مزایای خاصی دارد. سیستم امنیتی ویندوز NT، ویژگی‌های بیشتری از جمله تأیید صلاحیت و رمزگذاری کلمه‌های عبور به صورت ایمن‌تر، بررسی تاریخ انقضاء کلمه عبور، حداقل طول کلمه عبور و بستن حساب پس از چند Login نامعتبر را داراست.

توجه: حالت تأیید صلاحیت توسط ویندوز NT، هنگام اجرای SQL Server بر روی ویندوز ۹۵/۹۸ قابل دسترسی نیست.

تأیید صلاحیت توسط SQL Server به منظور سازگاری با نسخه‌های قبلی ایجاد شده، زیرا برنامه‌های کاربردی که با نسخه‌های قبلی SQL Server نوشته شده‌اند ممکن

است به استفاده از Loginها و کلمه‌های عبور SQL Server نیازمند باشند. علاوه بـر ایـن هنگام اجرای SQL Server بر روی ویندوز ۹۵/۹۸، تأیید صلاحیت توسـط SQL Server مورد نیاز است زیرا حالت تأیید صلاحیت توسـط ویـنـدوز NT بـر روی ویـنـدوز ۹۵/۹۸ پشتیبانی نمی‌شود، بنابراین SQL Server هنگام اجـرا بـر روی ویـنـدوز ۹۵/۹۸ از حـالت مخلوط استفاده می‌کند اما فقط تأیید صلاحیت توسط SQL Server را پشتیبانی می‌کند.

توسعه‌دهندگان برنامه‌های کاربردی و کاربران پایگاه داده ممکن اسـت بـه دلیـل آشنایی با عملکرد Login و کلمه عبور، تأیید صلاحیت توسـط SQL Server را ترجیـح دهند. تأیید صلاحیت توسط SQL Server برای برقراری ارتبـاط بـا اینـترنت و سـرویس گیرنده‌هایی به غیر از سرویس گیرنده‌های ویندوز NT مورد نیاز است.



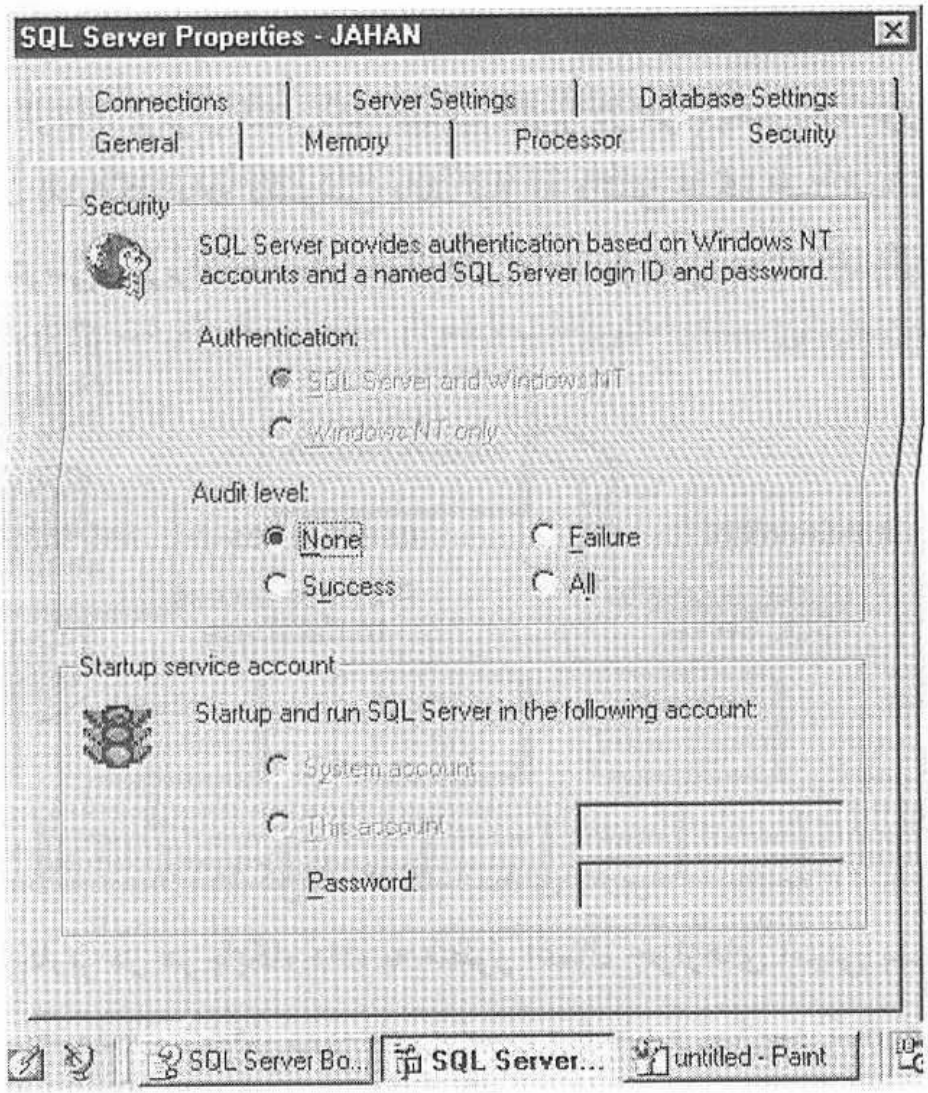
«شکل ۳-۶: درخت تصمیم امنیت در SQL Server»

تنظیم حالت تأیید صلاحیت توسط ویندوز NT

۱- یک گروه سرویس دهنده را باز کنید.

۲- روی یک سرویس دهنده رایت کلیک کنید و سپس properties را انتخاب کنید.

۳- روی دکمه **Security** کلیک کنید.



«شکل ۴-۶: انتخاب تأیید صلاحیت توسط ویندوز NT»

۴- **Windows NT Only** را انتخاب کنید.

۵- در قسمت **Audit Level**، سطحی را که دستیابی کاربر به **SQL Server** در فایل ثبت خطا درج می‌شود انتخاب کنید.

None: هیچ چیزی ثبت نمی‌شود.

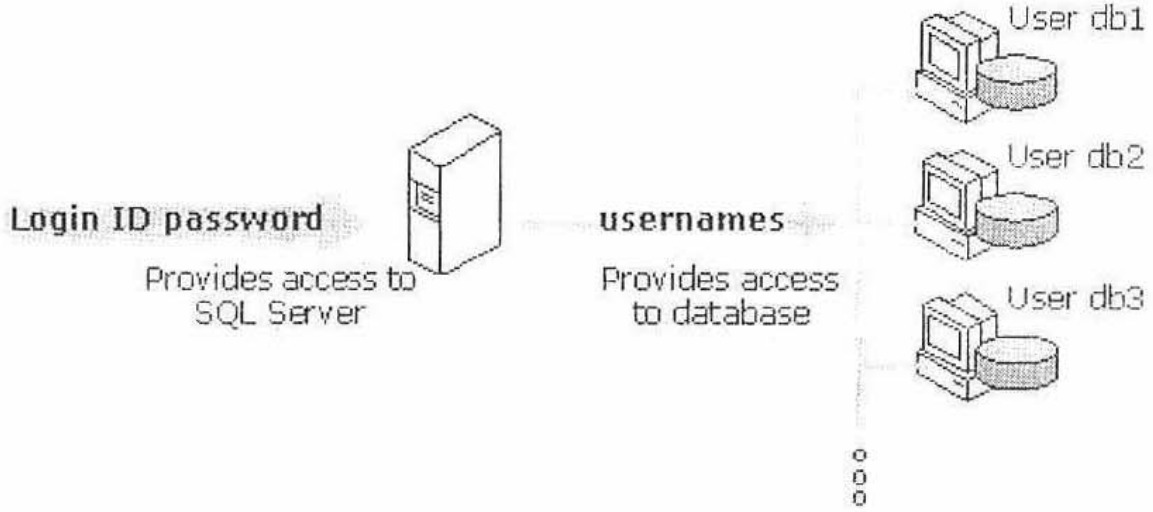
Success: باعث می‌شود که فقط اتصال‌های موفقیت آمیز ثبت شود.

Failure: باعث می‌شود که فقط اتصال‌های ناموفق ثبت شود.

ALL: باعث می‌شود که هم اتصال‌های موفق و هم اتصال‌های ناموفق ثبت شود.

برای تنظیم حالت مخلوط، تمام مراحل مانند حالت تأیید صلاحیت توسط ویندوز NT است با این تفاوت که درصفحه **Security**، باید **SQL Server and Windows NT** را انتخاب کنید.

پس از این که کاربر، تأیید صلاحیت شد و اجازه یافت تا به کمک **Login** خود با **SQL Server** ارتباط برقرار کند، یک حساب کاربر در هر پایگاه داده مورد نیاز است تا کاربر از طریق آن به پایگاه داده دسترسی یابد. نیاز به داشتن یک حساب کاربر در هر پایگاه داده از دسترسی کاربران به تمام پایگاه‌های داده سرویس دهنده جلوگیری می‌کند.



«شکل ۵-۶: حساب‌های کاربر»

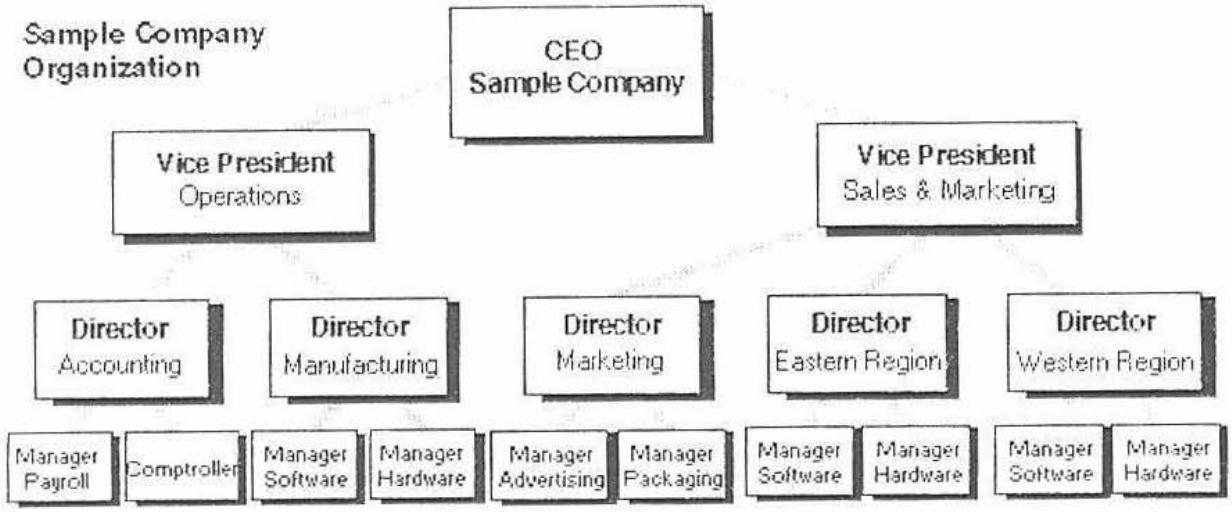
حساب کاربر در هر پایگاه داده به منظور اعمال جوازهای امنیتی برای شیءهای پایگاه داده (مانند جدول‌ها، دیدها، رویه‌های ذخیره شده و ...) به کار می‌رود. تمام فعالیت‌هایی که یک کاربر در پایگاه داده انجام می‌دهد از طریق دستورات **Transact-SQL** با **SQL Server** ارتباط برقرار می‌کند. وقتی **SQL Server** یک دستور **Transact-SQL** را دریافت می‌نماید از اجازه کاربر جهت اجرای این دستور در پایگاه‌های داده اطمینان حاصل می‌کند. اگر کاربر چنین جوازی نداشته باشد، **SQL Server** یک پیغام خطای جوازها را برمی‌گرداند.

مدیریت امنیت سلسله مراتبی

محیط امنیتی در ویندوز **NT** و **SQL Server** از طریق یک سیستم سلسله مراتبی کاربران، ذخیره، مدیریت و اعمال می‌شود. برای تسهیل مدیریت تعداد زیادی از کاربران، ویندوز **NT** و **SQL Server** از گروه‌ها و نقش‌ها استفاده می‌کنند. گروه عبارت است از یک واحد مدیریتی در ویندوز **NT** که حاوی کاربران ویندوز **NT** یا سایر گروه‌های آن است. نقش عبارت است از یک واحد مدیریتی در **SQL Server** که حاوی **Login**های

ویندوز NT، گروه‌ها یا سایر نقش‌هاست. قرار گرفتن کاربران در داخل گروه‌ها و نقش‌ها باعث اعطاء یا باز پس‌گیری ساده‌تر جوازها به بسیاری از کاربران می‌گردد. تنظیمات امنیتی تعریف شده برای یک گروه، به تمام اعضای آن گروه اعمال می‌شود. وقتی یک گروه، عضوی از یک گروه در سطح بالاتر باشد، تمام اعضای این گروه، تنظیمات امنیتی تعریف شده برای خود آن گروه یا حساب‌های کاربر را نیز دارا هستند.

نمودار سازمانی سیستم امنیتی اغلب با نمودار سازمانی یک شرکت متناظر می‌باشد.



«شکل ۶-۶: مقایسه سیستم امنیتی با نمودار سازمانی»

۳-۶- تنظیم حساب‌های امنیتی

یک Login به یک حساب کاربر در پایگاه داده‌ای که Login باید به آن دسترسی داشته باشد نگاشته می‌شود. اگر هیچ حساب کاربری در یک پایگاه داده وجود نداشته باشد، کاربر نمی‌تواند به پایگاه داده دسترسی داشته باشد حتی اگر بتواند به SQL Server متصل شود.

حساب‌های کاربر SQL Server که به Loginهای ایجاد شده در ویندوز NT یا SQL Server نگاشته می‌شوند و اجازه دسترسی به پایگاه داده می‌یابند همواره در داخل هر پایگاه داده SQL Server ایجاد می‌شوند.

قراردادهای مربوط به Login ها، کاربران، نقش‌ها و کلمه‌های عبور SQL Server

Loginها، کاربران، نقش‌ها و کلمه‌های عبور در SQL Server می‌توانند حاوی ۱ تا ۱۲۸ کاراکتر شامل حروف، علائم و ارقام باشند (مانند Andrew-Fuler،

Margaret Peacock یا 13&#57abc). بنابراین اسامی کاربران ویندوز NT یا ویندوز ۹۵/۹۸ می‌توانند به عنوان Loginهای SQL Server به کار روند، لیکن اگر Login، کاربر، نقش یا کلمه عبور در میان علائم (") یا ([]) قرار گیرند، علائم بخصوصی می‌توانند فقط در دستورات Transact-SQL به کار روند. اگر Login، کاربر، نقش یا کلمه عبور SQL Server شامل یکی از علائم زیر باشد از محدود کننده‌ها در دستورات Transact-SQL استفاده کنید:

- یک کاراکتر جای خالی بوده یا با آن شروع شود.
- با کاراکتر $ یا @ آغاز شود.

علاوه بر این، کاربر یا نقش در SQL Server:

- نمی‌تواند محتوی یک کاراکتر (\) باشد مگر آن که به کاربر یا گروهی در ویندوز NT ارجاع کند.
- نمی‌تواند قبلاً در پایگاه داده جاری (و فقط برای Loginها در master) وجود داشته باشد.
- نمی‌تواند NULL یا یک رشته خالی ("") باشد.

فعال کردن یک کاربر یا گروه ویندوز NT

به حساب‌های موجود در ویندوز NT (کاربران یا گروه‌ها) قبل از آن که بتوانند به یک پایگاه داده دسترسی یابند، جوازهایی برای اتصال به SQL Server اعطاء می‌شود. اگر تمام اعضای یک گروه ویندوز NT بخواهند به SQL Server متصل شوند، می‌توانید جواز اتصال به SQL Server را به کل گروه اعطاء نمایید. مدیریت جوازهای گروه بسیار ساده‌تر از مدیریت جوازهای تک تک کاربران است. اگر اعطای جواز به کل گروه ویندوز NT ضرورتی ندارد، می‌توانید جواز اتصال به SQL Server را برای هر کاربر ویندوز NT اعطاء نمایید.

هنگام اعطاء یک دستیابی کاربر ویندوز NT برای اتصال به SQL Server، ابتدا میدان یا نام کامپیوتر را که کاربر به آن تعلق دارد، سپس یک علامت (\) و در نهایت نام کاربر را مشخص کنید. مثلاً برای اعطاء دسترسی کاربری با نام Andrew در میدان LONDON عبارت LONDON\Andrew را به عنوان نام کاربر تعیین کنید.

در ویندوز NT دو نوع گروه وجود دارد: محلی و عمومی. گروه‌های عمومی حاوی حساب‌های کاربری از میدان سرویس دهنده ویندوز NT هستند که در آن ایجاد شده‌اند؛ بنابراین گروه‌های عمومی حاوی گروه‌ها یا کاربران دیگر از میدان‌های دیگر نمی‌باشند و نمی‌توانند بر روی کامپیوتری که ویندوز NT سرویس گیرنده در حال اجراست ایجاد شوند. گروه‌های محلی می‌توانند حاوی حساب‌های کاربر و گروه‌های عمومی از میدانی باشند که در آن ایجاد شده‌اند و یا این که در میدان‌های مطمئن دیگر باشند.

توجه: میدان‌های مطمئن مفهومی در معماری ویندوز NT می‌باشد و منظور میدان‌هایی است که با میدان جاری رابطه اطمینان دارند.

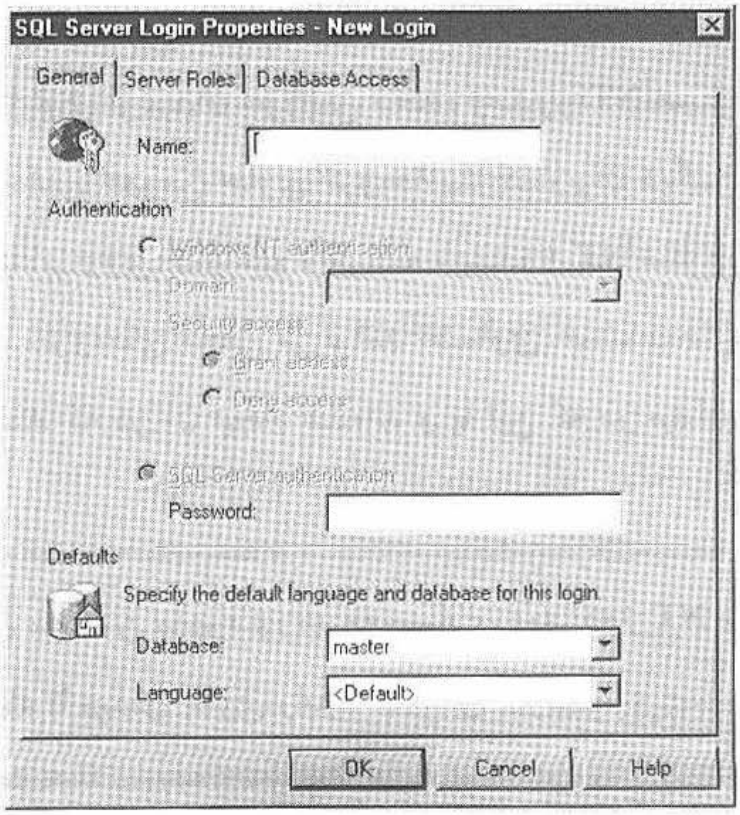
گروه‌های محلی نمی‌توانند حاوی گروه‌های محلی دیگر باشند. به علاوه، ویندوز NT حاوی گروه‌های محلی توکار از پیش تعریف شده‌ای نظیر Administrators، Users و Guests می‌باشد. به طور پیش فرض، این گروه‌های توکار تا زمانی که صراحتاً حذف نشده‌اند، همواره بر روی هر کامپیوتر ویندوز NT قابل دسترس می‌باشند. هنگام اعطاء دسترسی یک گروه محلی یا عمومی ویندوز NT برای اتصال به SQL Server، ابتدا میدان یا نام کامپیوتری که گروه در آن تعریف شده، سپس یک علامت (\) و در نهایت نام گروه را مشخص کنید اما برای اعطای دسترسی به یک گروه محلی توکار ویندوز NT، به جای تعیین میدان یا نام کامپیوتر، کلمه BUILTIN را به کار ببرید، مثلاً برای اعطاء دسترسی به گروه محلی توکار Administrators، عبارت BUILTIN\Administrators را به عنوان نام گروه جهت افزودن به SQL Server به کار ببرید.

اعطاء دسترسی یک گروه یا کاربر ویندوز NT به SQL Server

۱- یک گروه سرویس‌دهنده را باز کنید. سپس یک سرویس دهنده را باز کنید.

۲- Security را باز کنید. روی Logins رایت کلیک کنید و سپس روی New Login کلیک کنید.

۳- Windows NT Authentication را انتخاب کنید.



«شکل ۷-۶: اعطای توانایی دستیابی به SQL Server»

۴- در قسمت **Name**، حساب ویندوز **NT** را وارد کنید (به شکل **DOMAIN\User**)

۵- به دلخواه:

- در قسمت **Database**، پایگاه داده پیش فرضی که کاربر پس از اتصال به **SQL Server** با آن ارتباط برقرار می‌کند مشخص کنید.
- در قسمت **Language**، زبان پیش فرضی که پیغام‌ها به آن زبان برای کاربر نمایش پیدا کند، انتخاب کنید.

اعطاء دسترسی یک کاربر یا گروه ویندوز NT به یک پایگاه داده

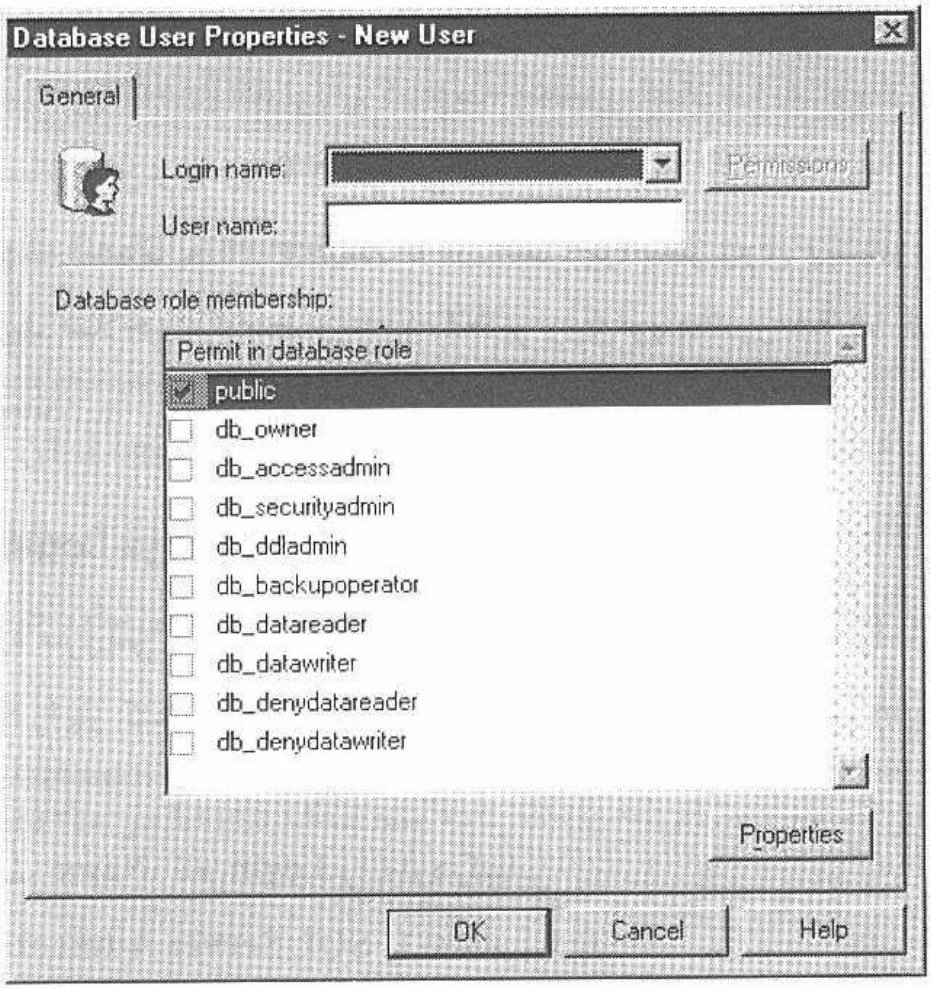
یک کاربر یا گروه ویندوز **NT** به منظور دسترسی به یک پایگاه داده **SQL Server** باید دارای یک حساب کاربر متناظر در هر پایگاه داده‌ای باشد که می‌خواهد به آن دسترسی یابد. ضرورتی ندارد که به ازاء هر کاربر موجود در یک گروه ویندوز **NT** که تمام اعضایش اعمال مشابه انجام می‌دهند، یک حساب کاربر در پایگاه داده اضافه کنید بلکه فقط کافی است یک حساب برای کل گروه ایجاد نمایید. در صورتی که یک کاربر ویندوز **NT**، فعالیتی متفاوت با سایر اعضای یک گروه ویندوز **NT** داشته باشد، باید به طور جداگانه به پایگاه داده اضافه شود.

اعطاء دستیابی به یک پایگاه داده به کاربر یا گروه ویندوز NT

۱- یک گروه سرویس دهنده را باز کنید، سپس یک سرویس دهنده را باز کنید.

۲- **Databases** را باز کنید، سپس پایگاه داده‌ای را که می‌خواهید کاربر یا گروه به آن دسترسی یابد، باز کنید.

۳- روی **Users** رایت کلیک کنید؛ سپس روی **New Database User ...** کلیک کنید.



«شکل ۸-۶: اعطاء دستیابی به پایگاه داده»

۴- در قسمت **Login Name**، نام کاربر یا گروه ویندوز NT را که می‌خواهید دستیابی به پایگاه داده را به آن اعطاء کنید، انتخاب نمایید.

۵- به دلخواه، در قسمت **User Name**، نام کاربر یا گروه را وارد کنید.

۶- به دلخواه، علاوه بر نقش **public**، یک نقش پایگاه داده را نیز انتخاب کنید.

اضافه کردن یک Login در SQL Server

اگر SQL Server برای کار در حالت مخلوط تنظیم شده باشد یا زمانی که SQL Server بر روی ویندوز ۹۵/۹۸ درحال اجراست، افزودن حساب‌های **Login** در

SQL Server که به کمک یک نام Login و کلمه عبور اجازه اتصال را می‌دهند، روش بهتری نسبت به استفاده از یک حساب کاربر یا گروه ویندوز NT جهت تأیید کاربران است.

افزودن یک Login در SQL Server می‌تواند در موارد زیر سودمند باشد:

- جهت سازگاری با برنامه‌های کاربردی ساخته شده برای نسخه‌های قبلی SQL Server که به منظور کار با حساب‌های ویندوز NT طراحی نشده‌اند.
- جهت برنامه‌های کاربردی که به منظور کار با کاربران عمومی فاقد حساب‌های ویندوز NT طراحی شده‌اند.
- جهت اتصال به SQL Server در حال اجرا بر روی ویندوز ۹۵/۹۸، زیرا تأیید صلاحیت توسط ویندوز NT بر روی ویندوز ۹۵/۹۸ قابل دسترسی نیست.

افزودن یک Login در SQL Server

۱- یک گروه سرویس دهنده را باز کنید، سپس یک سرویس دهنده را باز کنید.

۲- Security را باز کنید، روی Logins رایت کلیک کنید و سپس روی New Login کلیک کنید.

۳- در قسمت Name، نام Login را وارد کنید.

۴- SQL Server Authentication را انتخاب کنید.

۵- به دلخواه، در قسمت Password، یک کلمه عبور وارد کنید.

۶- کلمه عبور را تأیید کنید.

Login مدیر سیستم (sa)

sa یک Login مخصوص است که به طور پیش فرض به نقش ثابت سرویس‌دهنده sysadmin منتسب شده و قابل تغییر نمی‌باشد. اگر چه sa یک Login مدیر توکار است، لیکن نباید به صورت عادی استفاده شود. از sa فقط هنگامی استفاده کنید که هیچ راهی برای Login کردن به SQL Server وجود نداشته باشد، مثلاً وقتی که مدیران دیگر سیستم در دسترس نباشند یا کلمه‌های عبورشان را فراموش کرده باشند.

توجه: هنگام نصب SQL Server به sa کلمه عبوری منتسب نمی‌شود، بنابراین به منظور جلوگیری از دستیابی غیر مجاز به SQL Server به کمک sa توصیه می‌شود که بلافاصله یک کلمه عبور به آن منتسب گردد.

اعطاء جواز دسترسی به پایگاه داده به یک Login

برای هر Login که می‌خواهد به پایگاه داده دسترسی داشته باشد یک حساب کاربر SQL Server باید به آن پایگاه داده اضافه شود (همانند افزودن یک حساب کاربر برای یک کاربر یا گروه ویندوز NT).

توجه: برای اعطاء جواز دسترسی به پایگاه داده به یک Login، آن Login باید از قبل موجود باشد. این امر با کاربران یا گروه‌های ویندوز NT از این جهت متفاوت است که به آنها قبل از اعطاء جواز اتصال به SQL Server می‌توان جواز دسترسی به یک پایگاه داده را اعطاء نمود.

مراحل اعطاء دستیابی به پایگاه داده به یک Login در SQL Server، همانند اعطای دستیابی به کاربر یا گروه ویندوز NT می‌باشد.

مالک پایگاه داده (dbo)

هر عضو از نقش ثابت سرویس دهنده sysadmin در یک پایگاه به یک کاربر بخصوص نگاشته می‌شود که dbo خوانده می‌شود. هر شیء‌ای که توسط عضوی از sysadmin ایجاد شود، به طور خودکار متعلق به dbo است، مثلاً اگر کاربر Andrew عضوی از sysadmin باشد و جدولی با نام T1 ایجاد کند، T1 به dbo تعلق دارد و به عنوان dbo.T1 مشخص می‌شود نه Andrew.T1، اما اگر Andrew فقط عضوی از نقش ثابت پایگاه داده db_owner باشد و عضوی از sysadmin نباشد، T1 به Andrew تعلق دارد و به عنوان Andrew.T1 مشخص می‌گردد. dbo قابل حذف شدن نیست.

شئ‌های پایگاه داده عبارتند از: جدول‌ها، شاخص‌ها، دیدها، Triggerها و رویه‌های ذخیره شده. کاربری که شئ‌ای از پایگاه داده را ایجاد می‌کند، مالک آن شئ پایگاه داده است. مالک پایگاه داده یا مدیر سیستم در ابتدا باید جواز ایجاد نوع بخصوصی از شئ را به کاربر اعطاء کند؛ سپس مالک شئ پایگاه داده می‌تواند یک شئ را ایجاد کرده و جواز استفاده از آن را به سایر کاربران اعطاء نماید.

مالک‌های شئ پایگاه داده، شناسه‌های Login یا کلمه‌های عبور ندارند و به آنها به طور ضمنی تمام جوازها داده شده است اما باید قبل از دسترسی سایر کاربران به آن شئ، جوازها را به آنها اعطاء نمایند.

وقتی کاربران به شئ‌ای که توسط کاربر دیگری ایجاد شده دسترسی می‌یابند، آن شئ باید با نام مالکش مقید گردد وگرنه SQL Server ممکن است آن شئ را نشاسد، زیرا ممکن است شئ‌های همنام زیادی با مالک‌های مختلف وجود داشته باشند.

اگر به شئ‌ای بدون نام مالکش ارجاع گردد (مثلاً my-table به جای owner.my-table)، SQL Server به ترتیب زیر به دنبال آن شئ در پایگاه داده می‌گردد:

- مالکیت بوسیله کاربر جاری.
- مالکیت بوسیله dbo.

و اگر باز هم آن شئ پیدا نشد، یک پیغام خطا بر می‌گردد.

اگر یک مالک شئ پایگاه داده قرار است از پایگاه داده حذف گردد، ابتدا باید شئ‌های مربوط به آن مالک حذف شوند یا آن که مالکیت آنها به کاربر دیگری واگذار شود.

کاربر میهمان (guest)

حساب کاربر guest به یک Login اجازه می‌دهد تا بدون یک حساب کاربر به یک پایگاه داده دسترسی داشته باشد. یک login، موجودیت کاربر guest را در صورتی می‌پذیرد که تمام شرایط زیر محقق شود:

- آن Login به SQL Server دسترسی داشته باشد اما از طریق حساب کاربر خودش قابلیت دسترسی به پایگاه داده را نداشته باشد.

- پایگاه داده حاوی یک حساب کاربر **guest** باشد.

کاربر **guest** می‌تواند حذف شود و به پایگاه‌های داده بجز **master** و **tempdb** اضافه شود. به طور پیش فرض در پایگاه‌های داده‌ای که جدیداً بوجود می‌آیند، حساب کاربر **guest** وجود ندارد.

ایجاد نقش‌های پایگاه داده که بوسیله کاربر تعریف می‌شوند

وقتی یک گروه از کاربران، نیازمند انجام یک رشته اعمال بخصوص در **SQL Server** هستند و هیچ گروه ویندوز **NT** قابل کاربردی نیز وجود ندارد، یا زمانی که شما جوازهای مدیریت حساب‌های کاربر ویندوز **NT** را در اختیار ندارید، می‌توانید در پایگاه داده یک نقش اضافه کنید.

مزایای استفاده از نقش‌های پایگاه داده به قرار زیر است:

- برای تمام کاربران، بیش از یک نقش پایگاه داده می‌تواند در یک لحظه فعال باشد.
- نقش‌های **SQL Server** می‌توانند حاوی کاربران و گروه‌های ویندوز **NT**، همچنین کاربران **SQL Server** و سایر نقش‌ها باشند.
- یک کاربر می‌تواند به بیش از یک نقش در یک پایگاه داده تعلق داشته باشد.

از آنجایی که یک نقش، شیءای از پایگاه داده نیست، چند نقش همنام در یک پایگاه داده بوسیله چند کاربر (مالک) مختلف نمی‌توانند ایجاد گردند.

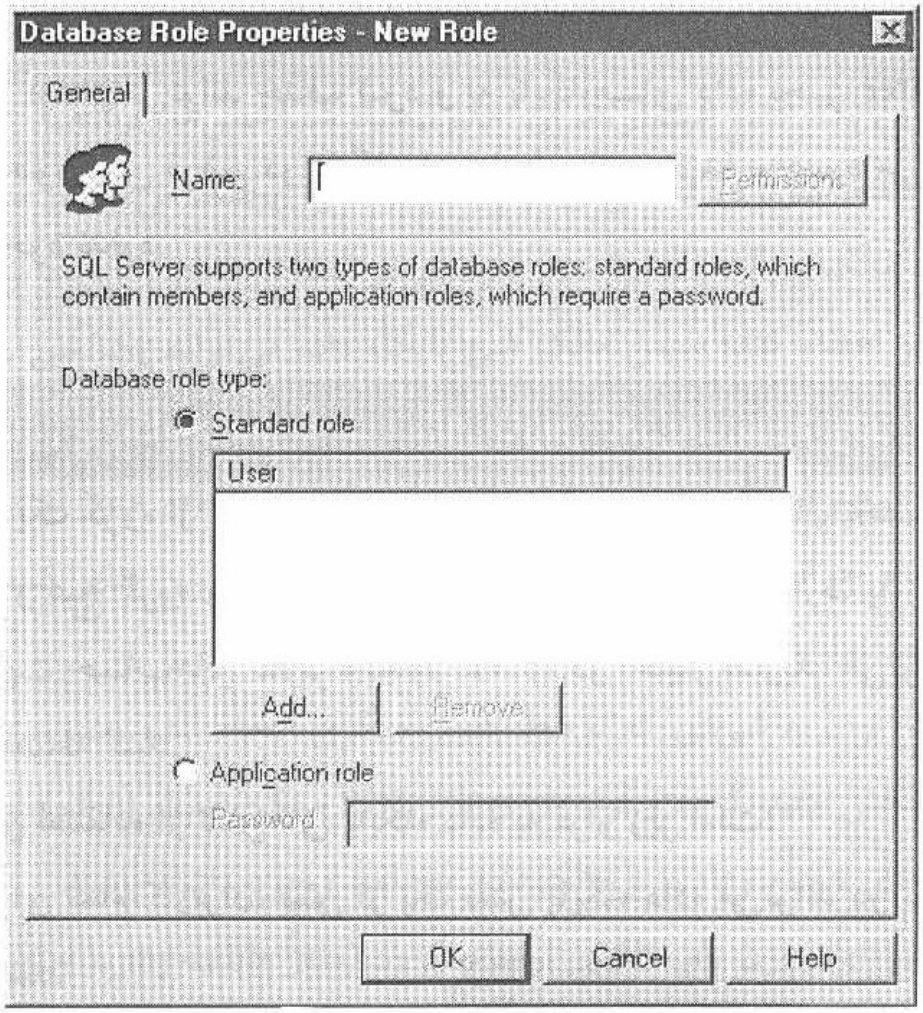
ایجاد یک نقش پایگاه داده

۱- یک گروه سرویس دهنده را باز کنید؛ سپس یک سرویس دهنده را باز کنید.

۲- **Databases** را باز کنید. پایگاه داده‌ای که نقش مورد نظر باید در آن ایجاد شود را نیز باز کنید.

۳- روی **Roles**، رایت کلیک کنید؛ سپس روی **New Database Role...** کلیک کنید.

۴- در قسمت **Name**، نام نقش جدید را وارد کنید.



«شکل ۹-۶: ایجاد یک نقش پایگاه داده»

۵- به دلخواه روی **...Add** کلیک کنید تا عضوهایی به نقش اضافه شود. فقط کاربران پایگاه داده جاری می‌توانند به نقش افزوده شوند.

اضافه کردن یک عضو به یک نقش پایگاه داده

هنگامی که حساب کاربر جدیدی را در **SQL Server** اضافه می‌کنید یا نیازمند اعمال تغییراتی در جوازهای یک کاربر موجود هستید می‌توانید به جای آن که مستقیماً جوازها را برای حساب تغییر دهید، کاربر را به یک نقش پایگاه داده اضافه کنید. کاربران **SQL Server**، کاربران ویندوز **NT**، گروه‌های ویندوز **NT** و سایر نقش‌های پایگاه داده می‌توانند به عنوان عضوی از یک نقش اضافه شوند.

از آنجایی که نقش به یک پایگاه داده محدود می‌شود، شما می‌توانید کاربران، گروه‌ها و نقش‌های بخصوص را فقط به همان پایگاه داده اضافه کنید. افزودن کاربران، گروه‌ها یا نقش‌ها از یک پایگاه داده به یک نقش در پایگاه داده دیگر امکان پذیر نیست.

یک حساب کاربر می‌تواند عضوی از نقش‌های مختلف در یک پایگاه داده باشد. مثلاً یک کاربر SQL Server می‌تواند عضو نقش admin و نقش users از یک پایگاه داده باشد که هر نقش، جوازهای مختلفی را اعطاء می‌کند. مثلاً نقش admin ممکن است جواز دستیابی به یک جدول را اعطاء کند در حالی که نقش users دستیابی به همان جدول را لغو کند.

افزودن یک عضو به یک نقش پایگاه داده

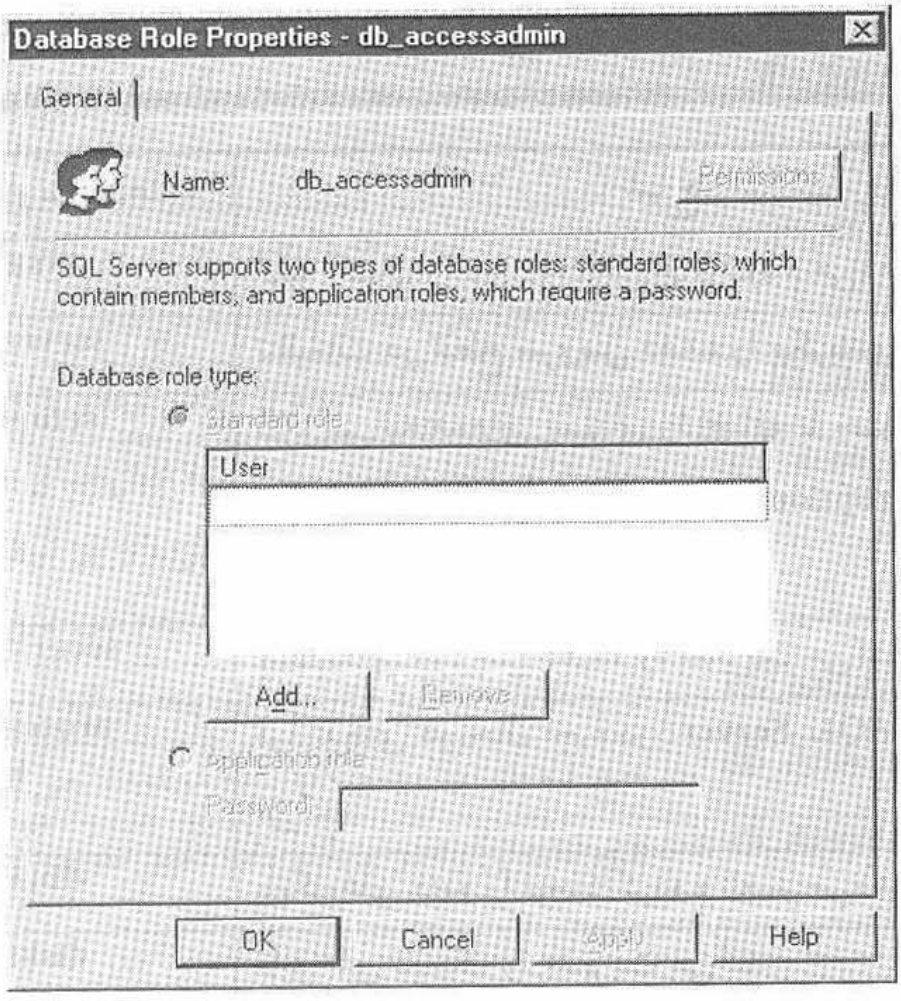
۱- یک گروه سرویس‌دهنده را باز کنید، سپس یک سرویس‌دهنده را باز کنید.

۲- Databases را باز کنید. پایگاه داده‌ای که نقش مورد نظر در آن قرار دارد را نیز باز کنید.

۳- روی Roles کلیک کنید.

۴- در پانل details روی نقش مورد نظر رایت کلیک کرده و سپس properties را انتخاب کنید.

۵- روی ... Add کلیک کنید و کاربرانی را به نقش اضافه کنید. فقط کاربران موجود در پایگاه داده جاری می‌توانند به نقش اضافه شوند.



«شکل ۱۰-۶: افزودن یک عضو به یک نقش پایگاه داده»

اضافه کردن یک عضو به یک نقش از پیش تعریف شده

مکانیزم امنیتی در SQL Server شامل تعدادی نقش از پیش تعریف شده با جوازهای بخصوصی است که نمی‌تواند به سایر حساب‌های کاربر اعطاء شوند. اگر شما دارای کاربرانی هستید که برای انجام اعمال بخصوصی نیازمند جوازهایی هستند که توسط این نقش‌ها پشتیبانی می‌گردند، باید حساب‌های آنها را به این نقش‌های از پیش تعریف شده اضافه کنید.

دو نوع نقش از پیش تعریف شده عبارتند از: ثابت سرویس‌دهنده و ثابت پایگاه داده. نقش‌های ثابت سرویس‌دهنده مانند sysadmin در سطح سرویس‌دهنده تعریف می‌شوند و خارج از پایگاه‌های داده وجود دارند. برای اضافه کردن یک کاربر به عنوان عضوی از یک نقش ثابت سرویس دهنده، کاربر باید یک حساب Login در SQL Server یا ویندوز NT داشته باشد.

توجه: کاربران ویندوز NT که عضو گروه BUILTIN\Administrator هستند، به طور خودکار عضو نقش ثابت سرویس دهنده sysadmin نیز می‌باشند.

نقش ثابت سرویس دهنده	شرح
sysadmin	کلیه اعمال را در SQL Server انجام می‌دهد.
serveradmin	تنظیمات در سطح سرویس دهنده را پیکربندی می‌کند.
setupadmin	سرویس دهنده‌های پیوندی را اضافه یا حذف کرده و تعدادی رویهٔ ذخیره شده مانند sp_serveroption را اجرا می‌کند.
securityadmin	loginهای سرویس دهنده را مدیریت می‌کند.
processadmin	فرآیندهای در حال اجرا در SQL Server را مدیریت می‌کند.
dbcreator	پایگاه‌های داده را ایجاد کرده و تغییر می‌دهد.
diskadmin	فایل‌های دیسک را مدیریت می‌کند.

توجه: امکان ایجاد نقش‌های از نوع ثابت سرویس دهنده وجود ندارد. نقش‌ها فقط می‌توانند در سطح پایگاه داده ایجاد شوند.

نقش‌های از نوع ثابت پایگاه داده مانند db_owner در سطح پایگاه داده تعریف شده و در هر پایگاه داده‌ای وجود دارند. شما می‌توانید هر حساب کاربر معتبری (کاربر یا گروه ویندوز NT یا کاربر یا نقش SQL Server) را به عنوان عضوی از یک نقش ثابت پایگاه داده اضافه کنید. هر عضو، جوازهای به کار رفته برای آن نقش ثابت پایگاه داده را بدست می‌آورد.

نقش ثابت پایگاه داده	شرح
db_owner	اعمال مربوط به تمام نقش‌های پایگاه داده به علاوه فعالیت‌های مربوط به نگهداری و پیکربندی در پایگاه داده را انجام می‌دهد.
db_accessadmin	گروه‌های ویندوز NT، کاربران ویندوز NT و کاربران SQL Server را در پایگاه داده حذف یا اضافه می‌کند.
db_datareader	داده‌های تمام جدول‌های کاربران را در پایگاه داده مشاهده می‌کند.
db_datawriter	داده‌ها را از تمام جدول‌های کاربر در پایگاه داده حذف کرده، تغییر می‌دهد و یا اضافه می‌کند.
db_ddladmin	شئ‌های پایگاه داده را اضافه یا حذف کرده و یا تغییر می‌دهد.
db_securityadmin	نقش‌ها و عضوهای مربوط به نقش‌های پایگاه داده در SQL Server را مدیریت کرده و می‌تواند جوازهای دستور و شئ را در پایگاه داده مدیریت نماید.
db_backupoperator	از پایگاه داده، نسخه پشتیبان تهیه می‌کند.
db_denydatareader	هیچ داده‌ای را در پایگاه داده مشاهده نمی‌کند.
db_denydatawriter	هیچ داده‌ای را در پایگاه داده تغییر نمی‌دهد.

نقش عمومی

نقش عمومی، یک نقش پایگاه داده ویژه است که به هر کاربر پایگاه داده تعلق دارد. خواص این نقش به صورت زیر است:

- این نقش، تمام جوازهای پیش فرض برای کاربران را در یک پایگاه داده تصاحب می‌کند.
- نمی‌توان کاربران، گروه‌ها یا نقش‌هایی را به آن منتسب کرد زیرا آنها به طور پیش فرض به این نقش تعلق دارند.
- در هر پایگاه داده از جمله **master**، **msdb**، **tempdb**، **model** و تمام پایگاه‌های داده کاربر شامل است.
- نمی‌تواند حذف گردد.

۴-۶- مدیریت حساب‌های امنیتی

پس از افزودن حساب‌های امنیتی به SQL Server، می‌توانید این حساب‌ها را در پایگاه داده مشاهده یا حذف کرده و یا این که آنها را تغییر دهید.

مشاهده Loginها

مشاهده Loginهای SQL Server به شما کمک می‌کند مشخص کنید آیا یک کاربر یا گروه ویندوز NT جواز اتصال به SQL Server را دارد یا خیر و این که آن Login به کدام پایگاه‌های داده می‌تواند دسترسی داشته باشد. مشاهده یک **Login** قبل از حذف آن می‌تواند برای تشخیص این که کدام کاربران پایگاه داده باید ابتدا حذف شوند به کار رود زیرا برای حذف یک **Login** ابتدا باید کاربران مرتبط با آن حذف شوند.

اطلاعاتی که درباره هر **Login** می‌توان مشاهده کرد عبارتند از:

- کاربران مرتبط با **Login** در هر پایگاه داده.
- پایگاه داده و زبان پیش فرض وقتی که کاربر ابتدا به SQL Server متصل می‌شود.
- شناسه امنیتی ویندوز NT.

توجه: مشاهده کلمه عبور Login امکان‌پذیر نیست مگر آن که NULL باشد. کلمه‌های عبور هنگام ذخیره سازی در SQL Server، رمزگذاری می‌شوند.

مشاهده یک Login در SQL Server و یا کاربر یا گروه ویندوز NT

۱- یک گروه سرویس‌دهنده را باز کنید؛ سپس یک سرویس‌دهنده را باز کنید.
۲- Security را باز کنید و سپس روی Logins کلیک کنید.
۳- در پانل details، روی Login مورد نظر کلیک کنید؛ سپس روی properties کلیک کنید.

مشاهده کاربران پایگاه داده

مشاهده یک حساب کاربر SQL Server در یک پایگاه داده، منجر به نمایش نقش‌هایی که کاربر عضو آنها است، Login مرتبط با کاربر و پایگاه داده پیش فرض می‌گردد.

مشاهده یک کاربر پایگاه داده

۱- یک گروه سرویس‌دهنده را باز کنید؛ سپس یک سرویس‌دهنده را باز کنید.
۲- Databases را باز کنید؛ سپس پایگاه داده‌ای که کاربر در آن قرار دارد را باز کنید.
۳- روی User کلیک کنید.
۴- در پانل details، روی نام کاربر، رایت کلیک کرده و سپس بر روی properties کلیک کنید.

تغییر Loginها

پس از ایجاد Loginها ممکن است تغییر کلمه عبور، پایگاه داده پیش فرض یا زبان پیش فرض لازم باشد. مثلاً یک کاربر ممکن است کلمه عبور خود را فراموش کرده باشد یا بخواهد به دلایل امنیتی کلمه عبور را تغییر دهد یا از پایگاه دادهٔ دیگری استفاده،

کند و یا این که پیغام‌ها را به زبان دیگری ببیند.

توجه: اگر کاربری، کلمه عبور خود را فراموش کند، عضوی از نقش ثابت سرویس‌دهنده sysadmin می‌تواند کلمه عبور او را بدون آن که کلمه عبور اولیه را بداند تغییر دهد اما خود کاربر در صورت فراموش کردن کلمه عبورش، قادر به تغییر آن نخواهد بود.

حذف نام‌ها و کاربران

حذف یک کاربر SQL Server یا کاربر یا گروه ویندوز NT از یک پایگاه داده SQL Server به طور خودکار منجر به حذف جوازهای تعریف شده برای آن کاربر یا گروه خواهد شد و از استفاده کاربر از پایگاه داده تحت حساب امنیتی قبلی جلوگیری خواهد کرد.

حذف یک کاربر باعث حذف خودکار یک Login نخواهد شد و بنابراین از اتصال به SQL Server جلوگیری نخواهد کرد. بعد از عمل حذف، کاربر می‌تواند فقط از طریق حساب guest با پایگاه‌های داده ارتباط برقرار کند.

برای جلوگیری از اتصال یک کاربر به SQL Server، Login او را حذف کنید.

برای حذف یک گروه یا کاربر از یک پایگاه داده و یا یک Login از SQL Server، بر روی کاربر، گروه و یا Login کلیک کرده و کلید Delete را فشار دهید. سپس عمل حذف را تأیید کنید.

باز پس گرفتن دستیابی به SQL Server از یک کاربر یا گروه ویندوز NT

۱- یک گروه سرویس‌دهنده را باز کنید؛ سپس یک سرویس‌دهنده را باز کنید.

۲- Security را باز کنید. سپس روی Logins کلیک کنید.

۳- در پانل details، روی نام کاربر یا گروه ویندوز NT رایت کلیک کنید و سپس کلید Delete را فشار دهید.

۴- عمل حذف را تأیید کنید.

جلوگیری از دسترسی Login به حساب‌های ویندوز NT

اگر یک کاربر ویندوز NT به گروهی از ویندوز NT تعلق داشته باشد که دارای یک حساب Login در SQL Server است، این کاربر اجازه اتصال از طریق Login گروه را خواهد داشت. شما می‌توانید از دسترسی Login به هر کاربر یا گروه ویندوز NT جلوگیری کنید تا بدینوسیله کاربرانی که حساب کاربرشان یا هر گروهی که به آن تعلق دارند و دسترسی Login آنها ابطال شده، نتوانند به SQL Server متصل شوند.

ابطال Login برای دسترسی به یک کاربر یا گروه ویندوز NT

۱- یک گروه سرویس‌دهنده را باز کنید؛ سپس یک سرویس‌دهنده را باز کنید.

۲- Security را باز کنید؛ سپس روی Logins کلیک کنید.

۳- در پانل details، روی کاربر یا گروه ویندوز NT رایت کلیک کرده و سپس روی properties کلیک کنید.

اگر گروه یا کاربر ویندوز NT در پانل ظاهر نمی‌شود، ابتدا به آن توانایی دسترسی به SQL Server را اعطاء کنید.

۴- در قسمت Security access، Deny access را انتخاب کنید.

مشاهده نقش‌ها

موضوعات متعددی در فرآیند ایجاد و استفاده از یک پایگاه داده هنگام یافتن اطلاعاتی درباره یک نقش پایگاه داده یا یک نقش ثابت سرویس دهنده وجود دارند؛ مانند مشاهده نقش‌های موجود در پایگاه داده جاری یا نقش‌های از نوع ثابت سرویس‌دهنده.

مشاهده نقش‌های پایگاه داده

۱- یک گروه سرویس‌دهنده را باز کنید؛ سپس یک سرویس‌دهنده را باز کنید.

۲- Databases را باز کنید؛ سپس پایگاه داده مورد نظر را باز کنید.

۳- روی Roles کلیک کنید.

مشاهدهٔ نقش ثابت سرویس دهنده

۱- یک گروه سرویس‌دهنده را باز کنید؛ سپس یک سرویس‌دهنده را باز کنید.
۲- **Security** را باز کنید؛ سپس روی **Server Roles** کلیک کنید.

مشاهده و تغییر اعضای نقش

ممکن است در حین استفاده از پایگاه داده، بخواهید اعضای یک نقش پایگاه داده یا یک نقش ثابت سرویس دهنده را مشاهده کنید. به علاوه، اگر یک کاربر **SQL Server** دیگر نیازی به جوازهای ثابت سرویس دهنده یا ثابت پایگاه داده‌ای که عضوی از آن می‌باشد، نداشته باشد می‌توانید به منظور حفظ سادگی سیستم امنیتی، کاربر را از نقش حذف کنید.

برای مشاهده اعضای یک نقش پایگاه داده یا نقش ثابت سرویس‌دهنده، بر روی نقش مورد نظر رایت کلیک کرده و سپس **properties** را انتخاب کنید. اگر بخواهید یک حساب کاربر یا **Login** را از نقش مزبور حذف کنید، حساب یا **Login** را انتخاب کرده و روی **Remove** کلیک کنید.

حذف یک نقش پایگاه داده

پس از حذف تمام کاربران موجود در یک نقش، می‌توانید نقش را حذف کنید. اگر احتمال می‌دهید که جوازهای نقش، مورد نیاز کاربر جدیدی می‌باشد، می‌توانید نقش‌های خالی را ذخیره کنید.

> **توجه: از سطوح عمیق نقش‌های تودرتو اجتناب کنید زیرا این امر بر روی کارآیی تأثیر می‌گذارد.**

برای حذف یک نقش پایگاه داده، روی نقش کلیک کرده و کلید **Delete** را فشار دهید سپس عمل حذف را تأیید کنید. نقش‌های ثابت سرویس‌دهنده قابل حذف نیستند.

۵-۶- مدیریت جوازها

وقتی کاربران به **SQL Server** متصل می‌شوند، فعالیت‌هایی که می‌توانند انجام

دهند توسط جوازهایی که به حساب‌های امنیتی آنها اعطاء می‌شود تعیین می‌گردد. کاربر باید جوازهای مناسبی برای انجام اعمالی که مستلزم تغییر تعریف پایگاه داده یا دسترسی به داده‌ها است داشته باشد. برخی از اعمال دیگر که مستلزم چنین تغییراتی نمی‌باشد به جواز احتیاج ندارند.

کار با داده‌ها یا اجرای یک رویه، نیازمند نوع خاصی از جواز موسوم به جوازهای شئ است. جوازهای شئ بر اساس یک جدول، دید یا رویه ذخیره شده است و قابلیت اجرای دستورات **SELECT**، **INSERT**، **UPDATE** و **DELETE** را روی جدول یا دید و جواز **EXECUTE** را بر روی یک رویه ذخیره شده کنترل می‌کند؛ مثلاً اگر کاربری بخواهد کل داده‌های یک جدول را بازیابی کند، جواز شئ **SELECT** باید به آن کاربر اعطاء شده باشد. جوازهای شئ به صورت زیر هستند:

- جوازهای دستور **SELECT**، **INSERT**، **UPDATE** و **DELETE** که می‌تواند برای کل جدول یا دید به کار روند.
- جوازهای دستور **SELECT** و **UPDATE** که می‌توانند برای ستون‌های انتخابی از یک جدول یا دید به کار روند.
- جوازهای دستور **INSERT** و **DELETE** که بر روی کل سطر تأثیر می‌گذارند و می‌توانند فقط برای جدول یا دید به کار روند.
- جوازهای دستور **EXECUTE** که می‌توانند فقط بر روی رویه‌های ذخیره شده تأثیرگذار باشند.

اعمالی که مستلزم ایجاد یک پایگاه داده یا یک شئ در یک پایگاه داده مثلاً یک جدول یا رویه ذخیره شده است به نوع متفاوتی از جوازها به نام جوازهای دستور نیازمندند. مثلاً اگر کاربری می‌بایست قادر به ایجاد یک جدول در داخل یک پایگاه داده باشد، جواز دستور **CREATE TABLE** باید به آن کاربر اعطاء شده باشد.

جوازهای دستور عبارتند از:

CREATE DATABASE	**CREATE DEFAULT**
CREATE PROCEDURE	**CREATE RULE**
CREATE TABLE	**CREATE VIEW**
BACKUP DATABASE	**BACKUP LOG**

آخرین نوع اعمالی که در SQL Server توسط جوازها کنترل می‌شوند، اعمالی هستند که فقط توسط اعضای نقش‌های سیستمی از پیش تعریف شده یا مالک‌های شئ‌های

پایگاه داده انجام می‌شوند. مثلاً یک عضو از نقش ثابت سرویس‌دهنده **sysadmin** به طور خودکار، جواز کامل انجام یا دیدن هر چیزی را در **SQL Server** به ارث می‌برد. نقش **sysadmin** حاوی جوازهایی است که نمی‌توان آنها را تغییر داد و همچنین حاوی جوازهای ضمنی نظیر قابلیت پیکربندی نصب **SQL Server** است که نمی‌تواند برای سایر حساب‌های کاربر به کار رود. مالک‌های شئ پایگاه داده نیز با استفاده از جوازهای ضمنی قادر به انجام کلیه اعمال مربوط به شئ‌ای هستند که مالک آن می‌باشند. مثلاً کاربری که مالک یک جدول می‌باشد می‌تواند داده‌ها را مشاهده، اضافه یا حذف کند، تعریف جدول را تغییر دهد یا جوازهایی که به سایر کاربران اجازه کار با آن جدول را می‌دهند، کنترل نماید.

اجازه دسترسی یا استفاده از اعطاء جوازها

شما می‌توانید جوازهای دستور و شئ را به یک حساب کاربر به منظور انجام فعالیت‌ها یا کار با داده‌ها در پایگاه داده جاری اعطاء کند. دقت در اعطاء جوازها به یک کاربر یا گروه، هنگام کار با سیستم‌های امنیتی پیچیده و بزرگ و داده‌های حساس بسیار اهمیت دارد. همچنین باید مطمئن شوید که جوازها به کاربران امکان انجام فعالیت‌های مورد نیازشان را در پایگاه داده جاری می‌دهند و از فعالیت‌ها یا اطلاعاتی که بخشی از عملکرد مورد نظرشان نیست جلوگیری می‌کند.

توجه: اعطاء جوازها فقط به حساب‌های کاربر پایگاه داده جاری برای شئ‌های موجود در پایگاه داده جاری امکان پذیر است. اگر کاربری به جوازهای شئ در پایگاه داده دیگری نیازمند باشد، حساب کاربر را در پایگاه داده دیگری ایجاد کنید یا این که علاوه بر پایگاه داده جاری، جواز دسترسی حساب کاربر به پایگاه داده دیگر را به او اعطاء کنید. رویه‌های ذخیره شده سیستمی استثنا هستند زیرا جوازهای **EXECUTE** قبلاً به نقش عمومی اعطاء شده‌اند که به هر شخصی اجازهٔ اعطاء آن را می‌دهد. لیکن بعد از صدور **EXECUTE**، رویه‌های ذخیره شده سیستمی عضویت نقش کاربر را بررسی می‌کنند. اگر کاربر عضوی از یک نقش ثابت سرویس‌دهنده یا نقش پایگاه داده مورد نیاز برای اجرای آن رویه ذخیره شده نباشد، رویه ذخیره شده ادامه نخواهد یافت.

اجازه دستیابی بوسیله اعطاء جوازها

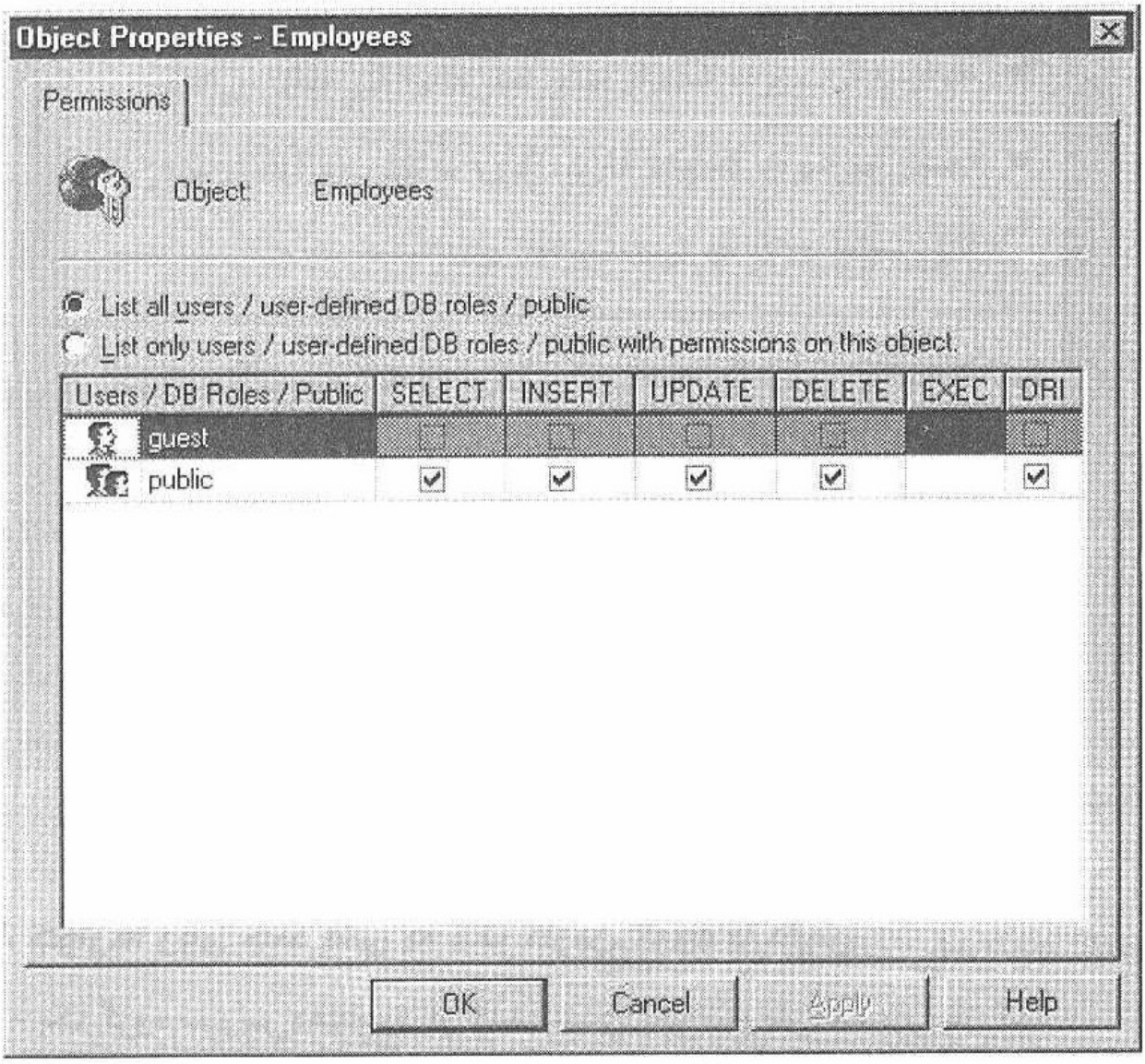
۱- یک گروه سرویس‌دهنده را باز کنید، سپس یک سرویس‌دهنده را باز کنید.

۲- **Databases** را باز کنید؛ سپس پایگاه داده مورد نظر را باز کنید.

۳- نوع شئ مورد نظر خود را انتخاب کنید (جدول، دید یا رویه ذخیره شده).

۴- در پانل **details**، روی شئ مورد نظر رایت کلیک کنید. **All Tasks** را انتخاب کرده و سپس روی **...Manage Permissions** کلیک کنید.

۵- **List all users/DB roles** را انتخاب کنید.



«شکل ۱۱-۶: اعطاء جوازها»

۶- جواز مربوط به هر کاربر را انتخاب کنید. علامت (✓) نشان دهنده اعطاء جواز است.

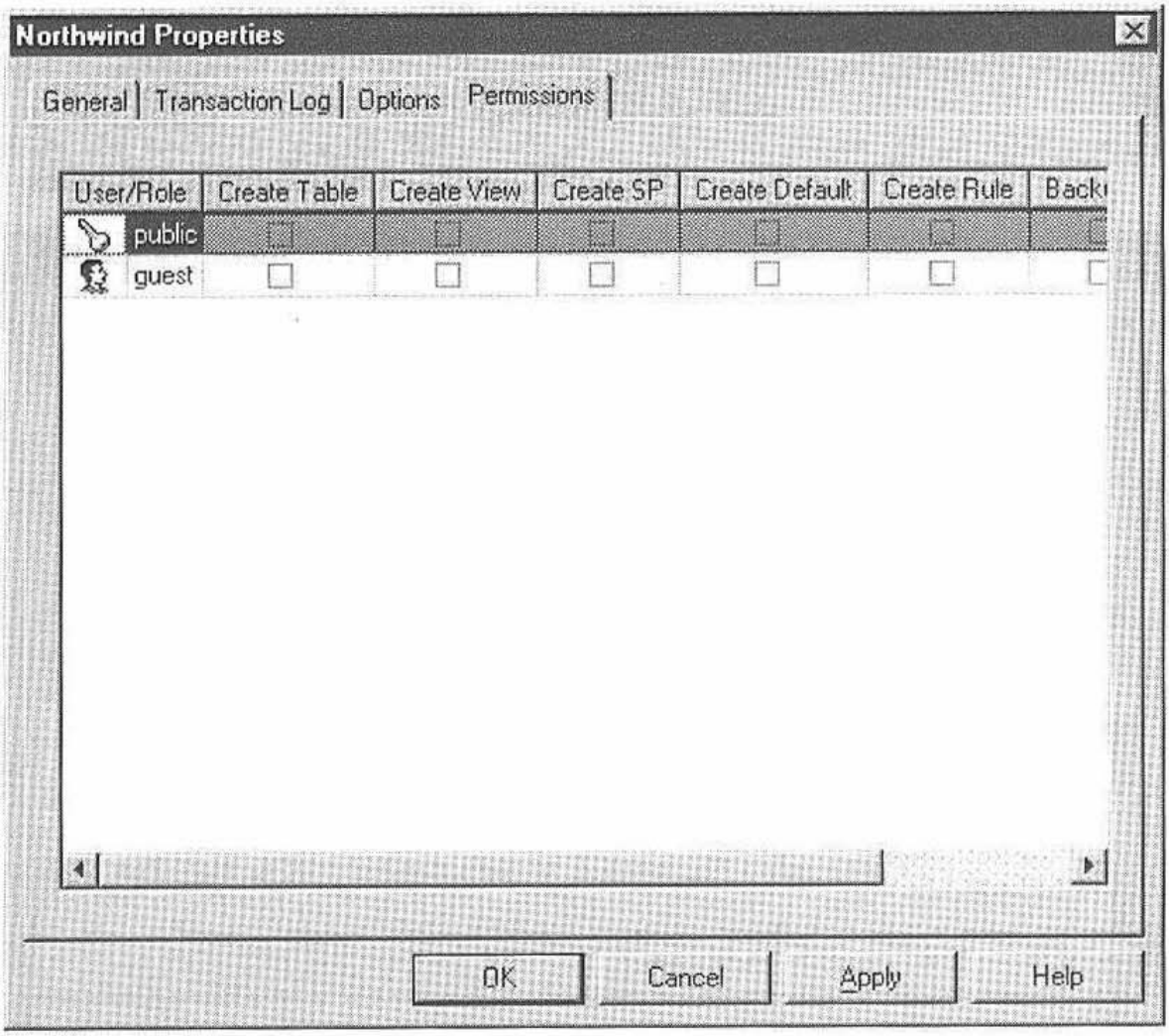
اعطاء جوازهای دستور به کاربران پایگاه داده

۱- یک گروه سرویس‌دهنده را باز کنید؛ سپس یک سرویس‌دهنده را باز کنید.

۲- Databases را باز کنید. پایگاه داده مورد نظر را باز کرده و سپس روی properties کلیک کنید.

۳- روی دکمه permissions کلیک کنید.

۴- جواز دستور مربوط به هر کاربر را انتخاب کنید.



«شکل ۱۲-۶: اعطاء جواز دستور به کاربران»

اعطاء جوازها روی چند شیء به یک کاربر، گروه یا نقش

۱- یک گروه سرویس‌دهنده را باز کنید؛ سپس یک سرویس‌دهنده را باز کنید.

۲- Databases را باز کنید؛ سپس پایگاه داده مورد نظر را باز کنید.

۳- بر حسب نوع کاربر، گروه یا نقش بر روی Users یا Roles کلیک کنید.

۴- در پانل details، روی کاربر، گروه یا نقش مورد نظر رایت کلیک کنید. All Tasks را انتخاب کرده و سپس روی ... Manage permissions کلیک کنید.

۵- List all objects را انتخاب کنید.

۶- جواز مربوط به هر شئ را انتخاب کنید.

SQL Server اجازه می‌دهد تا کاربران و گروه‌های ویندوز NT، کاربران SQL Server و نقش‌های پایگاه داده SQL Server، عضو سایر نقش‌ها باشند. این امر منجر به یک سیستم امنیتی سلسله مراتبی می‌شود که اجازه می‌دهد جوازها از طریق چند سطح از نقش‌ها و عضوها اعمال شوند. اما ممکن است مواقعی وجود داشته باشد که بخواهید جوازهای یک نقش یا کاربر را محدود کنید. هنگام ابطال جوازهای یک حساب کاربر، در واقع:

- جوازی که قبلاً به کاربر، گروه یا نقش داده شده را حذف می‌کنید.
- جواز به ارث برده شده از سایر نقش‌ها را غیر فعال می‌کنید.
- مطمئن می‌شوید که یک کاربر، گروه یا نقش در آینده، جواز را از گروه یا نقش سطح بالاتر به ارث نخواهد برد.

توجه: شما می‌توانید جوازهای حساب‌های کاربر را فقط در پایگاه داده جاری برای شئ‌های پایگاه داده جاری ابطال کنید.

جلوگیری از دستیابی بوسیله ابطال جوازها

تمام مراحل مانند اعطاء جوازها است با این تفاوت که هنگام انتخاب جواز، باید علامت (×) در کنار آن گذاشته شود.

ابطال جوازهای دستور از کاربران

تمام مراحل مانند اعطاء جوازها است، با این تفاوت که هنگام انتخاب جواز، باید علامت (×) در کنار آن گذاشته شود.

ابطال جوازها روی چند شئ برای یک کاربر، گروه یا نقش

تمام مراحل مانند اعطاء جوازها است، با این تفاوت که هنگام انتخاب جواز، باید علامت (×) در کنار آن گذاشته شود.

شما می‌توانید جوازی را که قبلاً اعطاء یا ابطال شده است باز پس بگیرید. باز پس‌گیری و ابطال از این جهت شباهت دارند که هر دو در یک سطح، باعث حذف جواز اعطاء شده می‌گردند. تفاوت آنها در این است که تا وقتی که باز پس‌گیری یک جواز باعث حذف جواز نشده است، از به ارث رسیدن جواز اعطاء شده از سطح بالاتر به کاربر، گروه یا نقش جلوگیری نمی‌کند. بنابراین اگرچه کاربر می‌تواند دارای جواز باز پس گرفته شده مشاهده مستقیم یک جدول باشد اما هنوز مشاهده جدول برای او امکان‌پذیر است؛ زیرا جواز مشاهده جدول به یک نقش که به آن متعلق است اعطاء شده است.

به عنوان مثال، برای حذف دسترسی **SELECT** به جدول **Employee** از نقش **HumanResource**، جواز را باز پس بگیرید تا **HumanResource** دیگر قادر به استفاده از جدول نباشد. اگر **HumanResource** عضوی از نقش **Administrator** باشد و شما بعداً جواز **SELECT** بر روی **Employee** را به **Administrator** اعطاء کنید، عضوهای **HumanResource** به دلیل عضویتشان در **Administrator** قادر به دیدن آن جدول هستند. اما اگر جواز را برای **HumanResource** ابطال کنید حتی در صورتی که آن را بعداً به **Administrator** اعطاء کنید، به ارث گذاشته نخواهد شد.

> **توجه: شما می‌توانید جوازهای حساب‌های کاربر را فقط در پایگاه داده جاری از شئ‌های پایگاه داده جاری باز پس بگیرید.**

باز پس گیری جوازها

تمام مراحل مانند اعطاء جوازها است، با این تفاوت که هنگام انتخاب جواز، باید جعبه خالی انتخاب شود.

باز پس‌گیری جوازهای دستور از کاربران

تمام مراحل مانند اعطاء جوازها است، با این تفاوت که هنگام انتخاب جواز، باید جعبه خالی انتخاب شود.

باز پس‌گیری جوازها روی چند شئ برای کاربر، گروه یا نقش

تمام مراحل مانند اعطاء جوازها است، با این تفاوت که هنگام انتخاب جواز، باید جعبه خالی انتخاب شود.

مراحل‌ها و حالت‌های جواز

جوازهای اعطاء شده به یک گروه یا نقش، توسط اعضای آن گروه یا نقش به ارث برده می‌شوند. اگرچه ممکن است در یک سطح به کاربری، جوازی اعطاء شده و یا از او باز پس گرفته شود، لیکن تداخل جوازها در یک سطح بالاتر، مثلاً به دلیل عضویت در یک نقش، می‌تواند دسترسی کاربر به یک شئ را تسهیل کرده و یا مانع از دسترسی او شود. جوازهای ابطال شده در هر سطح (کاربر، گروه یا نقش) باعث ابطال جواز اعطایی بر روی شئ، بدون توجه به جوازهای باز پس گرفته شده یا اعطایی موجود برای آن کاربر می‌گردد. مثلاً اگر کاربری مانند **John** که عضوی از نقش **sales** با جوازهای اعطایی **SELECT** بر روی جدول **Customer** می‌باشد، جوازهای **SELECT** او به طور صریح بر روی جدول **Cutomer** ابطال شود او هرگز نمی‌تواند به آن جدول دسترسی یابد. به طور مشابه، اگر دسترسی نقش **Sales** به **Customer** ابطال شود اما به **John** این دسترسی اعطاء شود، باز هم دسترسی او ابطال خواهد شد.

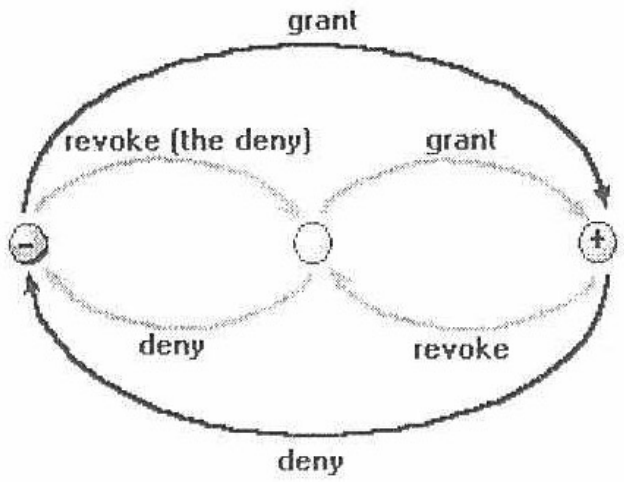
باز پس گرفتن جواز در یک سطح (کاربر، گروه یا نقش) فقط جواز اعطایی یا ابطال شده را حذف می‌کند، اما جواز مشابه اعطایی یا ابطال شده در سطح دیگر مانند یک گروه یا نقش محتوی کاربر، گروه یا نقش مورد نظر هنوز برقرار می‌باشد. مثلاً اگر به نقش **sales**، جوازهای **SELECT** بر روی جدول **Customer** اعطاء شده باشد و جوازهای **SELECT** بر روی جدول **Customer** مربوط به **John** (که عضوی از **sales** است) باز پس گرفته شود، او همچنان به دلیل عضویتش در نقش **sales** قادر به دسترسی به جدول خواهد بود. برای جلوگیری از دسترسی **John** به جدول **Customer**، جواز نقش **sales** یا باید باز پس گرفته شود (با فرض آن که هیچ جوازی درجای دیگری اعطاء نشده باشد) یا ابطال شود (که باعث جلوگیری تمام اعضای **sales** از دسترسی به جدول خواهد شد) و یا این که صراحتاً جوازهای **SELECT** مربوط به **John** بر روی **Cutomer**، ابطال گردد.

اعطای یک جواز در یک سطح (کاربر، گروه یا نقش)، جوازهای ابطال یا باز پس

گرفته شده را حذف می‌کند اما جواز مشابه ابطال شده در سطح دیگر مانندگروه یا نقش محتوی کاربر مورد نظر، همچنان برقرار می‌باشد و اگرچه جواز مشابه باز پس گرفته شده درسطح دیگر نیز همچنان برقرار می‌باشد، لیکن از دسترسی کاربر به شئ، جلوگیری نخواهد کرد. مثلاً اگر قبلاً جواز دسترسی **John** به **Cutomer** ابطال شده باشد و دسترسی **sales** نیز باز پس گرفته شده باشد و سپس به **John** به طور صریح، جواز دسترسی به **Customer** اعطاء گردد، او اکنون می‌تواند به **Customer** دسترسی یابد.

بنابراین، یک کاربر، مجموعه‌ای از تمام جوازهای اعطایی، ابطال شده یا باز پس‌گرفته شده بر روی یک شئ را دریافت می‌کند. شکل زیر نشان می‌دهد که چگونه فعالیت‌های مدیریتی سه جواز بر روی وضعیت یک جواز حساب کاربر تأثیر می‌گذارد.

State Diagram for a Permission



User is prevented from gaining the permission; cannot perform activity.

User does not have the permission but may gain it through a group or role; cannot perform activity.

User has permission and can perform activity.

«شکل ۱۳-۶: ابطال، باز پس گرفتن و اعطاء جوازها»

به عنوان مثالی از یک تداخل جواز، فرض کنید یک کاربر ویندوز **NT** به نام **LONDON\joe** به گروه‌های **LONDON\Clerks** و **LONDON\secretaries** تعلق دارد. **LONDON\ joe** می‌تواند به **SQL Server** متصل شود، زیرا به گروه **LONDON\Clerks**، جوازهای اتصال به **SQL Server** اعطاء شده است. به علاوه، **LONDON\joe** می‌تواند به پایگاه داده **secrets** دسترسی یابد، زیرا جوازهای دسترسی به پایگاه داده به گروه **LONDON\secretaries** اعطاء شده است.

توجه: در این حالت، هیچ مدخل مشخصی برای LONDON\joe در جدول‌های سیستمی sysusers و syslogin مربوط به SQL Server وجود ندارد. این جدول‌های سیستمی، فقط حاوی مدخل‌هایی برای گروه‌های LONDON\clerks و LONDON\secretaries می‌باشند.

LONDON \ joe جدولی با نام joetable در پایگاه داده secrets ایجاد می‌کند. در این حالت، یک مدخل جدید در جدول sysusers برای LONDON\joe به عنوان مالک شئ و نه به عنوان اعطاء دسترسی او به پایگاه داده ایجاد می‌شود. اگر LONDON\joe از گروه LONDON\secretaries حذف گردد، او با وجود این که مالک شئ joetable از پایگاه داده secrets است، اما هرگز نمی‌تواند به پایگاه داده secrets دسترسی یابد.

زنجیره‌های مالکیت

دیدها می‌توانند به دیدها یا جدول‌های دیگر وابسته باشند و رویه‌ها نیز می‌توانند به رویه‌ها، دیدها یا جدول‌های دیگری وابسته باشند. این وابستگی‌ها را می‌توان به عنوان یک زنجیره مالکیت در نظر گرفت. مثلاً، مالک یک دید، مالک شئ‌های مرتبط با آن (دیدها یا جدول‌های دیگر) نیز هست و مالک یک رویه ذخیره شده، اغلب مالک تمام رویه‌ها، دیدها و جدول‌های ارجاع شده نیز می‌باشد.

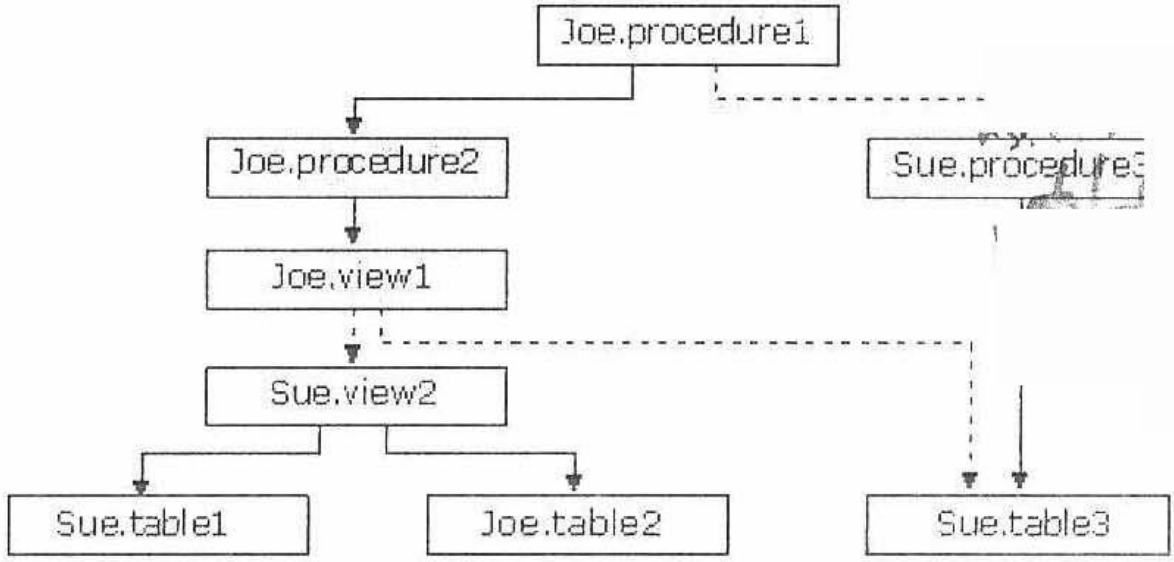
وقتی کاربری به یک دید دسترسی می‌یابد، SQL Server جوازهای مربوط به هیچ کدام از شئ‌های مرتبط با آن دید را بررسی نخواهد کرد به شرطی که این شئ‌ها و دید، تماماً در مالکیت همان کاربر بوده و همچنین آن دید و تمام شئ‌های مرتبط با آن، در یک پایگاه داده باشند. اگر همان کاربر، مالک یک رویه ذخیره شده و تمام دیدها یا جدول‌هایی باشد که رویه به آنها ارجاع می‌کند و اگر رویه و شئ‌ها، تماماً در یک پایگاه داده باشند، SQL Server فقط جوازهای رویه را بررسی می‌کند.

اگر زنجیره مالکیت یک رویه یا دید، قطع شود (یعنی تمام شئ‌های موجود در زنجیره متعلق به کاربر نباشد) SQL Server جوازهای هر شئ از زنجیره را که پیوند بعدی سطح پایین‌تر آن در مالکیت کاربر دیگری باشد بررسی می‌کند.

معمولاً، کاربری که دیدی را ایجاد می‌کند باید جوازها را فقط بر روی همان

دید، اعطاء کند. مثلاً فرض کنید mary دیدی با نام auview1 بر روی جدول authors ایجاد کرده که مالک آن هم هست. اگر mary اجازه استفاده از auview1 را به sue اعطاء کند، SQL Server بدون بررسی جوازهای authors به sue اجازه دسترسی به آن دید را خواهد داد.

یک کاربر که دید یا رویه‌ای را بر اساس شیءای که مالک آن، کاربر دیگری است ایجاد می‌کند باید بداند که هر جوازی که او اعطاء می‌کند به جواز اعطاء مالک دیگر بستگی خواهد داشت. مثلاً فرض کنید Joe یک رویه با نام procedaue1 که به procedure2 (که آن هم به Joe تعلق دارد) و procedure3 وابسته است ایجاد می‌کند. این رویه‌ها نیز به نوبه خود به جدول‌ها و دیدهای دیگری که Joe و Sue مالک آنها هستند بستگی دارند.



«شکل ۱۴-۶: مثالی از یک وابستگی»

اگر Joe جواز استفاده از procedure1 را به mary واگذار کند، SQL Server جوازهای procedure1، procedure3، View2، table2 و table3 را بررسی می‌کند تا ببیند که آیا mary اجازه استفاده از آنها را دارد یا خیر.