

جزوه آزمایشگاه پایگاه داده ها

آزمایشگاه پایگاه داده ها

1- سیستم های مدیریت پایگاه ها

سیستم مدیریت پایگاه داده ها (به انگلیسی database management system) به اختصار **DBMS**، یک نرم افزار است که با هدف مدیریت پایگاه داده ها طراحی شده است، به گونه ای که کاربر درگیر مسائل مربوط به ذخیره و بازیابی و شاخص بندی داده ها نمی شود و بر روی طراحی منطقی پایگاه تمرکز می نماید. این بسته به سازمان ها اجازه می دهد تا به راحتی پایگاه های داده را برای کاربردهای مختلف به وسیله ی مدیر پایگاه داده (به اختصار **DBAs**) و متخصصان دیگر توسعه دهد. سیستم مدیریت پایگاه داده اجازه می دهد تا برنامه های مختلف کاربردی کاربر به طور همزمان به یک پایگاه داده دسترسی داشته باشد. سیستم های مدیریت پایگاه داده ممکن است از مدل های مختلف پایگاه داده جهت سهولت در توصیف و حمایت از برنامه های کاربردی، استفاده کنند. یک سیستم مدیریت پایگاه داده امکان کنترل برای دسترسی به داده، اجرای تمامیت داده ها، مدیریت کنترل همزمانی و بازیابی پایگاه داده، پس از شکست و بازگرداندن آن از فایل های پشتیبان و همچنین حفظ امنیت پایگاه داده را فراهم می آورد.

معرفی چند سیستم مدیریت پایگاه داده :

1- **مایکروسافت اکسس** (به انگلیسی **Microsoft Access**) : یکی از اجزای مایکروسافت آفیس است که برای ایجاد پایگاه داده های رابطه ای مورد استفاده قرار می گیرد. این نرم افزار پایگاه داده جت را با یک واسط کاربری گرافیکی و ابزاری جهت تولید نرم افزار ترکیب نمود. نسخه ی ۱.۰ این نرم افزار در سال ۱۹۹۲ میلادی همراه با مایکروسافت ویندوز پا به عرصه ی وجود نهاد، در این نسخه این امکان فراهم شد تا بسته های پایگاه داده ی جداگانه بتوانند از طریق تکنولوژی اتصال پایگاه داده ی شی گرا (به انگلیسی **ODBC**) با یکدیگر ارتباط برقرار کنند. نسخه ی ۲.۰ اکسس در سال ۱۹۹۴ وارد بازار شد. یکی از ویژگی های مهم این نسخه افزوده شدن موتور پایگاه داده ی جت (به انگلیسی **Jet**

database engine) بود که باعث شد اجرای پرس و جو ها به صورت محسوسی سریعتر شود. نسخه ی کنونی اکسس، نسخه ی 2013 می باشد.

2- **اوراکل** : شرکت اوراکل (Oracle Corporation) یکی از بزرگ ترین شرکت های نرم افزاری در آمریکا و جهان است. این شرکت در سال ۱۹۷۷ میلادی با نام Relational Software Incorporated یا RSI شروع به کار کرد. در ویرایش ۳ نرم افزار، نام شرکت از RSI به اوراکل تغییر کرد. محصول اصلی آن نرم افزار پایگاه داده های اوراکل است. این شرکت پر قدرت ترین شرکت در زمینه ی سامانه مدیریت پایگاه داده ها است. در سال ۲۰۱۱ درآمد اوراکل از فروش محصولات و خدمات مختلف به رقم ۲.۴۰ میلیارد دلار رسید.

3- **مای اس کیوال** (به انگلیسی MySQL) : یک سامانه مدیریت پایگاه داده متن باز است، که توسط شرکت اوراکل توسعه، توزیع، و پشتیبانی می شود. سرور مای اس کیوال به چندین کاربر اجازه استفاده همزمان از داده ها را می دهد.

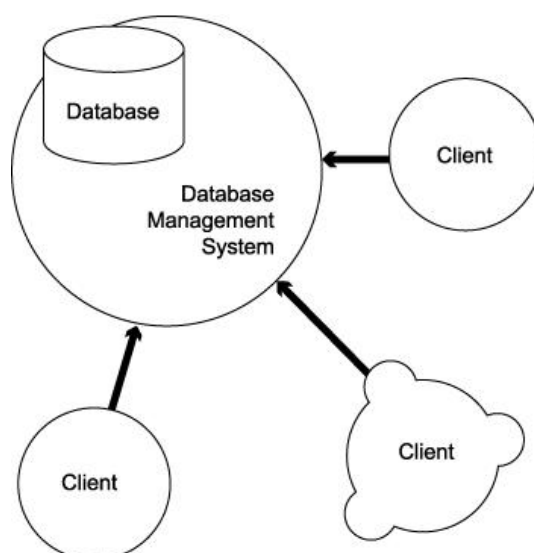
4- **دی بی تو** (به انگلیسی Database 2 - DB2) : به خانواده ای از محصولات شرکت آی بی ام در زمینه ی سیستم های مدیریت پایگاه داده ها اطلاق می شود.

5- **پُستگرس کیوال** (به انگلیسی PostgreSQL) : یا به طور ساده تر پُستگرس، یک سامانه مدیریت پایگاه داده های شی-رابطه ای است که برای سکوهای مختلفی از جمله لینوکس، فری بی اس دی، ویندوز، و مک اواس ده موجود است. پستگرس کیوال توسط گروه توسعه سراسری پستگرس کیوال توسعه داده می شود، که شامل تعداد زیادی از افراد داوطلب است. پستگرس کیوال بخش اعظم استاندارد اس کیوال: ۲۰۰۸ را پیاده سازی می کند و دارای افزونه های بسیاری است که توسط دیگران ایجاد شده است.

6- **مایکروسافت اس کیوال سرور** یا **مایکروسافت سی کول سرور** (به انگلیسی Microsoft SQL Server) : یک نرم افزار سیستم مدیریت بانک های اطلاعاتی است که توسط شرکت مایکروسافت توسعه داده می شود. برخی از ویژگی های این سیستم مدیریت پایگاه داده ها به این شرح است:

- بانک اطلاعاتی رابطه ای

- امکان استفاده از trigger, View, Stored procedure
- پشتیبانی از XML
- بسیار قدرتمند و بدون محدودیت حجم و تعداد رکورد
- پشتیبانی از FullText Search برای سرعت در بازیابی اطلاعات و استفاده از زبان طبیعی در جستجوها
- جدیدترین نسخه ی سرور SQL سرور SQL2008 می باشد در سال 2008 پیشنهاد و عرضه گردید. اهداف SQL Server 2008 ایجاد و مدیریت داده ها به شکل هماهنگی، سازماندهی و محافظت به شکل اتوماتیک می باشد و به صورت client/server عمل می کند.



به همراه SQL سرور ، ابزارهای گرافیکی نیز وجود دارد از جمله :

- SQL Server Management Studio : (بطور خلاصه SSMS) محیطی برای انجام تمام عملیات روی بانک اطلاعاتی و اجرای دستورات SQL
- Configuration Manager : محیطی برای انجام پیکر بندی های مختلف، محیطی و ...

SSMS اجازه می دهد که همزمان به یک یا چند سرور SQL موجود در شبکه متصل شده و بانک های اطلاعاتی روی آنها را مدیریت کنید. در پنجره Object Explorer آن لیست سرورها و اجزای داخل آن ها نمایش داده می شود. از جمله پوشه های databases که حاوی بانک های اطلاعاتی است.

2- مدیریت بانک اطلاعاتی

2-1- ایجاد بانک اطلاعاتی

برای ایجاد بانک های اطلاعاتی روی پوشه Databases کلیک راست کرده و گزینه New را اجرا می کنیم و در پنجره ی مشخصات بانک اطلاعاتی تنظیمات زیر را انجام مید هیم:

1- نام بانک اطلاعاتی

2- مالک بانک اطلاعاتی (owner)

3- مشخصات فایل های بانک اطلاعاتی که هر بانکی باید حد اقل دارای یک فایل برای نگهداری داده ها و یک فایل برای ثبت تراکنش ها (log) داشته باشد. برای هر فایل می توان مسیر آن و تنظیمات رشد خودکار آن (Autogrowth) را تنظیم کرد. همچنین برای هر فایل می توان اندازه ی اولیه ی آن را بر حسب مگابایت و همچنین نحوه ی رشد آن و حداکثر اندازه ی رشد آن را بصورت مطلق یا نسبی تعیین کرد.

4- Collation یعنی روش کد گذاری حروف

5- Recovery model یا مدل ترمیمی برای ثبت کردن وقایع به منظور ترمیم اطلاعات در صورت بروز مشکل

- Simple : مدل ساده

- Full : مدل کامل

- Bulk-Logged : بین مدل ساده و کامل

- 6 Compatibility سطح سازگاری با نسخه های مختلف SQL سرور
 - 7 برای تنظیماتی مانند Data base read-only: فقط خواندنی کردن بانک اطلاعاتی
 - 8 Restrict Access محدود کردن دست یابی
 - Single: در هر زمان فقط یک کاربر می تواند با بانک کار کند.
 - Restricted: کاربران خاصی که نقش مدیر دارند می توانند با بانک کار کنند.
 - Multi: چند کاربر می توانند با بانک کار کنند.
- برای ساخت بانک اطلاعاتی همچنین می توان دستور زیر را در پنجره پرس و جو اجرا کرد:

نام بانک اطلاعاتی create database

on

(name = 'class', filename = 'c:\class.mdf' ,size=2MB,maxsize=10MB ,filegrowth=2MB)

log on

(name = 'class', filename = 'c:\class.ldf' ,size=2MB,maxsize=10MB ,filegrowth=2MB)

2-2- پایگاه داده های پیش ساخته

هنگام نصب اس کیو ال سرور چهار بانک اطلاعاتی زیر بصورت پیش فرض ساخته می شوند:

-1 پایگاه داده master:

که برای اجرای sql سرور بسیار مهم است . اطلاعات مهمی مثل مشخصات پایگاه های داده، اجزای آنها، پیغام های خطا ، اطلاعات ورود به سیستم ، رویه های ذخیره شده و ... در آن ذخیره شده است.

-2 پایگاه داده model:

هنگام ایجاد یک بانک داده جدید یک کپی از این بانک ایجاد میشود و هرشی موجود در بانک اطلاعاتی model در بانک اطلاعاتی جدید نیز وجود خواهد داشت.

نکته: هرشی که به بانک اطلاعاتی model اضافه کنید به طور اتوماتیک به تمام بانک های اطلاعاتی که بعد از این ایجاد می کنید اضافه می شود.

3- پایگاه داده msdb :

برای تعیین برنامه های زمان بندی نگهداری سیستم و ثبت تاریخچه نسخه پشتیبان به کار میرود.

4- پایگاه داده tempdb :

محلی برای ترکیب و مرتب سازی و سایر اعمالی که به فضای ذخیره سازی موقت نیاز دارد و با هر بار اجرای sql به هنگام میشود و هنگامی که sql سرورس دهی خود را قطع می کند اطلاعات درون پایگاه نیز از بین می رود.

2-3- اشیاء یک بانک اطلاعاتی

هر بانک اطلاعاتی دارای اشیائی است، ایجاد بانک اطلاعاتی برای برآورده کردن اهداف تجاری مستلزم ایجاد و کارکردن با آن اشیا است که به اختصار معمول ترین آنها را توضیح میدهیم:

- table: حاوی تمام اطلاعات موجود در بانک اطلاعاتی است.

- view: یک جدول مجازی است که محتویات آن توسط یک تقاضا مشخص میشود و ممکن است از چند جدول به دست بیاید.

- stored proceduer (رویه های ذخیره شده): مجموعه ای از دستورات sql است که کاربر یک بار آنها را می نویسد و بارها و بارها آن را فراخوانی و اجرا می کند.

- roles: کاربرانی را با نیاز های دستیابی یکسان دسته بندی می کند.

- Rules: روش استاندارد برای محدود کردن مقادیر در یک ستون هستند در واقع مقرراتی را به ستون ها اعمال می کند.

- Defaults: وقتی مقادیری برای ستون ها در نظر گرفته نمی شود پیش فرض ها را برای ستون ها در نظر میگیرد.

برای تغییر تنظیمات پایگاه داده: راست کلیک روی نام بانک مورد نظر و گزینه properties و تغییر گزینه های مورد نظر

برای حذف پایگاه داده: راست کلیک روی نام بانک مورد نظر و گزینه delete و تغییر گزینه های مورد نظر

3- مدیریت جداول

برای ایجاد جدول روی پوشه Databases → Tables کلیک راست کرده و دستور New را اجرا می کنیم. سپس ستون های جدول را تعریف می کنیم. هر ستون در جدول دارای خصوصیات معین می باشد که آن را برای SQL Server تعریف می کند. مهمترین خصوصیت، نوع داده ستون میباشد، که تعریفی از نوع اطلاعاتی که در ستونها ذخیره خواهند شد می باشد. SQL Server یک محدوده وسیعی از انواع داده ها را فراهم آورد.

3-1- انواع داده ها

به طور کلی انواع داده ها به وسیله SQL Server فراهم می گردد، همچنین شما می توانید خودتان تعریف کنید.

متنی یا کاراکتری	char	کاراکتری با طول ثابت ۸۰۰۰بایت-هرکاراکتر ۱بایت
	nchar	کاراکتری با فرمت یونیکد با طول ثابت ۴۰۰۰بایت-هر کاراکتر ۲ بایت
	varchar	رشته کاراکتر با طول حداکثر ۸هزار کاراکتر
	nvarchar	رشته کاراکتر با فرم یونیکد با فرم متغییر حداکثر ۴ هزار کاراکتر
	text	کد گذاری نشده با طول متغییر حداکثر ۱میلیون کاراکتر(متن طولانی)
	ntext	کد گذاری شده با طول متغییر حداکثر ۱میلیون کاراکتر
تاریخ	datetime	مقادیر تاریخ و زمان که تا ۳۰۰۰ ثانیه وقت دارد
	smalltime	مقادیر تاریخ و زمان که به ازای ۱ دقیقه وقت دارد
عددی	bit	مقدارهای صفر و یک

	tiny int	داده عددی بسیار کوچک معادل ۰ تا ۲۵۵
	small int	مقادیر کوچک معادل ۲ بایت
	int	داده صحیح معادل ۴ بایت
	big int	داده صحیح بزرگ معادل ۸ بایت
	float	مقادیر اعشاری با دقت مضاعف معادل ۸ بایت تا ۱۵ رقم اعشار
	real	مقادیر اعشاری تک دقتی معادل ۴ بایت ۲۴ رقم اعشار
	numerical	اعداد صحیح در مبنای دیگر
	decimal	اعداد در مبنای ۱۰ با دقت ۳۸ رقم اعشار
پول	money	مقادیر پولی معادل ۱۵ و ۴ بایت تا ۱۵ رقم صحیح
	binary	داده باینری با طول ثابت حداکثر ۸۰۰۰
	varbinary	داده باینری با طول متغییر حداکثر ۸۰۰۰
باینری	image	(عکس را تبدیل به کد کرده و ذخیره می کند) با طول متغییر حداکثر ۲ گیگا بایت
	timestamp	درج مقادیر تاریخ و زمان تفاوت آن با datetime در این است که این نوع را زمانی به کار میبریم که می خواهیم تاریخ جاری سیستم به صورت مقادیر منحصر به فرد در جدول ذخیره شود

2-3. خصوصیات دیگر

خصوصیات دیگر فیلد ها عبارتند از:

- Allow Nulls یعنی مقدار دهی آن فیلد ها اجباری نیست.
- Default value or Binding یعنی مقدار پیش فرض
- Length: طول فیلد های رشته ای
- Collation استاندارد کد گذاری
- computed column specification: فرمول برای محاسبه مقدار فیلد.
- is identity افزایش خودکار مقدار فیلد.
- identity increment گام افزایش
- identy speed مقدار اولیه (شروع)
- Not for replication عدم استفاده از فیلد خود افزایشی در تکثیر داده ها.

روش دوم برای ساخت جدول اجرای دستوری مثل زیر در پنجره پرس و جو است:

```
create table student
(std_name varchar(20) not null ,
std_family varchar(20),
std_num int primary key identity(100)
city varchar(20),
birthday datetime ,
grade int ,
age int check age>15 ) ;
```

4- نمودار بانک اطلاعاتی

برای رسم نمودار بانک اطلاعاتی و ایجاد ارتباط بین جداول از بخش Database Diagrams استفاده کرده و در صفحه رسم نمودار :

- با کلیک راست و انتخاب گزینه Add table می توان جداول را به نمودار اضافه کرد.
- با درگ کردن کلید اصلی از جدول اصلی روی کلید خارجی در جدول فرعی می توان بین آن ها رابطه برقرار کرد.
- برای حذف رابطه روی آن کلیک راست کرده و delete را اجرا می کنیم.
- برای مدیریت کردن ارتباط های یک جدول روی جدول کلیک راست کرده و گزینه relationships را اجرا می کنیم و در کادر باز شده می توانیم ارتباط ها را حذف کرده، اضافه کنیم و یا تنظیمان آن ها را تغییر دهیم از جمله تنظیمات ارتباط داریم :

Tables and columns specifications : تعیین جداول دو طرفه ارتباط و ستون های متناظر با هم

enforce Foreign key constraints : قاعده ی جامعه ی ارجاعی در نظر گرفته شود یا خیر

Insert and update specification : می توان قواعد مورد نظر را هنگام حذف یا اصلاح یک سطر در جدول اصلی

را تعیین کرد. این قواعد عبارتند از:

1- No Action : هیچ عملی را در نظر نمی گیرد

2- CaseCade : حذف یک سطر از رابطه ی مرجع تمام سطر های رجوع کننده به آن در جدول فرعی نیز حذف شوند.

3- Set Null : با حذف یک سطر از رابطه ی مرجع مقدار کلید خارجی در جدول فرعی، Null گذاشته می شود.

4- Set Default : با حذف سطر مرجع کلید خارجی در سطر های رجوع کننده به آن با مقدار پیش فرض مقدار گذاری

می شود و

5- عملیات روی بانک اطلاعاتی

5-1- پشتیبان گیری و بازیابی اطلاعات

یکی از وظایف مسئول نگهداری بانک اطلاعات تهیه ی پشتیبان در بازه های زمانی معین است. برای گرفتن پشتیبان روی بانک اطلاعاتی کلیک راست کرده و گزینه ی Backup را اجرا می کنیم. در کادر باز شده باید بانک اطلاعاتی، وسیله پشتیبان گیری و آدرس همچنین نوع Backup (کامل تغییرات جدید یا تراکنش ها) بخش هایی که باید پشتیبان در نظر گرفته شود. file and file group گروه ها یا فایل های از بانک اطلاعاتی. تاریخ یا مدت انقضاء و آدرس و نام فایل مقصد را تعیین می کنیم.

برای برگرداندن پشتیبان روی بانک اطلاعاتی کلیک راست کرده و گزینه ی Restore را اجرا می کنیم. در کادر

Restore نام بانک اطلاعاتی مقصد، مسیر و نام فایل پشتیبان را تعیین کرده و در صفحه Option گزینه ی With

Replace را تیک می زنیم.

5-2- جدا کردن و چسباندن

برای کپی برداری یا انتقال فایل های بانک اطلاعاتی باید آن را از سرور جدا کنیم و مجدداً آنها را در سیستم مقصد، به سرویس دهنده بچسبانیم. برای جدا کردن بانک اطلاعاتی روی آن کلیک راست کرده و گزینه ی Detach را اجرا می کنیم. برای چسباندن روی پوشه Database کلیک راست کرده و گزینه ی Attach را اجرا می کنیم.

5-3- / ورود بانک اطلاعاتی (import/export)

برای تبادل اطلاعات بین دو نرم افزار مختلف مثلاً Access و SQL server استفاده می شود. Source یعنی مبدا را انتخاب می کنیم و Destination مقصد. علاوه بر تعیین مبدا و مقصد ، فرمت مبدا و مقصد نیز انتخاب می شود.

6- امنیت در SQL server

مقوله امنیت همواره یکی از مهم ترین شاخه های مهندسی نرم افزار و به تبع آن، یکی از حساس ترین وظایف مدیران سیستم به خصوص مدیران شبکه و یا مدیران بانک های اطلاعاتی است. با تنظیم سطوح دسترسی برای کاربران شبکه یا بانک های اطلاعاتی شبکه، امنیت اطلاعات یا به عبارتی عدم دسترسی افراد فاقد صلاحیت به اطلاعات، تضمین می گردد. هر سیستم عامل، پلتفرم یا بانک اطلاعاتی، شیوه هایی خاصی را برای برقراری قواعد امنیتی به کاربران معرفی می نماید. در SQL Server هم روش های خاصی برای این مقوله وجود دارد. SQL server سطوح مختلف امنیتی را برای دسترسی به بانک اطلاعاتی فراهم می کند.

6-1- : تائید هویت برای اتصال به سرور

در سطح اول کاربران برای اتصال به SQL سرور، باید با یکی از دو روش زیر تایید هویت شوند:

1- Windows Authentication

2- SQL server Authentication

در روش اول کاربران برای اتصال به سرویس دهنده SQL باید توسط ویندوز شناسایی شوند در نتیجه کاربرانی که با موفقیت وارد ویندوز شده اند می توانند به سرور متصل شوند کاربرانی که از طریق شبکه قصد اتصال به سرور را دارند باید نام کاربری و رمز عبور در سیستم داشته باشند.

در روش دوم کاربرانی می توانند به سرور متصل شوند که نام کاربری و کلمه ی عبور معتبر در SQL server داشته باشند. هنگام نصب SQL server به طور پیش فرض یک کاربر به نام sa ایجاد می شود. برای کار بردهای محلی روش دوم و برای کار بردهای شبکه روش اول امنیت بالاتری دارد.

برای مدیریت loginها (کسانی که می توانند به سرور متصل شوند) از پوشه ی security → logins استفاده می شود.

نکته : در حین نصب SQL سرور و نیز در کادر مشخصات سرویس دهنده می توان تعیین کرد که متد اول یا هر دو متد برای اتصال به سرویس دهنده استفاده شود.

6-2- نقش های از پیش تعریف شده سرویس دهنده

نقش های سرویس دهنده برای آن هستند تا برخی از کارهای مدیریتی سرویس دهنده را به اشخاص دیگر واگذار کنید .

هفت نقش از پیش تعریف شده سرویس دهنده به شرح زیر هستند :

sysadmin: اعضای این نقش می توانند هر عملی را در سرویس دهنده انجام دهند. هنگام نصب SQL سرور کاربری بنام sa بطور پیشفرض ایجاد شده و دارای این نقش است.

serveradmin: اعضای این نقش می توانند پیکربندی مشخصات سرویس دهنده را انجام دهند .

setupadmin: اعضای این نقش مجازند پیوندهای سرویس دهنده ها را حذف یا اضافه کنند .

Securityadmin: اعضای این گروه می توانند برقراری ارتباط با سرویس دهنده را مدیریت کنند .

processadmin: اعضای این گروه می توانند هر فرآیندی را که در SQL سرور اجرا می شود را مدیریت کنند .

dbcreator: این گروه مجوز ایجاد بانکهای اطلاعاتی در سرویس دهنده را دارد .

diskadmin: این گروه مجوز ایجاد و مدیریت فایل ها در دیسک را دارد .

3-6- تعیین کاربران مربوط به یک بانک اطلاعاتی :

هر کدام از login های موجود در سیستم می توانند به عنوان کاربر یک یا چند بانک اطلاعاتی تعیین شوند و در هر بانک اطلاعاتی دارای یک یا چند نقش باشند که با بر عهده گرفتن هر نقش مجموعه ای از اختیارات را در مورد آن بانک به دست می آورند. برای تعیین کردن یک login به عنوان کاربر بانک اطلاعاتی و دادن نقش ها به آن از صفحه ی user mapping در کادر مشخصات login استفاده می کنیم و با انتخاب کردن یک بانک اطلاعاتی می توان از پایین کادر نقش های مربوط به آن را انتخاب کرد این نقش ها عبارتند :

db-accessadmin

کاربران تعریف شده در این نقش قادر خواهند بود، سطوح دسترسی و امنیتی کلیه کاربران و نقش ها را در قسمت های مختلف پایگاه تعریف کنند .

db-backupoperator

این نقش مسؤل ایجاد نسخه های پشتیبان از سیستم و اطلاعات درون آن است .

db-datareader

این نقش قادر است کلیه اطلاعات تمام جداول بانک اطلاعاتی موجود در سیستم را بخواند. مگر آن که اطلاعات خاصی توسط مکانیسم Deny از دسترس او دور نگاه داشته شود .

db-datawriter

افراد تعریف شده در این نقش قادرند تا کلیه اطلاعات موجود در کلیه جداول بانک را با استفاده از دستورات سه گانه Insert، Update، Delete تغییر دهند. مگر آن که جدول یا فیلد خاصی توسط مکانیسم Deny از دسترس شان دور نگه داشته شود .

db-ddladmin

کاربران دارای این نقش می توانند ساختار جداول، دیدها، روتین ها و توابع یک بانک اطلاعاتی را با استفاده از دستورات سه گانه Create، Drop، alter، بسازند، تغییر دهند یا از بین ببرند .

db-denydatareader

این نقش قادر به خواندن هیچ اطلاعاتی از جداول یا سایر قسمت‌های بانک نیست.

db-denydatawriter

این نقش هم قادر به تغییر دادن هیچ یک از قسمت‌های بانک اطلاعاتی نیست.

db-owner

این نقش قادر به انجام هر عملی در بانک اطلاعاتی می‌باشد و بالاترین سطح موجود در یک بانک است.

db-securityadmin

مسئول تعریف و تنظیم نقش‌ها، کاربران و سطوح دسترسی در یک بانک است.

public

کاربران این نقش قادرند تمام جداول، دیدها و سایر قسمت‌هایی که توسط خودشان یا توسط کاربران متعلق به نقش

dbowner ساخته شده را بخوانند و بنویسند.

6-4- سطح چهارم: تعیین مجوزها برای هر کدام از اشیا

هر کدام از کاربران بانک اطلاعاتی می‌توانند به هر جدول یا اجزای دیگر آن دسترسی محدود داشته باشند به عنوان مثال

می‌توان تعیین کرد که کاربر خاصی از بانک اطلاعاتی مجوز درج، حذف، اصلاح و... داشته باشد یا نداشته باشد. برای

تعیین این مجوزها از صفحه ی permissions در کادر مشخصات جدول استفاده می‌کنیم.

7- زبان پرس و جوی ساختیافته (SQL)

1. پرس و جوی عمومی (انتخاب همه اطلاعات جدول)

نام جدول * from select *

مثال:

select * from student

تمامی رکورد های جدول دانش آموز را نمایش می‌دهد.

2- پرس و جوی فیلد های خاصی از جدول

نام جدول, from نام فیلد ۱, نام فیلد ۲, ...

3- انتخاب فیلد های خاص و نمایش آنها با نام های جدید در خروجی

نام جدول, ..., from نام جدید برای فیلداول as نام فیلد ۱

4- مرتب سازی پرس و جو

نام فیلد orther by نام جدول from نام فیلد ها

پرس جو براساس نام فیلدی که در جلوی orther by نوشته شده است مرتب می شود.

5- استفاده از عملگر های منطقی در شرط نمایش orther by

شرط ۲ (and/or/not) شرط ۱ where نام فیلد orther by نام جدول from *

6- نمایش query با حذف مقادیر تکراری

نام جدول, ..., from نام فیلد distinct

7- استفاده از عملگر like

توسط عملگر like می توانیم یک الگو یا شکل کلی برای ارزیابی مقادیری که می خواهیم تعریف کنیم که در صورت مطابقت با الگو نمایش داده شوند.

Select نام فیلد like '%مقدار%' where نام جدول from نام فیلدها

8- استفاده از group by

هرگاه بخواهیم نتیجه نمایش پرس و جو بر اساس یک فیلد دسته بندی شود نام آن را در جلوی عبارت group by می نویسیم. توجه کنید نام فیلد جلوی group by حتما باید در جلوی دستور select نوشته شود و در صورت استفاده از فیلد های دیگر باید از توابع جمعی استفاده کنیم.

نکته: همیشه و همیشه هنگامی که از توابع جمعی استفاده می کنیم حتما باید از کلمه group by استفاده شود.

توابع جمعی :

sum جمع

avg میانگین

max پیدا کردن بزرگترین مقدار

min پیدا کردن کوچکترین مقدار

count شمارش تعداد

نام فیلد group by نام جدول from *

مثال :

جمع نمرات هر دانش آموز با سن بالای ۱۵ (دسته بندی بر اساس نام و فامیل)


```
select std_name , std_family,sum(grade) from student  
where age>15 group by std_name,std_family
```

9- استفاده از group by اعمال شرط having بر روی آن

برای شرط دسته بندی رکورد ها استفاده می شود. عبارت having شرط نمایش مقادیر دسته ها را مشخص می کند و بعد از دسته بندی شدن اعمال می شود ولی شرط جلوی where روی تمامی رکورد ها و قبل از دسته بندی شدن اعمال می شود.

```
select * from نام جدول group by نام فیلد having شرط
```

مثال:

جمع نمرات دانش آموزانی بالای ۱۵ سال که جمع نمرات آنها از ۱۶ بیشتر شده است.

```
select std_name , std_family,sum(grade) from student  
where age>15 group by std_name,std_family having sum (grade)>16
```

نکته: معمولا having هر شرطی را با توابع جمعی بیان می کند در غیر این صورت فقط می توان در شرط از نام فیلدی استفاده کرد که در جلوی عبارت group by نوشته شده است.

10- استفاده از عملگر in

جهت تعیین شرط با محدوده ای از مقادیر (عملیات in شبیه عملگر or می باشد)

```
select * from نام جدول where نام فیلد in ( select ... )
```

11- استفاده از عملگر top

هرگاه بخواهیم حداکثر تعداد رکورد های نمایشی را در خروجی نمایش دهیم از این عملگر استفاده می کنیم.

```
select top 5 grade from student
```

5-نمره اول جدول دانش آموز را برمی گرداند.

تذکره: برای نمایش درصدی از رکورد ها در خروجی بعد از کلمه top ابتدا عدد و سپس کلمه percent را می نویسیم

```
select top 5 percent grade from student
```

5درصد اول نمرات جدول دانش آموز را برمی گرداند.

12- برای بررسی مقادیر فیلد ها با مقدار null در شرط نمایش رکورد ها

```
select * from نام جدول where نام فیلد is null/is not null
```

13- استفاده از عبارت compute by

توسط این عبارت یک عملیات را روی رکورد های دسته بندی شده توسط توابع جمعی انجام می دهیم. هنگام استفاده از این عملگر حتما باید عبارت by orther را نوشته و فیلد جلوی compute حتما باید جلوی orther آمده باشد.

```
select * from نام جدول where شرط
```

نام فیلد by order

نام فیلد دسته بندی شده by توابع جمع compute

تفاوت group by و compute by

compute نتیجه عملیات روی رکورد های دسته بندی شده را به صورت تفکیک شده نمایش می دهد.

14- استفاده از عبارت into

نتیجه پرس وجو در یک جدول موقت ذخیره میشود که نام آن جدول حتما باید ذکر شود. نشانه یک جدول موقت عبارت # می باشد. جدول موقت می تواند توسط دستور select مقادیر خود را نمایش دهد.

... نام جدول اصلی from نام جدول موقت into * select

ایجاد جستجوی های فرعی :

هر گاه قسمت where یک دستور select با یکی از عملگر های زیر select دیگری را بنویسیم یک پرس و جو فرعی ایجاد کرده ایم.

in : بررسی وجود مقادیر یک فیلد در یک مجموعه جواب که خروجی select فرعی است.

(دستورات پرس و جوی فرعی) in نام فیلد where نام جدول * select

any : مقادیر یک فیلد توسط عملگر "=" با مقدار جواب select داخلی مقایسه می شود.

(دستورات پرس و جوی فرعی) = نام فیلد where نام جدول * select

all : مقادیر یک فیلد توسط عملگرهای "<=>" با مقدار جواب select داخلی مقایسه می شود.

(دستورات پرس و جوی فرعی) =<> نام فیلد where نام جدول * select

exists/not exists: وجود مقادیر یک فیلد یا عدم وجود آن که با مقدار جواب select داخلی مقایسه می شود.

(دستورات پرس و جوی فرعی) exists/not exists نام فیلد where نام جدول * select

مثال :

ابتدا جداول زیر را بسازید.

جدول فروشندگان (saller):

با فیلد های: saller_name, saller_num

جدول محصولات (product):

با فیلد های: pro_name, pro_num

جدول فروش (salle):

با فیلد های: saller_num, pro_num, number

1- نام فروشندگانی که یکی از کالاهایی که احمدی فروخته است را فروخته اند نمایش داده شود.

پاسخ :

باتوجه به این که اطلاعات در دو جدول جداگانه ذخیره شده باید از پرس و جوی های فرعی استفاده کنیم.

```
select saller_name from saller  
where saller.saller_num=saller.num
```

(این دستور ارتباط بین جداول را ابتدا چک می کند)

```
and pro_num in
```

```
(select pro_num from saller)
```

```
where saller.saller_num=saller.num and
```

```
saller_name='ahmadi')
```

2- نام فروشندگانی که هیچ کالای نفروخته اند نمایش داده شود.

3- کد کالا هایی که بیش از یک بار فروخته شده است.

ارتباط بین جداول

یکی از مهمترین مزیت یک بانک اطلاعاتی نسبت به جدول های ساده امکان برقراری ارتباط بین جدول های آن

است. مزیت های برقراری ارتباط بین جداول:

1- می توان با استفاده از چند جدول گزارش هایی را تولید کرد.

2- اشتباهات در ورود اطلاعات را کاهش داد.

3- تسریع در ورود و بازیابی اطلاعات

4- موجب ایجاد یکپارچگی خواهد شد.

برای تعیین رابطه بین جداول باید از فیلد های مشترک استفاده کرد. فیلدی که براساس آن بین جداول ارتباط برقرار می

شود در جدول اصلی کلید اصلی و در جدول فرعی کلید خارجی می نامیم.

روش اول برای ایجاد ارتباط:

باز کردن جدول در حالت Design راست کلیک و انتخاب گزینه relationships

انتخاب گزینه **new** و انتخاب فیلد کلید اصلی از هر دو جدول اصلی و فرعی و انتخاب تنظیمات مناسب از گزینه های پایین پنجره

روش دوم برای ایجاد ارتباط:

عملگر های join

cross join : توسط این نوع الحاق لیستی از تمام رکورد های جدول فرعی تولید می شود به طوریکه هر رکورد جدول اصلی با رکورد جدول فرعی نظیر به نظیر ارتباط دارد.(ارتباط یک به یک)

نام جدول ۲ **cross join** نام جدول ۱ **from** فیلد های جدول اصلی **select**

inner join : توسط این عملگر رکورد های از هر دو جدول طوری واکنشی می شود که مقادیر غیر **null** مرتبط را نمایش می دهد.(نوعی ارتباط یک به چند)

شرط **on** نام جدول ۲ **inner join** نام جدول ۱ **from** فیلد های جدول اصلی **select**

مثال:(فیلد شماره دانش آموزی به عنوان کلید اصلی در نظر گرفته شده است)

```
select student.std_name,student.std_family from  
student innerjoin register on  
student.std_num = register . std_num
```

: outer join

که شامل **left,right,full** می شود:

در این نوع الحاق همه رکورد های مرتبط با جدول اصلی که در جدول فرعی وجود دارد حتی مقادیر **null** واکنشی می شود.

left outer join : تمام رکورد های جدول اصلی را برمی گرداند و رکورد های مرتبط با جدول اصلی در جدول فرعی نمایش داده می شود و فیلدهایی که در جدول اصلی وجود دارد و در جدول فرعی وجود ندارد، null می شود.

right outer join : تمام رکورد های جدول فرعی را برمی گرداند و رکورد های مرتبط با جدول اصلی در جدول فرعی نمایش داده میشود و فیلد های که در جدول اصلی وجود ندارد ولی در جدول فرعی وجود دارد، null می شود.

full outer join : همه رکورد ها را برمی گرداند. در صورتی که یک رکورد از دو جدول نظیری نداشته باشد، null می شود.

جدول ۱ from نام فیلد ها select

شرط on جدول ۲ [left | right | full] outer join

8- دیدگاه ها

دید یا نما مجموعه ای از رکورد های بانک است که جهت دسترسی کاربران بانک دسته بندی می شود. از مهمترین مزایای دید امنیت داده ها و عدم دسترسی کاربران غیر مجاز به داده ها و تسریع دسترسی به داده ها است. view یک جدول مجازی است.

روش اول:

بر روی شی view در بانک اطلاعاتی راست کلیک کرده و گزینه new view را انتخاب می کنید.

روش دوم:

نام دید create view

دستورات select

نکته ۱: تمامی دستورات (DDL) (alter-drop-create) که روی جدول ها انجام می شود برای دید های نیز قابل انجام است.

نکته ۲: استفاده از دستور `order by` و `compute by` در `select` تشکیل دهنده `view` غیر مجاز است مگر اینکه کلمه `top` را به کار ببریم.

نکته ۳: استفاده از عبارت `select into` در دیدها غیر مجاز است.

مثال:

ایجاد یک دید از جدول دانش آموز فقط ۵ رکورد مرتبط شده بر اساس شماره که به ۳ ختم نشوند.

```
create view v_std as
select top 5 std_name, std_family , std_num
from student innerjoin register on
student.std_num = register.std_num
where std_num not like '%3'
```

9- تراکنش ها

تراکنش عبارت است از مجموعه ای از دستورات SQL که همه با هم اجرا می شوند یا لغو میشوند می تواند شامل `if` ، `for` ، `select` و ... باشد.

ساختار کلی:

نام `begin transaction`

دستورات مورد نظر

`commit transaction`

مثال:

آنمره به نمره دانش آموزان درس ریاضی در قالب `t.s` اضافه کنید.

`begin transaction t`

`update student set grade = grade+2`

```
where std_num in  
(select std_num from register where lessen_name='math' )  
commit transaction
```

در مقابل commit دستور rollback transaction رو داریم که همه دستورات مربوط به t.S را لغو می کند و بانک را به حالت قبل از اجرای آن برمی گرداند.

علاوه براین دو دستور (rollback) و (commit) دستور save transaction را نیز داریم که یک نقطه ذخیره در داخل t.S تعیین می کند. بعد از این کلمه نام یک قسمت از برنامه را می آوریم. بعد از آوردن کلمه save transaction معمولاً با دستور rollback به نقطه ذخیره برمی گردیم. (به اولین نقطه)

```
begin transaction t  
...  
save transaction  
...  
save transaction  
...  
rollback transaction  
commim transaction
```

بعد از رسیدن به rollback به اولین نقطه ذخیره (save transaction1) برمی گردد.

تعریف متغییر در sql transaction ها

متغیر ها توسط دستور declare تعریف میشوند و توسط دستور set یا select مقدار دهی میشوند. به این صورت:

declare @ نام متغییر @ نوع داده

نکته: متغییر های محلی که با declare تعریف میشوند باید با علامت @ شروع شوند.

مثال(مقدار دهی متغییر با set):

پرس وجویی بنویسید که در آن نام تمام کسانی که شامل حرف a می شود برگردانده شود.

```
declare @name varchar
set @name = '%a%'
select std_name from student
where std_name like @name
```

مقدار دهی با **select**: این دستور حتما باید یک مقدار واحد را برگرداند.

```
declare @ave int
select @avg = ave(age) from student
print @avg
```

نکته: بادستور print میتوان محتویات یک متغیر را نمایش داد.

ساختارهای شرطی (**if, else**):

شکل کلی if ها در sql به صورت زیر میباشد.

if شرط یا همان دستورات

```
begin
```

دستورات

```
end
```

```
else
```

```
begin
```

دستورات

```
end
```

مثال: دستوری بنویسید که در صورت بزرگ بودن مجموع نمرات دانش آموز احمدی از ۱۰۰ پیغام مناسب نمایش دهد و

اگر نبود رکورد های مربوط به max و min نمرات نمایش داده شوند.


```
if (select sum(grade) from student
where std_family='ahmadi '
group by std_family)>100
print 'sum grade of ahmadi is more than 100'
else
select min(grade),max(grade) from student
where std_family='ahmadi'
group by std_family
```

ساختار تکرار:

شکل کلی دستور به این صورت است:

```
while شرایط یا دستورات
begin
    دستورات
end
```

دستور **break , continue**:

break: خروج از حلقه - دستورات حلقه دیگر اجرا نمیشود و کنترل برنامه برمی گردد به اولین دستور بعد از **end**

continue: دستورات حلقه به ازای مقدار بعدی حلقه اجرا می شود و برنامه برمی گردد به اولین دستور حلقه

مثال:

دستوری بنویسید که هر بار ۲ نمره به نمره دانش آموزان اضافه کند و اگر **max** آن بیشتر از ۱۸ شد دیگر اضافه نکند. (شرط میانگین نمرات کوچکتر از ۱۲ باشد)

```
<12 while (select ave(grade) from student )
```

```
begin
update student set grade =grade+2
8if (select avg(grade) from student)>1
break
end
```

توابع رشته ای در SQL :

تبدیل کاراکتر به کد اسکی-خروجی یک عدد است.

ascii(char)

تبدیل کد اسکی به کاراکتر-خروجی که کاراکتر است.

char(int)

موقعیت یک زیر رشته در رشته ای دیگر- خروجی یک عدد است.

char index(رشته,زیر رشته)

مقایسه دو رشته- خروجی یک عدد است- اگر خروجی تابع عدد ۳ بود به ان معناست که هر دو رشته نا برابر اند و اگر خروجی عدد ۴ بود رشته ها برابر و مساوی هستند.

diffrence(str1,str2)

از نقطه شروع به تعدادی که ما تعیین می کنیم از سمت چپ رشته یک زیر رشته برمی گرداند.

left(str,عدد,محل شروع یک عدد)

از نقطه شروع به تعدادی که ما تعیین می کنیم از سمت راست رشته یک زیر رشته برمی گرداند.

right(str,عدد,محل شروع یک عدد)

طول رشته را برمی گرداند.

len(str)

تبدیل رشته به حروف کوچک

lower(str)

تبدیل رشته به حروف بزرگ

upper(str)

حذف فضاهاى خالى يك رشته از سمت چپ

trim(str)

حذف فضاهاى خالى رشته از سمت راست

rtrim(str)

جايگزين كردن رشته دوم به جاي رشته اول

replace(رشته دوم,رشته اول,رشته اصلى)

تطابق دادن يك رشته با يك الگو-خروجى يك عدد است اگر عدد صفر بود رشته مانند الگو مى باشد.

pat index(رشته والگو)

تبدیل عدد به رشته عددی

str(int)

يك زیر رشته از نقطه شروع به طول دلخواه برمی گرداند.

substring(طول رشته ,نقطه شروع ,رشته اصلى)

معکوس يك رشته را برمی گرداند.

reverse(str)

نکته : در توابع رشته ای تمام توابعی که ورودی آنها يك رشته است مى تواند به جاي رشته مقدار يك فيلد در آن قرار گیرد.

مثال:

دستوری بنویسید که چهار کاراکتر اول فیلد شماره دانش آموزی یا با مقادیر ۱ تا ۴ جایگزین کند.

```
update student set  
std_num = replace(std_num,left(std_num,4),'123 ')
```

توابع تاریخ و زمان:

اضافه کردن قسمتی از تاریخ (ماه یا روز یا سال) به تاریخ

dateadd(month/date/year, مقدار تاریخ, تعداد)

تاریخ جاری سیستم را برمی گرداند.

get date()

تفاوت دو تاریخ

date diff (تاریخ ۲, تاریخ ۱, قسمتی از تاریخ)

برگرداندن نام قستی از تاریخ مثلا نام ماه یا روز

date name (تاریخ, قسمتی از تاریخ)

برگرداندن قسمت روز تاریخ

day (تاریخ)

برگرداندن قسمت ماه تاریخ

month (تاریخ)

برگرداندن سال تاریخ

year (تاریخ)

تاریخ جاری سیستم به فرم بین الملل

getutodate ()

10-روال های ذخیره شده

روال ذخیره شده (stored procedure) مجموعه ای از دستورات تحت یک نام است که در sql server تعریف شده و ذخیره می شود و می توان روال را از داخل SSMS(نرم افزار مدیریتی sql server)و یا از داخل برنامه های کاربردی در C# ، دلفی و ...فراخوانی کرد.

یک روال می تواند شامل یک دستور ساده باشد یا مجموعه ای از دستورات اس کیو ال، ورودی/خروجی و حتی حلقه ی تکرار و ساختار شرطی. استفاده از روال ذخیره شده در برنامه های کاربردی باعث افزایش سرعت اجرا و کاهش ترافیک شبکه می گردد.

هر روال ذخیره شده یک شی است که می تواند سیستمی یا غیر سیستمی باشد.روال های سیستمی قبلا تعریف شده اند و با ایجاد یک بانک اطلاعاتی جدید به آن بانک اضافه می شوند و ما فقط در T.sql ها آنها را فراخوانی می کنیم ولی روال های غیر سیستمی توسط خود کاربر تعریف شده و همانند نوع سیستمی می توانند بارها و بارها فراخوانی شوند. به مجموعه دستورات t.sql که شامل تعریف متغیر ها ، عملیات روی آنها و دستورات ورودی و خروجی میشود S.p گفته می شود.

ایجاد یک روال ذخیره شده:

پارامتر های ورودی نام روال create procedure

[output] نوع پارامتر ها

دستورات SQL

مثال :

create proc add_stud

```
@name nvarchar(50) ,  
@avgr float ,  
@fid int  
as  
Insert into studs (name ,avgr,fid) values (@name , @avgr , @fid )
```

پارامتر های ورودی روال ها با علامت @ و نوع داده ای تعریف می شوند و پارامتر های خروجی بعد از پارامتر های ورودی با علامت @ و نوع داده ای و کلمه کلیدی output می آیند.

مثال: (پارامتر های خروجی)

```
create proceduer grdesum @sum int output  
as  
select @sum = sum(grade) from student
```

موقع فراخوانی جمع نمرات به عنوان خروجی برگردانده می شود.

پارامتر های ورودی

```
create std_info @name nvarchar(20) , @family nvarchar  
as  
select std_name , std_family, age from student  
where std_name = @name and std_famil=@family
```

در موقع فراخوانی اطلاعات دانش آموزانی برگردانده می شود که برابر با مقادیر فرستاده شده می باشد.

فراخوانی procedure

اگر در قطعه کد t.sql دستور فراخوانی اولین دستور باشد می توان فقط نام پروسیجر را نوشت ولی در صورتی که اولین دستور نباشد باید حتما عبارت exec یا execute را همراه آن نوشت.

فراخوانی روال هایی که ورودی دارند:

مقدار فرستاده شده برای روال=نام پارامتر ورودی @ نام پروسیجر exec

مثال:

```
exec std_info @name='anis',@family='barmar'
```

فراخوانی روال ها بدون ورودی:

نام پروسیجر exec

11-توابع تعریف شده کاربر

تابع تعریف شده کاربر (user define function-UDF) مجموعه از t.sql ها است که یک بار نوشته می شود و چند بار فراخوانی می شود ولی تفاوت آن با پروسیجرهای قبلی در این است که حتما و حداقل یک خروجی را دارد و باید یک مقدار را به عنوان خروجی برگرداند.

```
create function نام تابع ( پارامترها ) returns نوع خروجی  
as  
begin  
....  
return نام متغییر خروجی  
end
```

روش فراخوانی:

select (نامی برای نمایش) as (پارامترها) نام تابع. نام مالک. نام بانک

مثال : تابعی بنویسید که یک عدد را دریافت و فاکتوریل آن را حساب کند.

```
create function fact (@num int)  
returns int
```

```
as
begin
declare @sum int
set @sum =1
declare @counter
set @counter = 0
while @counter<= @num
begin
set @counter = @counter+1
set @sum = @sum *@counter
end
return @sum
end
```

فراخوانی:

```
select class.dbo.fact(3) as 'fact'
```

نکته dbo: یک مالک عمومی است.

توابع جدولی یا: inline table

این نوع تابع نیز حتما خروجی دارد و می توان بارها و بارها از آن استفاده کرد اما تفاوت ان با تابع قبلی در این است که توابع جدولی مجموعه ای از رکورد های جدول را برمی گرداند.

ایجاد یک تابع جدولی:

```
create function (نام پارامترها) نام تابع return table
```

as

```
return (select دستورات)
```


فراخوانی توابع جدولی

```
select * from (پارامترها)نام تابع
```

مثال: اطلاعات دانش آموزانی را برگرداند که معلمشان همان معلمی باشد که به عنوان ورودی به تابع فرستاده می شود.

```
create function std_info(@teacher varchar) returns table
```

```
as
```

```
return (select student.std_name,student.std_family,register.teacher
```

```
where student.std_num =register.std_num
```

```
and register.teacher = @teacher)
```

روش فراخوانی:

```
select * from std_info('ahmadi')
```

راه انداز : Trigger

نوعی روال ذخیره شده است که در مواقع خاص اجرا می شود مثلا در زمان درج داده جدید یا ویرایش داده ها اجرا می شود . توجه کنید که trigger هیچ گاه فراخوانی نمی شود و بلکه اتوماتیک اجرا میشود(البته در زمانی که مشخص می کنید) توسط دستور زیر ایجاد میشود.

```
create trigger نام تریگر
```

```
on نام جدولی که تریگر روی آن عمل می کند
```

```
for insert/update/delete
```

```
as
```

دستوراتی بعد از عمل مشخص شده انجام میشود

نکته: به جای کلمه for می توان از کلمه af , inetred after , نیز استفاده کرد و کلمه after پیش فرض است.

after : بعد از انجام موفقیت آمیز عملیات (insert/update/delete)

for : در همان لحظه اجرا می شود.

ineterd of : تریگر را به یک قطعه کد T.SQL عادی تبدیل می کند و مانند یک پروسیجر معمولی تریگر را می سازد.

ایجاد یک تریگر به صورت ویزاردی:

راست کلیک روی جدول مورد نظر/گزینه all taks/mange trigger و نوشتن کد های مورد نظر

مثال:

```
create trigger std_trigger
on student
for insert/update
as
if (select max(grade) from student) >20
print 'a range is not valid'
```

نتیجه : در لحظه ورود یا تغییر داده ها اگر بزرگترین نمره در جدول از ۲۰ بیشتر بود پیام مناسب را نمایش می دهد.

مثال ۲: تریگری بنویسید که بعد از درج داده جدید در جدول student اگر داده ها بیشتر از ۱۰ رکورد شد یک پیام

خطا چاپ کند و رکورد های درج شده را حذف کند.

```
create trigger insert_trigger
on student
for insert
as
if (select count(*) from student) >10
begin
print 'you cannot insert'
rollback
end
```

مثال: با در نظر گرفتن جداول زیر پرس و جوهای مورد نظر را اجرا کنید

Books(bid , bname , price , pid)

Pubs(pid , pname)

1-لیست کتاب های منتشر شده توسط ناشر با کد 10 را نمایش دهید.

use lib:

```
select * from books where pid =10
```

2-نام کتابهای منتشر شده با ناشر شماره ی 11 را نمایش دهید.

```
select * from books where pid =11
```

3-ناشر تمام کتاب های منتشر شده از 10 به 12 تغییر کند.

```
select * from books set pid =10 where pid =10
```

4-نام کتاب با کد 2 را به B400 و کد ناشر آن را به 11 تغییر دهید.

```
select * from books set bname ="B400",pid=11 where bid =2
```

5-کتاب های با کد 4 را حذف کنید.

```
delete * from books where bid =10
```

6-لیستی به صورت زیر نمایش دهید.

```
bname | price|takhfif|pay
```

```
select bname,price,(price *10)/100 as tskhfif ,price-((price*10)/100) as pay form books
```

7-یک روال ذخیره شده ایجاد کنید که کد رشته را به عنوان پارامتر گرفته و آن رشته را از جدول رشته ها حذف کند و

همچنین دانشجویان آن رشته را از جدول دانشجویان حذف کند.(با دو دستور Delete)

```
create proc add_fit 1 @ fid int as
```

```
delete from field where fib =@fib:
```

delete from studs where fib =@fib:

8-روالی ایجاد کنید که مشخصات دانشجو را به عنوان پارامتر گرفته و سطر جدیدی به بانک اضافه کند.

```
create proc add_stubs2
```

```
@name nvarchar(50),
```

```
@avgr float,
```

```
@fit int
```

```
As
```

```
Insert into studs (name,avgr,fid,)values(@name,@avgr,@fid)
```

برای اجرای روال فوق باید دستور زیر را اجرا کنیم:

```
exec add_stubs2 'mina',17,2
```

با در نظر گرفتن جداول زیر پرس و جویهای زیر را اجرا کنید:

```
Kala ( kid , kname , price )
```

```
Froush ( kid , mid , qty )
```

```
Moshtary ( mid , mname )
```

1-لیست کالاهای فروخته شده به مشتری با کد 10 را به صورت زیر نمایش دهد.(با کد تعیین شده ، با کمک روال ذخیره

شده)

```
Select kala.*,froush .qty as total
```

```
From kala,froush
```

```
Where mid=10 and kala.kid=froush.kid
```

```
Select *from studs  
use eshop:  
create proc add_mid @mid int as  
select kala.*,froush.qty,qty*price as total  
from kala,froush  
where mid =@mid and kala.kid=froush.kid
```

2- لیست مشتریان یک کالا را به صورت زیر نمایش دهید.(با کمک روال دخیره شده)

```
create proc add_kid int as  
select moshtary.mid,moshtary.name,froush.qty,kala.price,qty*price az total  
from kala,froush,moshtary  
where kala.kid =@kid and kala.kid=froush.kid and froush.mid=moshtary . mid
```

3- مجموع فروش را با تفکیک کالا ها نمایش دهید.

```
select kid,sum(qty)total  
from froush group by kid
```

4- مجموع خرید کالا را با تفکیک مشتریان نمایش دهید.

```
select mid,sum(qty)total  
from froush group by mid
```

5- لیست کالاهایی را نمایش دهید که فروش نداشته اند.

6- لیست مشتریانی را نمایش دهید که خرید نداشته اند.

7- مشخصات کالاهایی را نمایش دهید که بیشترین فروش را داشته اند.