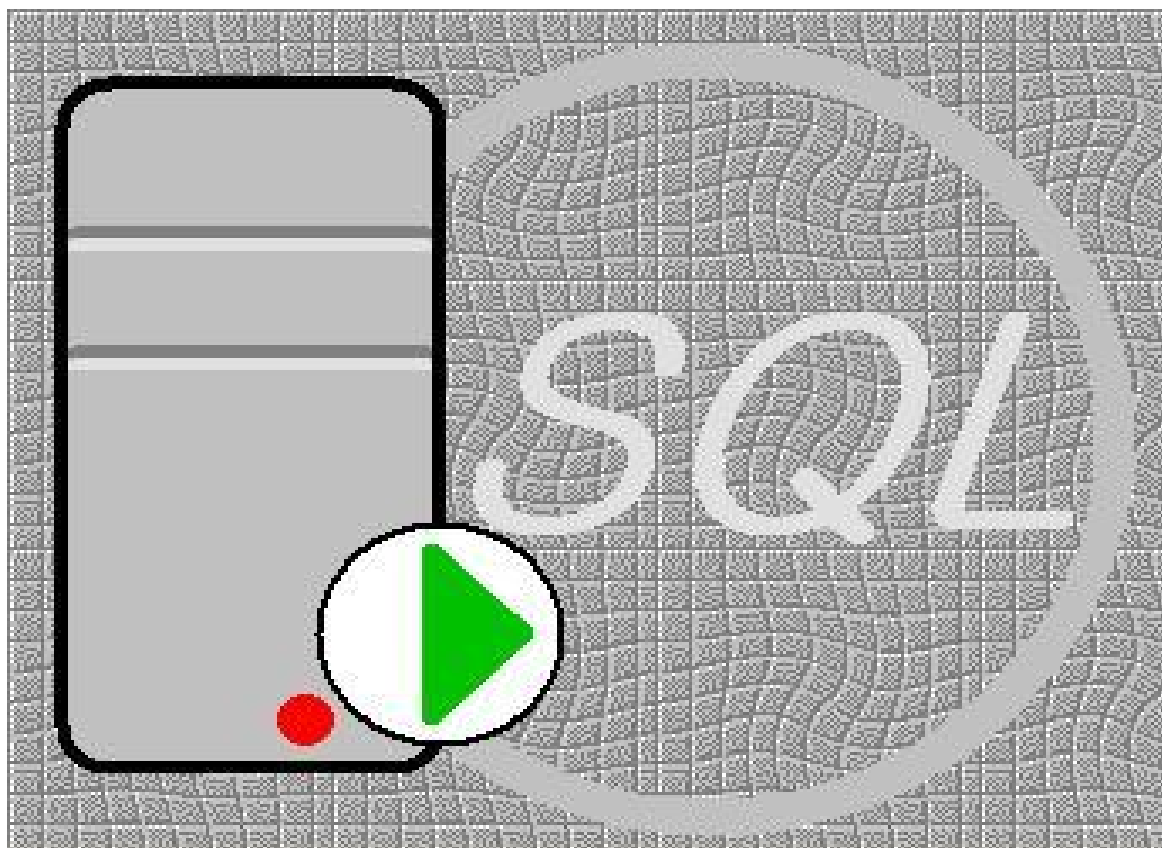


بسمه تعالی



برنامه نویسی بانک اطلاعاتی SQL Server

نویسنده: عباس صفای کفاش

Kaffash@RAI.IR

بهار 86

مقدمه نویسنده

یکی از ویژگیهای حائز اهمیت در یک پایگاه داده قدرتمند قابلیت برنامه نویسی است ، این ویژگی باعث کاهش میزان زیادی از سورش برنامه در قسمت برنامه کاربردی شده و وابستگی بخشی از پروژه را از برنامه کاربردی از بین خواهد برد. کتب زیادی درخصوص پایگاه داده SQL Server به رشته تحریر درآمده است اما اکثریت آنها تنها 20% از مطلب ارائه شده را به برنامه نویسی پایگاه داده اختصاص داده اند ، بدین منظور در این جزوه سعی برآن است که 100% موضوع به برنامه نویسی و تکنیک های برنامه نویسی در SQL Server پرداخته شود ، بدون شک مطالب گردآوری شده خالی از اشکال نبوده و شایسته است که خوانندگان محترم اشکالات فنی و نگارشی را مستقیماً به اینجانب منعکس نمایند. درضمن پیشاپیش از زحمات سرکار خانم بابازاده بابت تایپ و حروف چینی مطالب و پیگیری های مستمر سرکار خانم اباسینکی تشکر و قدردانی می نمایم. امید است مجموعه تهیه شده مفید برای برنامه نویسان کاربردی که از پایگاه داده SQL Server استفاده می نمایند واقع گردد.

کفاش

فصل اول

دستورات برنامه نویسی در

SQL Server

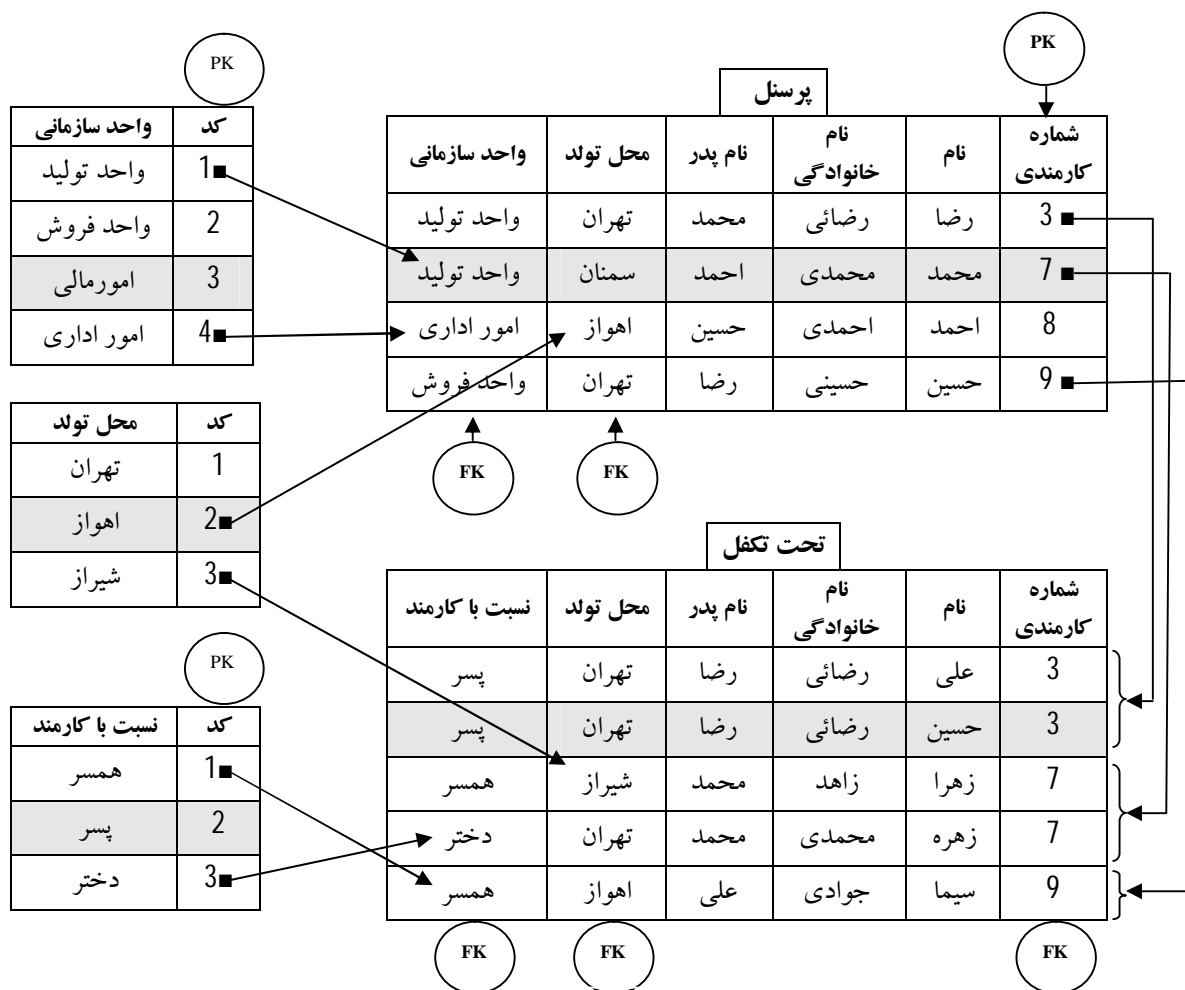
مفاهیم پایه

- **RDBMS** : در واقع RDBMS یا سیستم مدیریت بانک های اطلاعاتی رابطه ای (Relational Data Base Management System) موتوری است که وظیفه دریافت و ذخیره سازی و برقراری ارتباطات بین Object های درون پایگاه داده را بعهده دارد.

- **DBA** : بطور خلاصه شخصی است که وظیفه مدیریت سرویس دهنده پایگاه داده را بعهده دارد و معمولاً وظایفی نظیر مدیریت کاربران ، گروههای بانک اطلاعاتی ، مدیریت منابع قابل اشتراک ، تعیین مجوزها و مواردی مشابه را عهده دار است.

- **پایگاه داده** : مجموعه ای از Object هایی نظیر جداول ، View ها ، انواع داده ها ، Trigger ها و غیره است که به وسیله RDBMS مدیریت می گردد .

- **Table** : همانطور که از نام آن مشخص است یک جدول مجموعه ای از سطرها و ستونهاست هر جدول از تعدادی ستون که فیلد نامیده می شود تشکیل می گردد هر ردیف اطلاعاتی در یک جدول رکورد نامیده می شود. روابط بین جدول ، فیلد و رکورد را در مثال ذیل ملاحظه فرمائید:



- مطابق شکل موارد زیر از جدول قابل استخراج است :
- 1- کل پایگاه داده از پنج جدول تشکیل گردیده است که جدول اصلی جدول پرسنلی می باشد و هر جدول می بایستی یک نام داشته باشد.
 - 2- در هر یک از جداول به هرستون یک فیلد اطلاعاتی گفته می شود.
 - 3- هر فیلد اطلاعاتی در هر جدول حداقل می بایستی نام ، نوع و اندازه داشته باشد.
 - 4- به هر ردیف اطلاعاتی در هر جدول یک رکورد آن جدول گفته می شود که شامل اطلاعات کامل آن ردیف است.
 - 5- به هر مشخصه منحصر به فرد در یک جدول که در جدولی دیگر دارای تکرار باشد یک **Primary Key (PK)** گفته می شود و بعنوان مثال در جدول پرسنلی شماره کارمندی یک **PK** است و کد واحد سازمانی در جدول واحد سازمانی یک **PK** است.
 - 6- به هر فیلد اطلاعاتی در جدولی که به یک **PK** متصل گردد و رابطه ای $1 \times n$ داشته باشد یک **Foreign Key (FK)** گفته می شود.
 - 7- به منظور افزایش سرعت دسترسی به اطلاعات فیلدهای **PK** و **FK** مناسب است که شاخص بندی گردند.

- **View ها :** یک جدول مجازی است که می تواند ستونهای چند جدول را بطور همزمان با استفاده از روابط بین جداول به همراه داشته باشد و از دیدگاه ساده سازی و امنیت از ویژگیهای باارزشی برخوردارند.
- **ایندکس ها (Index) :** هر ایندکس در بانک اطلاعاتی ترتیب قرار گرفتن صعودی یا نزولی یک یا چند ستون را در یک جدول شامل می گردد و کاربرد عمده آن روش دسترسی سریع به اطلاعات می باشد و بهترین روش برای برقراری ارتباط بین **PK** و **FK** در چند جدول نیز محسوب می گردد.
- **Stored Procedures (SP) :** مجموعه ای از دستورات به زبان برنامه نویسی پایگاه داده می باشد و از نظر منطقی مشابه یک روتین در زبانهای برنامه نویسی متداول می باشد.
- **User Define Function (UDF) :** مجموعه ای از دستورات به زبان برنامه نویسی پایگاه داده می باشد که یک خروجی شامل یک داده یا یک جدول را ارائه می دهند و عملکرد آن از نظر منطقی بسیار مشابه توابع برنامه نویسی در زبانهای متداول است.
- **Extended Stored Procedures (ESP) :** این نوع از برنامه های قابل اجرا توسط پایگاه داده در واقع یک برنامه به زبان C و با نوع پروژه **ESP** در زبان C است که قادرند توابع یا عملیاتی را برای پایگاه داده به اجرا درآورده و در صورت نیاز خروجی لازم را به پایگاه داده منعکس نمایند و یک فایل از نوع **DLL** می باشد در این حالت لازم است که درون پایگاه داده یک نوع ارتباط **ESP** به **DLL** ایجاد شده تعریف گردد.

- Trigger : مجموعه ای از دستورات به زبان برنامه نویسی پایگاه داده می باشند که در صورتیکه یک Event از نوع اضافه ، حذف ، تصحیح ، ... روی یک جدول پایگاه داده اتفاق افتد اجرا می گردد. شایان ذکر است که در استفاده از Trigger ها بعلت اینکه ممکن است در مواردی باعث ایجاد سر بار گردد می بایستی دقت بیشتری بعمل آید.

- ظرفیت پایگاه داده SQL Server 200X : به منظور آشنائی بیشتر مناسب است یک برآورد از حجم محدودیت هائی که در یک پایگاه داده SQL Server 200X می تواند متصور باشد در ذیل ارائه می گردد.

ملاحظات	میزان محدودیت	موضوع محدودیت
	250 MB	اندازه سایز یک SP
	8060 Byte	طول یک رکورد اطلاعاتی دریافتی
	1	تعداد ایندکس کلاستری در یک جدول
	249	تعداد ایندکس غیر کلاستری در یک جدول
	16	تعداد ستونهای انتخابی در هر ایندکس
	16	تعداد فیلدهای درون یک PK یا FK
	1024	تعداد ستون های درون یک جدول
	4096	تعداد ستون قابل استخراج در دستور Select
	1024	تعداد ستون قابل ارائه در دستور Insert
	یک میلیون ترابایت	سایز یک بانک اطلاعاتی
	32767	تعداد بانک اطلاعاتی درون یک Server
	32	سطح پیشروی SP ها
	32	سطح پیشرو Sub Query
	حدود 2 میلیارد	تعداد Object های درون یک پایگاه داده
	به میزان محدودیت Object ها	تعداد Trigger درون یک جدول

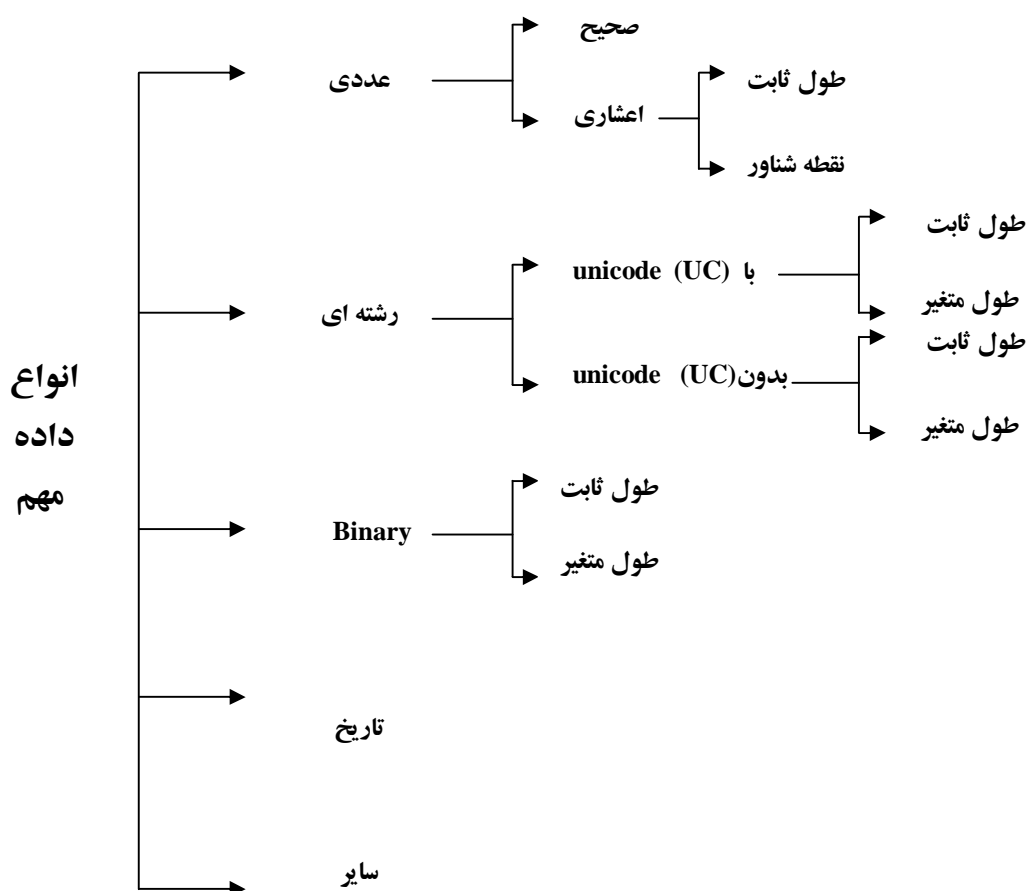
- برنامه نویسی SQL Server : اگر از برنامه نویسان باتجربه سؤال شود که چگونه می توان با سرعت و بادانستن یک زبان برنامه نویسی به زبان برنامه نویسی دیگر مهاجرت کرد احتمالاً یادگیری زبان برنامه نویسی جدید را به شکل ذیل تقسیم بندی خواهند کرد:

1) روش تعریف و مقداردهی به داده ها در زبان برنامه نویسی

- (2) دستورات شرطی در زبان برنامه نویسی
- (3) حلقه های تکرار در زبان برنامه نویسی
- (4) توابع داخلی درون زبان برنامه نویسی
- (5) طراحی توابع خارجی (UDF) و سابروتین ها در زبان برنامه نویسی
- (6) سایر دستورات مهم و کاربردی در زبان برنامه نویسی

حال با این رویکرد یک برنامه نویس باتجربه خیلی سریعتر می تواند در یک مجموعه به دنبال سئوالات خویش باشد و این طبقه بندی مراحل یادگیری زبان را کوتاه خواهد کرد حال با این رویکرد مطالب در زبان برنامه نویسی SQL Server را ادامه میدهم :

1 - انواع داده ها : در هر زبان برنامه نویسی دانستن انواع مهم داده ها و محدودیت های هر نوع داده از اهمیت ویژه ای برخوردار است بدین منظور انواع داده ها یا انواع فیلدهای یک جدول اطلاعاتی به شرح جدول ذیل قابل استفاده می باشد شایان ذکر است که در این جدول مهمترین انواع داده های متداول که کاربرد وسیع تری دارد مدنظر قرار گرفته است.



نام داده	نوع	محدودیت 1	محدودیت 2	ملاحظات
bit	عددی / صحیح	0 و 1		
TinyInt	عددی / صحیح	0	255	مناسب برای کد نسبت
SmallInt	عددی / صحیح	- 32768	+32767	مناسب برای کد واحد سازمانی
Int	عددی / صحیح	حدود منفی دو میلیارد	حدود مثبت دو میلیارد	مناسب برای فیلد شماره کارمندی
BigInt	عددی / صحیح	حدود عدد 18 رقمی منفی	حدود عدد 18 رقمی مثبت	مناسب برای فیلد مبلغ
Decimal یا Numeric	عددی / اعشاری با طول ثابت	$-10^{38} + 1$	$+10^{38} - 1$	
Float	عدد اعشاری با اعشار شناور	$-1,79 \times 10^{308}$	$+1,79 \times 10^{308}$	
Real	عدد اعشاری با اعشار شناور	$-3,4 \times 10^{38}$	$+3,4 \times 10^{38}$	
Char	رشته ای با طول ثابت	8000 ^{Byte}		مناسب برای فیلد تاریخ شمسی
VarChar	رشته ای با طول متغیر	8000 ^{Byte}		مناسب برای رشته های انگلیسی
Text	رشته ای با طول متغیر	2GB		مناسب برای رشته های انگلیسی
nChar	رشته ای با طول ثابت * با uc	4000 ^{Byte}		
nVarChar	رشته ای با طول متغیر با uc	4000 ^{Byte}		بسیار مناسب برای فیلدهای فارسی مانند نام خانوادگی
nText	رشته ای با طول متغیر با uc	1GB		بسیار مناسب برای متن های فارسی مثل متن نامه

* UC = UniCode

نام داده	نوع	محدودیت 1	محدودیت 2	ملاحظات
Binary	باینری با طول ثابت	8000 ^{Byte}		جهت ذخیره سازی فایل های باینری نظیر عکس ، فیلم ، متن ، ... مناسب است ولی فضای بیشتری نسبت به Varbinary اشغال خواهد کرد.
Varbinary	باینری با طول متغیر	8000 ^{Byte}		مناسب جهت ذخیره سازی فایل های binary نظیر فرمت Power Point ، Word , Excel ، تصاویر گرافیکی Tiff , Bmp , Gif , ...
Image	باینری با طول متغیر	2GB		مشابه Varbinary ولی با حجم خیلی مناسب تر
Table	-	-	-	برای تبادل بین توابع، SP ها مناسب است
Time Stamp	عددی	-	-	یک عدد منحصر به فرد در هر پایگاه که توسط پایگاه داده ایجاد می گردد
Unique Identifier	عددی	-	-	یک عدد منحصر به فرد در جدول که بطور اتوماتیک تولید می گردد (به شکل سریالی)

- عملگرهای ریاضی و منطقی: مشابه کلیه زبانهای برنامه نویسی زبان بانک اطلاعاتی SQL Server عملگرهایی برای انجام امور ریاضی و منطقی بهره می جوید که در جدول ذیل ارائه گردیده است:

عملگر مقایسه ای		عملگر منطقی		عملگر ریاضی		سایر	
مساوی	=	عملکرد	علامت	عملکرد	علامت	عملکرد	علامت
کوچکتر	<	و	and	جمع	+	مقایسه شباهت	IS
کوچکتر مساوی	<=	یا	or	تفریق	-	وجود در یک مجموعه	IN
بزرگتر	>	نیست	Not	ضرب	*	وجود در یک محدوده	Between
بزرگتر مساوی	>=			تقسیم	/	وجود شباهت رشته ای	Like
مخالف	<>			باقیمانده تقسیم	%		

- **تعریف یک متغیر**: نامگذاری متغیرها در SQL Server مشابه همه زبانهای برنامه نویسی از ترکیب حروف انگلیسی و اعداد و برخی از کارکترها مشروط به اینکه نام متغیرها با اعداد شروع نشود را شامل می گردد و براساس قالب ذیل قابل ارائه است و به منظور جلوگیری از شباهت نام فیلد و نام متغیر در SQL Server کلیه متغیرها با حرف @ معرفی می گردند.

نام متغیر @ Declare [, ...] نوع متغیر

بعنوان مثال در صورتیکه قرار است متغیری در دامنه 0 تا 255 تعریف شود بدین گونه عمل می نمائیم.

Declare @x TinyInt

مقداردهی به متغیرها برخلاف زبانهای برنامه نویسی غیر پایگاه داده ای از کلمات کلیدی Set یا Select به شکل ذیل صورت می گیرد ولی پیشنهاد می گردد بعلت شباهت به دستور Select از کلمه Set استفاده گردد.

عبارت = نام متغیر @ Set

یا

[, ...] عبارت = نام متغیر @ Select

بعنوان مثال داریم:

Set @x = 14

Set @x = @x + 2 * @y

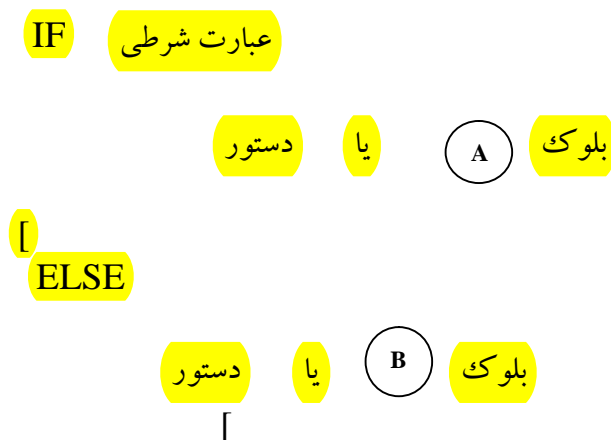
2- **دستورات شرطی**: دستورات شرطی در زبان SQL Server به دستور If ، تابع Case محدود می گردد قبل از تشریح هریک از دستورات فوق می بایستی مفهوم بلوک دستورات در زبان SQL Server تشریح گردد که به شکل ذیل یک بلاک دستورات ایجاد می گردد.

Begin

===== مجموعه دستورات
===== درون بلاک

End

و در صورتیکه تعداد دستورات یک بلاک تنها شامل یک دستور باشد از نوشتن کلمات Begin و End می بایستی صرف نظر گردد. با این توضیحات شکل کلی دستور شرطی If به شکل ذیل خواهد بود :



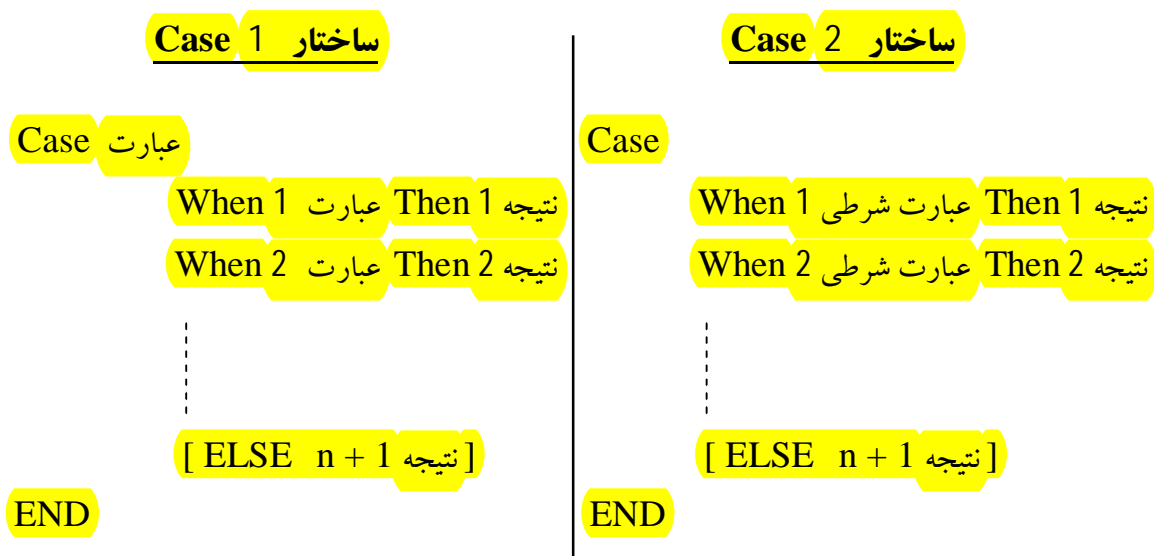
همانطور که از ظاهر دستور مشاهده می گردد در صورتیکه عبارت شرطی صحیح باشد دستورات بخش A و در غیراینصورت دستورات بخش B اجرا خواهد شد و در صورتیکه نیازی به ساختار ELSE نباشد می توان از این بخش صرف نظر نمود ، حال براساس توضیحات قبل هر بلوک در صورت نیاز می بایستی در محدوده یک Begin و End محصور گردد. بعنوان مثال داریم:

```

Declare  @x Int , @y Int
Set      @x= 200
Set      @y= 300 - @x
IF       @x > @y
      Set  @y = @x + 2
ELSE Begin
      Set  @y = @x + 3
      Set  @x = @x + 1
END
  
```

در این قطعه برنامه چون در بخش ELSE بیش از یک دستور بکار رفته است بنابراین از بلاک Begin ، End استفاده شده است.

- ساختار Case : با توجه به اینکه ساختار Case یک مقدار بازگشتی ارائه می دهد. بنابراین می توان گفت که ساختار Case عملکرد تابعی دارد و در دو شکل کلی ذیل قابل ارائه است.



اختلاف ظاهری دو ساختار نشان می دهد که ساختار 2 قدرتمندتر از ساختار 1 است چون بنحوی ساختار 1 را در خود جای می دهد برای درک بیشتر به مثال های ذیل دقت کنید:

```
Declare @x Int
Set @x = ( Case @y
           When 1 Then 102
           When 2 Then 203
           When 3 Then 304
           ELSE      1000
         END)
```

دقت گردد که چون ساختار Case عملکرد تابعی دارد و مقداری را باز می گرداند بنابراین از علامت () مناسب است استفاده گردد. حال اگر این قطعه برنامه با ساختار 2 ایجاد گردد خواهیم داشت:

```
Declare @x Int
Set @x = ( Case
           When @y=1 Then 102
           When @y=2 Then 203
           When @y=3 Then 304
           ELSE      1000
         END)
```

باتوجه به اینکه ساختار 2 از عبارت شرطی بهره می جوید قدرتمندی بیشتری را ارائه می دهد بعنوان مثال:

```
Declare @x Int
Set @x = Case When @y Between 2 and 10 Then 300
              When @y=15 or @y=17 Then 310
              When @y=18 or @y>=20 Then 320
              When @y+@z <= 30 Then 330
              When Not @z <= 40 Then @z
              ELSE @y END
```

بعدها خواهید دید که در دستوراتی نظیر Select استفاده از Case در مواردی بسیار مفید واقع خواهد شد.

مسئله (در قطعه برنامه ذیل نوع داده های A تا G را تعیین کنید:

```
Set @A = Case
          When @B Then 10
          When @C=0 Then @D
          When @D=@E Then Len (@F)
          When @F=@G Then 20
        END
```

3 - حلقه های تکرار : به منظور پیاده سازی ساختار حلقه تکرار در پایگاه داده SQL Server تنها از

حلقه While استفاده می گردد که در دو شکل ذیل قابل پیاده سازی است:



در شکل ساختار 2 تا زمانیکه عبارت شرطی درست باشد تک دستور اجرا می گردد. ولی در ساختار

پیچیده تر 1 کلید دستورات بلاک Begin , END در صورت صحیح بودن عبارت شرطی اجرا خواهند

شد و کلمه کلیدی Break جهت قطع حلقه و Continue حرکت به سمت ابتدای حلقه تکرار خواهد بود.

بعنوان مثال:

```
Declare @x Int , @y BigInt
Set @x = 0
Set @y = 1
While @x < 10 Begin
    Set @x = @x + 1
    Set @y = @y * @x
END
```

و یا در مثال ذیل از دستور Break استفاده گردیده است.

```
Declare @x Int , @y BigInt
Set @x = 0
Set @y = 1
While 1=1 Begin
    Set @x = @x + 1
    IF @x >= 10 Break
    Set @y = @y * @x
END
```

4 - توابع داخلی: در هر زبان برنامه نویسی آشنائی با توابع داخلی مهم زبان از اهمیت ویژه ای برخوردار است ، دانستن پارامترهای ورودی و عملکرد خروجی تابع داخلی قابلیت برنامه نویسی را افزایش داده و باعث طراحی نرم افزارهای بسیار پیچیده تر خواهد شد بدین منظور تعدادی از توابع داخلی مهم در SQL Server تشریح می گردد.

- **Left**: این تابع تعدادی کارکتر از سمت چپ یک رشته استخراج می نماید و در شکل کلی ذیل بکار می رود.

خروجی
 Left (تعداد کارکتر, رشته) → تعدادی کارکتر از سمت چپ رشته استخراج شده است.

- **Right**: این تابع تعدادی کارکتر از سمت راست یک رشته استخراج می نماید و در شکل کلی ذیل بکار می رود.

خروجی
 Right (تعداد کارکتر, رشته) → تعدادی کارکتر از سمت راست رشته استخراج شده است.

- **SubString** : این تابع تعدادی کارکتر از نقطه ای از یک رشته را استخراج می کند و در شکل کلی ذیل بکار می رود.

تعدادی کارکتر از نقطه شروع \longrightarrow ([تعداد , شروع استخراج , رشته) **SubString**
یک رشته استخراج می گردد

در این تابع در صورتیکه پارامتر تعداد ذکر نگردد از نقطه شروع تا انتهای رشته فرآیند استخراج زیر رشته صورت می پذیرد. بعنوان مثال در دستور ذیل مقدار ذخیره شده در متغیر x رشته Ali خواهد بود.

Set @x = Left ('And', 1) + SubString ('Hello', 3, 1) + Left ('in', 1)

- **تابع LTrim** : این تابع فضاهای خالی از سمت چپ یک رشته را حذف می کند و در شکل کلی ذیل بکار می رود:

حذف فضای خالی سمت چپ رشته \longrightarrow **خروجی**
LTrim (رشته)

- **تابع RTrim** : این تابع فضای خالی از سمت راست یک رشته را حذف می کند و در شکل کلی ذیل بکار می رود:

حذف فضای خالی سمت راست رشته \longrightarrow **خروجی**
RTrim (رشته)

بعنوان مثال در دستور ذیل کلیه فضاهای خالی دو طرف رشته حذف خواهند شد :

Set @x = LTrim (RTrim (' Hello '))

- **تابع Upper** : این تابع حروف یک رشته را به حروف بزرگ تبدیل خواهد کرد و به شکل کلی ذیل بکار می رود.

رشته به حروف بزرگ تبدیل خواهد شد \longrightarrow **خروجی**
Upper (رشته)

- تابع **Lower** : این تابع یک رشته را به حروف کوچک تبدیل خواهد کرد و به شکل کلی ذیل بکار می رود.

خروجی
رشته به حروف کوچک تبدیل خواهد شد $\xrightarrow{\text{Lower (رشته)}}$

- تابع **Len** : این تابع تعداد کاراکترهای یک رشته را محاسبه می کند و در شکل کلی ذیل بکار می رود.

خروجی
طول رشته به عدد $\xrightarrow{\text{Len (رشته)}}$

مثال : برنامه ای بنویسید که یک رشته را معکوس نماید.

```
Declare @x VarChar (50) , @C TinyInt , @y VarChar (50)
Set @x = 'Hello'
Set @y = ''
Set @C = 0
While @C < Len (@x) Begin
    Set @C = @C + 1
    Set @y = SubString (@x , @C , 1) + @y
END
```

- تابع **Char** : این تابع کاراکتر مورد نظر را با توجه به کد اسکی آن باز می گرداند و به شکل ذیل بکار می رود.

خروجی
کاراکتر $\xrightarrow{\text{Char (شماره کاراکتر)}}$

- تابع **Reverse** : این تابع یک رشته را معکوس می کند و در شکل کلی ذیل بکار می رود.

خروجی
رشته معکوس می گردد $\xrightarrow{\text{Reverse (رشته)}}$

- **تابع Replace** : این تابع یک مجموعه رشته ای را کاراکتر به کاراکتر ترجمه می کند و در شکل کلی ذیل بکار می رود.

خروجی

رشته اصلی کاراکتر به \longrightarrow (کاراکترهای ترجمه شونده، کاراکترهای قابل ترجمه، رشته اصلی) Replace کاراکتر ترجمه می شود

بعنوان مثال داریم :

Set @x = Replace ('Hello', 'ale', 'xyz')

که در نتیجه حروف ale نظیر به نظیر به حروف XYZ تبدیل خواهند شد و متغیر X شامل رشته 'Hzyyo' خواهد بود.

- **تابع Space** : این تابع به اندازه اعلام شده فضای خالی ایجاد خواهد کرد و در شکل کلی ذیل بکار می رود.

خروجی

به اندازه تعداد فضای خالی ایجاد می گردد \longrightarrow (تعداد) Space

- **تابع User_name** : این تابع به شکل (User_name) نام کاربری که هم اکنون به SQL Server متصل شده است را باز می گرداند.

- **تابع ABS** : این تابع قدر مطلق یک عدد را تعیین می کند و معادل |x| در ریاضی است: (عدد) ABS

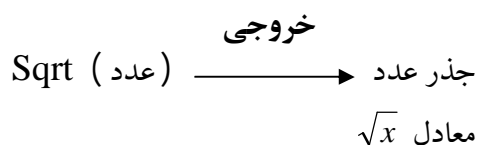
- **تابع Floor** : این تابع جزء صحیح یک عدد را تعیین می کند و معادل [x] در ریاضی است: (عدد) Floor

- **تابع Power** : در پایگاه داده SQL Server عملگری برای توان وجود ندارد و می بایستی با استفاده از تابع Power به شکل ذیل استفاده گردد.

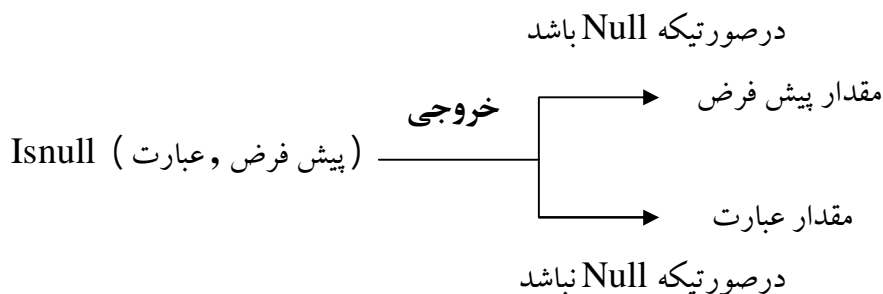
خروجی

عدد به توان خواهد رسید \longrightarrow (توان، عدد) Power

- **تابع Sqrt** : در صورتیکه در نظر باشد ریشه دوم یک عدد مثبت حاصل شود از تابع Sqrt به شکل ذیل استفاده می گردد.



- **تابع Isnull** : این تابع بسیار مفید اعتبارسنجی لازم جهت Null بودن یک مقدار را بعهدده دارد و به شکل کلی ذیل بکار می رود.



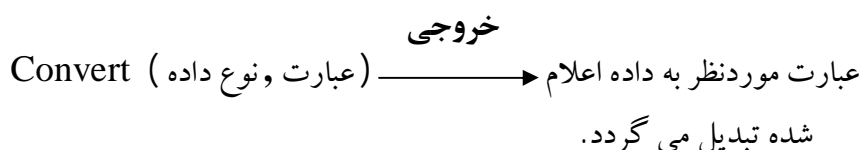
بعنوان مثال داریم :

Set @x = Isnull (@y,0)

در این حالت در صورتیکه متغیر y مقدار Null را داشته باشد مقدار x صفر خواهد بود در غیر این صورت مقادیر x و y مساوی خواهند شد.

- **تابع GetDate** : این تابع تاریخ و ساعت جاری Server را باز می گرداند.

- **تابع Convert** : به منظور تبدیل انواع داده ها به یکدیگر از این تابع تحت شرایط خاصی استفاده می گردد که قالب کلی آن به شکل ذیل خواهد بود.



بعنوان مثال خواهیم داشت:

```
Set @x = Convert (VarChar (10) , @y )
```

که در این دستور متغیر @y به یک نوع داده ای کاراکتری بدون Unicode و حداکثر طول 10 بایت تبدیل شده و در متغیر X جانشین می گردد.

- تابع **CharIndex** : از این تابع بسیار مفید به منظور تعیین مختصات قرار گرفتن یک رشته فرعی در یک رشته اصلی استفاده می گردد که در شکل کلی ذیل قابل ارائه می باشد.

CharIndex ([شروع جستجو] , رشته اصلی , رشته فرعی) $\xrightarrow{\text{خروجی}}$ موقعیت رشته فرعی در رشته اصلی را بدست می دهد

به مثال ذیل دقت کنید:

```
IF CharIndex (@x , @y) < > 0
    Set @z = 'Ok'
ELSE
    Set @z = 'Not'
```

5 - طراحی توابع خارجی و سابروتین :

- طراحی **(SP) Stored Procedure** : یک SP در واقع یک برنامه درون پایگاه داده است که از مجموعه ای از دستورات تشکیل شده است و می تواند مقدار یا جدولی را به برنامه صداکننده انتقال دهد و در شکل کلی ذیل قابل پیاده سازی است:

```
Create Proc[edure]      نام SP
[
    [پیش فرض =] نوع پارامتر 1 [AS] پارامتر 1
    [ [پیش فرض =] نوع پارامتر 2 [AS] پارامتر 2, ]
AS
    ]
```

_____ } دستورات

همانطور که مشاهده می گردد هر SP با یک نام منحصر به فرد قابل تعریف است و در صورتیکه این برنامه تعدادی ورودی داشته باشد می بایستی در ابتدای SP تعیین گردد و اگر قرار است در صورت عدم ارائه پارامتر مقدار پیش فرض به برنامه انتقال یابد می توانید از مقدار پیش فرض هر پارامتر استفاده نمائید پس از کلمه کلید AS کلید دستورات قابل اجرا خواهند بود.

- **دستور ساده Select** : دستور ساده Select قادر است مقداری را بازگرداند و به شکل ذیل بکار می رود.

Select [... عبارت 2 , عبارت 1]

بعنوان مثال خواهیم داشت :

Select 2*Len ('Hello') + 1

که در این حالت مقدار عدد 11 به برنامه اصلی باز خواهد گشت.

- **روش اجرای SP** : پس از طراحی هر SP نیاز است که SP طراحی شده اجرا گردد و به اشکال ذیل یک SP اجرا خواهد شد.

نام SP [پارامترهای SP
در صورت وجود]

EXEC [ute] نام SP [پارامترهای SP
در صورت وجود]

مثال : برنامه ای بنویسید که یک رشته ساعت با ساختار hh:mm:ss را به ثانیه تبدیل نماید.

```
Create Proc Time2Val
@T Char(8)
AS
Declare @h TinyInt , @m TinyInt , @s TinyInt
Set @h = Convert (TinyInt , SubString (@T,1,2))
Set @m = Convert (TinyInt , SubString (@T,4,2))
Set @s = Convert (TinyInt , SubString (@T,7,2))
Select @h*3600 + @m*60 + @s
```

- طراحی **UDF (User Define Function)** : یکی از قابلیت های پایگاه داده SQL Server ایجاد UDF است که کاربران قادرند متناسب با نیاز در هر پروژه به طراحی آن اقدام نمایند و دارای شکل کلی ذیل می باشد.

Create Function [...] [=] نوع پارامتر [AS] نام پارامتر (@) نام تابع

Returns نوع مقدار برگشتی

[AS]

Begin

_____ } دستورات
 _____ }
 _____ }
 _____ }
 _____ }

Return مقدار بازگشتی

End

هر UDF طراحی شده بعنوان یک تابع در کلیه دستورات SQL قابل استفاده است از طرفی در صورتیکه از EXECUTE استفاده گردد هر UDF می تواند مانند یک SP اجرا گردد همچنین در یک تابع UDF خروجی می تواند بصورت یک جدول باشد در این حالت از کلمه Table به جای نوع مقدار برگشتی استفاده خواهد شد و مقدار بازگشتی یک دستور Select خواهد بود که در مقابل دستور Return بکار خواهد رفت و تعداد پارامترهای تعریف شده در هر UDF حداکثر 1024 عدد خواهد بود و مشابه SP در صورتیکه پارامتر موردنظر ارائه نگردد پیش فرض بعنوان مقدار ورودی پارامتر مدنظر قرار خواهد گرفت نکته قابل ذکر در این خصوص به شرح ذیل می باشد:

- در هنگام اجرای یک تابع نام Owner می بایستی ذکر گردد بعنوان مثال (dbo)
- تعدادی از دستورات درون طراحی UDF قابل اجرا نمی باشند.
- از Extended Stored Procedure (ESP) درون تابع می توانید استفاده نمائید.

مثال: تابعی بنویسید که فاکتوریل یک عدد را محاسبه کند.

```

Create Function Fact (@n TinyInt)
Returns BigInt
AS
Begin
    Declare @RtnValue BigInt
    IF @n = 0 or @n = 1
        Set @RtnValue = 1
    ELSE
        Set @RtnValue = dbo.Fact (@n - 1) * @n
    Return @RtnValue
END

```

حال فرض کنید که قرار است تابع فوق در یک برنامه استفاده گردد و قرار باشد سری $\sum_{i=1}^{10} i!$ محاسبه گردد بدین منظور به شکل ذیل عمل خواهیم نمود:

```

Declare @S BigInt , @C TinyInt
Set @S = 0
Set @C = 0
While @C < 10 Begin
    Set @C = @C + 1
    Set @S = @S + dbo.Fact (@C)
END
Select @S

```

6 - سایر دستورات کاربردی :

- **دستور Goto** : مشابه اکثریت زبانهای برنامه نویسی این دستور ادامه برنامه را به نقطه ای دیگر که دستور Goto تعیین می کند انتقال می دهد و در شکل ذیل ارائه می گردد:

Goto نام برچسب

و نام برچسب تابع قانون نامگذاری متغیرهاست و نقطه ای که برچسب شروع می گردد با نام برچسب و علامت : معرفی می گردد. بعنوان مثال:

```

IF @x > 0 Goto Endof
ELSE
    Set @y = @y + 1
Set @x = ABS (@x)
Endof:

```

- دستور **Return** : این دستور ادامه اجرای یک SP را پایان می دهد و به برنامه اصلی بازگشت می نماید.

- دستور **EXEC[ute]** : در صورتیکه در نظر است دستوری در SQL Server اجرا گردد در دو حالت با استفاده از EXEC[ute] عمل انجام خواهد شد.

<u>ساختار 1</u>	<u>ساختار 2</u>
دستور EXEC[ute]	EXEC[ute] (متغیر رشته ای)

در ساختار 1 دستور اعلام شده توسط SQL Server اجرا خواهد شد در صورتیکه در ساختار 2 عبارت رشته ای که می تواند کاملاً متغیر باشد توسط دستور EXEC[ute] اجرا می گردد که قابلیت فوق العاده ای را ایجاد خواهد کرد بدین منظور به مثال های ذیل توجه فرمائید.

مثال : مقدار "12:13:14" را با استفاده از تابع Time2Val حساب کنید.

```
EXEC dbo.Time2Val ('12:13:14')
```

مثال : با استفاده از دستور EXEC[ute] مقدار دو متغیر را جمع کنید:

```
Declare @x Int , @y Int , @S VarChar (4000)
Set @x = 10
Set @y = 20
Set @S='Select'+ Convert (VarChar (10),@x) +'+'+ Convert (VarChar(10),@y)
EXEC (@S)
```

همانطور که مشاهده می گردد مقدار بازگشتی عدد 30 خواهد بود بدون اینکه از دستور Select بصورت مستقیم استفاده شده باشد.

- **دستور Begin Trans[Action]** : در صورتیکه قرار است عملیات حساسی روی پایگاه داده به خصوصی در زمان ذخیره سازی اطلاعات صورت گیرد استفاده از TransAction باعث می گردد که حاشیه اطمینان به 100% افزایش یابد و عملیاتی نظیر قطع ناگهانی برق و یا سایر موارد باعث تخریب عملیات نگردد (نظیر عملیات حساب بانکی) در این حالت با اعلام Begin TransAction به Server اعلام می گردد که شروع فرآیند حساس آغاز گردیده است.

- **دستور Commit [Trans[Action]]** : در صورتیکه کلیه فرآیندهای موردنظر در یک زیر برنامه با موفقیت صورت پذیرد با استفاده از Commit TransAction می توان انتقال اطلاعات را به پایگاه داده با حاشیه اطمینان کامل صادر نمود.

- **دستور RollBack [Trans[Action]]** : در صورتیکه در هر مرحله پس از صدور Begin TransAction به دلایلی تصمیم به انصراف از عملیات انجام شده مدنظر باشد با استفاده از RollBack TransAction کلیه فرامین ارسال اطلاعات به پایگاه داده پس از دستور Begin TransAction ابطال خواهد شد.

- **دستور Use** : با استفاده از این دستور یک بانک اطلاعاتی جهت انجام عملیات بعدی فعال می گردد و به شکل ذیل مورد استفاده قرار می گیرد.

نام پایگاه داده Use

- مسئله) تابعی بنویسید که سری ذیل را محاسبه کند ($n = 20$)

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots + \frac{x^n}{n!}$$

- مسئله) تابعی بنویسید که مقدار ماکزیمم دو عدد را محاسبه کند.

- مسئله) تابعی بنویسید که یک عدد صحیح را به ساختار hh:mm:ss تبدیل نماید.

فصل دوم

دستورات تبادل اطلاعات

با پایگاه داده

SQL Server

در پایگاه داده SQL Server دستوراتی به منظور اعمال تغییرات در داده ها تعبیه شده است دستوراتی که می توان داده ها را به یک جدول ارسال یا تغییراتی بر رکوردهائی در جداول اعمال نمود از طرفی اکثریت Object های درون پایگاه داده قابلیت تصحیح (Alter) ، حذف (Drop) و ایجاد (Create) را با استفاده از دستورات برنامه نویسی در اختیار قرار می دهند که در این فصل به این موضوع پرداخته خواهد شد اما مهمترین دستور بدون شک در پایگاه داده SQL Server دستور Select به منظور استخراج داده ها از یک یا چند جدول پایگاه داده است که بطور مفصل در این فصل مورد بررسی قرار خواهد گرفت.

- **دسترسی به Object ها** : در پایگاه داده SQL Server مشابه کلیه زبانهای برنامه نویسی OOP به هر Object درون پایگاه داده با ساختار کلی ذیل می توانید دسترسی داشته باشید.

[[DataBaseName.]Owner.]ObjectName[. ...]

بعنوان مثال در نظر بگیرید که در پایگاه داده Master که از پایگاه داده های مهم خود SQL Server است می خواهید به یک فیلد جدول درون پایگاه داده MyDB دسترسی داشته باشید بدین منظور می نویسیم :

MyDB.dbo.MyTBL.MyFld

در صورتیکه از نوشتن نام Owner صرف نظر گردد و پیش فرض کاربر ورودی خواهد بود و در صورتیکه با dbo به پایگاه داده متصل شده باشیم خواهیم داشت :

MyDB..MyTBL.MyFld

- **اعمال تغییرات در Object ها** : کلیه Object های اصلی درون پایگاه داده با استفاده از دستور Create (ایجاد) ، Alter (تصحیح) ، Drop (حذف) تغییرات را به پایگاه داده منعکس می نمایند اما باتوجه به اینکه عموماً در طراحی پایگاه داده به ندرت تغییرات در Object ها از طریق برنامه نویسی به وقوع می پیوندد و عموماً در مواردی اعمال تغییرات از طریق برنامه نویسی برای جدول متصور است بدین منظور جزئیات این دستورات برای جداول مدنظر قرار خواهد گرفت اما دستورات Drop,Alter,Create از طریق کدنویسی برای کلیه Object های اصلی قابل اجرا نمی باشد که جزئیات آن در جدول ذیل ارائه می گردد :

ESP	UDDT			UDF	SP				DB	
Extended Stored Procedure	User Define Data Type	Default	Trigger	User Define Function	Stored Procedure	View	Index	Table	Database	عملکرد
-	-	√	√	√	√	√	√	√	√	Create
-	-	-	√	√	√	√	-	√	√	Alter
-	-	√	√	√	√	√	√	√	√	Drop

بر اساس جدول Object از نوع Index دستور Alter جهت اصلاح ایندکس را شامل نمی شود و برای این منظور ابتدا می بایستی این Object توسط دستور Drop حذف و سپس مجدداً با دستور Create ایجاد گردد از طرفی این شرایط برای مقادیر پیش فرض برای یک فیلد بنام Default نیز صادق است و تنها عملیات Create و Drop برای آن متصور است در خصوص داده طراحی شده خاص توسط برنامه نویس UDDT و ESP ها که قبلاً به آنها اشاره شده است از طریق کدنویسی هیچ عملیاتی با استفاده از سه دستور Create , Alter , Drop قابل انجام نخواهد بود و برای ESP فرآیند ایجاد و حذف این Object از طریق روش های دیگری نظیر استفاده از SP های طراحی شده درون پایگاه داده مقدور خواهند بود.

Create Table - همانطور که قبلاً نیز اشاره شد ایجاد Object های اصلی در پایگاه داده در اکثریت موارد توسط برنامه نویسان توسط Wizard ها صورت می گیرد که محیط Enterprise Manager یک نمونه آن است بر اساس توضیحات ارائه شده در صورتیکه با استفاده از کدنویسی قرار است جدولی به پایگاه داده کنونی اضافه گردد. Create Table به شکل عمومی و خلاصه شده ذیل قابل ارائه می باشد:

(نام جدول Create Table

[Unique یا Primary Key] [Not Null یا Null] فرمول محاسباتی AS یا نوع داده نام ستون
 [مقدار پیش فرض Default] [NonClustered یا Clustered]
 [شرط محدودیت ستون Check]
 [,

 n]
)

- همانطور که ملاحظه می گردد با دستور **Create Table** یک جدول بصورت فیزیکی در پایگاه داده کنونی ایجاد می گردد که نکات ذیل در این خصوص می بایستی مدنظر قرار گیرد.
- در صورتیکه قرار است یک فیلد اطلاعاتی در جدول اشاره شده ایجاد گردد نوع داده می بایستی ارائه گردد ولی اگر بجای نوع داده از فرمول محاسباتی استفاده گردد درون جدول اشاره شده از نظر فیزیکی هیچ ستونی ایجاد نمی گردد.
 - هر ستون ایجاد شده در این دستور می تواند بصورت پیش فرض **Null** باشد.
 - در صورتیکه ستون ایجاد شده یک کلید است می توانید از **Primary Key** استفاده نمائید ولی ممکن است ستون مورد نظر تنها یک مقدار منحصر به فرد باشد که با استفاده از **Unique** قابل تنظیم خواهد بود.
 - در صورتیکه یک **Primary Key** ایجاد گردد یک ایندکس به جدول اشاره شده اضافه می گردد که می تواند یک ایندکس از نوع **Clustered** یا **NonClustered** باشد که بصورت پیش فرض در صورت عدم ارائه **NonClustered** در نظر گرفته خواهد شد.
 - هر ستون ایجاد شده در صورتیکه یک رکورد به جدول اضافه گردد در صورت عدم ارائه مقدار با مقدار تعیین شده توسط **Default** جانشین خواهد شد.
 - با استفاده از کلمه **Check** می توان محدودیت های لازم را برای فیلد تعریف شده در زمان ثبت اطلاعات ایجاد نمود.
 - تعداد ستونها و طول هر رکورد براساس ظرفیت های ارائه شده در بخش ظرفیت پایگاه داده قبلاً اعلام شده است.

مثال: با استفاده از دستور **Create Table** یک جدول برای سیستم پرسنلی ایجاد نمائید.
اگر فرض کنیم نام جدول **Person** باشد در این صورت خواهیم داشت:

```
Create Table Person (
  PID Int Not Null Primary Key Clustered ,
  PName nVarChar(20) Not Null ,
  PFamily nVarChar(30) Not Null ,
  PFather nVarChar(20) ,
  BCity SmallInt ,
  UnitID SmallInt Not Null )
```

در این دستور فیلد **PID** که شماره کارمندی پرسنل می باشد بعنوان **Primary Key** تعریف شده است که براین اساس نمی تواند **Null** باشد.

مثال : با استفاده از دستور Create Table یک جدول پایه واحد سازمانی در سیستم پرسنلی ایجاد نمائید.

در صورتیکه فرض شود که نام جدول مورد نظر Unit باشد بنابراین خواهیم داشت :

```
Create Table Unit (
  UnitID SmallInt Not Null Primary Key Clustered ,
  UnitName nVarChar (100) Not Null
)
```

مثال : یک جدول که دارای اطلاعات سه ضلع یک مثلث قائم الزاویه است با دستور Create Table ایجاد نمائید.

باتوجه به اینکه رابطه $a^2 = b^2 + c^2$ بین اضلاع مثلث برقرار است بنابراین ضلع a یک ستون محاسباتی تلقی می گردد. بنابراین به شکل ذیل عمل خواهیم نمود.

```
Create Table Polygon (
  b Real Not Null ,
  c Real Not Null ,
  a As SQRT (Power (b,2) + Power (c,2) )
)
```

همانطور که مشاهده می گردد چون ستون a تابعی محاسباتی از فیلهای b , c می باشد بنابراین طبق دستور بصورت یک ستون محاسباتی ظاهر شده است بدیهی است که این ستون درون جدول از نظر فیزیکی هیچ فضائی را اشغال نخواهد کرد و در صورت دریافت براساس فرمول ارائه شده تعیین می گردد.

مثال : با استفاده از دستور Create Table جدولی برای نسبت افراد تحت تکفل و افراد تحت تکفل شخص در سیستم پرسنلی طراحی نمائید. این جدول دارای دو فیلد اطلاعاتی کد نسبت و نام نسبت می باشد که کد نسبت بعنوان PK عمل می کند اگر نام جدول را Relate بنامیم آنگاه خواهیم داشت :

```
Create Table Relate (
  RelateID TinyInt Not Null Primary Key Clustered ,
  RelateName nVarChar (15) Not Null )
```

همچنین در صورتیکه نام جدول افراد تحت تکفل شخص PrsChild باشد خواهیم داشت :

```
Create Table PrsChild (
  PID Int Not Null ,
  RelateID TinyInt Default 1 Check (RelateID > = 1),
  CName nVarChar (20) Not Null ,
  CFamily nVarChar (30) Not Null ,
  CFather nVarChar (20) ,
  BCity SmallInt )
```

- **Alter Table** : دستور Alter Table به منظور تغییر در ساختار یک جدول بکار می رود و در اشکال خلاصه شده ذیل بادر نظر گرفتن محدودیت هائی مورد استفاده قرار می گیرد.

- تصحیح یک ستون جدول

نام جدول Alter Table

Alter Column نام ستون [Null یا Not Null] نوع ستون

- ایجاد ستون های جدید در جدول

نام جدول Alter Table

فرمول محاسباتی یا نوع ستون نام ستون ADD
[, ... n]

- حذف ستون هایی از جدول

نام جدول Alter Table

Drop Column [, ...] نام ستون

مثال : برنامه ای بنویسید که به جدول Person یک فیلد جدید بنام کد ملی اضافه کند. در اینصورت از ساختار ADD به شکل ذیل استفاده می کنیم.

```
Begin Tran
Alter Table Person
  ADD NationalCode VarChar (10)
Commit Tran
```

مثال: با استفاده از ساختار Alter فیلد PFather را افزایش طول دهید.
بدین منظور از ساختار Alter Column استفاده می‌نمائیم.

```
Begin Tran
Alter Table Person
    Alter Column PFather nVarChar (25) Not Null
Commit
```

مثال: با استفاده از ساختار Alter فیلد کد ملی را از جدول Person حذف نمائید.

```
Begin Tran
Alter Table Person
    Drop Column NationalCode
Commit TransAction
```

Drop Table - همانطور که قبلاً نیز اشاره شده از دستور Drop به منظور حذف یک Object از پایگاه داده استفاده می‌گردد دستور Drop برای حذف یک جدول از پایگاه داده به شکل کلی ذیل قابل ارائه می‌باشد.

نام جدول Drop Table

مثال: دستوری بنویسید که جدول پرسنلی از فهرست جداول سیستم پرسنلی حذف گردد. بدین منظور می‌نویسیم:

```
Drop Table Person
```

دستور Select - بدون شک دستور Select از مهمترین دستورات در SQL است، این دستور جهت استخراج داده‌ها از یک یا چند جدول تحت شرایطی بکار برده می‌شود شکل کلی دستور Select بصورت خلاصه و کاربردی بصورت ذیل قابل ارائه است:

```
... , ستون [= نام مستعار ستون] یا [ نام مستعار ستون [AS]] ستونها * یا * [Top n] Select
[ Into نام جدول جدید]
[ بخش اتصال به سایر جداول] [ نام مستعار [AS]] نام جدول اصلی From
[ عبارت شرطی Where]
[ ستون‌های قابل گروه بندی Group By]
[ عبارت شرطی در صورت استفاده از گروه بندی Having]
[ نام مستعار ستون یا شماره ستونهایی که بایستی مرتب شوند یا ستونهای قابل مرتب سازی Order By]
```

در این ساختار خیلی کلی اجزاء اشاره شده به شرح ذیل خواهند بود:

- با استفاده از دستور Into خروجی ستون ها به جدول اعلام شده ارسال خواهند شد.
- شرط عبارت است مقدار منطقی که می تواند یکی از مجموعه های ذیل را شامل گردد.

شرط	عملگرهای منطقی	{ and , or , Not
	عملگرهای مقایسه ای	{ = , <> , < , > , <= , >=
	سایر عملگرها	{ عبارت Is Null محدوده بالا and محدوده پائین Between عبارت عبارت In (مجموعه ای از مقادیر) عبارت In (SubQuery یک) استفاده از توابع داخلی نظیر ABS , Left , استفاده از توابع خارجی (UDF) طراحی شده
	عملگرهای ریاضی	{ استفاده از / , * , - , + , %

- که در اینجا "عبارت" می تواند یک SubQuery، فیلد، متغیر و یا ترکیبی از آنها باشد.
- ستون عبارت است یک مقدار استخراج شده تحت شرایط ذیل:

ستون	نام یک فیلد از جدول
	یک SubQuery
	یک مقدار محاسباتی
	یک تابع داخلی یا خارجی
	ترکیبی از همه موارد با رعایت نوع داده ها

و در صورتیکه از * استفاده گردد کلیه ستونهای جدول یا جداول درخواستی ارائه می گردد و در صورتیکه نام ستونها تعیین گردد تنها ستونهای ذکر شده ارائه خواهند شد و با استفاده از AS یا = می توان برای ستون بازگشتی یک نام مستعار نیز برای برنامه دریافت کننده ارسال نمود در غیر اینصورت نام فیلد یا یک نام خاص بصورت سریالی توسط SQL Server ارائه خواهد شد. از طرفی Top n تعداد n رکورد اولیه استخراج شده را تنها به درخواست کننده ارسال خواهد کرد که در مواردی بسیار مفید است.

- ستونهای قابل مرتب سازی می تواند به شکل ذیل ارائه گردد :

ستونهای مرتب شده	{	نام یک فیلد از
		یک SubQuery [ASC یا Desc] , ...
		شماره ستون
		ترکیبی از همه موارد

که کلمات ASC , Desc مرتب سازی صعودی و نزولی را برای ستون اشاره شده را تعیین خواهند کرد و در صورت عدم اعلام مقدار ASC یعنی صعودی در نظر گرفته خواهد شد.

نام جدول در واقع جدول اصلی که می بایستی داده از آن استخراج گردد را شامل می شود و در صورتیکه از نام مستعار استفاده گردد در کلیه بخش های اشاره شده نام مستعار جانشین نام جدول واقعی خواهد شد.

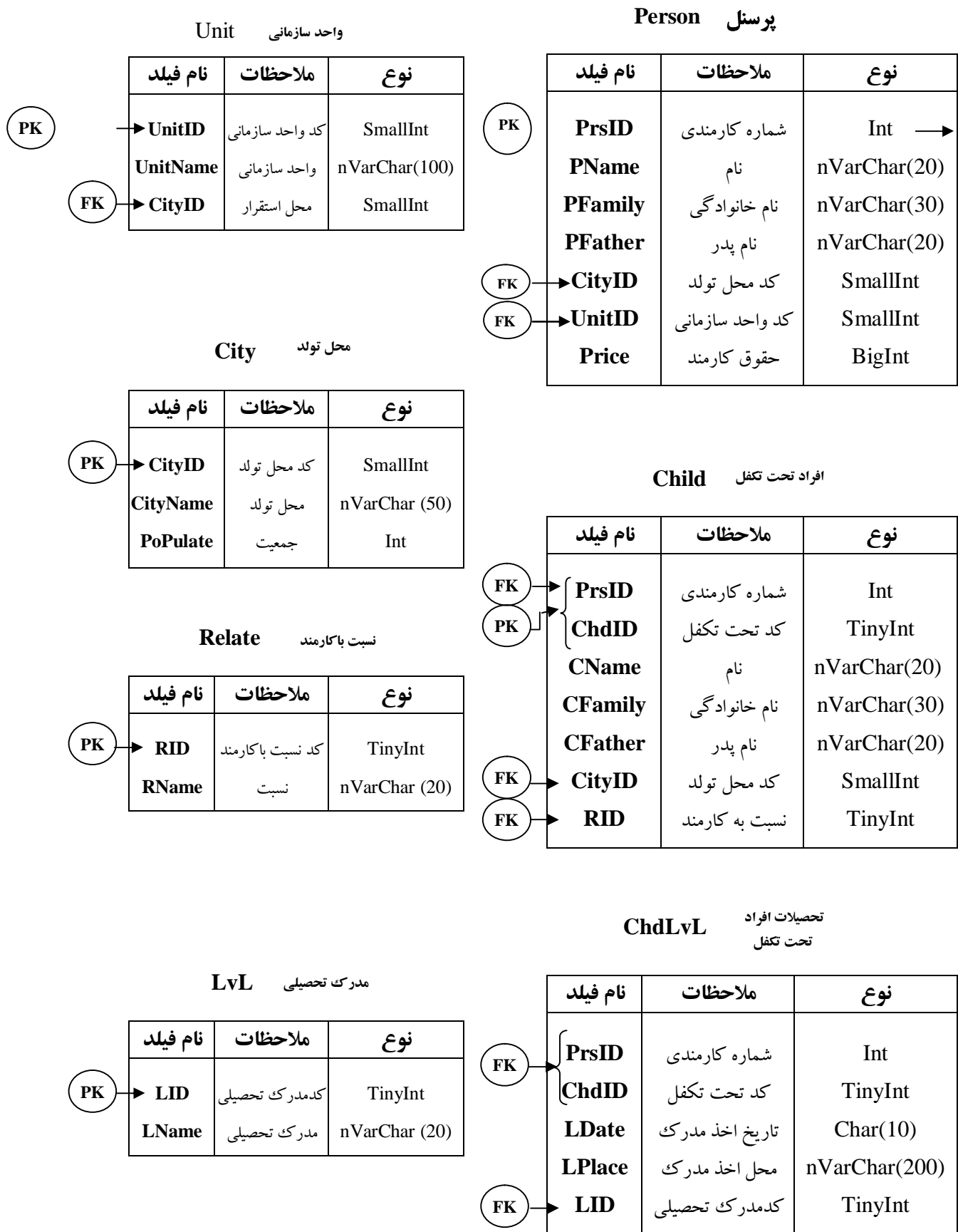
- بخش اتصال به سایر جداول به منظور تعیین روابط بین جداول لازم برای استخراج توسط دستور Select تعبیه شده است که دارای ساختار ذیل می باشد :

بخش اتصال به سایر جداول	{	[Inner] Join	(شرط اتصال به) On [نام مستعار [AS]] نام جدول جدول اصلی جدول ارتباطی
		یا	
		Left Join	
		یا	
		Right Join	
		[
		⋮	
		⋮	
		سایر جداول	
]	

- Inner Join زمانی بکار می رود که سطرهای معادل در دو جدول می بایستی وجود داشته باشد.
- Left Join زمانی بکار می رود که جدول سمت چپ یا اصلی ملاک انتخاب رکورد باشد.
- Right Join زمانی بکار می رود که جدول سمت راست یا متصل شده ملاک انتخاب رکورد باشد.

مقدار شرط اتصال عبارت منطقی است که رابطه بین جدول اصلی و جدول Join شده را تعیین می کند که عموماً از طریق PK و FK این ارتباط تعیین می گردد در غیر اینصورت به منظور افزایش سرعت شایسته است که ایندکس های مناسب برای ارتباط بین جداول در حجم داده های زیاد پیش بینی شده باشد. از طرفی در صورتیکه قرار است گروه بندی روی جدول اصلی صورت پذیرد می بایستی از دستور **Group By** استفاده گردد و شرط گروه بندی نیز در جلوی دستور **Having** اعلام می گردد. همانطور که مشاهده می کنید با تک دستور **Select** و مفاهیم اشاره شده یک برنامه نویس با تجربه می بایستی ستونهای مورد نیاز را در یک **Query** استخراج نماید که این موضوع با درک مفاهیم درست پایگاه داده و ممارست بیشتر محقق خواهد شد بدین منظور برای درک بیشتر به مثال های ذیل که براساس سیستم پرسنلی اشاره شده در فصل اول می باشد دقت فرمائید.

مثال: ارتباط شماتیک بین جداول سیستم پرسنلی اشاره شده در فصل اول را ترسیم نمایید.
ارتباط بین 7 جدول اشاره شده در فصل اول به شرح ذیل خواهد بود:



مثال : براساس جدول ارتباطی بین سیستم پرسنلی دستوری بنویسید که ستونهای ذیل را استخراج نماید.

واحد سازمانی ، محل تولد ، نام پدر ، نام خانوادگی ، نام ، شماره کارمندی

حل : براساس ستونهای اعلام شده جدول اصلی Person خواهد بود و جداول City و Unit جداول ارتباطی خواهند بود بدین منظور به دو روش این امر محقق خواهد شد.

روش اول : ستونهای واحد سازمانی و محل تولد بعنوان SubQuery در نظر گرفته خواهند شد بنابراین دستور Select موردنظر به شکل ذیل خواهد بود :

```
Select PrsID, PName, PFamily, PFather,
      (Select Top 1 CityName From City Where City.CityID=Prs.CityID)As CityN,
      (Select UnitName From Unit Where UnitID=Prs.UnitID) As UnitN
From Person As Prs
Order By PrsID
```

روش دوم : از دستور Join بدین منظور استفاده خواهد شد بنابراین خواهیم داشت :

```
Select Prs.PrsID, Prs.PName, Prs.PFamily, Prs.PFather,
      City.CityName As CityN, UnitN = Unit.UnitName
From Person As Prs
Left Join City As City On (City.CityID = Prs.CityID)
Left Join Unit On (Unit.UnitID = Prs.UnitID)
Order By 1
```

همانطور که ملاحظه می گردد در روش اول از یک SubQuery برای ارتباط با جدول City استفاده شده است و نام محل تولد تحت عنوان CityN باز خواهد گشت و جدول اصلی Person تحت نام مستعار Prs استفاده می گردد و همچنین از SubQuery نیز جهت استخراج ستون نام واحد سازمانی تحت UnitN استفاده گردیده است و در نهایت جدول بازگشتی روی فیلد PrsID مرتب شده ارائه خواهد شد اما در روش دوم چون از دو Join استفاده گردیده است در ابتدای هر فیلد استخراج شده نام جدولی که فیلد متعلق به آن است ذکر گردیده و ارتباط بین PK , FK جداول ارتباطی در مقابل شرط Join یا مقابل دستور On ارائه شده است و در نهایت اطلاعات بازگشتی براساس ستون اول بازگشتی یعنی PrsID مرتب شده است.

مثال : یک SP بنویسید که افراد تحت تکفل هر شخص را ارائه نماید.

حل : مشابه مثال قبل به دو روش این موضوع قابل انجام است اما جدول اصلی در این حالت Child می باشد.

روش اول : در این روش از SubQuery جهت ارتباط با جدول City و Relate استفاده خواهد شد.

```

Create Proc GetChild1
@ PrsID Int
AS
Select CName, CFamily, CFather
      (Select CityName From City Where CityID=Child.CityID)As CityName,
      (Select RName From Relate Where RID=Child.RID)AS RName
From Child
Where PrsID = @PrsID
Order By RName

```

روش دوم : در این روش از Join به دو جدول City و Relate استفاده می گردد.

```

Create Proc GetChild2
@PID Int
As
Select C.CName , C.CFamily , C.CFather,
      City. CityName , R.RName
From Child As C
Left Join City On (City.CityID = C.CityID)
Left Join Relate As R On (R.RID = C.RID)
Where C.PrsID = @ PID
Order By R.RName

```

ممکن است از ترکیب SubQuery , Join , بدین منظور استفاده گردد که در این شرایط SP فوق می تواند به شکل ذیل نیز طراحی گردد.

```

Create Proc GetChild3
@P Int
As
Select C.CName , C.CFamily , C.CFather , City.CityName,
      (Select Top 1 RName From Relate Where RID = C.RID ) As RName
Form Child As C
Left Join City On(City.CityID= C.CityID)
Where C.PrsID=@P
Order By (Select RName From Relate Where RID = C.RID ) Asc

```

که البته مناسب است که از ستون 5 یا RName بجای نوشتن SubQuery در مقابل دستور Order By استفاده گردد که باعث افزایش سرعت نسبی خواهد شد به خصوص در مواقعی که شرایط مناسبی جهت استخراج در SubQuery مهیا نباشد.

- **توابع جمعی** : این توابع عملیات روی مجموعه ای از رکوردها را بعهده دارند و عموماً در دستورات گروه بندی (Group By) ظاهر می گردند که اهم این توابع به شرح ذیل می باشند :

Min	مقدار می نیمم یک عبارت را محاسبه می کند
Max	مقدار ماکزیمم یک عبارت را محاسبه می کند
Avg	مقدار میانگین یک عبارت را محاسبه می کند
Sum	مقدار جمع یک عبارت را محاسبه می کند
Count	تعداد رکوردهای اشاره شده را شمارش می کند

اما این توابع در SubQuery ها می توانند بعنوان یک ستون اطلاعاتی منعکس شوند در این خصوص به مثال های ذیل توجه فرمائید.

مثال : بااستفاده از دستور Select ستونهای ذیل را به ترتیب فامیلی برای افرادی که بیش از یک تحت تکفل دارند استخراج نمائید.

تعداد فرزند , تعداد تحت تکفل , نام , نام خانوادگی , شماره کارمندی
(RID = 2 , 3)

روش اول : در این حالت جدول اصلی Person و جدول ارتباطی Child خواهد بود بنابراین خواهیم داشت :

```
Select PrsID , PFamily , PName,
      (Select Count (*) From Child Where PrsID=P.PrsID) As CT ,
      (Select Count (*) From Child Where PrsID = P.PrsID
      and RID In ( 2 , 3 ) ) As CC
From Person P
Where (Select Count (*) From Child Where PrsID = P.PrsID ) > 1
Order By 2, PName
```

روش دوم : در این حالت جدول اصلی Child و جدول ارتباطی Person خواهد بود بنابراین خواهیم داشت :

```
Select C1.PrsID,
      (Select PFamily From Person Where PrsID = C1.PrsID) As PFamily,
      (Select PName From Person Where PrsID = C1.PrsID) As PName,
      Count (*) As CT,
      (Select Count(*) From Child C2 Where PrsID=C1.PrsID and RID In(2,3)) As CC
From Child C1
Group By C1.PrsID
Having Count (*) > 1
Order By PFamily , 3
```

همانطور که ملاحظه می گردد در روش دوم شرط استخراج رکوردهای گروه بندی در جلوی دستور Having ظاهر شده است و از طرفی چون ستونهای 2 , 3 استخراج شده یک SubQuery می باشند بنابراین دستور Order By با ظاهر متفاوت جهت افزایش سرعت پیاده سازی شده است.

مثال : یک SP جهت استفاده در خروجی مدیریتی ذیل طراحی نمائید.

تعداد تحت تکفل	تعداد کارکنان	واحد سازمانی
—	—	—
—	—	—
—	—	—
—	—	—

همانطور که از ظاهر جدول مشاهده می گردد به دو روش گزارش مذکور قابل وصول است.

روش اول : جدول اصلی جدول Unit و جداول ارتباطی Child , Person خواهند بود بنابراین خواهیم داشت :

```
Create Proc MIS1 AS
Select UnitName ,
      (Select Count (*) From Person Where UnitID = U.UnitID )As PCnt ,
      (Select Count (*) From Child Where
      (Select UnitID From Person Where PrsID=Child.PrsID)=U.UnitID)As TCnt,
From Unit As U
Order By UnitName
```

روش دوم : جدول اصلی Person و جدول ارتباطی Child ,Unit خواهند بود بنابراین خواهیم داشت:

```

Create Proc MIS2 AS
Select (Select UnitName Form Unit Where UnitID = P1.UnitID)As UnitName,
      Count (*) As PCnt,
      (Select Count(*) From Child Where (Select UnitID From Person P2 Where PrsID
      = Child.PrsID ) = P1.UnitID ) As TCnt
From Person P1
Group By UnitID
Order By 1

```

مسئله : اختلاف اطلاعات نمایش داده شده در دو روش اول و دوم را تشریح نمایید.

مثال : یک SP طراحی کنید که فهرست کارکنانی را که بین 3 تا 6 تحت تکفل دارند را استخراج نماید.

به منظور سهولت در طراحی جدول Person را بعنوان جدول اصلی در نظر می گیریم بنابراین خواهیم داشت :

```

Create Proc MySP1 AS
Select P.*,
      (Select Count(*) From Child C Where PrsID = P.PrsID )As Cnt
From Person As P
Where (Select Count (*) From Child Where PrsID= P.PrsID) BetWeen 3 and 6
Order By PrsID

```

مثال : تابعی برای محاسبه تعداد افراد تحت تکفل یک شخص طراحی کنید و سپس با استفاده از آن یک SP جهت تعیین فهرست کارکنانی که بیش از n تحت تکفل دارند بنویسید :

ابتدا تابعی بنام NoChild که کد پرسنلی ورودی آن است به شکل ذیل طراحی می کنیم.

```

Create Function NoChild (@P Int )
Returns TinyInt
As
Begin
Declare @Cnt TinyInt
Set @Cnt = (Select Count(*) From Child Where PrsID = @P )
Return @Cnt
END

```


البته در این تابع بدون تعریف متغیر Cnt می توانید دستور SubQuery را مستقیماً توسط Return باز گردانید پس از طراحی تابع NoChild می توانید SP مورد نظر را به شکل ذیل با پارامترها ورودی n طراحی نمائید.

```
Create Proc MySP2 @n TinyInt
As
Select * , dbo.NoChild (PrsID)
From Person
Where dbo.NoChild (PrsID) > = @n
Order By PrsID
```

مثال : با استفاده از تابع فهرست کارکنان را براساس بالاترین مدرک تحصیلی افراد تحت تکفل آنها مرتب کنید.

ابتدا تابعی طراحی می کنیم که بزرگترین مدرک تحصیلی افراد تحت تکفل یک شخص را استخراج نماید که در این حالت فرض شده است که با افزایش مدرک تحصیلی میزان کد مدرک تحصیلی نیز افزایش خواهد یافت بنابراین خواهیم داشت:

```
Create Function dbo.MaxCL (@PID Int)
Returns TinyInt
As
Begin
Return (Select Max(LID) From ChdLvL
Where PrsID = @PID )
END
```

حال دستور SQL درخواست شده را می نویسیم بنابراین :

```
Select * , dbo.MaxCL(PrsID) AS CMCL ,
(Select LName From LvL Where LID=dbo.MaxCL(P.PrSID)) AS NMCL
From Person AS P
Order By CMCL Desc , PrsID
```

مثال : با استفاده از تابع تعیین کنید که هر واحد سازمانی در چه نسبت جمعیت شهری واقع شده است. ابتدا تابعی می نویسیم که تعیین کند که هر واحد سازمانی در چه جمعیتی واقع شده است بدین منظور به شکل ذیل عمل می کنیم:

```
Create Function SumPoP (@CID SmallInt)
Returns Int AS Begin
Return (Select PoPulate From City Where CityID = @CID)
END
```

براساس این تابع می توان دستور SQL موردنظر را به شکل ذیل پیاده سازی کرد.

```
Declare @S BigInt
Set @S = ( Select Sum( PoPulate ) From City )
Select * , dbo. Sumpop (CityID) / @S * 100 AS SPP
From Unit
Order By SPP Desc
```

- **دستور Insert** : به منظور ایجاد یک رکورد جدید در یک جدول از دستور Insert به شکل کلی ذیل استفاده می گردد.

```
Insert [ Into ] نام جدول
      [ ( فیلد 1 , فیلد 2 , ... ) ]
Values
      ( مقدار 1 , مقدار 2 , ... )
```

در صورتیکه اسامی فیلدها در دستور مذکور ارائه نگردد کلیه ستونهای جدول اشاره شده مدنظر قرار می گیرد و درخصوص مقادیر به ازای هر فیلد اعلام شده می بایستی مقدار متناسب با نوع آن ارائه گردد در این حالت می توان از کلمات Null و Default به منظور درج در ستون موردنظر استفاده کرد که Null مقدار تهی را به فیلد موردنظر ارسال نموده و مقدار Default از مقدار پیش فرض تعریف شده برای فیلد استفاده خواهد کرد.

مثال : یک رکورد جدید به جدول پرسنلی اضافه نمائید.

```
Insert Person
Values ( 200 , محمد , علیزاده , علی , 200 )
```

که در این حالت چون فیلدها در دستور Insert اعلام نشده اند بنابراین هر 6 فیلد جدول Person به ترتیب مدنظر قرار گرفته است.

مثال : یک رکورد جدید به جدول پرسنلی با اشاره به نام فیلدها اضافه نمائید.

```
Insert Into Person (PrsID , UnitID , CityID, PFather , PName , PFamily )
Values ( 201, 2 , 3 , حسین , رضا , رضائی )
```

- دستور Update : با استفاده از دستور Update رکوردهای اشاره شده تغییر خواهند کرد و در شکل کلی ذیل مورد استفاده قرار می گیرد.

Update Set [نام مستعار [AS] نام جدول

, مقدار 1 = فیلد 1

, مقدار 2 = فیلد 2

[Where عبارت شرطی]

در این حالت هر فیلد با مقدار تعیین شده که بانوع فیلد تناسب دارد جانشین خواهد شد و این عملیات تا زمانی که شرط صادق باشد ادامه می یابد. بدیهی است که در صورت عدم ارائه شرط کلیه رکوردهای جدول تحت تاثیر تغییرات واقع خواهند شد که شایسته است دقت بیشتری بعمل آید در واقع در مواقعی که قرار است یک رکورد خاص تحت تغییرات قرار گیرد فیلد PK در جدول مورد نظر بهترین کاندیدا جهت عبارت شرطی است.

مثال : با استفاده از دستور Update مقدار اطلاعات پرسنلی شخص با شماره کارمندی 200 را تغییر دهید.

```
Update Person Set
    PFamily = 'رضائی پور' ,
    UnitID = 1
Where PrsID = 200
```

که در این مثال ستونهای نام خانوادگی و واحد سازمانی تحت تغییرات برای شخص به کارمندی 200 قرار گرفته اند.

مثال : با استفاده از دستور Update حقوق کارکنان را 20% افزایش دهید.
اگر فرض کنیم در جدول پرسنلی فیلد Price حقوق کارمندان باشد بنابراین خواهیم داشت:

```
Update Person Set
Price = Convert ( BigInt , Price * 1.2 )
```

مثال : فرض کنید که کد تهران = 1 و کد اصفهان = 2 باشد دستوری بنویسید که حقوق کارکنان دو شهر را 25% افزایش دهد:

```
Update Person Set Price = Convert ( BigInt , Price * 1.25 )
Where CityID In (1,2) ( 39 )
```

مثال : دستوری بنویسید که حقوق کارکنان هر واحد را به نسبت درصد جمعیت هر شهر تغییر دهد.

```

Update Person P Set Price = Price * ( 1 +
  ( Select Isnull (C.PoPulate ,0) Form Unit U
    Left Join City C On (C.CityID = U.CityID)
    Where U.UnitID = P.UnitID)
  /
  ( Select Sum (PoPulate ) From City ) )

```

همانطور که مشاهده می گردد ابتدا برای هر شخصی به واحد سازمانی و از آنجا به شهری که آن واحد قرار دارد ارجاع داده می شد تا جمعیت آن شهر تعیین گردد و در مخرج کسر نیز جمعیت کل شهرستانها محاسبه می گردد در واقع فرمول ذیل مورد محاسبه قرار گرفته است:

$$\left(\text{جمعیت شهری که شخص در آن مشغول بکار است} + 1 \right) * \text{حقوق قبلی} = \text{حقوق جدید}$$

جمعیت کل شهرها

از طرفی مشاهده می گردد که مخرج کسر یعنی جمعیت کل شهرستانها برای هر فرد در حال محاسبه است که فرآیند به روزرسانی را کندتر خواهد کرد بدین منظور اگر از یک SP استفاده شود می توان افزایش سرعت را ایجاد نمود بنابراین خواهیم داشت:

```

Create Proc NewPrice AS
Declare @SumPoP BigInt
Set @SumPoP = ( Select Sum (PoPulate ) From City )
Begin Tran
Update Person P Set Price = Price * ( 1 +
  ( Select Isnull (C.PoPulate , 0 ) From Unit U
    Left Join City C On ( C.CityID = U.CityID)
    Where U.UnitID = P.UnitID ) / @ SumPoP )
Commit Tran

```

- **دستور Delete :** حذف رکوردهای یک جدول با استفاده از دستور Delete صادر می گردد که در قالب کلی ذیل مورد استفاده قرار می گیرد:

Delete [From] نام جدول [[AS] نام مستعار] [Where عبارتی شرطی]

و در صورتیکه از عبارت شرطی و دستور Where صرف نظر گردد حذف کلیه رکوردهای جدول را به همراه خواهد داشت.

مثال : در جدول تحصیلات افراد تحت تکفل دستوری بنویسید که رکوردهای کارکنانی که شماره کارمندی زوج دارند حذف شوند.

Delete ChadLvL C Where C.PrsID %2 = 0

مثال : جدول تحصیلات افراد تحت تکفل در سیستم پرسنلی را خالی از رکورد نمائید.

Delete ChdLvL

فصل سوم

مباحث تکمیلی در

SQL Server

در این فصل هدف مباحث تکمیلی و تکنیکی بیشتر مدنظر قرار خواهد گرفت. طراحی ایندکس ها با استفاده از تکنولوژی Rushmore و طراحی View و ESP و Trigger ها از اهم موضوعات این فصل می باشد اما قبل از هرچیز مواردی خاص که مباحث مطرح شده را تکمیل می کند در ابتدا مورد بررسی قرار می گیرد.

دستور Declare :

هنگام استفاده از دستور Declare به منظور ایجاد یک متغیر جدید در صورت لزوم اندازه داده می بایستی تعیین گردد اما در صورتیکه در خصوص اینگونه موارد طول داده اعلام نگردد پیش فرض از طرف SQL Server جانشین خواهد شد که می بایستی دقت گردد بدین منظور به جدول ذیل دقت نمائید.

عملکرد	طول پیش فرض		نوع داده
	باینری	رشته ای	
در دستورات تبدیل Cast و Convert پیش فرض طول رشته ای 30 است	1	1	Char
در دستورات تبدیل Cast و Convert پیش فرض طول رشته ای 30 است	2	1	NChar
در دستورات تبدیل Cast و Convert پیش فرض طول رشته ای 30 است	1	1	VarChar
در دستورات تبدیل Cast و Convert پیش فرض طول رشته ای 30 است	2	1	nVarChar
در دستورات تبدیل Cast و Convert پیش فرض طول 30 است	1	1	binary
در دستورات تبدیل Cast و Convert پیش فرض طول 30 است	1	1	Varbinary
طول حداکثر 38 می باشد	به اندازه ذخیره شده	18,0	Numeric
طول حداکثر 38 می باشد	به اندازه ذخیره شده	18,0	Decimal
طول حداکثر 38 می باشد	به اندازه ذخیره شده	18,0	Dec

مثال) در قطعه برنامه مقابل اگر بجای `Char` , `nChar` , `VarChar` , `nVarChar` از انواع داده هر یک از انواع داده باشد خروجی چقدر خواهد بود؟

```
Declare @x ? (15)
Set @x = 'Hello'
Select Len (@x) , DataLength (@x)
```

پس از اجرای خروجی برای چهار حالت ذکر شده پاسخ به شکل جدول ذیل خواهد بود.

نام داده	مقدار Len	تعداد Data Length
Char	5	15
nChar	5	30
VarChar	5	5
nVarChar	5	10

مثال) در قطعه برنامه مقابل مقدار خروجی چقدر است؟

```
Select Len ( Convert (nChar , 300)) ,
DataLength (Cast (300 As nChar))
```

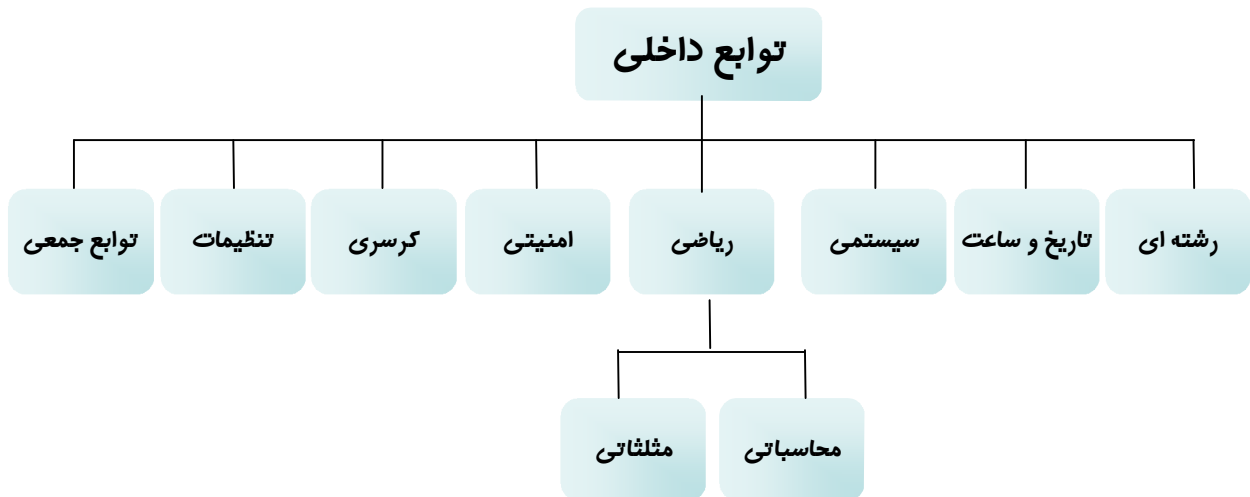
پس از اجرای برنامه به ترتیب خروجی 3,60 خواهد بود.

مثال) برنامه ای بنویسید که جمع طول رشته ای و باینری یک متغیر `nVarChar` را بدست دهد.

```
Declare @x nVarChar (4000) , @L1 SmallInt , @L2 SmallInt
Set @x = 'Hello'
Select @L1 = Len (@x) , @L2 = DataLength (@x)
Select @L1 + @L2 AS L
```


طبقه بندی توابع داخلی :

در فصول گذشته تعدادی تابع داخلی مفید جهت استفاده معرفی گردید تعداد توابع داخلی در هر زبان برنامه نویسی ابزارهای مفیدی جهت توسعه نرم افزار به شمار می روند ولی تعداد آنها در مواردی بسیار زیاد است که فراگیری آنها کمی مشکل به نظر می رسد بدین منظور تعدادی از توابع داخلی مهم در SQL Server جهت استفاده های بعدی طبقه بندی شده است.



مطابق شکل هر یک از توابع داخلی در SQL Server می تواند در یکی از گروههای 9 گانه قرار گیرد بدین منظور اهم توابع داخلی در جداول ذیل به اختصار درج گردیده است :

توابع رشته ای			
نام تابع	عملکرد تابع	مثال	نتیجه
Ascii	کداسکی یک کارکتر را باز می گرداند	Select Ascii ('A')	65
Char	کارکتر معادل یک کداسکی را باز می گرداند	Select Char (65)	'A'
CharIndex	موقعیت قرار گرفتن یک زیر رشته را در یک رشته اعلام می دارد	Select CharIndex ('o', 'Hello')	5
Left	تعدادی کارکتر از سمت چپ یک رشته را استخراج می کند	Select Left ('Hello', 2)	'He'
Len	طول یک رشته را باز می گرداند	Select Len ('Hello')	5

توابع رشته ای			
نتیجه	مثال	عملکرد تابع	نام تابع
'hello'	Select Lower ('Hello')	یک مجموعه رشته ای را به حروف کوچک تبدیل می کند	Lower
'Hello'	Select LTrim (' Hello')	فضای خالی یک رشته را از سمت چپ حذف می کند	LTrim
'A'	Select NChar (65)	کارکتر مربوط به یک عدد را در unicode استاندارد باز می گرداند و عددی بین 0 تا 65535 می باشد	NChar
'xbc'	Select Replace ('abc','a','x')	یک رشته را کارکتر به کارکتر ترجمه می کند	Replace
'okok'	Select Replicate ('ok',2)	یک رشته را به تعداد اعلام شده تکرار می کند	Replicate
'olleH'	Select Reverse ('Hello')	یک رشته کارکتری را معکوس می کند	Reverse
'lo'	Select Right ('Hello',2)	تعدادی کارکتر را از سمت راست یک رشته استخراج می کند	Right
'Hello'	Select RTrim ('Hello ')	فضای خالی یک رشته را از سمت راست حذف می کند	RTrim
' '	Select Space (5)	به تعداد اعلام شده کارکتر خالی ایجاد می نماید	Space
10	Select Len (Str (500.5))	یک مقدار عددی را به یک رشته با طول ثابت و نقطه شناور ثابت تبدیل می کند و پیش فرض طول 10 بایت و پیش فرض اعشار صفر است	Str
'500.3'	Select Str (500.3,5,2)		
'500.319'	Select Str (500.3188,7,3)		
'Hokllo'	Select Stuff ('Hello',2,1,'ok')	تعدادی کارکتر را در درون یک رشته حذف و جانشین می کند	Stuff
'll'	Select Substring ('Hello',3,2)	تعدادی کارکتر را از یک رشته استخراج می کند	Substring

-	Select Unicode (@Mystr)	یک عدد صحیح که نشانه مقدار Unicode یک رشته است را باز می گرداند	Unicode
'HELLO'	Select Upper ('Hello')	یک مجموعه رشته ای را به حروف بزرگ تبدیل می کند	Upper

توابع محاسباتی			
نتیجه	مثال	عملکرد تابع	نام تابع
1.7	Select ABS (-1.7)	معادل قدر مطلق یک عدد در ریاضی است	ABS
124	Select Ceiling (123.45)	یک عدد صحیح بزرگتر از عدد ورودی را ارائه می دهد	Ceiling
1	Select Exp (0)	معادل e^x در ریاضی است	Exp
-3	Select Floor (-2.7)	معادل جزء صحیح x یعنی $[x]$ در ریاضی می باشد	Floor
1	Select Log (Exp (1))	مقدار لگاریتم در مبنای عدد e (نپر) را ارائه می دهد	Log
2	Select Log10 (100)	مقدار لگاریتم در مبنای 10 را ارائه می دهد	Log10
100	Select Power (10,2)	یک عدد را به توان عدد دیگر می رساند که معادل a^b می باشد	Power
-	Select Rand ()	یک عدد تصادفی بین 0 تا 1 تولید می کند	Rand
123.4600	Select Round (123.4567,2)	یک عدد را تحت شرایطی گرد می کند	Round
-1	Select Sign (-3.7)	علامت یک عدد را باز می گرداند و شامل اعداد 0, -1, +1 خواهد بود	Sign
25.0	Select Square (-5)	توان دوم یک عدد را محاسبه می کند	Square
1.2	Select SQrt (1.44)	جذر یک عدد مثبت را باز می گرداند	SQrt

توابع مثلثاتی			
نتیجه	مثال	عملکرد تابع	نام تابع
0.0	Select ACos (1)	آرک کسینوس یک عدد را به رادیان محاسبه می کند	ACos
0.0	Select ASin (0)	آرک سینوس یک عدد را به رادیان محاسبه می کند	ASin
0.785398	Select ATtan (1)	آرک تانژانت یک عدد را به رادیان محاسبه می کند	ATAN
1.0	Select Cos (0)	کسینوس یک عدد را محاسبه می کند	Cos
0.64209	Select Cot (1)	کوتانژانت یک عدد را محاسبه می کند	Cot
90	Select Degrees (PI() /2)	یک مقدار زاویه را از رادیان به درجه تبدیل می کند	Degrees
3.141592	Select PI ()	مقدار عدد پی را باز می گرداند	PI
3.141592	Select Radians (180.0)	یک زاویه به درجه را به رادیان تبدیل می کند	Radians
1.0	Select Sin (Radians (90.0))	سینوس یک عدد را محاسبه می کند	Sin
0.999999	Select Tan (PI () /4)	تانژانت یک عدد را محاسبه می کند	Tan

توابع سیستمی و تبدیل داده ای و امنیتی			
نتیجه	مثال	عملکرد تابع	نام تابع
		نام برنامه اجراکننده را باز می گرداند که بسیار برای موارد امنیتی مفید است	APP_Name ()
		یک نوع داده ای را به نوع دیگری تبدیل می کند	Cast
		مشابه تابع Cast یک نوع داده ای را به نوع دیگر داده ای تبدیل می کند	Convert

توابع سیستمی و تبدیل داده ای و امنیتی			
نتیجه	مثال	عملکرد تابع	نام تابع
		نام Owner کنونی را باز می گرداند	Current_User
4	Select DataLength (100)	طول داده ای برای یک نوع داده را باز می گرداند	DataLength
		در صورتیکه در فرآیند اجرای یک فرمان خطایی ایجاد گردد شماره خطا را باز می گرداند	@@Error
		شماره شناسائی مربوط به یک کامپیوتر متصل شده را باز می گرداند	Host_ID ()
		نام کامپیوتر متصل شده به پایگاه داده را باز می گرداند	Host_Name ()
		در صورتیکه مقدار ورودی Null باشد از مقدار پیش فرض استفاده خواهد کرد	IsNull
1	Select IsNumeric ('12')	در صورتیکه مقدار رشته محتوای عددی داشته باشد مقدار 1 باز می گردد	IsNumeric
-	Select NewID ()	یک مقدار منحصر به فرد از نوع داده ای UniquIdentifier را تولید می کند	NewID
		تعداد رکوردهای تحت تاثیر دستوراتی نظیر Select یا Update و ... را باز می گرداند	@@RowCount
		مشابه تابع @@RowCount آخرین رکوردهای تحت تاثیر را باز می گرداند	RowCount_Big
	Select System_User	در صورتیکه دسترسی به SQL Server و ویندوز معادل باشند در این صورت نام کاربر ویندوز و در غیر این صورت نام کاربر متصل به SQL Server خواهد بود	System_User
	Select User_Name (1)	بدون پارامتر معادل تابع Current_User خواهد بود و در صورتیکه در عدد ورودی شماره User اعلام گردد نام کاربر اشاره شده را باز می گرداند	User_Name
		با این تابع می توان یک عدد از محتوای فیلدهای خاص در یک رکورد را از نظر امنیتی کنترل کرد	Checksum

مثال) در سیستم پرسنلی طراحی شده بر روی اطلاعات افراد تحت تکفل مقدار Check Sum را محاسبه کنید.

Select * , CheckSum (*) From Child

طراحی View :

View ها در واقع جداول مجازی هستند که محدودیت در سطرها و ستون های یک یا چند جدول رابطه ای یا ستون های محاسباتی را شامل می شوند، در طراحی View استفاده از Order By (بدون Top), Into در دستور Select بازگشتی مجاز نبوده و هنگام استفاده از جدول مجازی و با استفاده از دستور * Select کلیه ستونهای اعلام شده در View در اختیار درخواست کننده قرار خواهد گرفت و درخصوص قابلیت دسترسی به سطرها و ستون های View می بایستی موارد دسترسی به جدول مجازی از طریق روش های امنیتی ایجاد گردد ، در نگاه اول شاید بنظر رسد که View ها شباهت به SP ها دارند ولی با دقت بیشتر سطوح امنیتی و محدودیت های ایجاد شده و قابلیت تعیین مشروط و مرتب سازی اختصاصی و مواردی از این دست که توسط دستور Select قابل وصول است در محدوده تعیین شده بسیار انعطاف فوق العاده ای را ایجاد خواهد کرد که در جای خود بسیار حائز اهمیت است با این اوصاف شکل کلی طراحی یک View به شکل ذیل خواهد بود.

[(نام ستون ها)] نام ویو Create View

[With SchemaBinding]

As

Select دستور

همانطور که از شکل ظاهری دستور مشاهده می گردد نتیجه جدول بازگشتی در دستور Select بعنوان جدول مجازی توسط کاربر مورد استفاده قرار خواهد گرفت در این دستور در صورتیکه ستون بازگشتی در دستور Select یک فیلد محاسباتی یا SubQuery باشد می بایستی حتماً نامی برای آن ستون در نظر گرفته شود. درضمن هیچ دستور دیگری جز دستور Select در این ساختار قابل استفاده نمی باشد که این موضوع قدرت SP را در برنامه نویسی نسبت به View نشان می دهد.

(مثال) برای سیستم پرسنلی یک View طراحی کنید که ستون های ذیل را شامل گردد.

[تعداد تحت تکفل] , [محل تولد] , [نام] , [نام خانوادگی] , [شماره کارمندی]

همانطور که مشاهده می گردد ستون های تعداد تحت تکفل و محل تولد می تواند از طریق Join با جداول مربوطه به شکل ذیل حاصل گردد.

Create View V1 As

Select No=PrsID,PFamily,PName,C.CityName As CN ,

(Select Count(*) From Child Where PrsID=P.PrSID) As Cno

From Person P

Left Join City C On(C.CityID=P.CityID)

مثال) براساس View طراحی شده بنام V1 دستوری بنویسید که ستون های ذیل را به ترتیب نام و نام خانوادگی برای تحت تکفل بزرگتر از 2 ارائه دهد.

[تعداد تحت تکفل] , [نام خانوادگی] , [نام] , [شماره کارمندی]

در این حالت خواهیم داشت :

```
Select No,PName , PFamily , Cno
From V1
Where Cno > 2
Order By PFamily , PName
```

مثال) در سیستم پرسنلی یک View طراحی کنید که براساس گروه بندی جدول ذیل را برای تعداد کارکنان بیشتر از 10 نفر ارائه دهد.

[تعداد کارکنان] , [واحد سازمانی]

در این مثال چون موضوع گروه بندی مطرح است مناسب از Group By براساس جدول Person طراحی گردد بنابراین می نویسیم:

```
Create View V2
AS
Select (Select UnitName From Unit Where UnitID= P.UnitID )AS UN,
      Count (*) AS No
From Person P
Group By UnitID
Having Count (*) >10
```

مثال) براساس View طراحی شده V2 یک SP طراحی کنید که واحدهائی را که بین 30 تا 50 کارمند دارند را ارائه نماید.

```
Create Proc GetUnitNo As
Select * From V2
Where No Between 30 and 50
Order By No Desc
```


مثال) براساس View طراحی شده V2 تعداد کارکنان واحد سازمانی را گروه بندی نمائید.
همانطور که مشاهده می گردد ستون تعداد کارکنان یعنی No قابل گروه بندی است که موضوع به افزایش قدرت با View اشاره می کند بنابراین خواهیم نوشت :

```
Select No ,Count (*) From V2
Group By No
Having Count (*) > 1
Order By No Desc
```

پارامتر Union در دستور Select :

در مواردی نیاز است که چند جدول با ساختارهای یکسان به یکدیگر متصل یا تلفیق شوند پارامتر Union در دستور Select در دو حالت ادغام جداول بصورت منحصر بفرد و یا بصورت تجمعی این فرآیند را انجام می دهد ، در صورتیکه تنها از کلمه Union استفاده گردد ادغام جداول و Union All ترکیب جداول بصورت تجمعی را به همراه خواهد داشت شکل کلی پارامتر Union به صورت ذیل خواهد بود.

```
Select دستور
[ Union [All] ]
Select دستور
:
```

مثال) در سیستم پرسنلی جدول محل تولد را با استفاده از Union تلفیق نمائید.
ابتدا تنها از پارامتر Union استفاده می نمائیم بنابراین خواهیم داشت :

```
Select * From City
Union
Select * From City
```

حال اگر از پارامتر Union All استفاده نمائیم خواهیم داشت :

```
Select * From City
Union All
Select * From City
```

اختلاف دو جدول بازگشتی در این است که چون در حالت اول از پارامتر Union تنها استفاده شده است و جدول City با خودش ادغام شده است پس نتیجه بازگشتی تنها جدول City خواهد بود ولی در حالت دوم جدول City به اندازه دو برابر رکوردهای جدول City تکرار خواهد شد و ادغام منحصر بفرد مقادیر رکوردهای دو جدول صورت نمی گیرد.

مثال) بدون ایجاد جدول فیزیکی ، جدول ذیل را به کاربر درخواست کننده منعکس کنید.

A	B	C	D
1	10	Ok	3.7
2	15	Yes	4.8
3	30	No	5.6
4	40	Ok	6.7

بدین منظور از پارامتر Union در دستور Select به شکل ذیل استفاده می نمایم.

Select A=1 , B=10 , C=' Ok ' , D=3.7 Union

Select A=2 , B=15 , C=' Yes ' , D=4.8 Union

Select A=3 , B=30 , C=' No ' , D=5.6 Union

Select A=4 , B=40 , C=' Ok ' , D=6.7

پارامترهای توابع جمعی در دستور Select :

هنگام استفاده از توابع جمعی نظیر Count , Sum , ... احتمال وجود مقدار Null در یک

عبارت، بر فرآیند عملیات محاسبات توابع جمعی مستقیماً اثرگذار خواهد بود بدین منظور پارامترهای توابع جمعی به شکل ذیل قابل اعلام خواهد بود بعنوان مثال در تابع Count خواهیم داشت:

کلیه رکوردهای اعلام شده شمارش می شوند (*)

رکوردهائی که مقدار عبارت آن غیر

Null باشند شمارش می شوند (عبارت All)

رکوردهائی که مقدار عبارت آن غیر

Null باشند بصورت منحصر بفرد شمارش می شوند (عبارت Distinic)

اشکال دیگری از Insert Into :

دستور Insert در شکل بسیار ساده تنها یک رکورد به یک جدول اضافه می کند اما در صورتیکه در نظر باشد مجموعه ای از رکوردها به یک جدول افزوده شود از یکی از اشکال ذیل می توان بدین منظور استفاده نمود.

Insert	[Into]	نام جدول	Select دستور
Insert	[Into]	نام جدول	EXEC[ute] SP نام [پارامترها در صورت نیاز]
Insert	[Into]	نام جدول	EXEC[ute] (متغیر رشته ای)
Insert	[Into]	نام جدول	View نام

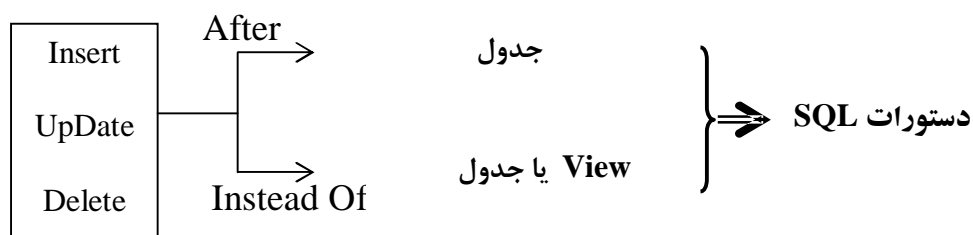
همانطور که مشاهده می گردد مستقیماً می توان مجموعه ای از ستون های استخراج شده از یک SP یا View یا Select را به یک جدول ارسال نمود که این شکل از دستور Insert بر قدرت دستور می افزاید.

(مثال) فرض کنید که اطلاعات محل تولد در سیستم پرسنلی در جدولی بنام MyCity و در پایگاه داده MyPerson قرار دارد دستوری بنویسید که اطلاعات به جدول City منتقل گردد.
در این صورت می توانیم بنویسیم:

Insert Into City Select * From MyPerson..MyCity

طراحی Trigger :

از نظر تئوری یک Trigger یک Event روی یک جدول یا View است که در صورتیکه هر یک از عملیات اضافه ، تصحیح و حذف روی Object مورد نظر صورت پذیرد بطور اتوماتیک برنامه طراحی شده اجرا خواهد شد. همچنین بر روی هر جدول یا View به هر تعداد می توان Trigger طراحی کرد دو نوع Trigger بنام های After , Instead Of روی هر یک از عملیات Insert , UpDate , Delete متصور است که به شرح ذیل می باشد:



(54)

همانطور که مشاهده می گردد Trigger های از نوع After تنها برای جداول متصور هستند و پس از اینکه هر یک از عملیات Insert , Update , Delete صورت پذیرد این Event اجرا خواهد شد اما Trigger از نوع Instead Of هم برای جداول و هم برای View قابل طراحی است این Trigger جانشین دستور طراحی شده می گردد در واقع در صورتیکه این نوع Trigger مثلاً برای Insert طراحی گردد هیچ رکوردی به جدول اشاره شده اضافه نمی شود و در صورتیکه در نظر است عملیات جانشین رکوردها در جدول صورت پذیرد می بایستی درون Trigger این عمل انجام شود بنابراین می توان گفت که Trigger از نوع Instead Of می تواند برای کنترل های قبل از تغییرات در جدول بکار رود براساس توضیحات ارائه شده شکل کلی طراحی یک Trigger به صورت ذیل خواهد بود:

```

Create Trigger Trigger نام ON نام جدول یا View نام
After یا Instead Of
[ Insert ] [ , ] [ UpDate ] [ , ] [ Delete ]
As
_____ }
_____ }   دستورات SQL
_____ }
_____ }

```

براساس شکل کلی دستور Trigger نکات ذیل قابل توجه است:

- برای View ها تنها می توان از Trigger از نوع Instead Of استفاده نمود.
- برای هر ترکیبی از Insert , Update , Delete می توان Trigger روی جدول یا View طراحی کرد.
- تعداد Trigger های طراحی شده به هر تعداد است پس از وقوع Event کلیه Trigger های طراحی شده برای اتفاق مورد نظر یکی پس از دیگری اجرا می گردد و پیش فرض آن ترتیب ایجاد Trigger هاست و در صورتیکه نیاز است ترتیب تغییر داده شود از یک SP بنام SP_SettriggerOrder و با پارامترهای نام Trigger , Firs/Last و نوع اتفاق می بایستی استفاده گردد.
- برای هر دستور با هر تعداد رکوردی که مورد پردازش قرار می گیرد تنها یکبار Trigger اجرا می گردد بعنوان مثال در صورتیکه 100 رکورد در یک جدول در حال Update شدن باشند تنها یک بار Trigger مربوط به Update اجرا خواهد شد.

- درون یک Trigger طراحی شده دو جدول مقیم در حافظه بنام های Deleted , Inserted در دسترس می باشند که رکوردهای مورد پردازش را در اختیار برنامه نویس تحت شرایط ذیل قرار می دهند.

نوع Trigger	جدول Inserted	جدول Deleted
Insert	کلیه رکوردهایی که قرار است به جدول اضافه شوند	خالی
UpDate	کلیه رکوردهای پس از تغییر	کلیه رکوردهای قبل از تغییر
Delete	خالی	کلیه رکوردهائی که قرار است حذف شوند

همانطور که ملاحظه می گردد از تعداد رکوردهای دو جدول مذکور می توان به نوع Trigger پی برد.

- اطلاعات درون جداول Deleted , Inserted تحت هیچ شرایطی توسط برنامه نویس قابل تغییر نمی باشند.

- درون یک Trigger نمی توان عملیات ایجاد ، تغییر و حذف یک پایگاه داده را انجام داد و تنها دسترسی به Object های پایگاه داده کنونی مقدور است.

- به منظور سازگاری با نسخه های قبلی SQL Server از کلمه For بجای After نیز می توان استفاده نمود.

- در Trigger از نوع Instead Of در صورتیکه روی جدول FK وجود داشته باشد قابل پیاده سازی نمی باشد.

- تابع (نام ستون) UpDate تعیین می کند که در فرآیند UpDate آیا ستون اشاره شده در حال تغییر است.

- تابع () Columns_UpDate که یک مجموعه بیتی است در فرآیند UpDate تعیین می کند که چه ستون هایی در حال تغییر هستند.

- درون یک Trigger مقدار @@RowCount تعداد رکوردهای مورد پردازش را تعیین می کند.

- موضوع Recursive در زمانی که دو جدول که دارای Trigger می باشند و در Trigger یکی تغییرات دیگری مدنظر قرار گرفته است باید پیش بینی گردد.

مثال) در سیستم پرسنلی فرض کنید جدولی بنام LogFile پیش بینی شده است که تغییرات در جدول پرسنل را نگهداری می کند. یک Trigger به منظور ثبت حذف پرسنل طراحی کنید.

در این مثال قرار است تنها برای عملیات Delete روی جدول Person طراحی گردد و فرض کنیم که جدول LogFile دارای سه فیلد نوع عملیات و شماره کارمندی و زمان باشد بدین منظور خواهیم نوشت:

```
Create Trigger DelPrs On Person
```

```
After Delete
```

```
AS
```

```
Insert Into LogFile Select 'D' AS LogType , PrsID , GetDate ( ) As LogDate
From Deleted
```

مثال) یک Trigger جهت ثبت کلیه تغییرات در جدول پرسنلی طراحی کنید که در جدول LogFile منعکس نماید.

به دو روش می توان موضوع را پیاده سازی کرد.

روش اول) در این حالت سه Trigger برای Insert , Update , Delete به صورت منفرد طراحی می کنیم.

```
Create Trigger I_Prns On Person
```

```
After Insert AS
```

```
Insert Into LogFile Select 'I' AS LogType,
PrsID ,
GetDate ( ) AS LogDate
From Inserted
```

```
Create Trigger U_Prns On Person
```

```
After UpDate AS
```

```
Insert Into LogFile Select 'U' AS LogType,
PrsID,
GetDate ( ) AS LogDate
From Deleted
```

```
Create Trigger D_Prns On Person
```

```
After Delete AS
```

```
Insert Into LogFile Select 'D' AS LogType,
PrsID,
GetDate ( ) AS LogDate
From Deleted
```

روش دوم) در این حالت یک Trigger طراحی می گردد و درون آن Trigger نوع Delete , UpDate , Insert براساس رکوردهای درون جداول موقت تشخیص داده می شود.

```

Create Trigger IUD_PrS On Person
After Insert , UpDate , Delete AS
Declare @I BigInt , @D BigInt , @ T Char (1) , @ S VarChar (500)
Set @I = (Select Count (*) From Inserted )
Set @D = (Select Count (*) From Deleted )
Set @T = Case
    When @ I <> 0 and @ D = 0 Then 'I'
    When @ I <> 0 and @ D <> 0 Then 'U'
    When @ I = 0 and @ D <> 0 Then 'D'
END
If @T = 'I'
    Insert Into LogFile Select @T As LogType, PrsID , GetDate ( )
    As LogDate From Inserted
Else
    Insert Into LogFile Select @T As LogType , PrsID, GetDate ( )
    As LogDate From Deleted

```

براساس Trigger طراحی شده پس از تشخیص نوع فرآیند که در متغیر T تعیین می گردد جدول Inserted یا Deleted بصورت پارامتریک انتخاب می شود و سپس در جدول LogFile ذخیره می گردد.

مثال) یک Trigger طراحی کنید که از ورود شماره کارمندی کمتر از 1000 جلوگیری کند. بدین منظور از یک Trigger از نوع Instead of استفاده می کنیم بنابراین می نویسیم:

```

Create Trigger PrsValid On Person
Instead Of Insert , UpDate AS
IF (Select Count (*) From Inserted Where PrsID < 1000 ) = 0
    Insert Into Person Select * From Inserted

```

همانطور که مشاهده می گردد در این حالت شرایط ReCursive اتفاق می افتد ولی عملیات بطور کامل انجام می گیرد البته شایسته است که به منظور کنترل شماره کارمندی بیشتر از 1000 از قید Check استفاده گردد.

در انتها متذکر می شود که استفاده نادرست از Trigger می تواند سربار در عملیات تغییر در اطلاعات را با جدول ایجاد نماید و در صورتیکه یک شرط می تواند با قید Check انجام شود استفاده از Trigger مناسب نخواهد بود.

طراحی Index :

ایندکس ها افزایش سرعت دسترسی به داده ها را محقق می سازند از طرفی به منظور ایجاد ارتباط بین PK و FK در جداول رابطه ای نیز باعث افزایش کارایی می شوند و در صورتیکه در نظر باشد از طریق کدنویسی یک ایندکس به یک جدول یا View اضافه نمائید شکل کلی ذیل بدین منظور بکار برده می شود:

Create [Unique] [Clustered یا NonClustered] Index
([, ...] [Asc یا Desc] نام ستون) نام ویو یا نام جدول ON نام ایندکس

- هر ایندکس ایجاد شده یا از نوع Clustered و یا از نوع NonClustered خواهد بود که در صورت عدم اعلام پیش فرض NonClustered مدنظر خواهد بود. اختلاف این دو ایندکس در بخش بعدی بحث خواهد شد.
- پارامتر اختیاری Unique در مواردی بکار می رود که در نظر است ایندکس ایجاد شده مقداری منحصر به فرد در جدول را مدیریت نماید.
- هر ایندکس روی یک جدول یا View می بایستی یک نام منحصر به فرد داشته باشد.
- ستون هایی که به ترتیب ایندکس روی آنها می بایستی ایجاد گردد در داخل پرانتز و با در نظر گرفتن صعودی یا نزولی روی ستون تعریف می گردد که در صورت عدم اعلام پیش فرض Asc خواهد بود.
- در صورتیکه پارامتر SchemaBinding در ایجاد یک View بکار رفته باشد قابلیت ایجاد ایندکس روی ویو امکان پذیر خواهد بود.
- به منظور ایجاد ایندکس روی یک View می بایستی نام Owner جدول اعلام گردد.

مثال) یک ایندکس منحصر به فرد روی جدول واحد سازمانی در سیستم پرسنلی ایجاد نمائید:
مناسب است که ایندکس ایجاد شده روی فیلد کد واحد سازمانی از نوع Clustered ایجاد گردد با این توضیحات می نویسیم:

Create Unique Clustered Index Ix_UID ON Unit (UnitID)

مثال) یک ایندکس مناسب در جدول افراد تحت تکفل در سیستم پرسنلی ایجاد نمائید.

در صورتیکه با دقت به جدول افراد تحت تکفل نگاه کنیم متوجه خواهیم شد که ترکیب شماره کارمندی و کد تحت تکفل یک PK می باشد که مناسب است یک ایندکس کلاستری و منحصر به فرد روی آن ایجاد گردد در این صورت خواهیم داشت:

Create Unique Clustered Index Ix_PC
ON Child (PrsID , ChdID)

همانطور که از شکل طراحی ایندکس مشاهده می گردد ایندکس ها می توانند یکی از دو نوع کلاستر یا غیر کلاستری باشند و از طرفی می دانیم که در هر جدول تنها یک ایندکس کلاستری می تواند وجود داشته باشد اما سؤال اینجاست که در زمان طراحی تشخیص نوع کلاستر یا غیر کلاستری چگونه ممکن است. بدین منظور به موارد نسبتاً تجربی ذیل دقت نمائید.

ایندکس کلاستری :

این نوع ایندکس داده ها را با همان ترتیبی که در ایندکس مرتب است در جدول ذخیره می سازد که در واقع ذخیره سازی بصورت فیزیکی اتفاق می افتاد و به همین علت است که تنها یک ایندکس کلاستری در یک جدول می توان ایجاد نمود و در دستور Select بدون Order By ترتیب نمایش داده براساس این ایندکس خواهد بود.

ایندکس غیر کلاستری :

در این نوع ایندکس ترتیب ایندکس با ترتیب داده ها در جدول یکسان نبوده و به تعداد محدودیت اعلام شده از طرف پایگاه داده قابل ایجاد است.

چه ایندکسی در چه شرایطی مناسب تر است؟

همانطور که قبلاً نیز گفته شد ایندکس ها به منظور افزایش سرعت دسترسی به داده ها پیاده سازی می شوند ولی در هنگام طراحی یک ایندکس تشخیص نوع ایندکس کلاستری و غیر کلاستری از اهمیت ویژه ای برخوردار است به مواردی در این خصوص توجه فرمائید:

- هنگامیکه تعداد ستون های ایندکس کم و ترکیب ایندکس یک PK باشد ایندکس کلاستری بسیار مناسب است.

- ایندکس روی فیلدهائی که شامل مقادیر خیلی تکراری مثل صفر و یک باشند اصلاً مناسب نمی باشند.
- اگر محدوده ای از یک ستون با مقادیر نسبتاً منحصر به فرد وجود دارد یک ایندکس کلاستری می تواند مناسب باشد.
- اگر یک ترکیب از فیلدهای جدول مکرراً مورد دستیابی قرار می گیرد یک ایندکس کلاستری می تواند مناسب باشد.
- جدولی که به دفعات روی یک ترکیب از ستون ها مرتب می گردد می تواند یک کاندید مناسب جهت ایندکس کلاستری باشد.
- هنگامیکه در یک جدول یک رکورد از جدول مورد جستجو قرار می گیرد می تواند یک کاندید مناسب برای ایندکس کلاستری باشد.
- اگر داده های جدول تغییرات کمی را به مرور زمان شامل می شود می تواند یک ایندکس کلاستری مناسب خواهد بود.
- در صورتیکه ستون های یک جدول مقادیر غیرمنحصر به فردی را به خود می گیرند شاخص غیر کلاستری مناسب تر خواهد بود.
- ستون هائی که مکرراً در جلوی عبارت Where ظاهر می گردد می تواند یک کاندید جهت یک ایندکس غیر کلاستری باشد و در صورتیکه ترکیب یک PK باشد همانطور که قبلاً گفته شد ایندکس کلاستری مناسب تر خواهد بود.
- در صورتیکه برنامه کاربردی شما از ارتباط با سایر جداول بهره می جوید مناسب است که برای جداول ارتباطی یک ایندکس کلاستر یا غیر کلاستر مناسب با موارد اشاره شده طراحی گردد.

تکنولوژی Rushmore :

همانطور که اشاره شد طراحی ایندکس روی جداول تنها به مرتب سازی رکوردها، جهت مشاهده یا گزارش گیری محدود نمی شود بلکه شاخص ها مستقیماً بر روی کارائی برنامه تاثیر گذار خواهند بود. از زمانیکه شرکت مایکروسافت تکنولوژی Rushmore را معرفی کرد این تکنیک بهینه سازی کارایی برنامه کاربردی را به دلیل سرعت بازیابی سریع رکوردها به طور قابل توجهی افزایش داد. بسیاری از برنامه نویسان پس از بکارگیری تکنولوژی فوق اعلام کردند که استفاده از این تکنیک نسبت به گذشته کارایی برنامه ها را از 100 تا 1000 برابر افزایش داده است. به عنوان نمونه در یک عملیات مدت یافتن 2217 رکورد در بین 42818 رکورد در یک آزمایش 4/61 ثانیه بوده است که این زمان بسته به سرعت پردازنده، دیسک گردان، موقعیت فیزیکی سکورها و چند عامل دیگر در کامپیوتر بیشتر یا کمتر خواهد شد

اما در روش بکارگیری از تکنیک این زمان به 0/16 ثانیه تقلیل یافته است اگرچه در این حالت سرعت 29 بار افزایش یافته است اما در تعداد زیاد داده ها قطعاً اختلاف فرآیند استخراج بسیار بیشتر از این مقدار خواهد بود. اما تکنولوژی Rushmore کارایی برنامه را به چه صورت افزایش می دهد؟

ابتدا ایندکس مناسب جهت فرآیند استخراج جستجو می شود سپس نخستین رکورد تحت شرایط موردنظر استخراج می گردد و با استفاده از ایندکس مناسب تا زمانیکه شرط موردنظر تغییر نکرده است فرآیند استخراج داده های موردنظر ادامه می یابد. در واقع دستوراتی که در مقابل کلمات Where و Order By قرار می گیرند Rushmore را تحت تاثیر قرار می دهند در استفاده از تکنولوژی Rushmore موارد ذیل مهم است که مدنظر قرار گیرد:

- چون Rushmore باید عبارت شاخص را بایک متغیر یا مقدار ثابت مقایسه کند بنابراین قادر به بهینه سازی مواردی که این موضوع را تحت شعاع قرار دهد نخواهد بود مثلاً استفاده از توابع می تواند در بهینه سازی Rushmore تاثیر گذار باشد.
- این تکنولوژی تنها قادر به بهینه سازی عبارتی است که براساس شاخص های معمولی ایجاد شده اند می باشد.
- در صورتیکه یک ایندکس از فیلدها روی یک جدول ایجاد شده باشد و از سمت چپ یک دستور تنها بخشی از شرط با ترتیب ایندکس شباهت داشته باشد Rushmore قادر به بهینه سازی خواهد بود که این موضوع نشان می دهد که از ایجاد ایندکس هایی که بخشی از آنها تکراری است می بایستی جلوگیری کرد چون علاوه بر بی استفاده بودن باعث ایجاد سربار زمانی در زمان حذف، تصحیح و اضافه کردن رکورد به جدول خواهد شد.
- استفاده از برخی توابع داخلی می تواند عملکرد تکنولوژی Rushmore را از حالت بهینه خارج نماید.
- Rushmore کسری از زمان کاری خود را صرف بررسی عبارت با شاخص های موجود می نماید و در صورتیکه دستورات به گونه ای پیش بینی گردد که این مرحله حذف گردد Rushmore شاخص بهینه را بهتر انتخاب خواهد کرد.

درانتها متذکر می شویم که هیچ روش ایده آلی جهت پیاده سازی ایندکس ها وجود نداشته و به شرایط بستگی دارد که این شرایط شامل کارهای موردنظر، ساختار جاری داده ها، دستورات خاص استفاده شده خواهد بود.

مثال) در صورتیکه در سیستم پرسنلی از دستور ذیل به تکرار استفاده می گردد چه ایندکسی بسیار مناسب است ؟

Select * From Child Where PrsID = @ P Order By ChdID

همانطور که از شکل دستور مشاهده می گردد محدوده انتخاب یک PrsID خاص خواهد بود و در این محدوده مرتب سازی براساس ChdID صورت می پذیرد علیهذا همان PK این جدول یک ایندکس مناسب است یعنی یک ایندکس براساس ابتدا PrsID و سپس ChdID می تواند باعث بهینه سازی سرعت دسترسی به داده شود.

طراحی Extended Stored Procdures (ESP) :

همانطور که قبلاً اشاره شد یک ESP در واقع یک پروژه از نوع Extended Stored Procdures است که به زبان VC++ پیاده سازی می گردد خروجی این نرم افزار یک DLL است که توسط پایگاه داده SQL Server قابل اضافه شدن به توابع خواهد بود ، هنگامیکه یک DLL از این نوع که می تواند شامل تعدادی تابع باشد ایجاد می گردد ابتدا می بایستی به پایگاه داده Master از طریق Wizard یا از طریق یک SP بنام SP_addextendedproc معرفی گردد سپس با استفاده از EXEC قابل اجرا خواهد بود در این خصوص نکات ذیل قابل توجه است:

- ایجاد یک ESP از طریق کدنویسی با استفاده از SP_addextendedproc صورت می گیرد.
- فایل DLL با ساختار پروژه ESP در صورتیکه در مسیر Bin سرور SQL قرار داشته باشد در هنگام معرفی نیازی به ارائه مسیر قرار گرفتن DLL ندارد.
- در صورتیکه یک ESP قرار است با کدنویسی حذف شود از SP_addextendedproc صورت می گیرد.
- نام گذاری حروف کوچک و بزرگ برای نام ESP بسیار مهم است.
- با دستور EXEC تابع درون یک ESP اجرا می گردد.

مثال) یک تابع برای SQL بنویسید که یک شمارنده عددی Public از نوع TinyInt در حافظه Sever ایجاد نماید.

بدین منظور به ترتیب مراحل ذیل عمل می نمایم:

- یک پروژه از نوع ESP در VC++ ایجاد می‌نمائیم و در بخش Globals کد ذیل را می‌نویسیم:

```
#include <stdafx.h>
unsigned char ID = 0;
#ifdef __cplusplus
extern "C" {
#endif
RVRETCODE __declspec(dllexport) Xp_MyID (SRV_PROC *srvproc);
#ifdef __cplusplus
}
#endif
//--- Get New ID ----
SRVRETCODE __declspec(dllexport) Xp_MyID (SRV_PROC *srvproc)
{
    if (ID == 255) ID = 0;
    ++ ID;
    return ID;
}
```

- فرض می‌کنیم که DLL ایجاد شده بنام MyProjID در مسیر C:\MyID\ ایجاد شده باشد.
 - با استفاده از SP_addextendedproc می‌توان DLL ایجاد شده را به شکل ذیل برای SQL Server ثبت کرد.

```
Use Master
EXEC SP_addextendedproc Xp_MyID , `C:\MyID\MyProjID.dll`
```

در اینجا در واقع Xp_MyID نام ESP ایجاد شده خواهد بود.

- تابعی بنام GNID طراحی می‌کنیم که از ESP ایجاد شده استفاده می‌کند بنابراین داریم:

```
Create Function GNID ( )
Returns TinyInt
As
Begin
    Declare @R TinyInt
    EXEC @R = master..Xp_MyID
    Return @R
End
```

(64)

تعدادی ESP مفید در SQL Server :

تعدادی ESP توسط SQL Server به منظور استفاده برنامه نویسان موجود است که کاربردی ترین

آنها به شرح جدول ذیل می باشد.

مثال	عملکرد	نام ESP
Master .. xp_cmdshell 'Dir C:\'	یک خط فرمان Dos را روی Server اجرا کرده و نتیجه را باز می گرداند	xp_Cmdshell
	در صورتیکه mail in box پایگاه داده نصب باشد یک mail خوانده می شود	xp_Readmail
	نتیجه یک Query یا یک پیام قابل اتصال به یک پیغام قابل mail شدن است	xp_Sendmail
master .. xp_msver	یک Query از وضعیت SQL Server منعکس می کند	xp_msver

تعدادی SP مفید در SQL Server :

مشابه ESP تعداد بسیار زیادی SP به منظور استفاده برنامه نویسان کاربرد در SQL Server

پیاده سازی گردیده که کاربردی ترین آنها به شرح جدول ذیل قابل ارائه است:

عملکرد	نام SP
یک کاربر برای پایگاه داده جاری ایجاد می نماید	SP_adduser
از یک مسیر اعلام شده یک پایگاه داده را به سرور معرفی می کند	SP_attach_db
ستون های یک جدول را در پایگاه داده کنونی ارائه می دهد	SP_columns
فهرست پایگاه داده های سرور جاری را باز می گرداند	SP_databases
فهرستی از انواع داده ها به همراه محدودیت های هر نوع داده را باز می گرداند	SP_datatype_info
ارتباط یک پایگاه داده را با سرور قطع می کند	SP_detach_db
اطلاعاتی درخصوص FK روی یک جدول در پایگاه داده کنونی را ارائه می دهد	SP_fkkeys
مشابه SP_databases ولی در ساختاری متفاوت فهرستی از پایگاه داده های سرور ارائه می دهد	SP_helpdb
فهرستی از مسیرهای Back Up ایجاد شده در SQL را ارائه می دهد	SP_helpdevice
اطلاعاتی نظیر موقعیت فیزیکی ، سایز و ... از پایگاه داده کنونی ارائه می دهد	SP_helpfile
فهرستی از وضعیت ایندکس روی یک جدول از پایگاه داده را ارائه می دهد	SP_helpindex
اطلاعاتی درخصوص User های پایگاه داده کنونی را ارائه می دهد	SP_helpuser
وضعیت عملکردی SQL Server را بطور خلاصه نمایش می دهد	SP_monitor
کلمه عبور یک کاربر یا کاربر فعلی را با داشتن کلمه عبور قبلی تغییر می دهد	SP_password
اطلاعاتی درخصوص PK روی یک جدول در پایگاه داده کنونی را ارائه می دهد	SP_Pkeys
نام یک Object نظیر جدول ، ستون ، ... را در پایگاه داده کنونی تغییر می دهد	SP_Rename
نام یک پایگاه داده را تغییر می دهد	SP_Renamedb
فهرستی از SP و UDF های پایگاه داده کنونی را ارائه می دهد	SP_Stored_Procedures
فهرستی از جداول موجود در پایگاه داده کنونی را ارائه می دهد	SP_Tables
یک Device جدید جهت Back Up گیری برای SQL Server معرفی می کند	SP_addumpdevice

: Cursor

- هنگام استفاده از دستورات SQL نظیر Select کلیه رکوردهای درخواستی بطور کامل استخراج می گردد اما در مواردی نیاز است که رکوردهای استخراج شده تحت شرایطی خاص مورد پردازش مجدد قرار گرفته و به برنامه درخواست کننده ارسال گردد در این صورت استفاده از کرسرها بسیار حائز اهمیت خواهد بود در واقع برای استفاده از یک کرسر می توان به ترتیب مراحل ذیل عمل نمود.
- یک متغیر از نوع کرسر تعریف می گردد. که شامل دستور Select درخواستی خواهد بود.
 - با استفاده از دستور Open یک کرسر آماده استفاده می گردد.
 - با استفاده از دستور Fetch حرکت درون یک کرسر امکان پذیر می گردد که در این حالت مقدار فیلدهای اعلام شده در رکورد جاری در دسترس می باشد.
 - با استفاده از دستور Close کرسر فعال شده بسته می شود.
 - با استفاده از Deallocate فضای اختصاص داده شده برای کرسر آزاد می گردد.

روش ایجاد کرسر :

شکلی از دستور Declare قادر به ایجاد نوع داده ای تحت عنوان کرسر خواهد بود که در شکل کلی ذیل قابل ارائه می باشد:

```

Declare      نام کرسر      Cursor [ Local یا Global ]
[ Forward_Only یا Scroll ] [ Static یا Keyset یا Dynamic ]
[ Read_Only یا Scroll_Locks یا Optimistic ]
For Select دستور
[ For UpDate [ Of ستون ها ] ]

```

براساس این دستور موارد ذیل می بایستی مدنظر قرار گیرد:

- Local که پیش فرض دستور است باعث ایجاد یک کرسر محلی در یک SP یا Trigger خواهد بود که با خروج از آنها بطور اتوماتیک حافظه مربوطه آزاد خواهد شد.
- Global یک کرسر با زاویه دید عمومی ایجاد می کند که در کلیه SP ها یا دستورات اجرائی در دسترس خواهد بود و در صورت پایان ارتباط بصورت اتوماتیک حافظه مربوطه آزاد خواهد شد.

- Forward_Only که پیش فرض دستور است یک کرسر رو به جلو ایجاد می کند که در این حالت حرکت اشاره گر رکورد به عقب مجاز نخواهد بود و این موضوع سرعت عملیات را نیز افزایش می دهد و در مقابل Scroll قابلیت جلو و عقب را برای کرسر ممکن می سازد.
- Static که پیش فرض دستور می باشد کرسری است که یک کپی موقت درون پایگاه داده TempDB ایجاد می کند و در این حالت هیچ عکس العملی را به داده های اصلی منعکس نمی کند.
- Keyset کرسری ایجاد می کند که تغییرات ایجاد شده توسط دیگران به شما منعکس می گردد و حذف و افزودن رکورد توسط دیگران به این کرسر منعکس نخواهد شد.
- Dynamic یک کرسر متغیر ایجاد می کند که کلیه تغییرات توسط دیگران به کرسر منعکس می گردد.
- Read_Only کرسری ایجاد می کند که تغییرات در داده ها را منعکس نخواهد کرد.
- Scroll_Locks نوعی کرسر ایجاد می کند که تضمین می نماید که سطرها را به همان طریقی که خوانده است جهت تغییرات بعدی قفل نماید.
- Optimistic نوعی کرسری است که تغییرات در داده ها را تضمین نمی کند و اطلاعات قفل نمی شود و جهت اعمال تغییرات در صورت وجود فیلد از نوع TimeStamp استفاده کرده و در غیراینصورت از مقدار CheckSum رکورد استفاده می نماید.
- دستور Select اعلام شده همان فرمان استاندارد Select می باشد و استفاده از فرامینی مشابه Into درون یک کرسر مجاز نخواهد بود.
- پارامتر اختیاری For UpDate جهت تعیین ستونهای قابل ویرایش پیش بینی شده است.

حرکت روی رکوردها (Fetching) :

مشابه همه ابزارهای ارتباط با پایگاه داده نظیر ADO حرکت روی رکوردهای بازگشتی با استفاده از دستور Fetch امکان پذیر است و شکل کلی دستور Fetch به صورت ذیل می باشد:

```
Fetch [ Next یا Prior یا First یا Last یا
        مقدار نمو Relative یا شماره رکورد Absolute
From نام کرسر یا متغیر از نوع کرسر
[ Into فهرست متغیرها ]
```

پارامترهای بکار رفته در دستور به شکل ذیل خواهد بود:

- Next که پیش فرض دستور است حرکت به رکورد بعدی را میسر می کند.
- Prior حرکت اشاره گر به رکورد قبلی خواهد بود.
- First حرکت اشاره گر به رکورد اولیه خواهد بود.
- Last حرکت اشاره گر به رکورد انتهائی خواهد بود.
- Absolute اشاره گر را به شماره رکورد اعلام شده هدایت می کند.
- Relative اشاره گر را به اندازه تغییرات اعلام شده نسبت به رکورد جاری هدایت می کند.
- در صورتیکه قرار است ستونهای رکورد جاری در تعدادی متغیر جانشین شود از فرمان Into به همراه فهرست متغیرها که قبلاً اعلام شده اند استفاده می گردد.
- مقدار تابع @@Fetch_Status وضعیت کنونی کرسر را مورد تحلیل قرار می دهد و در صورتیکه مقدار صفر را داشته باشد نشان می دهد که عملیات دریافت داده توسط کرسر موفقیت آمیز بوده است.

مثال) در سیستم پرسنلی تابعی بنویسید که متوسط تحصیلات افراد تحت تکفل هر شخص را محاسبه کند.

همانطور که از ظاهر درخواست معین است جدول ChdLvL جهت انجام عملیات Fetch

مناسب ترین انتخاب است بدین منظور می نویسیم:

```

Create Function AvgLvL (@PrsID Int )
Returns Real
AS Begin
Declare @i TinyInt , @L Real , @A Real
Set @i = 0 Set @A = 0
Declare TmPCursor Cursor For
Select LID From ChdLvL Where PrsID = @PrsID
Open TmPCursor
While 1=1 Begin
Fetch Next From TmPCursor Into @L
If @@ Fetch_Status <> 0 Break
Set @i = @i + 1
Set @A = @A + @L
End
Close TmPCursor
Deallocate TmPCursor
If @i <> 0 Set @A = @A / @i
Return @A
END

```

در برنامه طراحی شده شماره کارمندی بعنوان پارامتر ورودی به تابع AvgLVL وارد می گردد و در خط ششم کرسری بنام TmPCursor براساس دستور Select که افراد تحت تکفل PrsID خاصی را باز می گرداند ایجاد شده است و درون حلقه مقدار کلیه تحصیلات افراد تکفل جمع می شود و براساس @@Fetch_Status انتهای کرسر کنترل می گردد ، سپس کرسر ایجاد شده بادستور Close بسته می شود و دستور Deallocate حافظه ایجاد شده توسط کرسر را آزاد می کند و در انتهای برنامه تابع میانگین سطح تحصیلات را باز می گرداند ، شایان ذکر است که میانگین سطح تحصیلات افراد تحت تکفل با استفاده از یک Select نیز قابل محاسبه بود ولی در این تابع تنها هدف استفاده از کرسرها می باشد چون در مواردی با استفاده از یک دستور Select نمی توان درخواست موردنظر را محقق کرد ، بعنوان مثال در مواردیکه نتیجه ای وابسته به مقایسه هر رکورد با رکورد ماقبل یا مابعد باشد استفاده از کرسرها یک الزام خواهد بود.

ارتباط با سایر پایگاه داده ها :

در پایگاه داده SQL Server دستوراتی تعبیه شده است که بتوان از طریق کدنویسی ارتباط با جداول در سایر پایگاه داده ها را امکان پذیر نمود استفاده از دستوراتی نظیر OpenDataSource , OpenRowSet , LinkedServer از جمله روش های متعارف در این خصوص می باشند در این حالت براساس وجود یک نوع Provider و ارائه یک ConnectionString ارتباط با یک جدول درون پایگاه داده خارجی امکان پذیر خواهد بود.

: OpenDataSource

با استفاده از تابع OpenDataSource که پارامترهای ConnectionString , Provider را بعنوان ورودی می پذیرد ارتباط با یک جدول در یک پایگاه داده راه دور برقرار می گردد که Provider از یک درایور OLE_DB استفاده می نماید و عبارت ConnectionString رشته ای شامل سرور و پایگاه داده و کاربر و کلمه عبور جهت اتصال خواهد بود. شایان ذکر است که چون Provider مربوط به OLE_DB برای ODBC نیز موجود است بنابراین اتصال ODBC نیز می تواند مورد استفاده قرار گیرد.

مثال) در سیستم پرسنلی واحد سازمانی را از جدول دیگری در یک سرور راه دور دریافت نمائید:

فرض کنید که یک سرور تحت نام RServer نیز پایگاه داده پرسنلی پروژه مورد بحث ما را شامل شده است بنابراین می توانیم بنویسیم:

```
Select P.* , R.UnitName
From Person P
Left Join OpenDataSource ('SQLOLEDB','Data Source = RServer ;
User ID = sa;Password=123'). PersonDB.dbo.Unit R
On ( R.UnitID = P.UnitID )
```

حال اگر فرض کنیم بجای یک پایگاه داده راه دور از نوع SQL Server یک پایگاه داده اکسل در مسیر خاصی از سرور قرار دارد دستور به شکل ذیل قابل پیاده سازی است.

```
Select P.* , R.UnitName
From Person P
Left Join OpenDataSource ('Microsoft.Jet.OLEDB.4.0',
'C:\ PersonDB.XLS';User ID=Admin;Password=123;Extended
Properties=Excel 5.0')...Unit R
On ( R.UnitID = P.UnitID )
```

که در این مثال Unit یک Spread Sheet درون فایل PersonDB.XLS است که جهت اتصال از Engine مربوط به اکسس استفاده گردیده است.

: OpenRowSet

تابع OpenRowSet با دریافت یک Query و پارامترهای ارتباطی نظیر Provider و محل سرور و کاربر و کلمه عبور و سایر پارامترهای ارتباطی متناسب با Provider اتصال به یک سرور راه دور را مهیا خواهد ساخت و اختلاف آن با OpenDataSource در نحوه ارائه پارامترهای Provider و میزان کارایی در دریافت Query است. از طرفی این تابع می تواند در هر یک از فرامین Select , Insert , Update , Delete ظاهر گردد.

مثال) با استفاده از OpenRowSet به یک جدول پرسنلی در سرور راه دور RServer و یک جدول واحد سازمانی در یک پایگاه داده اکسس متصل شوید.

```
Select P.* , U.UnitName , C.CityName From
OpenRowSet('SQLOLEDB' , 'RServer' ;'sa';'123',
'Select * From Person DB.dbo.Person Order By PrsID'
) As P
Left Join OpenRowSet ('Microsoft.Jet.OLEDB.4.0' ,
'C:\PersonDB.MDB';'Admin';'123',Unit ) As U
On (U.UnitID = P.UnitID )
Left Join City C On (C.CityID = P.CityID )
```

در خصوص نوع ارائه پارامترهای ورودی توابع OpenRowSet , OpenDataSource مناسب است که به مستندات Books OnLine مراجعه شود.

: Linked Server

یکی از روش های ارتباط با یک پایگاه داده دور استفاده از Linked Server می باشد که از جنبه امنیتی حائز اهمیت است و مشابه توابع OpenDataSource , OpenRowSet از طریق یک Provider برای OLE_DB ارتباط با پایگاه داده موردنظر برقرار می گردد. شایان ذکر است که جهت ارتباط با پایگاه داده هایی که از ODBC پشتیبانی می نمایند می باید از Provider مربوط به OLE_DB برای ODBC استفاده گردد. در هر حال جهت راه اندازی یک Linked Server می بایستی نکات ذیل مورد توجه قرار گیرد.

- با استفاده از SP تحت نام SP_addLinkedServer و ارائه پارامترهای ورودی یک ارتباط با یک پایگاه داده از طریق OLE_DB برقرار می گردد.
- با استفاده از تابع OpenQuery یک دستور SQL قابل اجرا خواهد بود.
- با استفاده از SP_LinkedServers فهرستی از ارتباط از نوع Linked Server ارائه خواهد شد.
- با استفاده از SP_dropServer یک ارتباط از نوع Linked Server از فهرست ارتباطات حذف خواهد شد.
- برخلاف OpenDataSource و OpenRowSet از نظر امنیتی استفاده از Linked Server بسیار حائز اهمیت است و از طرفی از OpenQuery در درون یک SP نیز می توان با ارائه نام Linked Server ایجاد شده استفاده نمود.

- اگر دستور SQL شامل یک SP باشد از فرمان EXEC استفاده نمائید.

مثال) در سیستم پرسنلی اگر جدول واحدهای سازمانی در یک سرور دیگر تحت نام RServer وجود داشته باشد یک SP جهت ارائه فهرست کارکنان طراحی نمائید.

بدین منظور اگر فرض کنیم که در سرور RServer یک پایگاه داده از نوع SQL Server وجود دارد مراحل ذیل را انجام می دهیم.

- ابتدا یک ارتباط با سرور RServer ایجاد می گردد.

```
Use Master
Go
SP_AddLinkedServer 'MyLinked', ' ', 'MSDASQL', Null, Null,
'Driver = {SQL Server};Database=PersonDB;Server=Rserver;
UID=sa;PWD=123;'
```

- براساس دستور قبل یک Linked Server تحت نام MyLinked ایجاد شده است بنابراین خواهیم داشت.

```
Create Proc GetPrs
As
Select P.*, U.UnitName From Person P
Left Join OpenQery(MyLinked, 'Select * From Unit') U
On (U.UnitID=P.UnitID)
```

حال اگر بجای سرور RServer فرض شود که پایگاه داده یک بانک اطلاعاتی اکسس است بنابراین تنها می بایستی MyLinked به شکل ذیل پیاده سازی گردد و بقیه مراحل هیچ تفاوتی ندارد.

```
Use Master
Go
SP_AddLinkedServer 'MyLinked', 'OLE DB Provider for Jet',
'Microsoft.Jet.OLEDB.4.0', 'C:\PersonDB.mdb'
```

در انتها متذکر می شویم که درخصوص چگونگی ارائه پارامترهای لازم جهت اجرای SP_AddLinkedServer می بایستی به مستندات Books OnLine مراجعه شود.

فصل چهارم

نمونه سؤالات پایگاه داده

SQL Server

نمونه سؤالات فصل اول

نوع امتحان: کتاب بسته

زمان: ۴۵ دقیقه

1- در قطعه برنامه ذیل مقدار بازگشتی چقدر است؟

```
Declare @x TinyInt , @y BigInt
Set @y = 1
While @x < 5 Begin
    Set @x = @x + 1
    Set @y = @y * @x
END
Select @y
```

(1) 120 (2) 24 (3) 1 (4) هیچکدام

2- در صورتیکه هیچکدام از متغیرهای بکار رفته در خروجی و ورودی توابع از نوع SQL – Variant نباشند نوع خروجی تابع Fy را در صورت وجود تعیین کنید.

```
Select Case
    When @D = dbo.Fy( ) Then .....
    When @A = @B Then dbo.Fx (@A)
    When @D = @B + @C Then dbo.Fx (dbo.Fx (@A) )
    ELSE IsNull (@E, '')
END
```

(1) رشته ای (2) عددی (3) نمی توان نظر داد (4) این دستور اشتباه است

3- در قطعه برنامه مقابل خروجی چقدر است؟

```
Declare @x nVarChar Set @x = 'SQL'
Select Len (@x)
```

(1) 1 (2) 2 (3) 3 (4) 6

4- در طراحی یک نرم افزار حمل و نقل ریلی به منظور تعیین نوع فیلد اطلاعاتی کد ایستگاه راه آهن چه نوع داده ای مناسب تر است؟

TinyInt (1) SmallInt (2) Int (3) BigInt (4)

5- در قطعه برنامه ذیل مقدار متغیر X در نهایت چقدر خواهد بود ؟

```

Declare @x TinyInt
Set @x = 10
IF @x%2 = 0
    Set @x = @x - 2
ELSE
    Set @x = Case When @x = Power (2,2) Then @x-3 end
    Set @x = Case When @x = Power (4,2) Then @x-2 end
    Set @x = Case When @x = Power (3,2) Then @x-1
END
Select @x

```

(4) هیچکدام

(3) 4

(2) 5

(1) 8

6- تابعی بنویسید که مقدار عبارت ذیل را محاسبه کند ؟

$$T(x) = \frac{x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots}{1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \dots}$$

نمونه سئوالات فصل دوم

نوع امتحان: کتاب بسته

زمان: ۴۵ دقیقه

1- در قطعه برنامه ذیل مقدار بازگشتی را تعیین کنید. (0/5)

```
Declare @x , @y BigInt
Set @y = 1
While @x < 5 Begin
    Set @x = @x + 1
    Set @y = @y * @x
END
Select @y
```

(4) برنامه اشتباه است 1 (3) 24 (2) 120 (1)

2- در قطعه برنامه مقابل مقدار بازگشتی چقدر است؟ (0/5)

```
Declare @x TinyInt , @S Int
Set @x = 0 Set @S = 0
While @x < 3
    Set @x = @x + 1
    Set @S = @S + Power (@x , 2 )
Select @S
```

(4) هیچکدام 14 (3) 11 (2) 9 (1)

3- در طراحی یک نرم افزار پرسنلی چه نوع داده ای را برای تعیین کد استان محل خدمت مناسب تر می دانید؟ (0/5)

BigInt (4) Int (3) SmallInt (2) TinyInt (1)

4- در قطعه برنامه مقابل مقدار آخرین بازگشتی چقدر است؟ (0/5)

```
Declare @x TinyInt
Set @x = 0
Select Case @x
    When 0 Then @x + 1
    When 1 Then @x + 2
END Select @x
```

0 (1) 1 (2) 2 (3) 4 (دستور اشتباه است)

5- در قطعه برنامه مقابل مقدار بازگشتی کدام متغیر خواهد بود؟ (0/5)

```
Declare @x TinyInt, @y TinyInt, @z TinyInt
Select @x = 10, @y = 13, @z = 5
If @x < @y
If @x < @z Select @x
Else Select @y
Else Select @z
```

x (1) y (2) z (3) 4 (دستور اشتباه است)

6- یک SP بنویسید که حاصل عبارت ذیل را محاسبه کند. (2/5)

$$L(x) = \left(x - \frac{x^3}{2!} + \frac{x^5}{4!} - \dots\right) \times \left(x^2 - \frac{x^4}{3!} + \frac{x^6}{5!} - \dots\right)$$

نمونه سؤالات فصل دوم

نوع امتحان: کتاب بسته

زمان: ۱۵': ۱

1- کدام Object با استفاده از دستور Alter قابلیت اصلاح دارد؟ (0/5 نمره)

Index (1 Trigger (2 ESP (3 Default (4

2- تعداد اشکالات موجود در دستور ذیل را نام ببرید؟ (1 نمره)

```
Select PrsID,(Select Top 2 From City C Where P.CityID = CityID)
UnitID
From Person P
Join Unit U On (UnitID = P.UnitID)
Order By Select Top 3 UnitName From Unit
Where CityID = P.CityID
Where X(Price)
```

3- دستوری بنویسید که خروجی ذیل را به ترتیب نزولی تعداد کارکنان ارائه دهد؟ (2 نمره)

نام شهرستان محل تولد ، تعداد کارکنان ، تعداد افراد تحت تکفل

4- بدون استفاده از SubQuery دستوری بنویسید که جدول ذیل را به ترتیب نام و نام خانوادگی ارائه دهد؟ (1/5 نمره)

تحت تکفل			کارمند		
محل تولد	نام خانوادگی و نام	نسبت با کارمند	محل تولد	نام خانوادگی و نام	شماره کارمندی
.....

Unit واحد سازمانی

نام فیلد	ملاحظات	نوع
UnitID	کد واحد سازمانی	SmallInt
UnitName	واحد سازمانی	nVarChar(100)
CityID	محل استقرار	SmallInt

Person پرسنل

نام فیلد	ملاحظات	نوع
PrsID	شماره کارمندی	Int
PName	نام	nVarChar(20)
PFamily	نام خانوادگی	nVarChar(30)
PFather	نام پدر	nVarChar(20)
CityID	کد محل تولد	SmallInt
UnitID	کد واحد سازمانی	SmallInt
Price	حقوق کارمند	BigInt

City محل تولد

نام فیلد	ملاحظات	نوع
CityID	کد محل تولد	SmallInt
CityName	محل تولد	nVarChar (50)
PoPulate	جمعیت	Int

Child افراد تحت تکفل

نام فیلد	ملاحظات	نوع
PrsID	شماره کارمندی	Int
ChdID	کد تحت تکفل	TinyInt
CName	نام	nVarChar(20)
CFamily	نام خانوادگی	nVarChar(30)
CFather	نام پدر	nVarChar(20)
CityID	کد محل تولد	SmallInt
RID	نسبت به کارمند	TinyInt

Relate نسبت با کارمند

نام فیلد	ملاحظات	نوع
RID	کد نسبت با کارمند	TinyInt
RName	نسبت	nVarChar (20)

ChdLvL تحصیلات افراد تحت تکفل

نام فیلد	ملاحظات	نوع
PrsID	شماره کارمندی	Int
ChdID	کد تحت تکفل	TinyInt
LDate	تاریخ اخذ مدرک	Char(10)
LPlace	محل اخذ مدرک	nVarChar(200)
LID	کد مدرک تحصیلی	TinyInt

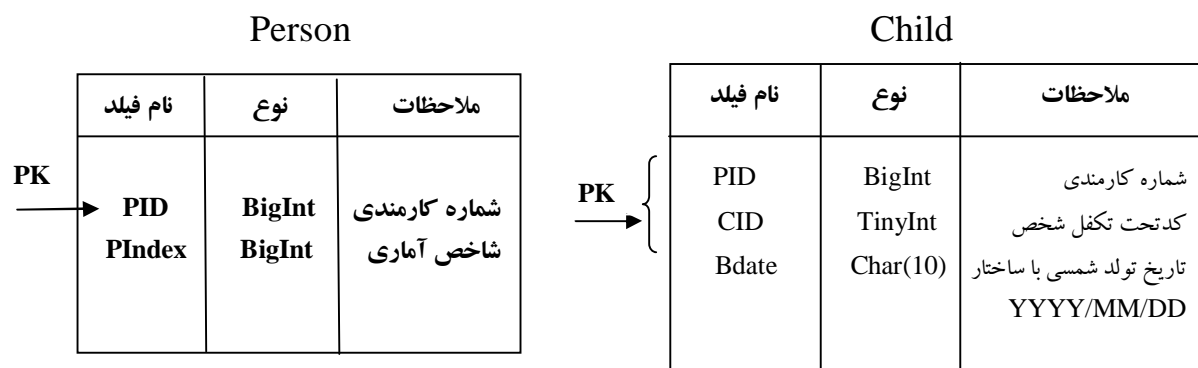
LvL مدرک تحصیلی

نام فیلد	ملاحظات	نوع
LID	کد مدرک تحصیلی	TinyInt
LName	مدرک تحصیلی	nVarChar (20)

نمونه سؤالات فصل سوم

نوع امتحان : کتاب باز	زمان: ۳۰': ۱
<p>1- خروجی دستور $Select Round (123,-2)/3+Len(Str(@x))/3$ چقدر است؟ (0/25)</p>	
<p>(1 36.66666 (2 36 (3 به مقدار متغیر x بستگی دارد 4 پارامتر 2- مجاز نیست</p>	
<p>2- کدامیک از فرمول های ذیل برای محاسبه تعداد روزهای یک ماه شمسی (@M) صحیح است (بدون احتساب سال کبیسه)؟ (0/25)</p>	
<p>Select 31-@M/7-@M/12 (1</p>	
<p>Select 31- Round (@M/7,0)-Round (@M/12,0) (2</p>	
<p>Select 31-Floor (@M/7)-Floor(@M/12) (3</p>	
<p>(4 هر سه روش</p>	
<p>3- به منظور طراحی یک متغیر عمومی بین کلیه کاربران که هم اکنون به SQL Server متصل هستند استفاده از چه نوع Object مناسب است؟ (0/25)</p>	
<p>SP (1 View (2 ESP (3 4) قابل پیاده سازی نمی باشد</p>	
<p>4- به منظور ارتباط با یک پایگاه داده راه دور یک SP طراحی شده است در این حالت کدامیک از توابع ذیل را مناسب تر می دانید؟ (0/25)</p>	
<p>(1 OpenDataSource (2 OpenRowSet (3 OpenQuery 4) فرقی نمی کند</p>	
<p>5- یک برنامه نویس شرایطی را در یک پایگاه داده ایجاد کرده است که یک جدول خاص عملکردی کاملاً غیرقابل تغییر دارد اگر موضوع دسترسی به اطلاعات مطرح نباشد علت موضوع را توضیح دهید؟ (0/5)</p>	

6- در یک سیستم پرسنلی با ساختار ارائه شده و به منظور تحلیل های آماری شاخص ذیل برای محاسبه فیلد PIndex درخواست شده است، برنامه ای بنویسید که درخواست موردنظر را پاسخ دهد. (3/5)



$$PIndex = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n-1} \Delta t^2}$$

n = تعداد تحت تکفل هر شخص

Δt = اختلاف تاریخ تولد (به روز) هر تحت تکفل با تحت تکفل قبلی
براساس ترتیب تاریخ تولد افراد تحت تکفل

در نوشتن برنامه درخواستی می بایستی موارد ذیل مدنظر قرار گیرد:

الف) هر نوع تغییرات در جدول Child بطور اتوماتیک در محاسبه شاخص منعکس گردد.

ب) نوع داده های بکار رفته در برنامه می بایستی متناسب با محدوده ها و نوع فیلدهای جداول ایجاد گردد.

پ) برنامه طراحی شده می بایستی در بهینه ترین شکل پیاده سازی گردد.