دوره آموزشی
SQL Server 2016 Implementation

مدرس: سعید رجایی
آموزشگاه سماتک

SQL Server 2016 Implementation
آشنایی با سرویس های SQL Server

- SQL Server (MSSQLSERVER) (SSMS):
  - این سرویس به عنوان سرویس اصلی SQL Server شناخته می شود و به صورت کلی تمامی فعالیت ها و خدمات مربوط به بانکهای اطلاعاتی موجود در SQL Server

    به عهده این سرویس است
  - امکانات و سرویس های زیر مجموعه:
    - Database Engine
    - Service Broker
    - Notification Services
    - Replication
    - Full Text Search

- SQL Server Agent
  - این سرویس وظیفه اجرای فعالیت های زماندار، تکرار شونده و ابزار های مدیریتی است که DBA ها (مدیران بانک های اطلاعاتی) بر روی بانک های اطلاعاتی موجود تنظیم و تعریف می کنند

SQL Server 2016 Implementation
آشنایی با سرویس های SQL Server

- Analysis Service(SSAS):
  - از این سرویس به منظور Data Warehousing استفاده می شود. این سرویس ابزار هایی را در اختیار کاربر قرار می دهد تا بتواند این ذخیره سازی اطلاعات و همچنین داده کاوی هایی که بر روی این بانک های اطلاعاتی مخصوصا در برنامه ها واپلیکیشن های BI استفاده می شود را مدیریت نماید
- Reporting Service(SSRS):
  - این سرویس ابزار هایی را در اختیار کاربر قرار می دهد تا وی بتواند بر اساس آن گزارش های مورد نظر خود را با استفاده از یک Report Designer تولید و اجرا نماید
- Integration Service(SSIS):
  - از این سرویس برای جابجایی، استخراج، تجمیع اطلاعات در بانک های اطلاعاتی که با استفاده از تکنیک Data Warehousing تولید شده اند استفاده می شود

SQL Server 2016 Implementation

آشنایی با سرویس های SQL Server

- آشنایی با SQL Server Configuration Manager
- آشنایی با محیط (IDE) برنامه Management Studio
  - آشنایی با Object Browser
    - تعریف Connection و مدیریت آنها
    - آشنایی با System Databases
      - آشنایی با بانک اطلاعاتی master
      - آشنایی با بانک اطلاعاتی model
      - آشنایی با بانک اطلاعاتی msdb
      - آشنایی با بانک اطلاعاتی tempdb
    - آشنایی با Security و آیتم های دیگر
  - آشنایی با پنجره Property
  - معرفی ابزار های موجود در نوار ابزار Toolbar

Microsoft® SQL Server®

SQL Server 2016 Implementation

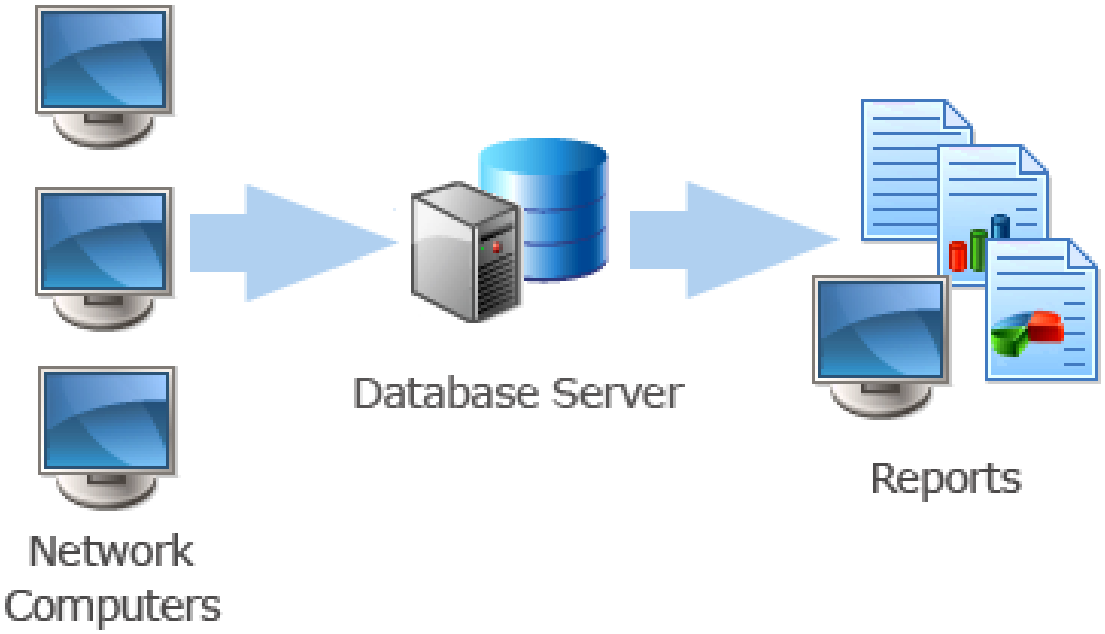
ایجاد بانک اطلاعاتی

- تعریف بانک اطلاعاتی
  - استفاده آنها با توجه به نوع نرم افزار ها:
    - به صورت متمرکز
    - به صورت توزیع شده
  - آشنایی با نحوه ایجاد بانک اطلاعاتی
    - آشنایی با فایل های بانک اطلاعاتی
      - Primary Group Rows Data(.mdf)
      - None Primary Group Rows Data(.ndf)
      - Transaction Log(.ldf)
    - آشنایی با File Group

SQL Server 2016 Implementation

ایجاد بانک اطلاعاتی

- بانک های اطلاعاتی متمرکز

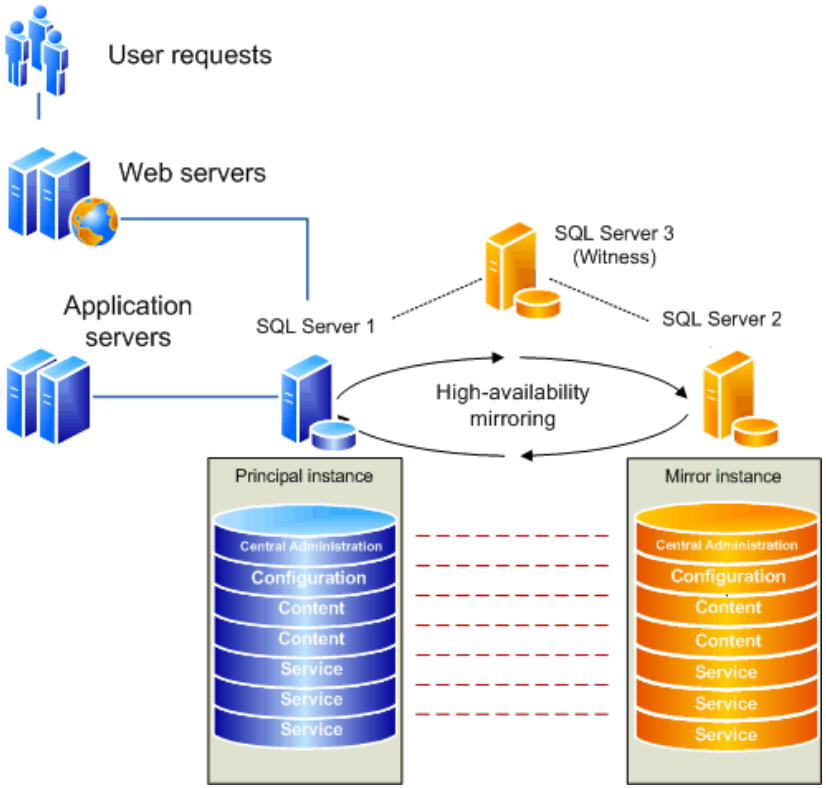


Microsoft SQL Server

SQL Server 2016 Implementation

ایجاد بانک اطلاعاتی

- بانک های اطلاعاتی توزیع شده

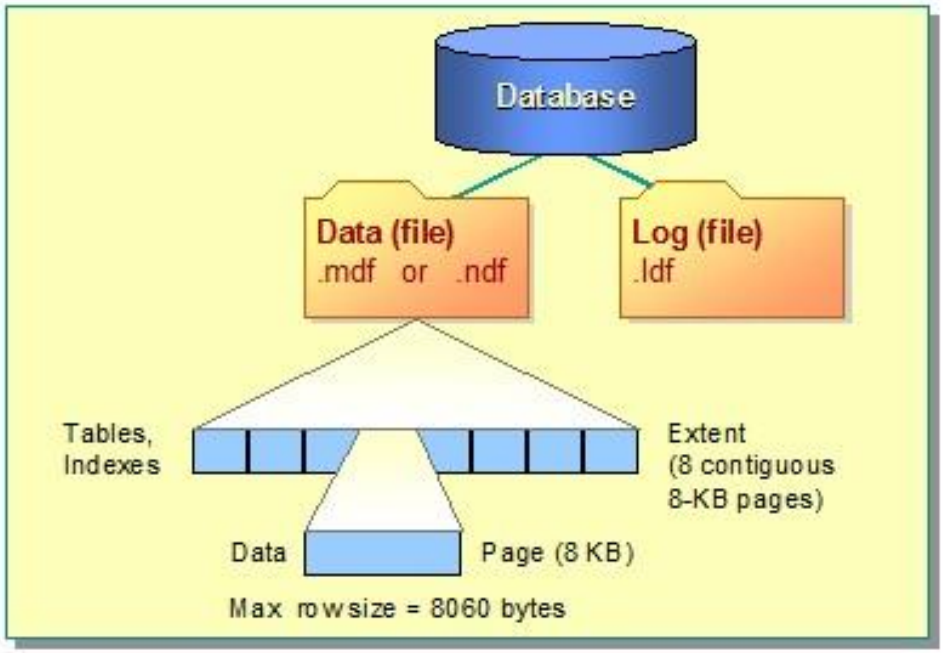


Microsoft SQL Server

SQL Server 2016 Implementation

ایجاد بانک اطلاعاتی

- فایل های بانک های اطلاعاتی

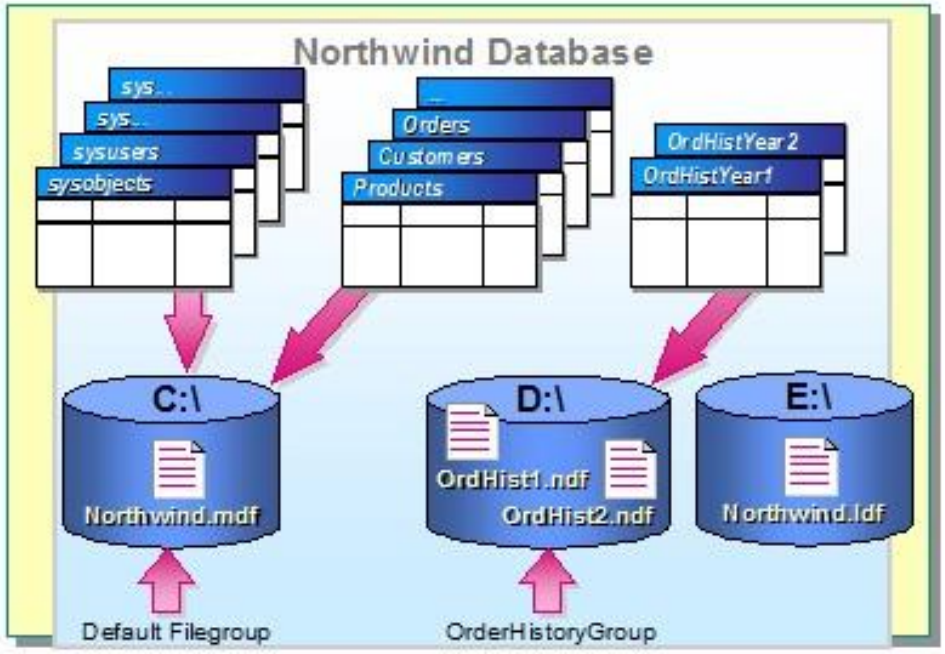


Microsoft SQL Server

SQL Server 2016 Implementation

ایجاد بانک اطلاعاتی

- استفاده از File Group ها



Microsoft® SQL Server®

SQL Server 2016 Implementation
جداول در SQL Server

- **آشنایی جداول و نحوه ایجاد آنها**
  - معرفی انواع جداول
  - معرفی ملحقات یک جدول
  - معرفی انواع ستون ها
  - معرفی مشخصات یک ستون
- **آشنایی با Data Type های استاندارد در SQL Server**

Microsoft® SQL Server®

SQL Server 2016 Implementation
جداول در SQL Server

- **تعریف جدول:**
  - **جدول به مجموعه ای از سطر ها و ستون ها گفته می شود که هدف آن ذخیره سازی اطلاعات می باشد**
    - هر جدول اختصاص به ذخیره سازی یک موجودیت یا Entity دارد
    - هر ردیف از جدول یک نمونه از موجودیت، و هر ستون یک خصوصیت و یا Property از آن آن موجودیت است

ستون
(Column)

سطر (ردیف)
(Row)

Country of origin of visitors	Number of visitors	Average length of stay (nights)
Italy	51 737	42
China	308 452	48
United States of America	456 084	24
United Kingdom	734 244	34
Canada	109 843	42
New Zealand	1 075 797	14

اطلاعات
Data

Microsoft® SQL Server®

SQL Server 2016 Implementation
جداول در SQL Server

- **جداول از دو نوع زیر تشکیل می شوند:**
  - **جداول سیستمی:**
    - این جداول حاوی اطلاعات سیستم ای است و کاربر نمی تواند به صورت مستقیم اطلاعات این جداول را تغییر دهد. از جمله جداول سیستم ای پر کاربرد، می توان به جداول: Sysdatabases, Syscolumns , SysObjects اشاره نمود. این جداول در بانک های اطلاعاتی سیستمی (از جمله master) قرار دارند
  - **جداول تعریف شده توسط کاربر:**
    - این دسته از جداول توسط کاربر برای ذخیره سازی اطلاعات ساخته می شود

Microsoft® SQL Server®

SQL Server 2016 Implementation
جداول در SQL Server

- **به طور کلی هر جدول از لحاظ ساختاری به اجزای زیر تقسیم می شوند:**
  - **ستون ها:**
    - بدنه اصلی جداول را تشکیل می دهند. هر ستون مشخص کننده یک خصوصیت از Entity و یا موجودیتی است که قرار است اطلاعات آن در جدول ذخیره گردد
  - **قید ها (Constraints):**
    - قیدها به منظور تضمین صحت اطلاعات مورد پذیرش برای ستون ها و جداول تعریف می شوند. به طور مشخص از قید ها برای جامعیت داده ها به روش Decretive استفاده می شود. شامل موارد زیر:
      - Allow Null
      - مقادیر پیش فرض (Defaults)
      - کلید اصلی (Primary Key)
      - کلید خارجی (Foreign Key)
      - کلید غیر تکراری (Unique Key)
      - کلید کنترلی (Check Constraint)

Microsoft® SQL Server®

SQL Server 2016 Implementation
جداول در SQL Server

- **جداول از دو نوع ستون زیر تشکیل می شوند:**
  - **ستون های عادی:**
    - این نوع ستون حاوی داده (Data) به ازای هر ردیف می باشد که توسط کاربر یا سیستم وارد می گردد.
  - **ستون های محاسباتی(Computed Column):**
    - داده های این ستون ها توسط کاربر وارد ویا ذخیره نمی شود (Read-only) و به طور معمول توسط فرمولی که برای آن مشخص می شود بر اساس داده ها و اطلاعات مابقی ستون ها در زمان لازم توسط SQL Server محاسبه می گردد.

Microsoft® SQL Server®

SQL Server 2016 Implementation
جداول در SQL Server

- خصوصیات مهم یک ستون :

Attribute	Description
Name	نام ستون
DataType	نوع داده
Length	طول نوع داده
Scale	تعداد ارقام اعشار
Precision	تعداد ارقام صحیح
Identity	شمارنده اتوماتیک
Allow Null	پذیرش مقدار Null
Default Value	مقدار پیش فرض

Microsoft SQL Server

SQL Server 2016 Implementation
جداول در SQL Server

- انواع داده های استاندارد:

Group	Types	Range & Storage
Exact Numeric (Fixed)	bigint	-2^63 (-9,223,372,036,854,775,808) through 2^63-1 (9,223,372,036,854,775,807). (8 Bytes)
	int	-2^31 (-2,147,483,648) through 2^31 - 1 (2,147,483,647) (4 Bytes)
	smallint	-2^15 (-32,768) through 2^15 - 1 (32,767). (2 Bytes)
	tinyint	0 through 255 (1 Byte)
	decimal	-10^38 +1 through 10^38 –1 (5 to 17 Byte Based on Precision Length)
	numeric	-10^38 +1 through 10^38 –1 (5 to 17 Byte Based on Precision Length)
	money	-2^63 (-922,337,203,685,477.5808) through 2^63 - 1 (+922,337,203,685,477.5807) (8 Bytes)
	smallmoney	-214,748.3648 through +214,748.3647 (4 Bytes)
	bit	1 or 0 (1 Byte)

Microsoft® SQL Server®

SQL Server 2016 Implementation
جداول در SQL Server

- انواع داده های استاندارد:

Group	Types	Range & Storage
Approximate Numeric	Float	-1.79E + 308 through -2.23E - 308, 0 and 2.23E + 308 through 1.79E + 308. 1 (4 to 8 Byte Based on Precision Length)
	real	-3.40E + 38 through -1.18E - 38, 0 and 1.18E - 38 through 3.40E + 38 (4 Byte)
Date and Time	datetime	January 1, 1753, through December 31, 9999 and time range 00:00:00 through 23:59:59.997 (8 Bytes)
	date	Just Date without time (3 Bytes)
	smalldatetime	January 1, 1900, through June 6, 2079 (4 Bytes)
	datetimeoffset	0001-01-01 through 9999-12-31 and time range 00:00:00 through 23:59:59.9999999 (10 Bytes)
	time	00:00:01 through 23:59:59 (5 Bytes)
Character Strings	char	maximum length of 8,000 characters
	varchar	maximum of 8,000 characters
	text	maximum length of 2^31 - 1 (2,147,483,647) characters
Unicode Character Strings	nchar	maximum length of 4,000 characters
	nvarchar	maximum length of 4,000 characters
	ntext	maximum length of 2^30 - 1 (1,073,741,823) characters
Binary Strings	binary	maximum length of 8,000 bytes
	varbinary	a maximum length of 8,000 bytes
	image	a maximum length of 2^31 - 1 (2,147,483,647) bytes

Microsoft® SQL Server®

SQL Server 2016 Implementation
جداول در SQL Server

- طراحی جدول ناشران:

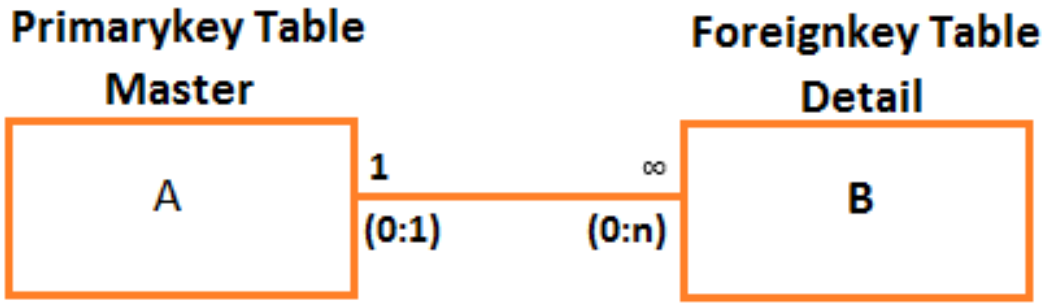
Name	Manager	Date of Publishing	Address	Tel	Fax	Web Site
Wrox	Natasha Parkin	21 Jul 1989	628 Desoto Avenue, Clarksdale, MS 38614, United States	(707) 827-7000	(707) 829-0104	www.wrox.com
MIcrosoft	Bill Gates	11 Feb 1972	1005 Gravenstein Highway North Sebastopol, CA 95472 USA	+1 (0) 1252 721284	+1 (0) 2552 721284	www.microfoft.com
O'Reily	Pitt Brakstone	23 Aug 1990	Suite 807, Building C, Cheng Ming Mansion, No.2 Xizhimen South Street, Xicheng District, Beijing, 100035 PRC	8610-88097475	8610-88097463	www.Oreily.com

Microsoft® SQL Server®

SQL Server 2016 Implementation

روابط جداول با یکدیگر

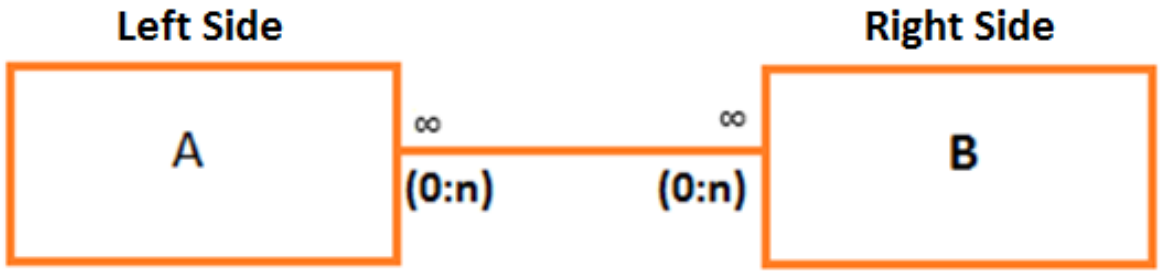
- **تعریف رابطه بین جداول**
  - هدف: ایجاد رابطه بین اطلاعاتی است که در هر جدول ذخیره می گردد
- **انواع رابطه ها:**
  - رابطه یک به چند (One –To-Many)
  - در این رابطه به ازای هر رکورد از جدول A صفر یا چند رکورد در جدول B وجود دارد و بالعکس، به ازای هر ردیف در جدول B صفر یا یک رکورد در جدول A وجود دارد. به شکل زیر توجه کنید:



SQL Server 2016 Implementation

روابط جداول با یکدیگر

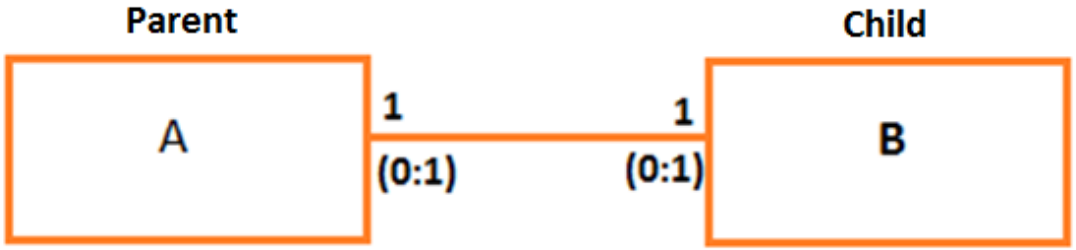
- **انواع رابطه ها:**
  - رابطه چند به چند (Many-To-Many)
  - در این رابطه به ازای هر رکورد از جدول A صفر یا چند رکورد در جدول B وجود دارد و بالعکس، به ازای هر ردیف در جدول B صفر یا چند رکورد در جدول A وجود دارد. به شکل زیر توجه کنید:



SQL Server 2016 Implementation

روابط جداول با یکدیگر

- **انواع رابطه ها:**
  - رابطه با یک به یک (One-To-One)
  - در این رابطه به ازای هر رکورد از جدول A صفر یا چند رکورد در جدول B وجود دارد و بالعکس، به ازای هر ردیف در جدول B صفر یا چند رکورد در جدول A وجود دارد. به شکل زیر توجه کنید:

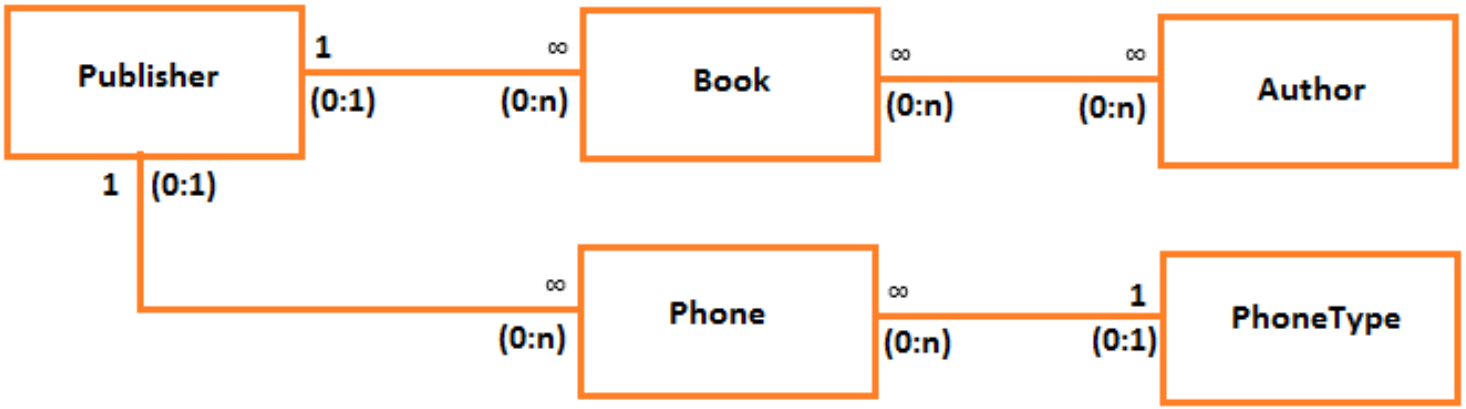


Microsoft® SQL Server®

SQL Server 2016 Implementation

روابط جداول با یکدیگر

- جمع بندی: تعیین رابطه ها در بانک اطلاعاتی نمونه (کتاب فروشی):

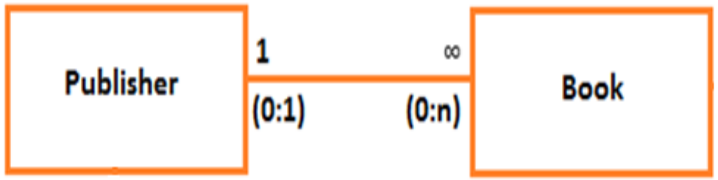


Microsoft SQL Server

SQL Server 2016 Implementation

روابط جداول با یکدیگر

- **مرحله سوم: تعیین رابطه بین جداول**
  - پیاده سازی رابطه یک به چند
    - مرحله اول: افزودن ستون به جدول Detail
      - افزودن ستون از جنس bigint و با ساختار نامگذاری:
        - <MasterTableName+ID>



Publisher	
P.K	ID
U.K	Title
	Address

Book	
P.K	ID
	Title
	Print Year
	Pages
U.K	ISBN
F.K	PublisherID

Microsoft® SQL Server®

SQL Server 2016 Implementation

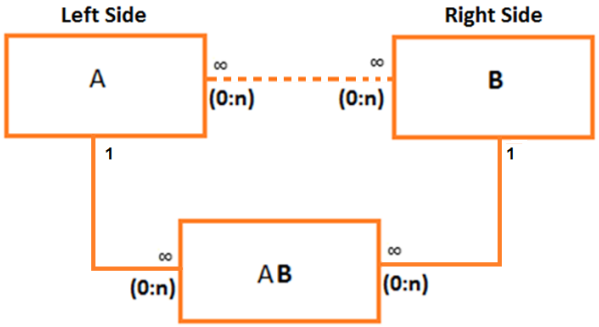
روابط جداول با یکدیگر

- **پیاده سازی رابطه یک به چند**
  - مرحله دوم: افزودن Foreign key به جدول Detail
  - نحوه تعریف: با استفاده از امکان Database Diagram در Management Studio
    - مدیریت ارتباط:
      - INSERT and UPDATE Specification

- Delete Rule
  - No Action, Cascade, Set Null, Set Default
- Update Rule
  - No Action, Cascade, Set Null, Set Default

پیاده سازی رابطه چند به چند

- امکان پیاده سازی رابطه چند به چند به صورت مستقیم بین دو جدول وجود ندارد. برای این منظور از یک جدول کمکی (که در اصطلاح به آن Junction Table نیز گفته می شود) استفاده می نماییم
- برای این منظور:
  - جدول جدیدی با ترکیب نام دو جدول اصلی ایجاد می نماییم
  - به ازای هر یک از جداول اولیه، ستون Foreign key در جدول جدید قرار می
  - ترکیب دو ستون Foreign key ایجاد شده را به عنوان Primary key تعری



Book	
P.K	ID
	Name

Author	
P.K	ID
	Name

BookAuthor		
P.K	F.K	BookID
	F.K	AuthorID

SQL Server 2016 Implementation

روابط جداول با یکدیگر

- **پیاده سازی یک به یک**
  - از دو روش زیر می توانیم برای تعریف این رابطه استفاده نماییم
  - روش اول: Foreignkey Approach
  - روش دوم: Cross Reference
- **روش اول:**
  - در این روش مانند پیاده سازی رابطه یک به چند عمل می شود. به این منظور که در جدول Child یک Foreign key اضافه می نماییم تنها با این تفاوت که یک قید از نوع U.K بر روی آن قرار می دهیم تا از تکراری شدن ردیف ها جلوگیری نماییم.

Parent	
P.K	ID
	Title

۱ ∞

Child		
P.K		ID
U.K	F.K	ClassID

SQL Server 2016 Implementation

روابط جداول با یکدیگر

- **پیاده سازی یک به یک**
- **روش دوم:**
  - در این روش نیازی به ایجاد ستون کلید خارجی نیست، تنها ستون ID جدول Child از حالت Identity خارج شده (البته هنوز Primary key باید باشد) و با ارتباط دادن Primary key ها این نوع ارتباط ایجاد می گردد.

Parent			Child		
P.K	ID	۱ ۱	F.K	P.K	ID(is not Identity)
	Title				

Microsoft® SQL Server®

SQL Server 2016 Implementation

واکشی اطلاعات از جداول

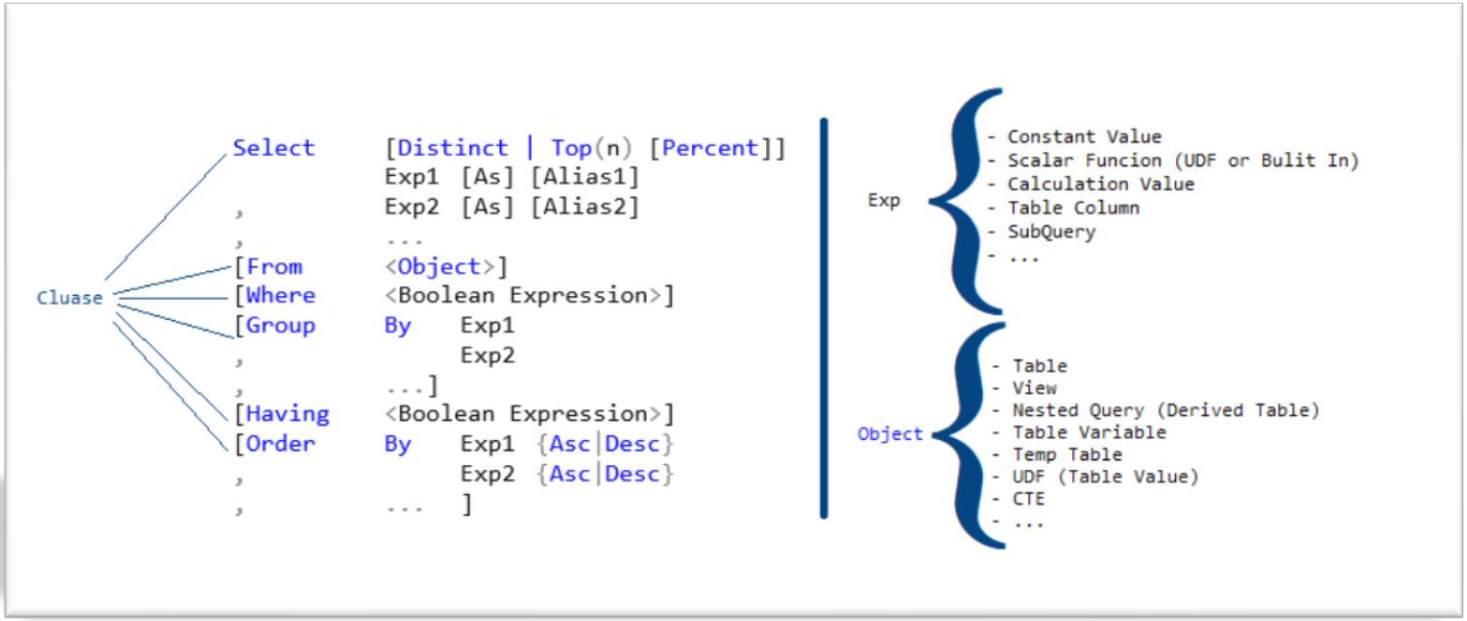
- **واکشی اطلاعات از جداول**
- **دستور Select**
  - قالب دستور
  - محدود سازی اطلاعات فراخوانی شده (فیلتر کردن اطلاعات)
  - تخصیص نام مستعار به فیلد ها و جداول
  - معرفی Distinct
  - آشنایی با گذاره Top

Microsoft® SQL Server®

SQL Server 2016 Implementation

واکشی اطلاعات از جداول

- واکشی اطلاعات از جداول
- قالب دستور **Select**



Microsoft® SQL Server®

SQL Server 2016 Implementation

واکشی اطلاعات از جداول

- **واکشی اطلاعات از جداول**
- چنانچه بخواهید تمامی ستون های یک جدول را در دستور Select انتخاب نمایید می توانید از کارکتر * استفاده نمایید. در غیر اینصورت می بایست نام ستون ها با استفاده از جداکننده کاما (,) در دستور آورده شود
- در کنار جمله Select از جملات دیگر نیز استفاده می شود که به آنها Clouse گفته می شود
- دستور From و ملحقات آن یک Clouse برای دستور Select به حساب می آید
- جملات یاد شده می تواند به یک دستور Select کم یا اضافه شوند
- ترتیب نگارش آنها مهم است

SQL Server 2016 Implementation

واکشی اطلاعات از جداول

- **واکشی اطلاعات از جداول**
  - **محدود سازی اطلاعات فراخوانی شده (فیلتر کردن اطلاعات)**
  - **قالب دستور:**

```
Select      [*] |
            [<Expr1>],
            [<Expr2>],
            …
            [<Expr n>],
            [From       <object>]
            [Where      Boolean Expression]
```

- به دلیل استفاده از گذاره های منطقی در دستور Where به منظور ترکیب آنها می بایست از عملگر های منطقی And و Or استفاده نمود

Microsoft® SQL Server®

SQL Server 2016 Implementation

واکشی اطلاعات از جداول

- واکشی اطلاعات از جداول
  - اپراتور های جهت استفاده در Where
    - اپراتور های مقایسه ای

Operator	Description	Sample
=	برابر	Where ID = 10
>	بزرگتر از	Where ID >200
<	کوچکتر از	Where ID <963
>=	بزرگتر مساوی از	Where ID >= 10
<=	کوچکتر مساوی از	Where ID <= 100
<>	مخالف	Where ID <> 10
!=	مخالف	Where ID != 10
!<	کوچکتر نباشد	Where ID !< 1000
!>	بزرگتر نباشد	Where ID !> 1000

Microsoft® SQL Server®

SQL Server 2016 Implementation

واکشی اطلاعات از جداول

- **واکشی اطلاعات از جداول**
  - **اپراتور های جهت استفاده در Where**
    - **اپراتور های منطقی**

Operator	Description
And	بین دو گذاره منطقی، زمانی True است که هر دو True باشند
Between	مقدار آن زمانی True است که مقدار در بین محدوده داده آن باشد
Exists	مقدار آن زمانی True است که اطلاعاتی بر اساس Select موجود باشد
IN	زمانی True است که مقدار داده شده در بین مجموعه داده شده باشد
Like	با استفاده از کارکتر های خاص زمانی True می شود که شرط آن برقرار باشد
Not	به منظور معکوس کردن یک عبارت منطقی استفاده می شود
Or	بین دو گذاره منطقی، زمانی True است که یکی از آنها True باشند

Microsoft® SQL Server®

SQL Server 2016 Implementation

واکشی اطلاعات از جداول

- **واکشی اطلاعات از جداول**
  - **استفاده از نام های مستعار در ستون ها و جداول**

```
Select      [<Column1>] [Expression1] As [<YourName1>],
            [<Column1>] [Expression2]  As [<YourName2>],
            …
            [<ColumnN>] [ExpressionN]       As [<YourNameN>],
            [From       <TableName> <ViewName> <TableValue UDF> <NestedQuery>]
      As [<YourTableName>]
            [Where      Boolean Expression]
```

- از این تکنیک برای خوانایی بیشتر دستورات نوشته شده همچنین جهت نامگذاری در ستون های محاسباتی و ترکیبی استفاده می شود

Microsoft® SQL Server®

SQL Server 2016 Implementation

واکشی اطلاعات از جداول

- **واکشی اطلاعات از جداول**
  - **معرفی دستور Distinct**

```
Select      [Distinct][*] |
            [<Column1>] [Expression1],
            [<Column1>] [Expression2],
            …
            [<ColumnN>] [ExpressionN],
            [From      <TableName> <ViewName> <TableValue UDF> <NestedQuery>]
            [Where     Boolean Expression]
```

- این دستور، اطلاعات تکراری هر سطر که ترکیب اطلاعاتی که در ستون ها انتخاب می شود را حذف می نماید

SQL Server 2016 Implementation

واکشی اطلاعات از جداول

- **واکشی اطلاعات از جداول**
  - **آشنایی با گذاره Top**

```
Select[Top {n | m Percent}][*]
        [<Column1>] [Expression1],
        [<Column1>] [Expression2],
        …
        [<ColumnN>] [ExpressionN],
        [From <TableName> <ViewName> <TableValue UDF><NestedQuery>]
        [Where      Boolean Expression]
```

- از این دستور، بیشتر در کنار دستور Order By که در جلسات بعدی به آن اشاره می شود استفاده می شود
- به صورت تعدادی و درصدی ردیف های اطلاعاتی را انتخاب می کند
- موارد کاربرد
  - انتخاب ردیف های تصادفی
  - به منظور Paging اطلاعات
  - انتخاب ترین ها

Microsoft® SQL Server®

SQL Server 2016 Implementation

واکشی اطلاعات از جداول

- واکشی اطلاعات از جداول (امکانات پیشرفته جهت فیلتر کردن اطلاعات)
- آشنایی با دستور Case و استفاده از آن
- آشنایی با اپراتور Like
- آشنایی با اپراتور های In و Not In
- آشنایی با اپراتور Exists
- آشنایی با اپراتور Union

Microsoft® SQL Server®

SQL Server 2016 Implementation

واکشی اطلاعات از جداول

- **واکشی اطلاعات از جداول (امکانات پیشرفته جهت فیلتر کردن اطلاعات)**
  - **آشنایی با دستور Case و استفاده از آن**
  - **قالب دستور اول Simple Case:**

```
Case    [Expression]
        When <value1>   Then   [Expression]
        When <value2>   Then   [Expression]
        …
        [Else]  [Expression]
End
```

  - **قالب دستور دومSearch Case:**

```
Case
        When [Boolean Expression]   Then    [Expression]
        When [Boolean Expression]   Then    [Expression]
        …
        [Else]  [Expression]
End
```

Microsoft SQL Server

SQL Server 2016 Implementation

واکشی اطلاعات از جداول

- واکشی اطلاعات از جداول (امکانات پیشرفته جهت فیلتر کردن اطلاعات)
  - آشنایی با اپراتور **Like**
  - از این اپراتور برای جستجو در اطلاعات از جنس رشته استفاده می شود
  - همراه اپراتور **Like** می توان از کارکترهای خاصی به منظور انعطاف در عملیات جستجو طبق جدول زیر استفاده نمود:

Character	Description	Sample
%	هر چیزی با طول ۰ تا n	FirstName Like 'Jo%' FirstName Like '%on' FirstName Like '%Jo%'
_	هر چیزی با طول ۱	FirstName Like '_ra%' FirstName Like '__r%'
[مجموعه]	مجموعه	FirstName Like '[rt]%'
[کران بالا – کران پایین]	کران	FirstName Like '[A-Z]%' FirstName Like '[0-6]%'
[[]	[	FirstName Like '%[-]%'
[%]	%	
[_]	_	
]	]	

Microsoft® SQL Server®

SQL Server 2016 Implementation

واکشی اطلاعات از جداول

- **واکشی اطلاعات از جداول (امکانات پیشرفته جهت فیلتر کردن اطلاعات)**
- **آشنایی با اپراتور های In و Not In:**
  - **از اپراتور In جهت کنترل وجود یک مقدار در یک مجموعه استفاده می شود و مشخصا در دستورات Where می بایست از آنها استفاده نمود**
  - **از اپراتور Not In به منظور کنترل عدم وجود یک مقدار در یک مجموعه استفاده می شود**
  - **مجموعه قابل جستجو میتواند به صورت Static و یا Dynamic تولید شود. برای تولید مجموعه های Dynamic باید از Sub Query ها استفاده نمود که در جلسات بعدی به آن اشاره می شود**
  - **قالب دستور:**

عبارت محاساباتی In (مجموعه)

- **مثال:**

Code In (‘001’ , ‘1235’ , ‘0203’)

ID Not In (12 , 14 , 192, 2369)

Microsoft® SQL Server®

SQL Server 2016 Implementation

واکشی اطلاعات از جداول

- واکشی اطلاعات از جداول (امکانات پیشرفته جهت فیلتر کردن اطلاعات)
- آشنایی با اپراتور **Exists**:
  - از این اپراتور جهت کنترل وجود یک رکورد (ردیف اطلاعاتی) استفاده می شود.
  - این اپراتور یک دستور **Select** می پذیرد و وجود ردیف (براساس شرایط آن **Select**) را مورد بررسی قرار می دهد
  - در صورت وجود رکورد از نتیجه **Select** مقدار **True** و در غیر اینصورت مقدار **False** را بر می گرداند
  - قالب دستور:

```
Exists (Select Statement)
```

Microsoft® SQL Server®

SQL Server 2016 Implementation

واکشی اطلاعات از جداول

- **واکشی اطلاعات از جداول**
  - **آشنایی با اپراتور Except | Intersect | Union**
  - از Union جهت تجمیع چند دستور Select و نمایش آن در یک خروجی استفاده می شود
  - **قالب دستور**

```
Select Statement
Union | Intersect | Except
Select Statement
```

- اپراتور Union به صورت درونی رکورد هایی که همه اطلاعات آنها با هم یکسان است را حذف می نماید. چنانچه تمایل دارید این رکورد ها حذف نشوند باید از دستور Union All استفاده نمایید
- تعداد ستون ها همچنین جنس و ترتیب ستون ها در Select ها می بایست یکسان باشد
- نام ستون ها (در خروجی نهایی) از روی نام ستون های دستور اول برداشته می شود
- از دستور Intersect برای مشترکات، و از Except برای خارج کردن مشترکات دستور دوم از اول استفاده می شود

Microsoft® SQL Server®

SQL Server 2016 Implementation

توابع کاربردی در SQL

- **توابع کاربردی کار با اطلاعات**
  - **توابع Aggregate**
    - COUNT: شمارنده تعداد رکورد های یک جدول
    - MIN: ورودی از جنس ستون (معمولا غیر رشته ای)، برای استخراج مقدار کمینه آن ستون
    - MAX: ورودی از جنس ستون (معمولا غیر رشته ای)، برای استخراج مقدار بیشینه آن ستون
    - SUM: ورودی از جنس ستون (معمولا عددی)، برای استخراج مقدار جمع همه مقادیر ستون
    - AVG: ورودی از جنس ستون (معمولا عددی)، برای استخراج مقدار میانگین ستون

SQL Server 2016 Implementation
توابع کاربردی در SQL

- **توابع کاربردی کار با اطلاعات**
- **توابع Scalar – توابع تاریخی**
  - توابع دریافت تاریخ کامل

Function	Syntax	Return value	Return data type	Determinism
SYSDATETIME	SYSDATETIME ()	Returns a **datetime2(7)** value that contains the date and time of the computer on which the instance of SQL Server is running. The time zone offset is not included.	**datetime2(7)**	Nondeterministic
SYSDATETIMEOFFSET	SYSDATETIMEOFFSET ( )	Returns a **datetimeoffset(7)** value that contains the date and time of the computer on which the instance of SQL Server is running. The time zone offset is included.	**datetimeoffset(7)**	Nondeterministic
SYSUTCDATETIME	SYSUTCDATETIME ( )	Returns a **datetime2(7)** value that contains the date and time of the computer on which the instance of SQL Server is running. The date and time is returned as UTC time (Coordinated Universal Time).	**datetime2(7)**	Nondeterministic

Microsoft® SQL Server®

SQL Server 2016 Implementation

توابع کاربردی در SQL

- **توابع کاربردی کار با اطلاعات**
- **توابع Scalar – توابع تاریخی**
- توابع دریافت تاریخ خلاصه شده

Function	Syntax	Return value	Return data type	Determinism
CURRENT_TIMESTAMP	CURRENT_TIMESTAMP	Returns a **datetime** value that contains the date and time of the computer on which the instance of SQL Server is running. The time zone offset is not included.	**datetime**	Nondeterministic
GETDATE	GETDATE ( )	Returns a **datetime** value that contains the date and time of the computer on which the instance of SQL Server is running. The time zone offset is not included.	**datetime**	Nondeterministic
GETUTCDATE	GETUTCDATE ( )	Returns a **datetime** value that contains the date and time of the computer on which the instance of SQL Server is running. The date and time is returned as UTC time (Coordinated Universal Time).	**datetime**	Nondeterministic

Microsoft® SQL Server®

SQL Server 2016 Implementation

توابع کاربردی در SQL

- **توابع کاربردی کار با اطلاعات**
- **توابع Scalar – توابع تاریخی**
  - توابع دریافت قسمتی از تاریخ

Function	Syntax	Return value	Return data type	Determinism
DATENAME	DATENAME (*datepart* , *date*)	Returns a character string that represents the specified *datepart* of the specified date.	**nvarchar**	Nondeterministic
DATEPART	DATEPART (*datepart* , *date*)	Returns an integer that represents the specified *datepart* of the specified *date*.	**int**	Nondeterministic
DAY	DAY (*date*)	Returns an integer that represents the day day part of the specified *date*.	**int**	Deterministic
MONTH	MONTH (*date*)	Returns an integer that represents the month part of a specified *date*.	**int**	Deterministic
YEAR	YEAR (*date*)	Returns an integer that represents the year part of a specified *date*.	**int**	Deterministic

Microsoft® SQL Server®

SQL Server 2016 Implementation

توابع کاربردی در SQL

- **توابع کاربردی کار با اطلاعات**
- **توابع Scalar – توابع تاریخی**
- توابع برای دریافت اختلاف دو تاریخ

Function	Syntax	Return value	Return data type	Determinism
DATEDIFF	DATEDIFF (*datepart* , *startdate* , *enddate*)	Returns the number of date or time *datepart* boundaries that are crossed between two specified dates.	**int**	Deterministic

Microsoft® SQL Server®

SQL Server 2016 Implementation

توابع کاربردی در SQL

- **توابع کاربردی کار با اطلاعات**
  - **توابع Scalar – توابع تاریخی**
    - توابع برای تغییرات در تاریخ

Function	Syntax	Return value	Return data type	Determinism
DATEADD	DATEADD (*datepart* , *number* , *date*)	Returns a new **datetime** value by adding an interval to the specified *datepart* of the specified *date*.	The data type of the *date* argument	Deterministic

    - تابع برای بررسی صحت تاریخ

Function	Syntax	Return value	Return data type	Determinism
ISDATE	ISDATE (*expression*)	Determines whether a **datetime** or **smalldatetime** input expression is a valid date or time value.	**int**	ISDATE is deterministic only if you use it with the CONVERT function, when the CONVERT style parameter is specified, and when style is not equal to 0, 100, 9, or 109.

- **توابع کاربردی کار با اطلاعات**
- **توابع Scalar – توابع منطقی**
- تابع Choose: ساختار دستوری این تابع به شکل زیر است

Syntax

```
CHOOSE ( index, val_1, val_2 [, val_n ] )
```

- نکته: از این تابع می توان بجای دستور Case در مواردی که از مقادیر ثابت در آنها استفاده می شود استفاده نمود و همچنین مقدار Index از عدد یک شروع خواهد شد
- تابع IIF: به منظور کنترل یک شرط به صورت زیر استفاده می شود

Syntax

```
IIF ( boolean_expression, true_value, false_value )
```

Microsoft® SQL Server®

SQL Server 2016 Implementation

توابع کاربردی در SQL

- توابع کاربردی کار با اطلاعات
- توابع **Scalar** – توابع ریاضی

ABS	DEGREES	RAND
ACOS	EXP	ROUND
ASIN	FLOOR	SIGN
ATAN	LOG	SIN
ATN2	LOG10	SQRT
CEILING	PI	SQUARE
COS	POWER	TAN
COT	RADIANS	

Microsoft® SQL Server®

SQL Server 2016 Implementation

توابع کاربردی در SQL

- **توابع کاربردی کار با اطلاعات**
- **توابع رشته ای**

ASCII	CHAR	CHARINDEX
CONCAT	CONCAT_WS	DIFFERENCE
FORMAT	LEFT	LEN
LOWER	LTRIM	NCHAR
PATINDEX	QUOTENAME	REPLACE
REPLICATE	REVERSE	RIGHT
RTRIM	SOUNDEX	SPACE
STR	STRING_AGG	STRING_ESCAPE
STRING_SPLIT	STUFF	SUBSTRING
TRANSLATE	TRIM	UNICODE

Microsoft® SQL Server®

SQL Server 2016 Implementation

توابع کاربردی در SQL

- **توابع کاربردی کار با اطلاعات**
- **توابع تبدیل نوع داده ها**
  - تابع CAST
  - تابع CONVERT

Syntax

```
Syntax for CAST:
CAST ( expression AS data_type [ ( length ) ] )
```

```
Syntax for CONVERT:
CONVERT ( data_type [ ( length ) ] , expression [ , style ] )
```

Microsoft® SQL Server®

SQL Server 2016 Implementation

مرتب سازی و گروه بندی اطلاعات

- **مرتب سازی و گروه بندی اطلاعات**
- **مرتب سازی اطلاعات**
  - معرفی دستور Order By
  - ترکیب استفاده آن با دستور Top
  - بررسی نکات استفاده
- **دسته بندی گروه بندی اطلاعات**
  - معرفی دستور Group By
  - استفاده از Aggregate Function ها در گروه بندی
  - استفاده از دستور Having (فیلتر کردن اطلاعات بر اساس توابع Aggregate)

- **مرتب سازی و گروه بندی اطلاعات**
- **مرتب سازی اطلاعات**
  - **معرفی دستور Order By**

```
Order By    <Expr1> {Asc,Desc},
            <Expr2> {Asc,Desc},
...
```

- **نکات:**
  - از عناوین مستعار ستون ها می توان در دستور Order By استفاده نمود(برعکس دستور Where)
  - اگر تنها قصد مرتب سازی ستون ها را داشته باشیم، می توانیم بجای نام ستون ها از اندیس آنها (براساس آنچه در دستور Select نوشته ایم) استفاده نماییم
  - در دستورات Select ای که از Union استفاده شده است نمی توانیم مستقیما از دستور Order By استفاده نماییم (به دلیل اینکه خود دستور Union مرتب سازی را به صورت درونی انجام می دهد) و برای این کار باید حتما از Nested Query ها که در آینده به آنها اشاره خواهیم کرد استفاده نمود
  - در زمانی که دستوراتی با استفاده از ترکیب دو دستور Top و Order By به منظور استخراج اطلاعات تولید می نماییم، رکورد های هم مرتبه حذف خواهند شد برای جلوگیری از این اتفاق می توان از گزینه With Ties در دستور Top استفاده نمود

Microsoft® SQL Server®

SQL Server 2016 Implementation

مرتب سازی و گروه بندی اطلاعات

- **مرتب سازی و گروه بندی اطلاعات**
- **گروه بندی اطلاعات**
  - معرفی دستور Group By

```
Group By  <Expr1>, <Expr2>,...
```

- **نکات:**
  - از این دستور می بایست بعد از دستور Where استفاده شود
  - استفاده مفرد و تنها از توابع Aggregate در دستورات امکان پذیر است اما در صورتیکه بخواهیم هر یک از مقادیر ستون های دیگر را در کنار این توابع برای نمایش خروجی بهتر استفاده نماییم، می بایست از این دستور استفاده نماییم
  - در صورتیکه بخواهیم از توابع Aggregate جهت فیلتر کردن اطلاعات استفاده کنیم آنها را می بایست با استفاده از دستور Having انجام دهیم

استفاده از With Cube و With Rollup در دستور Group By

- با استفاده از دستور With Cube می توانید تمامی حالات ممکن در دستور Group By را پیاده سازی کنیم. با اجرای این دستور، دسته بندی ها بر اساس حالات ممکن، و حالت تجمعی نمایش داده می شود. در این دستور بر اساس تمامی ستون هایی که در Group Byنوشته می شود حالت تجمعی آورده می شود

```
Select  Publisher.Title         As PublisherName
,       Subject.Name            As SubjectName
,       Count(Book.ID)          As BookCount
From    Subject
Left Outer Join
        Book
On      Subject.ID = Book.SubjectID
Inner Join
        Publisher
On      Publisher.ID = Book.PublisherID
Group By        Publisher.Title
,               Subject.Name
With Cube
```

استفاده از With Cube و With Rollup در دستور Group By

- با استفاده از دستور With Rollup می توانید تمامی حالات ممکن در دستور Group By را پیاده سازی کنیم. با اجرای این دستور، دسته بندی ها بر اساس حالات ممکن، و حالت تجمعی نمایش داده می شود. در این دستور بر اساس ستون اول در دستور که در Group Byنوشته می شود حالت تجمعی آورده می شود

```
Select  Publisher.Title         As PublisherName
,       Subject.Name            As SubjectName
,       Count(Book.ID)          As BookCount
From    Subject
Left Outer Join
        Book
On      Subject.ID = Book.SubjectID
Inner Join
        Publisher
On      Publisher.ID = Book.PublisherID
Group By        Publisher.Title
,               Subject.Name
With Rollup
```

SQL Server 2016 Implementation

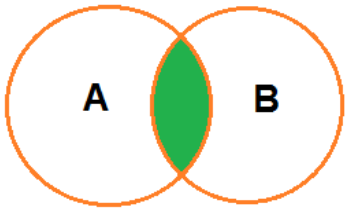
مرتب سازی و گروه بندی اطلاعات

- **واکشی اطلاعات از چندین جدول (تلفیق جداول)**
- **معرفی Join و انواع آن**
  - دسته اول:Inner Join
  - دسته دوم: Outer Join
  - دسته سوم: Cross Join
  - دسته چهارم: Cross Apply
  -
- **معرفی Inner Join**
- **معرفی Outer Join**
  - Left Outer Join
  - Right Outer Join
  - Full Outer Join
- **حل چند مسئله**

- **واکشی اطلاعات از چندین جدول (تلفیق جداول)**
- **معرفی Join و انواع آن:**
  - به ایجاد رابطه منطقی بین رکورد های دو جدول در اصطلاحJoin گفته می شود. دستورات Join و انواع آنها در عبارت مقابل From می آیند.
- **معرفی Inner Join**
  - قالب دستور

```
From <obj1>      Inner Join  <obj2>
On     <Boolean Expression>
```

  - شرح: تمامی ردیف های شیئ اول که در شیئ دوم رکورد متناظر داشته باشند در خروجی ظاهر می شوند
  - عبارت منطقی: به طور کلی در عبارت منطقی Join ها رابطه بین Primary key جدول Master با Foreign key جدول Detail برقرار می گردد
  - نتیجه نموداری



Microsoft SQL Server

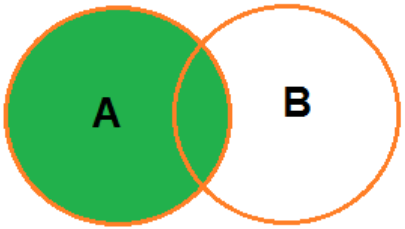
SQL Server 2016 Implementation

واکشی اطلاعات از چندین جدول

- **واکشی اطلاعات از چندین جدول (تلفیق جداول)**
- **معرفی Outer Join**
  - Left Outer Join
    - **قالب دستور**

```
From <obj1> Left Outer Join  <Obj2>
On     <Boolean Expression>
```

- شرح: تمامی ردیف های شیئ اول که در شیئ دوم رکورد متناظر داشته باشند همچنین تمامی ردیف های شیئ اول که رکورد متناظر در شیئ دوم نداشته باشند نیز در خروجی ظاهر می شوند
- نتیجه نموداری



Microsoft SQL Server

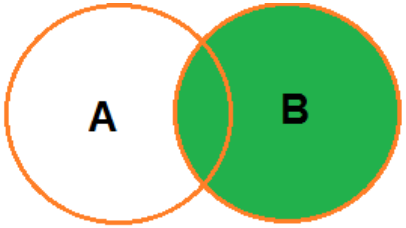
SQL Server 2016 Implementation

واکشی اطلاعات از چندین جدول

- **واکشی اطلاعات از چندین جدول (تلفیق جداول)**
- **معرفی Outer Join**
  - Right Outer Join
    - **قالب دستور**

```
From <Obj1> Right  Outer Join  <Obj2>
On     <Boolean Expression>
```

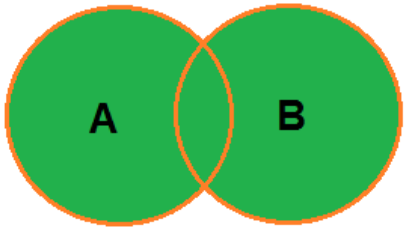
- شرح: تمامی ردیف های شیئ اول که در شیئ دوم رکورد متناظر داشته باشند همچنین تمامی ردیف های شیئ دوم که رکورد متناظر در شیئ اول نداشته باشند نیز در خروجی ظاهر می شوند
- نتیجه نموداری



- **واکشی اطلاعات از چندین جدول (تلفیق جداول)**
- **معرفی Outer Join**
  - Full Outer Join
    - **قالب دستور**

```
From <Obj1>  Full  Outer Join  <Obj2>
On     <Boolean Expression>
```

- شرح: تمامی ردیف های شیئ اول که در شیئ دوم رکورد متناظر داشته باشند همچنین تمامی ردیف های شیئ اول و دوم که رکورد متناظر ها در شیاء دیگر وجود نداشته باشند نیز در خروجی ظاهر می شوند
- نتیجه نموداری



SQL Server 2016 Implementation

واکشی اطلاعات از چندین جدول

- **واکشی اطلاعات از چندین جدول**
- **معرفی Cross Join**
- **واکشی اطلاعات از چندین جدول**
- **معرفی Sub Query**
  - استفاده آنها در دستورات Select
  - استفاده آنها در Where Clause ها
  - استفاده از آنها به عنوان منعکس کننده حاصل های توابع تجمعی
- **معرفی Nested Query ها**
- **معرفی CTE**

SQL Server 2016 Implementation

واکشی اطلاعات از چندین جدول

- **واکشی اطلاعات از چندین جدول**
- **معرفی Cross Join**
  - از این Join به منظور برقراری ارتباط بین دو جدولی که هیچ ارتباط مستقیمی با یکدیگر ندارند استفاده می شود. در عمل خروجی یک دستور Cross Join ضرب دکارتی و ترکیب ردیف های هر دو جدول است
  - **قالب دستور**

```
From <Obj1> Cross Join  <Obj2>
```

  - **نکته:**
    - این دستور شرط on و عبارت منطقی ندارد
    - با شروطی که در دستور Where برای این تلفیق میتوان قرار داد قادر به پیاده سازی تمامی انواع Join های گذشته خواهیم بود
    - از این دستور بیشتر برای استخراج عدم وجود رابطه بین اطلاعات استفاده می شود

Microsoft® SQL Server®

SQL Server 2016 Implementation

واکشی اطلاعات از چندین جدول

- **واکشی اطلاعات از چندین جدول**
  - **واکشی اطلاعات از چندین جدول (بیش از دو جدول)**
  - قالب دستور ثابت است و می توان بیش از دو جدول را نیز با همان قالب دستورات مطرح شده در تلفیق ها شرکت داد
  - در استفاده از تلفیق جداول برای تولید خروجی، محدودیت هایی وجود دارد که بر اساس Resource های مورد استفاده مورد بررسی قرار داده می شوند. این موضوع در ورژن های مختلف SQL Server با یکدیگر متفاوت است. اما به صورت کلی معمولا تا ۳۲ جدول می توانند به صورت عادی در عملیات تلفیق شرکت کنند
  - آنچه که در استفاده از تلفیق های متعدد مورد اهمیت است نوع شناخت از خروجی جداول و بررسی بیشتر صورت مسئله می باشد.

Microsoft® SQL Server®

SQL Server 2016 Implementation

واکشی اطلاعات از چندین جدول

- **معرفی Sub Query**
  - Sub Query یک دستور Select است که یک مقدار Scalar را بر می گرداند(قابلیت نمایش اطلاعات یک ستون). عملا مانند یک ستون عمل می نماید
  - از Sub Query ها می توان مانند یک ستون در دستورات Select استفاده نمود. در بعضی مواقع می توان انتخاب ستون ها با این دستور را جایگزین نوشتن Join ها بین جداول نمود.

```
Select<Column1> ,
       <Column2>,
       (Select …)      As Column3,
       …
From …
```

  - در استفاده از Sub Query ها در دستورات Select به عنوان یک ستون، باید به این نکته توجه داشت که به ازای هر رکورد مقدار خروجی Sub Query هم باید یک رکورد را در خروجی نمایش دهد
  - از این دستور می توان در شرط Where نیز استفاده نمود. استفاده بارز آن را می توان برای تولید خروجی داینامیک در استفاده آنها در گذاره In مشاهده نمود
  - از این دستور می توان برای منعکس کردن خروجی توابع Aggregation در دستورات Select استفاده نمود.

Microsoft® SQL Server®

SQL Server 2016 Implementation

واکشی اطلاعات از چندین جدول

- **معرفی Nested Query**
  - برخلاف Sub Query ها، خروجی Nested Query (جستجو های تو در تو) می تواند حاوی چندین ستون باشد
  - معمولا برای تولید جداول سفارشی برای شرکت در تلفیق ها از این دستور استفاده می شود
  - برای تهیه خروجی های خاص از روی لیست ای از ستون ها نیز از این دستور استفاده می نمایند.
  - برای استفاده از Nested Query ها در دستورات، باید برای آنها نام مستعار در نظر گرفته شود
  - ستون هایی که در Nested Query ها نوشته می شود باید دارای نام باشند و چنانچه از دستورات خاصی برای استخراج ستون در Nested Query استفاده می نمایید برای آنها باید از نام های مستعار لحاظ نمایید

SQL Server 2016 Implementation

استفاده از CTE

- استفاده از **CTE (Common Table Expression)**
- تعریف: **CTE** یک جدول مجازی است که می توان برای تهیه با استفاده از آن اطلاعات سلسله مراتبی و یا جستجو های تو در تو نوشت.
- قالب دستور به منظور نوشتن **Recursive Query**:

```
;With <CTEName> [(<Columns>)]
As (
        <Parent Statement>
        Union All
        <Child Stament>
)
Select  *
From            <CTEName>
```

- قبل از دستور **with** باید از ; استفاده شود
- بعد از نوشتن **CTE** حتما از باید دستور **Select** به منظور نمایش خروجی استفاده نمود
- در دستوری که برای **Child** استفاده می شود، باید نتیجه با خود جدول **CTE** تلفیق شود تا نتیجه به صورت تودرتو در خروجی ظاهر شود
- شما در یک **CTE** نمی توانید بیش از ۱۰۰ مرحله اطلاعات تو در تو تولید نمایید

Microsoft® SQL Server®

SQL Server 2016 Implementation

استفاده از CTE

- استفاده از **CTE (Common Table Expression)**
- **مثال: با استفاده از CTE ها، ساختار درختی موضوعات را پیمایش کنید و در کنار هر عنوان موضوع، عنوان پدر(ان) را نیز نمایش دهید**

```
;With AllSubjects
As(
        Select  ID
        ,       Name
        ,       ParentSubjectID
        ,       Cast(Name       As nVarChar(Max))       As RootPath
        From    Subject
        Where   ParentSubjectID Is Null
        Union All
        Select  Subject.ID
        ,       Subject.Name
        ,       Subject.ParentSubjectID
        ,       AllSubjects.RootPath + '/' +  Cast(Subject.Name As nVarchar(Max))
        From    Subject
        Inner Join
                AllSubjects
        On      Subject.ParentSubjectID = AllSubjects.ID

)

Select  *
From    AllSubjects
Order By RootPath
```

Microsoft® SQL Server®

SQL Server 2016 Implementation

استفاده از CTE

- استفاده از **CTE (Common Table Expression)**
- قالب دستور به منظور نوشتن داده های سلسله ای:

```
;With <CTEName1> [(<Columns>)]
As (
        <Statement>
),
<CTEName2> [(<Columns>)]
As (
)
...
Select  *
From            <CTEName[1,2,3,...]>
```

- قبل از دستور **with** باید از ; استفاده شود
- در صورتیکه اطلاعات مد نظر شما با استفاده از پردازش های مرحله ای تولید می شود می توانید از این قالب استفاده کنید
- استفاده از هر **CTE** تنها برای یکبار امکان پذیر است.

Microsoft® SQL Server®

SQL Server 2016 Implementation
Data Manipulation Language

- **Data Manipulating Language(DML)**
- **تغییر در داده های جداول:**
  - **افزودن اطلاعات جدید**
    - آشنایی با دستور Insert
      - با استفاده از مقادیر ثابت
      - با استفاده از داده جداول دیگر
    - آشنایی با دستور Select Into
  - **ویرایش اطلاعات**
    - آشنایی با دستور Update
  - **حذف اطلاعات**
    - آشنایی با دستور Delete
    - آشنایی با دستور Truncate

- **افزودن اطلاعات جدید**
- **قالب دستور با استفاده از مقادیر ثابت (به صورت ساده):**

```
Insert [Into] <Object> [Column1,Column2 ,...]
Values(<Value1,Value2>, ...)
```

- **قالب دستور برای افزودن چند ردیف:**

```
Insert [Into] <Object> [Column1,Column2 ,...]
Values(<Value1,Value2>, ...), (<Value1,Value2>, ...),...
```

- **نکات:**
  - در صورتیکه نام ستون ها در دستور **Insert** آورده نشود تمامی ستون های جدول در نظر گرفته می شود
  - دستور **Insert** مجوز ورود اطلاعات در ستون های **Identity** و ستون های محاسباتی **Computed Column** را ندارد
  - مقادیر نوشته شده در دستور **Values** باید به ترتیب نام ستون های دستور **Insert** و همجنس آنها باشد
  - در مقادیر **Values** ها می توان از مقادیر ثابت و یا توابع **Scalar** استفاده نمود
  - به منظور اطمینان از ورود صحیح مقادیر نوشتاری فارسی از **N** قبل از مقادیر استفاده می نماییم

Microsoft® SQL Server®

SQL Server 2016 Implementation
Data Manipulation Language

- افزودن اطلاعات جدید
- قالب دستور با استفاده از مقادیر جداول دیگر:

```
Insert [Into] <Object> [Column1,Column2 ,…]
Select [*] |    <Column1> ,
                <Column2> ,…
          From        <Object>
…
```

- نکته: با استفاده از دستور **Top** در دستور **Insert** می توان رکورد های انتخاب شده از جداول را برای افزودن به جدول محدود نمود

```
Insert [Top (n)] [Into] <Object> [Column1,Column2 ,…]
Select [*] |    <Column1> ,
                <Column2> ,…
          From        <Object>
…
```

Microsoft® SQL Server®

SQL Server 2016 Implementation
Data Manipulation Language

- **افزودن اطلاعات جدید (دستور Select Into)**
- **قالب دستور:**

```
Select   <Column1>,
         <Column2> , …
Into <New Object>
From   <Object>
```

- از این دستور برای انتخاب رکورد ها و ایجاد یک جدول جدید و افزودن آن ردیف ها به جدول جدید اضافه می شود
- در مقابل دستور Into نمی توان از جداول موجود در بانک استفاده نمود و به دلیل اینکه این دستور جدول جدید ایجاد خواهد نمود.
- جدول ساخته شده با این دستور، هیچ یک از Constraint های جدول اصلی را ندارد و تنها جدولی که ستون هایی با همان جنس اطلاعات مبدا دارد را تولید می گردد

Microsoft® SQL Server®

SQL Server 2016 Implementation
Data Manipulation Language

- **ویرایش اطلاعات**
- **قالب دستور:**

```
Update   <Object>
    Set  <Column1> = <Value1>,
         <Column2> = <Value2>,
         ....
Where <Boolean Expression>
```

- نکات: ستون های Identity و محاسباتی (همانند دستور Insert) در این دستور قابلیت استفاده ندارند
- در گذاره Where می توانید مشخص نمایید که چه رکورد هایی می بایست با این دستور ویرایش شوند

SQL Server 2016 Implementation
Data Manipulation Language

- **حذف اطلاعات (دستور Delete)**
- **قالب دستور:**

Delete <Object>

Where <Boolean Expression>

- در گذاره Where می توانید مشخص نمایید که چه رکورد هایی می بایست با این دستور از جدول مورد نظر حذف شوند

Microsoft® SQL Server®

SQL Server 2016 Implementation
Data Manipulation Language

- **حذف اطلاعات (دستور Truncate)**
- **قالب دستور:**

Truncate Table <Object>

- از این دستور برای حذف تمامی رکورد های یک جدول استفاده می گردد
- اطلاعات حذف شده log نمی شود و سرعت حذف در آن بالاتر از دستور Delete است
- در صورت وجود ارتباط با جداول دیگر این دستور قابلیت اجرا نخواهد داشت
- در صورتیکه جدول فوق دارای ستون Identity باشد بعد از انجام این دستور Identity آن Reset شده و از مقدار Seed دوباره پیگیری می شود
- در صورتیکه جدول شما دارای ارتباط با جداول دیگر است وپس از حذف اطلاعات تمایل به Reset کردن مقدار ستون Identity داشتید می توانید از دستور زیر برای Reset کردن آن استفاده نمایید:

DBCC CheckIdent('<TableName>' , Ressed , <number>)

Microsoft® SQL Server®

SQL Server 2016 Implementation

بررسی امکانات T-SQL

- **بررسی امکانات T-SQL**
  - **تعریف متغیر ها**
  - **دستور If**
  - **استفاده از حلقه While**
  - **معرفی و استفاده از متغیر های جدولی**
  - **معرفی جداول موقتی (Temporary Table)**

Microsoft® SQL Server®

SQL Server 2016 Implementation

بررسی امکانات T-SQL

- **بررسی امکانات T-SQL**
- **تعریف متغیر ها**
  - **به منظور تعریف متغیر از ساختار دستور زیر استفاده نمایید:**

```
Declare @<VariableName> <DataType> [=DefaultValue]
```

  - متغیر های تعریفی باید با @ شروع شوند
  - برای مقدار دهی به متغیر ها از دستور Set می توانید استفاده نمایید
  - متغیر ها می توانند در دستورات Select نیز مقدار دهی شوند
  - متغیر های سیستمی نیز وجود دارند که با @@ شروع می گردند. چند نمونه از آنها مانند:
    - @@RowCount
    - @@Version
    - @@Identity
    - @@ServerName
    - @@Connections
    - ....

SQL Server 2016 Implementation

بررسی امکانات T-SQL

- **بررسی امکانات T-SQL**
- **دستور If:**
  - **از این دستور برای کنترل یک مقدار Boolean استفاده می شود**
  - **قالب دستور**

```
If  <Boolean Expression>
```

- در صورت کنترل چندین خط برنامه به ازای شرط If از Begin و End استفاده می نماییم
- همانند دستورات برنامه نویسی دیگر از Else با همان ساختار نیزمی توانیم در صورت عدم صحت شرط استفاده نمود
- **دستور While**
  - **قالب دستور**

```
While   <Boolean Expression>
Begin
…
End
```

- از دو دستور Break و Continue می توان برای ایجاد وقفه در حلقه While (مانند زبان های برنامه نویسی دیگر) استفاده نمود

- **بررسی امکانات T-SQL**
- **تعریف و استفاده از متغیر های جدولی:**
  - این متغیر ها یک جدول را در حافظه ایجاد می نماید و با اتمام اجرای برنامه (Query) از حافظه سیستم خارج می گردند
  - متغیر های جدولی نمی توانند تمامی Constrain های مربوط به جداول واقعی را داشته باشند
  - قالب دستور:

```
Declare @<VariableName>   Table (<Column1> DataType, <Column2> DataType , …)
```

  - در استفاده از این متغیر ها به عنوان جدول در عملیات های مختلف مربوط به جداول هیچ محدودیتی وجود ندارد
  - برای افزودن، ویرایش و حذف اطلاعات از آنها باید از دستورات جدولی که قبلا اشاره شده است استفاده شود
  - این متغیر ها مانند جداول واقعی قابلیت شرکت در Join های و ... را دارند

Microsoft® SQL Server®

SQL Server 2016 Implementation

بررسی امکانات T-SQL

- **بررسی امکانات T-SQL**
- **تعریف و استفاده از جداول موقتی:**
- این جداول گزینه دیگری است که می توان از آنها برای نگهداری داده های موقت استفاده نمود. با این تفاوت که این جداول در بانک اطلاعاتی tempdb ساخته و نگهداری می شوند
- دو نوع جدول موقتی الف)Local و ب) Global وجود دارد
- قالب دستوری جدول موقتی Local

```
Create #<TableName> Table (Column Definitions)
```

- قالب دستوری جدول موقتی Global

```
Create ##<TableName> Table (Column Definitions)
```

- این دو جدول از نظر طول عمر با یکدیگر متفاوتند. جداول Local تا زمانی که Connection کاربر سازنده آن بر روی SQL وجود داشته باشد پابرجا خواهند بود. اما جداول Global تا زمانی که سرویس SQL دوباره اجرا شود، وجود داشته و قابل استفاده خواهند بود.

SQL Server 2016 Implementation

استفاده از View ها

- تعریف و استفاده از View ها در SQL Server
  - انواع View ها
  - ایجاد و استفاده از View ها
  - محدودیت ها ونکات مربوط به View ها

Microsoft® SQL Server®

SQL Server 2016 Implementation

استفاده از View ها

- **تعریف و استفاده از View ها در SQL Server**
- تعریف: View ها جداولی هستند مجازی، که اطلاعات ذخیره شده در جداول را به گونه ای متفاوت ارائه می دهند.
- **انواع View ها**
  - View های سیستمی
  - View های استاندارد
  - Indexed View های
  - View های موقتی

- **View های سیستمی:**
  - این **View** ها توسط SQL Server ایجاد شده و اطلاعات مخصوصی را در اختیار ما قرار می دهند و از جمله این View ها می توان به View هایی که در **INFORMATION_SCHEMA** قرار دارد اشاره نمود. در این Schema می توانید View های متعددی (ازجمله: **INFORMATION_SCHEMA.COLUMNS** که در آن اطلاعات ستون های تعریف شده در جداول را مشخص می نماید) را بیابید که اطلاعات مهمی از اجزای بانک اطلاعاتی جاری را در اختیاراتان قرار می دهد.

SQL Server 2016 Implementation

استفاده از View ها

- **View های استاندارد:**
- **این View ها توسط کاربر به منظور و اهداف زیر تولید می گردد:**
  - فیلتر کردن ستون ها :
  - در View ها می توانید نمایش تعداد ستون ها را در اختیار داشته باشید. با این کار می توانید امنیت اطلاعات را نیز به نوعی کنترل و مدیریت کنید تا کاربر ستون های مهم و استراتژیکی که شما در جداول قرار داده اید و معمولا کاربر با آنها تماسی ندارد را پنهان نمایید
  - فیلتر کردن رکورد ها:
  - به دلیل توانایی قرار دادن فیلتر بر روی نمایش ردیف ها در یک View می توانید رکوردهای مورد نیاز را به کاربر نمایش داده و از نمایش تمامی آنها خودداری نمایید
  - تلفیق اطلاعات مهم و پر کاربرد از جداول
  - جداول پر کاربرد را می توان به صورت تلفیقی در View هایی از پیش تعریف شده قرار داد تا از بانویسی آنها در Object های دیگر جلوگیری شود.
  - حفظ استقلال داده ای لایه های پایین تر در زمان Denormalization
  - با استفاده از View ها می توان تغییراتی را که در جداول می دهیم، تاثیر کمتری بر روی لایه اطلاعاتی که به کاربر نمایش می دهیم داشته باشد

Microsoft® SQL Server®

SQL Server 2016 Implementation

استفاده از View ها

- **در زمان ساخت View ها به نکات زیر توجه نمایید:**
  - نام ستون ها در View باید منحصر به فرد باشد
  - امکان استفاده از دستور Order By بدون دستور Top در View ها وجود ندارد

- **نحوه تولید View ها:**
  - با استفاده از Management Studio
  - با استفاده از دستورات DDL (که در درس های آتی به آن پرداخته می شود)

- **کاربرد View ها:**
  - بازیابی اطلاعات : یک View می تواند به عنوان یک Object در دستور From قرار داده شود. بنابراین می تواند در تلفیق با جداول و یا View های دیگر نیز از آنها استفاده نمود
  - تغییرات در اطلاعات: با استفاده از View ها می توان اطلاعات درون جداول را نیز با دستوراتی مانند Update و Delete و Insert تغییر داد. اما تغییرات اطلاعات با استفاده از View ها دارای محدودیت هایی است که باید به آنها توجه داشت

Microsoft® SQL Server®

SQL Server 2016 Implementation

استفاده از View ها

- **تعریف و استفاده از View ها در SQL Server**
- **View های استاندارد:**
- **شرایط لازم برای View به منظور تغییر در داده ها:**
  - چنانچه بر روی View ها Trigger تعریف شده باشد عملیات امکان پذیر است
  - در View حداقل یک جدول استفاده شده باشد
  - ستون های موجود در View قابلیت ویرایش داشته باشند (Read Only نباشند)
  - در صورتیکه از دستور Group By در View استفاده شده باشد امکان انجام هیچ گونه عملیاتی بر روی آنها امکان پذیر نیست
  - ستون هایی که در View استفاده نشده اند قابلیت پذیرش Null را داشته باشند و یا برای آنها مقادیر پیش فرض تعریف شده باشد.

- **تعریف ابتدایی Index**
  - مفهوم Page در SQL Server:
  - همانطور که اشاره شد هر Page از اطلاعات می تواند ۸ کیلو بایت اطلاعات را در خود جای دهد. به طور مشخص چنانچه یک صفحه پر شده باشد و شما به اقدام ثبت رکورد بعدی نمایید SQL صفحه بعدی را ایجاد و رکورد جدید را در آن صفحه قرار می دهد.

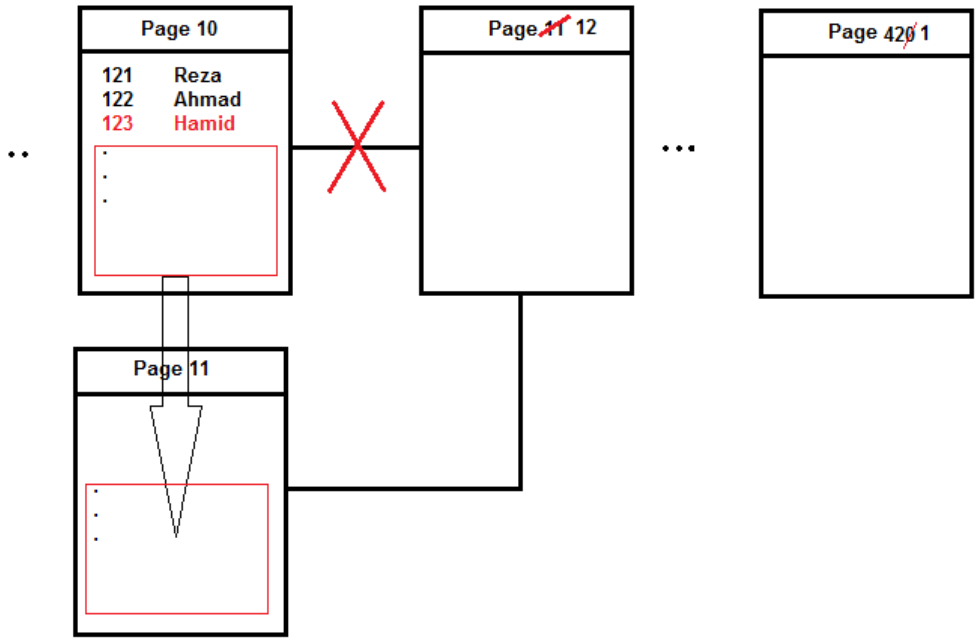
Page 1	
10	Ali
12	Reza
14	Hasan
18	Mina
20	Aria

Page 2	
22	Pedram

SQL Server 2016 Implementation
Index

- **تعریف ابتدایی Index**
  - مفهوم Page Splitting در SQL Server:
  - در مثال قبل فرض کنید اطلاعاتی را که می خواهید Insert کنید بر اساس تنظیمات Index تعریفی در Page1 باید قرار داده شود اما Page1 پر شده است. در این زمان SQL اقدام به تکه کردن Page ها می نماید. یک راه حل این است که محل جدید رکورد را بیابد، مابقی رکورد ها را تا آخرین صفحه Shift نماید. اما این عمل در زمانی که تعداد صفحات زیادی وجود داشته باشد بسیار سنگین و ممکن است به کندی انجام شود.
  - اما آنچه SQL انجام می دهد به این شکل است که فرض کنید تشخیص داده شد این رکورد جدید باید در صفحه ۱۰ قرار داده شود و کل صفحات نیز ۴۲۰ صفحه می باشد. یک صفحه جدید ایجاد شده نصف از رکورد های این صفحه را به صفحه جدید منتقل می گردد و رکورد جدید را با توجه به فضای باز شده در صفحه اصلی قرار می دهد حال این صفحه جدید را بین دو صفحه قدیم قرار می دهد و ارتباط صفحات با هم را دوباره برقرا می نماید با این کار دیگر لازم به جابجایی رکورد ها از صفحه ۱۱ تا صفحه ۴۲۰ نیست و این عمل با سرعت بیشتری اتفاق می افتد

- تعریف ابتدایی **Index**
- مفهوم Page Splitting در SQL Server:



Microsoft® SQL Server®

SQL Server 2016 Implementation
Index

- **Clustered Index در مقابل Non Clustered Index**

- ایندکس اشاره گری است که به هر داده اشاره می نماید. زمانی که Clustered تعریف شوند مستقیما اشاره گر به خود هر رکورد در جدول اشاره می نماید و برای ذخیره سازی اشاره گر نیازی به فضای دیگر نمی باشد. این نوع Index ها را Clustered می گویند. با توجه به این تعریف هر جدول تنها می تواند یک Index از جنس Clustered داشته باشد و مابقی ایندکس ها باید Non Clustered تعریف شوند.
- هر جدول نهایتا می تواند شامل ۲۵۰ Index باشد که یکی از آنها Clustered و مابقی باید از نوع Non Clustered تعریف شوند
- ایندکس هایی که از نوع Non Clustered تعریف شوند نیاز به یک فضای مجزا برای ذخیره سازی دارند.
- چنانچه برای یک جدول قید Primary Key تعریف نمایید، یک ایندکس از نوع Clustered بر روی آن تعریف می شوند و مابقی ایندکس ها می بایست از نوع non Clustered تعریف شوند

Microsoft® SQL Server®

SQL Server 2016 Implementation
Index

- **Ascending Index در مقابل Descending Index**
  - شما میتوانید در زمان تعریف یک ایندکس نوع مرتب سازی را Ascending و یا Descending انتخاب نمایید. مشخص است که بر اساس نوع داده و اهمیت نوع مرتب سازی آن می توانید این نوع را به صورت مناسب برای ایندکس انتخاب نمایید
- **Unique Index در مقابل Non Unique Index**
  - شما می توانید در زمان تعریف ایندکس مشخص نمایید که ان ایندکس از ثبت اطلاعات تکراری جلوگیری نماید یا خیر. در اینصورت این دو نوع ایندکس نیز تولید می شود. به طور مشخص قرار دادن قید Unique Key برای یک جدول یک ایندکس از نوع Unique را برای جدول تولید می نماید که البته Non Clustered نیز می باشد
- **Single Column Index در مقابل Multi Column Index**
  - چنانچه یک ایندکس بر روی یک ستون تعریف شود در اصطلاح Single Column Index و در غیر اینصورت Multi Column Index نامیده می شوند. حداکثرتعداد ۱۶ ستون را می توانید در یک Index شرکت دهید. چنانچه بیشتر از آن تعداد را مورد نیاز دارید می توانید از امکان Included Columns در تعریف Index استفاده نمایید.

Microsoft® SQL Server®

SQL Server 2016 Implementation
Index

- **نحوه ساخت Index**
  - امکانات جدول مورد نظر خود در بانک اطلاعاتی را با کلیک بر روی + کنار آن می توانید مشاهده نمایید.
  - از قسمت Indexes ها می توانید Index های تعریف شده بر روی هر جدول را مشاهده نمایید.
  - برای ایجاد یک Index جدید می توانید با کلیک سمت راست از منوی باز شده نوع ایندکس مورد نظر برای ایجاد را انتخاب نمایید
  - در قسمت General از پنجره باز شده می توایند مشخصات عمومی Index را تعیین نمایید و از روی قسمت Options می توانید خصوصیات این Index را نیز تعیین نمایید

Microsoft® SQL Server®

SQL Server 2016 Implementation

Index

- **بازبینی نتیجه کار و مدیریت Index ها**
  - با کلیک بر روی هر Index در پنجره باز شده قسمت Fragmentation اضافه می شود. شما در این Page می توانید اطلاعات مهمی را از Index ها دریافت نمایید:
  - **Page Fullness** :
  - میزان پر بودن صفحات از لحاظ اطلاعات را نمایش می دهد
  - **Total Fragmentation**:
  - میزان تکه تکه شدن صفحات را نشان می دهد. بعد از استفاده از جداول، بر اساس عملیاتی مانند حذف و ... می تواند فضاهای خالی بین صفحات ایجاد شود. این درصد میزان فضای خالی بین صفحات را نشان دهد. هرچه این عدد بیشتر باشد میزان فضای خالی بیشتر و به طبع تعداد صفحات نیز بیشتر است و عملیات جستجو با سرعت کمتری انجام می شود. عمل Rebuild کردن و یا Reorganize یک Index می تواند این فضاهای خالی را از بین برده و تعداد صفحات را کاهش دهد و در نتیجه جستجوی سریعتری را به همراه داشته باشد
  - نکته:
  - برای درصد های بین ۰ تا ۳۰ معمولا ایندکس در وضعیت مطلوب است
  - برای درصد های بین ۳۰ تا ۵۰ درصد آن را Reorganize کنید
  - برای درصد بالای ۵۰ درصد آن را Rebuild کنید

Microsoft® SQL Server®

SQL Server 2016 Implementation
Stored Procedure

- روال های ذخیره شده یا **Stored Procedure**
- تعریف **Stored Procedure**
- بررسی انواع **Stored Procedure** ها
- ایجاد و ویرایش **Stored Procedure**
  - تعریف پارامتر ورودی
  - تعریف پارامتر خروجی
- اجرای **Stored Procedure** ها و استفاده از مقادیر خروجی

Microsoft® SQL Server®

SQL Server 2016 Implementation
Stored Procedure

- **تعریف Stored Procedure**
  - یک SP روال و دستورات ای است که در بانک اطلاعاتی ذخیره می شود
  - از قابلیت های مهم این Object ها دربانک اطلاعاتی این است که یکبار نوشته می شوند و بارها می توان از آنها استفاده نمود
  - SP ها، یکبار در زمان تعریف و تغییر در متن آنها Compile می شود و در دفعات اجرای بعدی دیگر نیازی به کامپایل مجدد نیست و به همین دلیل سرعت اجرای آنها بیشتر از Query های معمولی است
  - از آنجا که SP ها دارای پارامتر هستند، به همین دلیل Execution Plan ای که برای آنها تولید می شود یکسان است و به همین دلیل دارای سرعت بالاتری در اجرا هستند.
  - در SP ها میتوانید از تمامی دستورات معرفی شده در T-SQL استفاده کنید. دستورات DML نیز در SP ها قابل استفاده اند.(این دستورات در UDF ها و View هاقابلیت استفاده ندارند)
  - SP ها به طور کامل از تراکنش ها پشتیبانی می نمایند. به همین خاطر بهترین گزینه برای انجام محاسبات و تغییر برروی اطلاعات جداول به ازای انجام آن محاسبات می باشند
  - با توجه به اینکه SP ها توسط سرویس های SQL اجرا می شوند، معمولا در انجام محاسبات سنگین و پیچیده از آنها استفاده می شود و برنامه نویسان ترجیح می دهند این دستورات بیشتر سمت سرور پردازش شود تا در سمت کاربر (در محیط برنامه نویسی)

SQL Server 2016 Implementation
Stored Procedure

- **انواع Stored Procedure ها**
  - سیستمی (System)
  - عادی یا کاربردی (Normal)
  - توسعه یافته (Extended)
  - CLR
- **پروسیجر های سیستمی:**
  - این SP ها توسط SQL سرور تهیه شده و می توانید از آنها برای عملیات های مختلفی استفاده نمایید. از جمله این SP ها می توان به: sp_help, sp_rename, sp_who , ... اشاره نمود. در نامگذاری این SP ها معمولا از sp_ در ابتدای آنها استفاده شده است
- **پروسیجر های عادی یا نرمال:** این SP توسط کاربر در بانک اطلاعاتی به منظور عملیات خاص نوشته می شود.
- **پروسیجر های توسعه یافته:**
  - پروسیجر هایی هستند که معمولا با زبان های برنامه نویسی دیگر (به غیر از T-SQL) نوشته شده می شود. زبان هایی مانند C++. از جمله این پروسیجر ها می توان به xp_logininfo, xp_sendMail, xp_fixedDrives و... اشاره نمود. از پیشوند xp_ در نامگذاری پروسیجر ها توسعه یافته استفاده می شود. این SP ها معمولا در بانک master قرار داده می شود ومعمولا اطلاعاتی از ویندوز و کامپیوتر سرور را در اختیار ما قرار می دهند. همچنین می توان با استفاده از آنها عملیاتی را نیز به صورت فیزیکی بر روی سرور انجام داد (مانند حذف فایل، اطلاعات هارد و ...)
- **پروسیجر های CLR:** این پروسیجر ها معمولا به زبان های برنامه نویسی در Visual Studio تولید شده و قابلیت اجرا بر روی بانک اطلاعاتی را دارند

Microsoft® SQL Server®

SQL Server 2016 Implementation
Stored Procedure

- **دستور ایجاد Stored Procedure**

```
Create Procedure [<Schema>.]<ProcedureName>
<Parameters Definitions>
As
<Operation Statements>
GO
<Parameter Definition>:
@<ParameterName> <DataType> [=DefaultValue] [Input | Output] [0...n]
```

- چنانچه نیاز به ایجاد SP در یک Schema خاص دارید، نام Schema را نیز قبل از نام SP ذکر کنید
- چنانچه نوع یک پارامتر را (ورودی/خروجی) تعیین ننمایید، پیش فرض به عنوان پارامتر ورودی در نظر گرفته می شود.
- در بدنه یکSP می توانید از دستور return جهت خروج از SP استفاده نمایید
- یک SP می تواند دارای یک مقدار خروجی باشد. معمولا از این نوع خروجی به منظور پی بردن به نتیجه اجرای SP استفاده می شود

Microsoft® SQL Server®

SQL Server 2016 Implementation
Stored Procedure

- روال های ذخیره شده یا **Stored Procedure**
  - اجرای یک **Stored Procedure**

```
Exec <Stored Procedure Name> <Parameters Value>

Sample:   Exec sp_rename 'MyTB', 'Table1'
```

- چنانچه یک پاراتر ورودی دارای مقدار پیش فرض باشد، می توان مقداری در آن به عنوان پارامتر ارسال نکرد
- چنانچه یکی از پارامترهای میانی دارای مقدار پیش فرض باشد، و شما قصد استفاده از مقدار پیش فرض برای آن دارید، می توانید از دستور Default بجای مقدار دادن به آن استفاده نمایید و نمی توانید آن را بدون مقدار رها کنید
- در صورت استفاده از پارامتر خروجی یک SP، می بایست متغیری از همان جنس تعریف کرده و آن را به SP پاس دهید و در اجرا مشخص نمایید که این متغیر خروجی است

```
Declare @Result int

Exec InsertPerson 'Test' , @Result Output

Print @Result
```

Microsoft® SQL Server®

SQL Server 2016 Implementation

Stored Procedure

- **امکان اجرای یک SP در یک SP دیگر وجود دارد. اما تنها می توان تا ۳۲ SP را به صورت Nested یا تو در تو بر روی SQL Server اجرا نمود.**
- **در صورتیکه خروجی نوع ساده از داده باشد می توان با دستور Return خروجی را مشخص نمود (بدون اینکه از متغیری برای این کار استفاده نمایید) در این صورت با دستور Exec می توانید خروجی یک SP را در یک متغیر برگردانید:**

```
Declare @Var
Exec @Var = <SPName> <Parameters>
```

- **ویرایش و حذف Stored Procedure**
- به منظور ویرایش از دستور Alter در قالب زیر استفاده کنید:

```
Alter Procedure <ProcedureName>
<Parameters Definition>
As
<Statements>
GO
```

- به منظور حذف پروسیجر از دستور Drop با ساختار زیر اااستفاده نمایید

```
Drop Procedure <ProcedureName>
GO
```

Microsoft® SQL Server®

SQL Server 2016 Implementation
Function - UDF

- توابع در SQL Server(User Define Functions - UDF)
- تعریف Function
- بررسی انواع Function ها
  - توابع سیستمی
  - توابع تعریف شده توسط کاربر
- انواع توابع تعریف شده توسط کاربر
  - توابع اسکالر (Scalar-Value)
  - توابع جدولی (Table-Value)
  - توابع جمع (Aggregate-Value)
- اجرای انواع توابع

Microsoft® SQL Server®

SQL Server 2016 Implementation

Function - UDF

- **تعریف Function**
- **همانطور که از نام آن مشخص است، تابع می تواند با دریافت ورودی خروجی هایی را برای ما مهیا نماید**
- **توابع سیستمی:**
  - دسته اول: توابع جمع (Aggregate): از این توابع برای جمع بندی اطلاعات استفاده می شود (Sum, Avg , Count , Min , Max ,..)
  - دسته دوم: توابع اسکالر: این توابع یک مقدار خاص را بر اساس وظیفه بر می گرداند که می توان از آن در محاسبات استفاده کرد. مانند: GetDate() , IsNull() , Stuff() , ...
  - دسته سوم: توابع رتبه بندی(Ranking): از این توابع جهت دادن رتبه و وزن به رکورد ها استفاده می شود. مانند: Row_Number(), Rank(), Dense_Rank(), ...
  - دسته چهارم: توابع جدولی(RowSet): از این توابع می توان در دستور From استفاده نمود. توابعی مانند: OpenRowset, OpenQuery و... که می توان در Distributed Query ها از آنها استفاده نمود

- **کار با توابع رتبه بندی:**
  - تابع **Row_Number**:
  - **این تابع یک شماره متوالی به هر ردیف تخصیص می دهد. این تابع بر اساس دو پارامتر زیر شماره بندی را انجام می دهد:**
    - مرتب سازی (Order By)
    - بخش بندی(Partition By)
    - **قالب کلی دستور آن به شکل زیر است:**

Row_Number() Over ([Partition By <ColumnNames>] Order By <ColumnNames>)

- وجود دستور Order By در استفاده از این تابع الزامی است
- اما دستور Partition By می تواند به صورت دلخواه به آن افزوده شود

- **کار با توابع رتبه بندی:**
- **تابع Rank و Dense_Rank:**
  - این تابع یک بر اساس ستونی که در دستور مرتب سازی به آن می دهید یک رتبه به هر رکورد اختصاص می دهد. لازم به ذکر است که رتبه های تکراری نیز تولید می شود. این تابع بر اساس دو پارامتر زیر رتبه بندی را انجام می دهد:
    - مرتب سازی (Order By)
    - بخش بندی(Partition By)
  - قالب کلی دستور آن به شکل زیر است:

Rank() Over ([Partition By <ColumnNames>] Order By <ColumnNames>)

  - دستور Order By در این تابع بایدبه صورت الزامی استفاده شود
  - اما دستور Partition By می تواند به صورت دلخواه به آن افزوده شود
  - این دستور چنانچه ردیف های هم رتبه وجود داشته باشد اعداد یکسان را به آنها اختصاص می دهد اما عدد بعدی بر حسب جایگاه آن رکورد است و ممکن است شماره رتبه ها با این دستور بدون فاصله (gap) نباشد. برای اطمینان از اینکه اعداد پشت سر هم باشند می توانید از دستور Dense_Rank با همان قالب استفاده نمایید

Microsoft® SQL Server®

SQL Server 2016 Implementation

Function - UDF

- **توابع اسکالر:**
- **این توابع به ازای دریافت ورودی(که دلخواه می باشد) یک مقدار را به عنوان خروجی تحویل می دهند**
- **قالب دستور:**

```
Create Function <SchemaName>.<FunctionName>(<Parameters Definitions>)
Returns   <ReturnDataType>
As
Begin
        Return <Return Value>
End
```

- این تابع میتواند به عنوان یکExpression در تمامی دستورات Select و یا Where و یا ... دیگر استفاده شود
- برای فراخوانی توابع باید نام Schema آن نیز حتما ذکر شود (به منظور جلوگیری از ادغام با نام جداول)
- مثال: تابعی بنویسید که یک ورودی متنی از شما دریافت و خروجی همان متن بدون کارکتر Space باشد (حق استفاده از تابع Replace را ندارید)
- توابعی برای تبدیل تاریخهای میلادی به شمسی و بالعکس نمونه های دیگری از این نوع توابع هستند

Microsoft® SQL Server®

SQL Server 2016 Implementation
Function - UDF

- **توابع جدولی:**
- **خروجی این توابع می تواند یک جدول باشد (بالعکس توابع اسکالر که تنها می تواند یک مقدار باشد)**
- **این توابع خود به دو دسته زیر تقسیم بندی می شوند:**
  - Inline Table Value Functions
  - Multi Statement Table Value Functions
- **تفاوت این دو تابع در این است که دسته اول تنها می تواند حاوی یک دسته دستور Select برای بازگرداندن خروجی باشد اما دسته دوم می تواند حاوی چندین دستور متفاوت باشد**
- **قالب دستور برای ایجاد توابع Inline Table Value**
  - **Create Function <FunctionName>(<Parameters Definition>)**
  - **Returns Table**
  - **As**
  - **Return (Select Statement)**
- **نکته: این توابع از جهاتی شبیه به View هستند اما باتفاوت های زیر:**
  - تابع پارامتر می پذیرد اما View نمی تواند
  - دستکاری اطلاعات توسط View با شرایطی امکان پذیر است اما در UDF ها امکان پذیر نیست

- توابع **Multi Statement Table Value**:
- **قالب دستور :**

```
Create Function <FunctionName>(<Parameters Definition>)
Returns <TableVariableName> Table (<Column Definitions>)
As
Begin
    Return
end
```

- در واقع در این دسته از توابع، می بایست نتیجه تابع را در متغیر جدولی که در تعریف تابع معرفی می کنید، قرار دهید
- با توجه به تعریف متغیر جدولی و قرار دادن نتیجه تابع در آن، دستور Return آن متغیر را به خروجی ارسال می نماید

- **نکات استفاده از توابع:**
  - تمامی مسیر های اجرایی در تابع باید به دستور Return ختم شود (اطمینان از اینکه در هر حالتی تابع خروجی را ارسال می نماید)
  - در بدنه تابع نمی توان از دستورات DML به منظور دستکاری اطلاعات ذخیره شده در جداول استفاده نمود
  - قوانین مطرح شده در رابطه به اجرای SP های تو در تو برای توابع نیز صادق اند و بیش از ۳۲ تابع تو در تو را نمی توانید اجرا نمایید
  - برای اجرای توابع اسکالر نام بردن Schema الزامی است
  - برای اعمال تغییرات بر روی یک Function می توانید از دستور Alter وبرای حذف آن می توانید از دستور Drop استفاده نمایید
  - امکان اجرای توابع در SP ها وجود دارد اما بالعکس آن صادق نیست.

UDF – Cross Apply

- **واکشی اطلاعات از چندین جدول**
- **معرفی Cross Apply**
  - از این دستور زمانی استفاده می شود که بخواهیم از اطلاعات ستون شیئ اول در انتخاب ردیف های شیئ دوم استفاده نماییم
  - قالب دستور

```
From <Obj1> Cross Apply   <Obj2>
```

- **نکته: از این Join بیشتر در زمانی که شیئ دوم در حالات زیر است استفاده می شود:**
  - شیئ دوم با استفاده از Nested Query ها استخراج شود
  - شیئ دوم با استفاده از Table-Value Function ها استخراج گردد
  - در درس های آتی که Nested Query ها و Table-Value Function ها معرفی می گردد استفاده این دستور بیشتر مورد بررسی قرار داده خواهد شد

SQL Server 2016 Implementation

استفاده از Trigger

- ایجاد و استفاده از **Trigger** ها
- تعریف **Trigger**
- بررسی انواع **Trigger** ها
  - DML Trigger
  - DDL Trigger
- انواع **Trigger** های **DML**
  - Instead Of
  - After
- کاربرد **Trigger** ها

Microsoft® SQL Server®

SQL Server 2016 Implementation

استفاده از Trigger

- **ایجاد و استفاده از Trigger ها**
- **تعریف Trigger**
- **تریگر، مانند یک Procedure است با این تفاوت که که بر اساس اتفاقات خاص به صورت خودکار اجرا می گردد**
- **انواع تریگر ها**
  - DML: از این تریگر ها برای زمانی که می خواهید در صورت اجرای یک دستور DML به صورت خودکار اجرا شود استفاده نمایید
  - DDL: از این تریگر ها برای زمانی که می خواهید به واسطه تغییر بر روی Object های بانک به صورت خودکار اجرا گردد استفاده نمایید
- **انواع تریگر های DML**
  - Instead Of: زمانی که بخواهید به جای یک دستور خاص DML دستور شما اجرا شود از این نوع تریگر ها استفاده نمایید
  - After: از این تریگر ها به منظور اجرای دستورات خود بعد از دستورات DML می توانید استفاده نمایید

Microsoft® SQL Server®

SQL Server 2016 Implementation

استفاده از Trigger

- ایجاد و استفاده از **Trigger** ها
- نکات خاص در مورد انواع تریگر های **DML**

موضوع	Instead of	After
اعمال تغییرات به صورت فیزیکی	قبل از ذخیره سازی اطلاعات	بعد از ذخیره سازی اطلاعات
قابل تعریف بودن	Table/ View	Table
موافقت با اعمال تغییرات	تنها با Table	-
اولویت نسبت به Constraint	قبل از چک کردن Constraint ها	بعد از چک کردن Constraint

- برای ایجاد تریگر ها از قالب دستور زیر استفاده نمایید

```
Create Trigger  <TriggerName>
On      <TableName | ViewName>
{Instead of  |  After  |  For} [Insert] [,] [Update] [,] [Delete]
As
Begin
End
```

Microsoft® SQL Server®

SQL Server 2016 Implementation

استفاده از Trigger

- ایجاد و استفاده از **Trigger** ها
  - نکات خاص در مورد انواع تریگر های **DML**
  - یک تریگر می تواند به ازای بیش از یک دستور **DML** بر روی یک جدول ایجاد گردد
  - به منظور ویرایش دستورات یک تریگر از دستور **Alter** برای حذف آن از دستور **Drop** استفاده نمایید
  - امکان فعال و غیر فعال کردن یک تریگر با استفاده از دستور زیر نیز مهیا می باشد:

```
Alter {Table | View}
{Enable | Disable} Trigger <TriggerName>
```

- **ایجاد و استفاده از Trigger ها**
  - **نکات خاص در مورد انواع تریگر های DML**
    - در تریگر های از نوع DML جهت دسترسی به اطلاعات در حال تغییر از دو جدول Inserted و Deleted می توانید استفاده نمایید.

Table \ Action	Count(*) From Inserted	Count(*) From Deleted
Insert	>0	= 0
Update	>0	>0
Delete	= 0	> 0

Microsoft® SQL Server®

SQL Server 2016 Implementation

استفاده از Trigger

- **ایجاد و استفاده از Trigger ها**
- **نکات خاص در مورد انواع تریگر های DML**
- **به منظور پی بردن به تغییر در یک ستون خاص در تریگر ها می توانید از دستور Update به شکل زیر استفاده نمایید**

```
If Update(<ColumnName>)
Begin
End
```

- **کاربرد Trigger ها**
  - پیاده سازی جامعیت داده ها به صورت Procedural
  - تهیه سابقه و یا Log از روی فعالیت های کاربر
  - سفارشی کردن متن خطاها (استفاده از Instead Of)
  - ایجاد محرک در پرداز شها (با استفاده از After)
  - تغییر در ماهیت دستور ها (با استفاده از Instead of)

SQL Server 2016 Implementation
مدیریت خطا ها و تراکنش ها

- **مدیریت خطاها و تراکنش ها**
  - **استفاده از متغیر های عمومی مناسب**
  - **تعریف بلوک Try Catch**
  - **تعریف تراکنش**
    - تایید تراکنش
    - بازگرداندن تراکنش
  - **استفاده دستور RiasError به منظور بازگرداندن خطاها**
  - **استفاده از دستور Throw**

- **مدیریت خطاها و تراکنش ها**
- **استفاده از متغیر های عمومی مناسب**
- **متغیر @@Error:**
  - در این متغیر شماره خطای آخرین دستوری که اجرا شده است بازگردانده می شود. چنانچه آخرین دستور اجرا شده همراه با خطا باشد، عدد این متغیر بزرگتر از صفر و در غیر اینصورت صفر خواهد بود. به همین منظور برای اطمینان از اجرای صحیح آخرین دستور اجرا شده می توان از این متغیر استفاده نمود.
- **متغیر @@RowCount**
  - در این متغیر تعداد رکورد هایی که در آخرین دستور اجرا شده متاثر می شوند بازگردانده می شود. به منظور اطلاع از انجام عملیاتی خاص و اطمینان از تعداد رکورد های تحت تاثیر قرار داده شده می توان از این متغیر استفاده نمود.

Microsoft® SQL Server®

SQL Server 2016 Implementation
مدیریت خطا ها و تراکنش ها

- **مدیریت خطاها و تراکنش ها**
- **بلوک Try Catch:**

```
Begin Try
    <Your Statement>
End Try
Begin Catch
    < Your Statement when an error has been occurred>
End Catch
```

- در صورتیکه اجرای دستورات در بلوک Try با خطا مواجه شود، روند اجرای دستورات در بلوک Catch قرار می گیرد و چنانچه نیاز به نوشتن دستورات خاصی داشتید می توانید از این بلوک استفاده نمایید

- **مدیریت خطاها و تراکنش ها**
  - **استفاده از توابع خاص در صورت بروز خطا:**
  - در صورتیکه جریان اجرای دستورات (بواسطه ایجاد یک خطا) در بلوک Catch قرار داده شود شما می توانید با استفاده از توابع زیر به اطلاعات مناسبی از خطا دستیابی پیدا کنید:

Function Name	Description
Error_Number()	کد خطای رخ داده شده
Error_Severity()	میزان اهمیت خطای رخ داده شده
Error_State()	وضعیت خطای رخ داده شده
Error_Line()	شماره خط دستوری که باعث ایجاد خطا شده است
Error_Message()	متن خطای رخ داده شده
Error_Procedure()	بازگرداندن نام SP یا تریگری که باعث ایجاد خطا شده است

Microsoft® SQL Server®

SQL Server 2016 Implementation
مدیریت خطا ها و تراکنش ها

- **مدیریت خطاها و تراکنش ها**
  - **تعریف تراکنش ها Transaction ها:**
  - با تعریف تراکنش ها می توانیم مجموعه ای دستورات را به صورت موفق با هم اجرا کرده و یا هیچ کدام اجرا نگردند. درصورت بروز مشکل در اجرای یک تراکنش، عملیات Recovery انجام داده خواهد شد و داده های تغییر یافته به وضعیت قبل از اجرای تراکنش باز خواهند گشت.
  - انواع تراکنش ها
  - دسته اول: تراکنش های صریح (Explicit)
  - دسته دوم: تراکنش های ضمنی (Implicit)
  - دسته سوم: تراکنش های خودکار(Auto Complete)

- **مدیریت خطاها و تراکنش ها**
- **دسته اول: تراکنش های صریح (Explicit)**
  - در این دسته از تراکنش ها دستورات تراکنش به صورت صریح و واضح نوشته می شود. این دستورات عارتند از:

```
Begin Transaction
    <Your Statement>
Commit Transaction
RollBack Transaction
```

  - در صورتیکه دستورات با موفقیت اجرا شوند از دستور Commit برای ثبت تغییرات استفاده نمایید
  - درصورتیکه دستورات با خطا مواجه شد، از دستور RollBack استفاده نمایید تا تغییرات به قبل از دستور Begin Transaction باز گردد (تاثیر دستورات اجرا شده از بین برود)

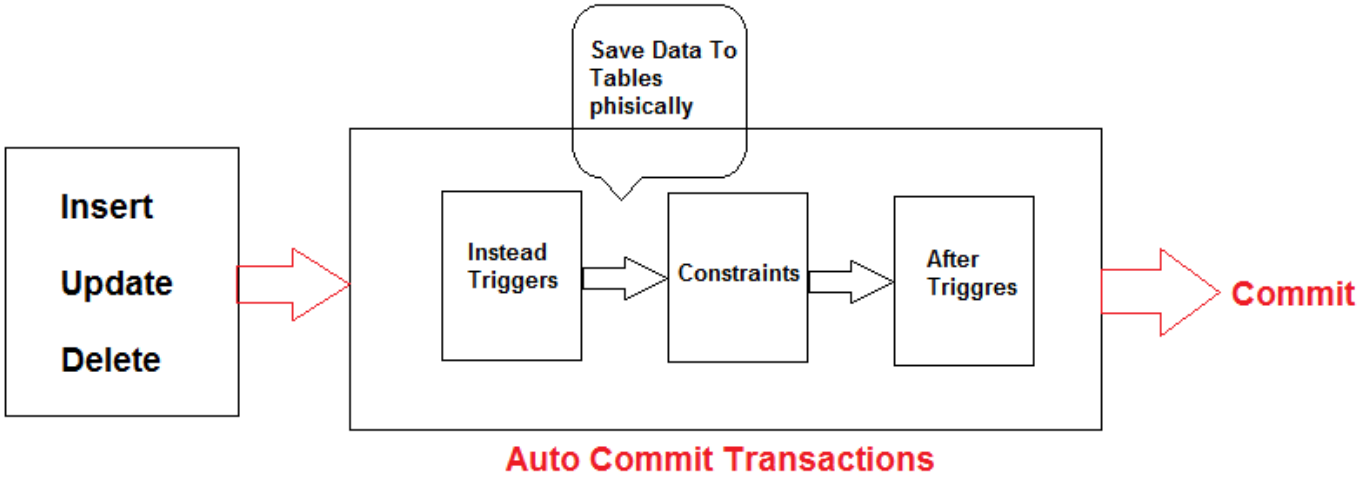
- **مدیریت خطاها و تراکنش ها**
- **دسته دوم: تراکنش های ضمنی (Implicit)**
  - این تراکنش ها مجموعه ای از دستورات صریح هستند که از تکرار آنها در کد جلوگیری می نماید
  - از دستور زیر می توانید برای فعال و غیر فعال کردن آن استفاده نمایید

```
Set Impilicit_Transactions {ON | OFF}
```

  - در صورتیکه از این نوع تراکنش استفاده نمایید هر یک از دستورات زیر یک تراکنش را شروع می نماید:

ALTER TABLE	FETCH	REVOKE
BEGIN TRANSACTION	GRANT	SELECT (See exception below.)
CREATE	INSERT	TRUNCATE TABLE
DELETE	OPEN	UPDATE
DROP		

- **مدیریت خطاها و تراکنش ها**
- **دسته سوم: تراکنش های خودکار(Auto Complete)**
  - این تراکنش ها توسط خود SQL Server اجرا و مدیریت می شوند. چنانچه شما از تراکنش های صریح و ضمنی در دستورات DML استفاده ننمایید، خود SQL Server به منظور جامعیت اطلاعات این تراکنش ها را به اجرا در می آورد. به طور کلی قالب آن در شکل زیر مشخص است:



Microsoft® SQL Server®

SQL Server 2016 Implementation
مدیریت خطا ها و تراکنش ها

- **مدیریت خطاها و تراکنش ها**
- **نکات استفاده از تراکنش ها:**
  - شما می توانید با استفاده از متغیر عمومی @@TranCount در هر لحظه از تعداد تراکنش های باز مطلع شوید
  - از ترکیب دستور Try Catch و تعریف تراکنش های صریح جهت مدیریت خطاها و تراکنش ها استفاده نمایید
  - برای مشخص کردن مدت زمان Timeout در یک تراکنش از دستور زیر استفاده نمایید:

```
Set Lock_Timeout   <Value In Second>
```

- امکان استفاده از تراکنش های تو در تو تا ۱۶ مرحله امکان پذیر خواهد بود. برای تراکنش های تو در تو می توانید در جلوی دستورات خود از یک عنوان برای هر تراکنش استفاده نمایید. با این کار می توانید دستورات RollBack و Commit را برای یک تراکنش مشخص اجرا نمایید.
- در صورت استفاده از تراکنش های صریح، مطمئن شوید که دستورات شما به RollBack و یا Commit منتهی می گردد. در غیر اینصورت بعد از انتظار زمان مشخصی (و تعیین نکردن شما برای تایید یا بازگردندن)SQL خود دستور RollBack را فراخوانی کرده و تغییرات شما را باز می گرداند

- **مدیریت خطاها و تراکنش ها**
  - **استفاده از دستور RaisError()**
  - در مواقعی که نیاز به اعلام خطای خاص در Object هایی که دستورات شما را به صورت خودکار یا غیر خودکار انجام می دهند دارید می توانید از این دستور در قالب زیر استفاده نمایید:

```
RaisError(<MessageText>,<Severity>,<State>)
Exp: RaiseError('Code is not Valid' , 10 , 1)
```

  - در تریگر هایی که به منظور کنترل جامعیت داده ها استفاده می کنید به منظور بازگرداندن خطا های اتفاق افتاده می توانید از این دستور استفاده نمایید.
  - لازم به ذکر است که یکی از روش هایی که شما می توانید پیغام های Customize به کاربر دهید نیز استفاده از این دستور برای بازگرداند متن خطا ترجمه شده می باشد
  - در SP ها نیز به منظور آگاهی از آنچه اتفاق می افتد (در بدنه SP) می توانید از این دستور برای بازگردندن و برافراشتن یک خطا استفاده نمایید

- **مدیریت خطاها و تراکنش ها**
  - **استفاده از دستور RaisError()**
- چنانچه خطای ای که می خواهید برافراشته نمایید، می بایست از ادامه کار جلوگیری نماید می بایست با Severity مناسب استفاده نمایید

Severity	Description
1-10	Information
11-15	Warning
16-20	Error
21-25	Fatal Error

- **مدیریت خطاها و تراکنش ها**
  - **استفاده از دستور Throw**
  - درصورتیکه در بلوک Try Catch می خواهید خطای اتفاق افتاده در خروجی ظاهر شود می توانید از دستور Throw بدون پارامتر در بلوک Catch استفاده نمایید
  - استفاده از این دستور در تریگر ها (مخصوصا از نوع After) می تواند علاوه بر نمایش خطا تراکنش فعال را نیز RollBack نمایید

```
THROW [ { error_number | @local_variable },
        { message | @local_variable },
        { state | @local_variable } ]
[ ; ]
```

  - میتوانید پیغام خاص خود را در قسمت message در زمان برافراشتن خطا ذکر کنید تا پیغام ترجمه شده و با مفهوم را به کاربر نمایش دهد

- استفاده از اپراتور **Pivot** و **Unpivot** در دستورات **Select**
- اپراتور **Pivot**
- زمانی که بخواهید یک نمونه اطلاعات سطری را در ستون به ازای یک ستون خاص نمایش دهید می توانید از این دستور استفاده نمایید.
- قالب دستور:

```
Select <PivotColumn> , [Value1] , [Value2] ...
(<Select Statement>) As <AliasName>      From
Pivot(<Aggregate <ColName>>
For <ColName> In ([Value1] , [Value2] ...)) As <ResultAlias>
```

- به صورت معمول عملیات **Pivot** بر روی یک ستون انجام می شود (نسبت به آن ستون اطلاعات از سطر به ستون ها منتقل می شود)
- مقادیر ای که قرار است از سطر به ستون ها منتقل شود باید ثابت و مشخص باشند (می بایست مقادیر را ذکر کنید)
- از این دستور به منظور تهیه خروجی مناسب برای نمایش در نمودارهای اطلاعاتی استفاده می شود

- استفاده از اپراتور **Pivot** و **Unpivot** در دستورات **Select**
- اپراتور **Pivot**
- مثال: نام کتاب ها را به همراه تعداد نویسنده، مترجم و مولف (در ستون) مشخص نمایید:

```
Select  BookName , [1] As [نویسنده], [2] As [مولف] , [3] As [مترجم]
From    (
Select  Book.Title                      As BookName
,       BookAuthor.ActivityTypeID       As ActivityTypeID
From    Book
Inner Join
        BookAuthor
On      Book.ID         = BookAuthor.BookID
)       As Result
Pivot (Count(Result.ActivityTypeID) For Result.ActivityTypeID In ([1] , [2] , [3])) As PivotResult
```

- **استفاده از اپراتور Pivot و Unpivot در دستورات Select**
- **اپراتور UnPivot**
- درصورتیکه بخواهید اطلاعات ستون ها را به صورت سطری نمایش دهید از این اپراتور می توانید استفاده نمایید.
- از این دستور برای مواجهه با اطلاعاتی که نرمال نیستند استفاده نمایید (اطلاعات غیر نرمال: اطلاعاتی است که بجای اینکه به صورت سطری ارائه شوند به صورت ستونی ارائه می شوند)
- قالب دستور:

```
Select <FixedColName> , <UnpivotColName1>,...
For(<Sleect Statement>) As <AliasName>
Unpivot( <UnpivotColName1> For <UnpivotColName2> In ([Values])) As <UnpivotAlias>
```

- استفاده از اپراتور **Pivot** و **Unpivot** در دستورات **Select**
- اپراتور **UnPivot**
- مثال: فرض کنید اطلاعات جدول زیر را در مورد تعداد استخدام مشاغل مختلف داشته باشید:

Year	Specialist	Manager	Supervisor	Staff
۱۳۸۵	۴	۳	۵	۱۳
۱۳۸۶	۲	۱	۶	۲۰
۱۳۹۰	۲	۲	۴	۱۲
۱۳۹۴	۳	۱	۲	۱۱

- **استفاده از اپراتور Pivot و Unpivot در دستورات Select**
  - اپراتور UnPivot
  - آنچه مشخص است شما نیاز دارید که اطلاعات فوق را به صورت سطری نمایش دهید. به این معنی که مشخص نمایید در هر سالاز چه شغلی چه تعدادی را به صورت سطری داشته باشید:

Year	Job	Amount
۱۳۸۵	Specialist	۴
۱۳۸۵	Manager	۳
۱۳۸۵	Supervisor	۵
۱۳۸۵	Staff	۱۳
۱۳۸۶	Specialist	۱
۱۳۸۶	Manager	۱
...		

- استفاده از اپراتور **Pivot** و **Unpivot** در دستورات **Select**
- اپراتور **UnPivot**

```
Declare @Tabl1 Table([Year] int , Specialist int , Manager int , Supervisor int , Staff int )
Insert @Tabl1
 Values  (1385 , 3 , 1 , 5 , 10),
         (1386 , 2 , 2 , 3 , 9 ),
         (1390 , 1 , 5 , 8 , 14 ),
         (1391 , 2 , 2 , 5 , 8 ),
         (1395 , 0 , 4 , 5 , 3 )

Select  *
 From    @Tabl1

Select  [Year] , Job , Amount
 From    (       Select  *
                 From    @Tabl1
         )       As Result
 Unpivot (Amount For Job In ([Specialist] , [Manager] , [Supervisor] , [Staff])) As UnPivotResult
```

SQL Server 2016 Implementation

نکات و دستورات تکمیلی

- استفاده از **Merge**
- تعریف: چنانچه دو منبع داده ای دارید، وقصد دارید از یکی از منابع، اطلاعات منبع دیگر را به روز رسانی نمایید می توانید از این دستور استفاده نمایید.
- در این قالب دستور یک منبع به عنوان **Source** و دیگری به عنوان **Target** در نظر گرفته می شود.
- مشخصا اطلاعات در جدول **Target** از منبع **Source** به روز آوری می شود. به همین دلیل معمولا جدول **Target** از جدول های بانک اطلاعاتی است و منبع **Source** بر اساس پردازش اطلاعات خاصی است که آن را مهیا نموده اید.
- قالب دستور:

```
Merge <Source Table> As <Target Alias Name>
Using (
    <Select Statement>
As <Source Alias Name>                    )
On (<Boolean Expressions>)
When Matched Then
    <Match Statement>
When Not Matched
<Not Matched Statement>;
```

- **استفاده از Merge**
  - بر اساس شرایط شما می توانید در حالت Matched و Not Matched از دستورات Insert, Update , Delete برای اعمال تغییرات بر روی جدول Targetاستفاده نمایید
  - می توانید عبارات Matched و Not Matched را با گزینه های کنترلی دیگر And و Or نمایید
  - مثال: فرض کنید که لیست کتاب ها، هر روز از طریق یک وب سایت در یک جدول بانک ریخته می شود. شما موظف هستید در یک زمان خاص از شبانه روز، اطلاعات این جدول را خوانده، و جدول Book را با آن همسان سازی کنید (یعنی کتاب هایی که وجود ندارند را Insert و برای آنهایی که وجود دارند، اطلاعات کتاب را را به روز رسانی کنید) این امکان را بانوشتن دستور Merge پیاده سازی نمایید. (کلید مشترک شما ISBN هر کتاب می باشد)

- **استفاده از دستور For XML**
  - از این دستور می توان برای ساخت انواع فایل های XML از روی اطلاعات جداول استفاده کرد.
  - قالب دستور:

```
Select      * | <Column1Name>,
<Column1Name>,
…
From <Object Name>
Where <Boolean Expression>
For XML {Raw | Auto | Explicit | Path}
```

  - بر اساس نیاز می توان با هر یک از حالت های یاد شده قالب فایل XML را تغییر و تولید کرد.

- **Cursor ها و نحوه استفاده از آنها**
  - از Cursor ها برای بررسی رکورد به رکورد یک ResultSet تولید شده توسط دستورات Select استفاده می شود
  - به طور کلی Cursor ها منابع زیادی از SQL را در اختیار می گیرند و تا حد امکان باید از استفاده از آنها برای Result هایی که تعداد رکورد آنها زیاد است پرهیز کرد
  - به طور کلی یک Cursor از قسمت های زیر تولید می شود:
  - تعریف اولیه که شامل نام Cursor است:
  - Declare <CursorName> Cursor For (<Select Statement>)
    - نام یک Cursor در دسته ای از دستورات باید Unique باشد
    - با انجام این کار نتیجه خروجی از Select Statement در کرسر قرار داده می شود و آماده استفاده می باشد
  - باز کردن Cursor: به منظور استفاده از Cursor ها باید آنها را با دستور Open به شکل زیر باز نمود
  - Open <CursorName>

Microsoft® SQL Server®

SQL Server 2016 Implementation

نکات و دستورات تکمیلی

- **بررسی امکانات T-SQL**
- **Cursor ها و نحوه استفاده از آنها**
  - به منظور خواندن رکورد به رکورد موجود در Cursor از دستور Fetch Next به شکل زیر استفاده می شود
  - Fetch [Next| Prior | Last | First] From <CursorName> Into <Variable Names>
  - نکته: بعد از باز کردن یک کرسر، اشاره گر آن به رکورد قبل از اولین رکورد اشاره می نماید (در اصطلاح در BOF قرار دارد) بنابراین برای خواندن اولین رکورد نیز باید از دستور Fetch Next استفاده نمود
  - این دستور هر رکورد از Cursor را در متغیر های تعریفی قرار می دهد. به طور کلی این دستور باید قبل و درون حلقه استفاده شود تا بتوان با استفاده از Cursor ها ردیف به ردیف موجود در Cursor را خواند و عملیات مورد نظر را بر روی رکورد ها انجام داد.
  - هر ردیف اطلاعات از Cursor ها در متغیر های موجود قرار داده می شود و می توان بر اساس مقادیر متغیر ها عملیات مورد نظر را انجام داد
  - به منظور خواندن رکورد به رکورد اطلاعات Cursor می توان از دستور While به شکل زیر استفاده نمود
  - While @@Fetch_Status = 0
  - Begin
  - Fetch Next From <Cursor> Into <Variable Names>
  - End

- **بررسی امکانات T-SQL**
- **Cursor ها و نحوه استفاده از آنها**
  - به منظور کنترل خواندن اطلاعات از Cursor از متغیر Global به نام @@Fetch_Status استفاده می شود. زمانی که اشاره گر کرسر به انتها رسیده باشد این متغیر مقدارش مخالف صفر خواهد شد
  - بعد از پایان پذیرفتن عملیات بر روی Cursor باید توسط دستور Close به شکل زیر کرسر را ببندید:
  - Close <CursorName>
  - در انتها به منظور آزاد سازی منابع استفاده شده در Cursor باید از دستور زیر استفاده نمایید.
  - Deallocate <CursorName>
  - تا زمانی که یک Cursor توسط دستور Deallocate پایان نپذیرد، نمی توان کرسر دیگری با این نام تعریف نمود.

- **بررسی امکانات T-SQL**
- **Cursor ها و نحوه استفاده از آنها**
  - انواع Cursor
  - کرسرها دارای انواع مختلفی است که در زمان تعریف می توانید از آنها استفاده نمایید. قالب کلی دستور به شکل زیر است:

```
DECLARE cursor_name CURSOR [ LOCAL | GLOBAL ]
    [ FORWARD_ONLY | SCROLL ]
    [ STATIC | KEYSET | DYNAMIC | FAST_FORWARD ]
    [ READ_ONLY | SCROLL_LOCKS | OPTIMISTIC ]
    [ TYPE_WARNING ]
    FOR select_statement
    [ FOR UPDATE [ OF column_name [ ,...n ] ] ]
[;]
```

  - براساس نوع Cursor تعریف شده، کرسر می تواند از منابع مختلفی استفاده نماید

- بررسی امکانات T-SQL
- Cursor ها و نحوه استفاده از آنها

Type	Description
FORWARD_ONLY	این نوع Cursor ها تنها می توانند به جلو حرکت کنند و نمی توان با استفاده از دستورات First , Last, Prior ,... در کنار Fetch برای حرکت به ابتدا و یا عقب استفاده نمود. این نوع پیش فرض دستور است.
SCROLL	این نوع Cursor ها می توانند به جلو و عقب حرکت کنند و می توان با استفاده از دستورات First , Last, Prior ,... در کنار Fetch استفاده نمود
STATIC	این نوع که پیش فرض تعریف Cursor است یک کپی موقت از رکورد ها را در بانک tempdb قرار می دهد و در این حالت هیچ تغییری را بر روی داده های اصلی منعکس نمی نماید
KEYSET	در این Cursor تغییر داده ها توسط کاربران به Cursor منتقل شده ولی افزودن و حذف کردن رکورد ها به Cursor منتقل نمی گردد
DYNAMIC	این Cursor اطلاعات رکورد ها را داینامیک در اختیار می گیرد یعنی چتاتچه کاربر دیگری تغییراتی را بر روی داده های ایجاد نماید به Cursor منتقل می گردد
READ_ONLY	کرزی ایجاد می نماید که تغییرات بر روی داده ها را امکان پذیر نمی نماید
SCROLL_LOCK	نوعی Cursor ایجاد می نماید که تضمین می نماید رکورد ها را همانطور که خوانده است برای تغییرات دیگر کاربران قفل نماید
FAST_Forward	تولید Cursor ای از نوع Forward_Only و Read_Only به صورت ترکیبی