دستورات SQL

درس ایجاد بانک های اطلاعاتی

محمدعلی عظیمی

SQL چیست؟

- SQL (Standard Query Language، یا زبان پرس و جوی استاندارد)
- SQL یک زبان استاندارد برای ایجاد، دسترسی و دستکاری پایگاه‌های داده رابطه‌ای است. با وجود اینکه SQL استاندارد است، اما تولید کنندگان DBMS،پیاده سازی‌های مختلفی از آن ارائه داده اند، که هر چند با اغلب دستورات استاندارد ANSI/ISO سازگار هستند، (به خصوص در دستورات پرکاربرد مانند Select، Update، Delete و Insert)، ولی در برخی از دستورات تفاوت دارند.

انواع داده ای در SQL Server

✓ انواع داده های تعریف شده در SQL Server

- اعداد صحیح
- داده های متنی
- داده های دسیمال
- داده های ارزی
- داده های با ممیز شناور
- داده های نوع تاریخ
- داده های ویژه

داده های نوع صحیح

۵ نوع داده از انواع داده های صحیح برای ذخیره سازی اعداد صحیح در SQL Server در اختیار کاربران قرار دارد:

- bit - می توانند مقادیر ۰ ، ۱ یا تهی را ذخیره کنند. برای ذخیره سازی Yes/NO یا True/False
- tinyint - می توانند اعداد صفر تا 255 ، یا تهی را ذخیره کنند.
- smallint - می توانند اعداد ۳۲۷۶۸- تا ۳۲۷۶۷ یا تهی ذخیره کنند.
- int - می توانند اعداد ۲۱۴۷۴۸۳۶۴۸- تا ۲۱۴۷۴۸۳۶۴۷ یا تهی ذخیره کنند.
- bigint - می توانند اعداد ۹۲۲۳۳۷۲۰۳۶۸۵۴۷۷۵۸۰۸- تا

۹۲۲۳۳۷۲۰۳۶۸۵۴۷۷۵۸۰۷ یا تهی ذخیره کنند.

داده های نوع متنی

۴ نوع مختلف برای نگهداری داده های متنی در SQL Server در اختیار کاربران است.

- char – تعداد محدودی کاراکتر غیر یونی کد را در خود جای می دهند. یعنی ستونی از نوع char(20) همیشه ۲۰ کاراکتر را ذخیره می کند ، حتی اگر رشته ای کوتاهتر از ۲۰ کاراکتر به آن اختصاص یابد.

- varchar- تعداد کاراکترهای قابل ذخیره در ستون های varchar متغیر است ، یعنی مثلا ستونی از نوع varchar(20) حداکثر ۲۰ کاراکتر را ذخیره خواهند کرد. حداکثر اندازه این نوع داده ها ۸۰۰۰ کاراکتر است و اگر از فرمت varchar(max) استفاده کنیم می توان ۲ به توان ۳۱ کاراکتر را ذخیره کرد.

داده های نوع متنی

• nchar– تعداد محدودی کاراکتر یونی کد را در خود جای می دهند. یعنی ستونی از نوع char(20) همیشه ۲۰ کاراکتر را ذخیره می کند ، حتی اگر رشته ای کوتاهتر از ۲۰ کاراکتر به آن اختصاص یابد. چون هر کاراکتر یونی کد نیاز به ۲ بایت برای ذخیره سازی دارد حداکثر اندازه این نوع ۴۰۰۰ کاراکتر است.

•nvarchar- تعداد کاراکترهای یونی کد قابل ذخیره در ستون های nvarchar متغیر است ، یعنی مثلا ستونی از نوع nvarchar(20) حداکثر ۲۰ کاراکتر را ذخیره خواهند کرد. حداکثر اندازه این نوع داده ها ۴۰۰۰ کاراکتر است .

*** نکـــتـــــــــــه ***

- ✓ بهتر است برای ذخیره سازی داده های متغیر از انواع nchar , nvarchar استفاده شود.
- ✓ داده هایی که تمامی مقادیر آن یک اندازه است بهتر است از char , varchar استفاده شود زیرا به مراتب سریعتر عمل می کند.
- ✓ از انواع text , ntext برای داده های بیشتر از ۸۰۰۰ کاراکتر استفاده می شود.
- ✓ انواع داده های یونی کد ntext , nvarchar , nchar تنها در صورت احتمال وجود کاراکترهای ویژه در داده ها استفاده می شود.

داده های نوع دسیمال

SQL Serverنوع واحدی از داده ها را برای کار با اعداد اعشاری دارای ممیز شناور ، بدون گرد کردن آن با دو نام numeric , decimal در اختیار کاربران قرار می دهد.

- زمانی که ستونی از نوع decimal , numeric تعریف می شود دقت و تعداد ارقام اعشاری آن را باید مشخص کنیم.

- منظور از دقت تعداد کل ارقام قابل ذخیره است.
- منظور از تعداد رقم اعشاری ، تعداد ارقام قابل دخیره در سمت راست ممیز اعشار است.

مثلا : ستونی که به صورت decimal(20) عددی چون ۳۴۵/۱۲۱ را می تواند در خود جای دهد، حداکثر دقت این ستون ها ۳۸ است.

داده های ارزی

۲ نوع از داده ها برای ذخیره سازی داده های ارزی در SQL Server وجود دارد:

- smallmoney – محدوده مقادیر قابل ذخیره در این ستون ها ۲۱۴۷۴۸/۳۶۴۸- تا ۲۱۴۷۴۸/۳۶۴۷ می باشد . داده هایی که در این ستون ها ذخیره می شود همیشه دقیقا ۴ رقم در سمت راست ممیز اعشار خواهند داشت.

- money- محدوده مقادیر قابل ذخیره در این ستون ها ۹۲۲۳۳۷۲۰۳۶۸۵۴۷۷/۵۸۰۷- تا ۹۲۲۳۳۷۲۰۳۶۸۵۴۷۷/۵۸۰۷ می باشد . داده هایی که در این ستون ها ذخیره می شود همیشه دقیقا ۴ رقم در سمت راست ممیز اعشار خواهند داشت.

داده های با نوع ممیز شناور

بر خلاف داده های نوع دسیمال چنانچه داده های با ممیز شناور با دقت لازم در فرمت باینری داخلی سرور قابل ارائه نباشد SQL Server آنها را گرد می کند.

- float – محدوده مقادیر قابل ذخیره در این ستون ها 1/79E+38- تا 2/23E-38 می باشد .
  وقتی این نوع ستون ها تعریف می شود تعداد بیت ها از یک تا ۵۳ متغیر خوهد بود.

- real- real در SQL سرور اکسپرس مترادف float(24) است.

داده های با نوع تاریخ

دو نوع از داده ها برای ذخیره سازی تاریخ و زمان دارد ، تفاوت این دو نوع ، در محدوده تاریخ ها و دقت مورد استفاده برای ذخیره سازی است.

- smalldatetime – تاریخ های اول ژانویه ۱۹۰۰ تا شش ژوئن ۲۰۷۹ را با دقت یک دقیقه در خود جای دهند.

- datetime- ستون های از این نوع می توانند تاریخ های اول ژانویه ۱۷۵۳ تا ۳۱ دسامبر ۹۹۹۹ را با دقت ۳/۳۳ میلی ثانیه در خود جای دهند.

داده های باینری

برای ذخیره سازی داده های باینری مورد استفاده قرار می گیرد.

- binary – هر ستون باینری می تواند ۸۰۰۰ بایت از داده های باینری را ذخیره کند.

- varbinary - می توانند داده های باینری را با طول متغیر تا حداکثر اندازه تعیین شده ذخیره نمایند.

انواع داده های ویژه

انواع دیگری از داده ها در SQL Server وجود دارد که برای مواردی خاص استفاده می شود همانند: cursor , table , timestamp , uniqueidentifier , XML ...

- XML نوع جدیدی از داده هاست این نوع ستون ها ، برای ذخیره سازی کل یک سند XML به کار می روند. XML قابلیت پیشرفته ای همچون جستجو از ساختار گرامری Xquery را فراهم می سازد. ستون های XML در بیشتر موارد جایگزین بسیار بهتری برای ستون های nvarchar(max) هستند.

انواع عملگرها در SQL

- **عملگرهای محاسباتی:** عبارتند از +، -، * (ضرب) و / (تقسیم).
- **عملگرهای مقایسه ای**

در جدول زیر (جدول XXX) انواع عملگرهای مقایسه توضیح داده شده است.

شکل عملگر	معنی
>	بزرگ تر
<	کوچک تر
=	مساوی
>=	بزرگ‌تر مساوی
<=	کوچک‌تر مساوی
<>	نامساوی

انواع عملگرها در SQL

- **عملگرهای منطقی**
- SQL از چهار عملگر منطقی: AND ، OR ، NOT و IS پشتیبانی میکند. هر یک از عملگرهای فوق را در ادامه مورد بررسی قرار خواهیم داد.
- **AND**: در صورتی که همه‌ی قسمت‌ها درست باشد نتیجه درست است در غیر اینصورت نتیجه اشتباه است. جدول صحت عملگر AND در جدول زیر نمایش داده شده است.

AND	TRUE	FALSE	UNKNOWN
TRUE	True	False	Unknown
FALSE	False	False	False
UNKNOWN	Unknown	False	Unknown

انواع عملگرها در SQL

- **OR**: در صورتی که حداقل یک قسمت درست باشد نتیجه درست است در غیر اینصورت نتیجه اشتباه است. جدول صحت عملگر OR در جدول زیر نمایش داده شده است.

OR	TRUE	False	UNKNOWN
TRUE	True	True	True
False	True	False	Unknown
UNKNOWN	True	Unknown	Unknown

- **NOT**: نقیض عبارت منطقی. درست را به خراب و خراب را به درست تبدیل می‌کند.

انواع عملگرها در SQL

- **IS**: عملگری منطقی است و مشخص می کند که آیا مقدار یک داده درست، غلط یا نامشخص است. جدول صحت عملگر IS در جدول زیر نمایش داده شده است.

IS	TRUE	FALSE	UNKNOWN
TRUE	True	False	False
False	False	True	False
UNKNOWN	False	False	True

- **عملگر الحاق دو رشته**

از عملگر || برای الحاق دو رشته به یکدیگر استفاده می شود. به عنوان مثال:

'This is a ' || 'Computer' = 'This is a Computer'

در MS SQL Server از عملگر + برای الحاق دو رشته استفاده می شود.

انواع عملگرها در SQL

- **عملگرهای کار با زمان**
- عملگرهای معتبر برای نوع‌های زمان و Interval و همچنین نوع حاصل از عملگر در جدول زیر نشان داده شده.

عملوند اول	عملگر	عملوند دوم	نوع داده‌ای نتیجه
Datetime	–	Datetime	Interval
Datetime	+ or –	Interval	Datetime
Interval	+	Datetime	Datetime
Interval	+ or –	Interval	Interval
Interval	* or /	Numeric	Interval
Numeric	*	Interval	Interval

- در جدول فوق منظور از Datetime هر یک از نوع های زمان یعنی DATE، TIME و TIMESTAMP می باشد.

ایجاد جدول

- برای ایجاد جدول از ساختار دستوری زیر استفاده می شود:

CREATE TABLEنام جدول
(نام ستون اولنوع ستون اول
[, نام ستون دومنوع ستون دوم , ...]**)**

در دستور فوق، نوع ستون می تواند یکی از انواع داده ای، دامنه یا نوع تعریف شده توسط کاربر باشد.

- **مثال)** با استفاده از دستور زیر جدول نمره (Score) را ایجاد کرده ایم.

CREATE TABLE Score
(St#**CHAR(**10**),**
Pr#**CHAR(**5**),**
Cr#**CHAR(**5**),**
Mark Grade**)**

- در مثال فوق از دامنه Grade برای مشخص کردن نوع Mark (نمره) استفاده شده.

تعیین کلید اصلی

- **مثال)** در این مثال جدول درس را ایجاد می‌کنیم.

CREATE TABLE Course
(Cr# **CHAR(**5**)** **PRIMARY KEY,**
CrName **VARCHAR(**20**),**
Type**VARCHAR(**10**),**
Unit **NUMERIC(**1**))**

- **مثال)** در زیر مجدداً دستور ایجاد جدول نمره آورده شده، با این تفاوت که کلید اصلی نیز تعریف شده است.

CREATE TABLE Score
(St#**CHAR(**10**),**
Pr# **CHAR(**5**),**
Cr# **CHAR(**5**),**
MarkGrade
PRIMARY KEY (St#, Pr#, Cr#**))**

تعیین کلید اصلی

- در این مثال مجدداً جدول درس را تعریف می‌کنیم با این تفاوت که نام درس نمی‌تواند تکراری وارد شود.

```
CREATE TABLE Course
(Cr# CHAR(5) PRIMARY KEY,
 CrNameVARCHAR(20) UNIQUE,
 TypeVARCHAR(10),
 Unit NUMERIC(1))
```

جدول دانشجو (Student)							
St#	StFName	StLName	Gender	Phone	Degree	Field	Address
881015	کوروش	یزدانیان	مرد		کارشناسی	کامپیوتر	تهران
882719	سهراب	وطن دوست	مرد		کاردانی	هوا-فضا	کردستان
891430	نگار	نقشینه	زن		کاردانی	فیزیک	اصفهان
891614	علی	لاری	مرد	۰۹۱۶	کاردانی	کامپیوتر	
891615	محمدعلی	شیخی	مرد	۰۹۱۶	کاردانی	کامپیوتر	اهواز
892418	رستم	وطن دوست	مرد	۰۹۱۷	کاردانی	فیزیک	بوشهر
901034	سیدعلی محمد	بلوچی	مرد	۰۹۱۵	کارشناسی	کامپیوتر	
901214	سارا	شیرازی	زن		کارشناسی	کامپیوتر	شیراز
901345	تهمینه	شیردل	زن	۰۹۱۴	کارشناسی	فیزیک	تبریز
902316	علیرضا	ترک	مرد	۰۹۱۴	کارشناسی	فیزیک	

جدول استاد (Professor)				
Pr#	PrFName	PrLName	Phone	Address
۱۰	محمد	مصدق	۰۹۱۲	تهران
۱۱	مریم	پاکدل	۰۹۱۷	بندرعباس
۱۲	آرش	کمانگیر	۰۹۱۱	ساری
۱۳	سینا	دانشمند	۰۹۱۸	همدان
۱۴	محمد	فارابی	۰۹۱۵	مشهد

جدول درس (Course)			
Cr#	CrName	Type	Unit
۸۱	پایگاه داده	تخصصی	۳
۸۴	وب	اختیاری	۳
۸۵	مدارمنطقی	اصلی	۲
۸۶	اپتیک	اصلی	۲
۸۷	زبان تخصصی	تخصصی	۲
۸۹	ارتعاشات	تخصصی	۳

جدول نمره (Score)				
St#	Pr#	Cr#	Semester	Mark
۸۸۱۰۱۵	۱۰	۸۱	۸۸۱	۹
۸۸۱۰۱۵	۱۰	۸۱	۸۹۱	۱۸,۲۵
۸۸۲۷۱۹	۱۴	۸۹	۹۰۱	۲۰
۸۹۱۴۳۰	۱۱	۸۶	۸۹۱	۱۶
۸۹۲۴۱۸	۱۱	۸۶	۹۰۱	۱۷,۵۰
۹۰۱۰۳۴	۱۰	۸۱	۹۰۲	۲۰
۹۰۱۲۱۴	۱۰	۸۱	۹۰۲	۱۹
۹۰۱۲۱۴	۱۲	۸۴	۸۹۱	۲۰
۹۰۱۲۱۴	۱۳	۸۵	۹۰۱	۱۶,۵۰
۹۰۱۳۴۵	۱۱	۸۶	۸۹۱	۱۹,۵۰

ساختار ساده دستور Select:

SELECT نام ستون‌ها **FROM** نام جدول [شرط**WHERE**]

بعد از WHERE شروط محدود کردن جستجو نوشته می‌شود.

مثال) دستور زیر شماره دانشجویی، نام و نام خانوادگی کلیه دانشجویان را نمایش می‌دهد.

SELECT St#, StLName, stfname **FROM** Student

St#	StFName	StLName
881015	کوروش	یزدانیان
882719	سهراب	وطن دوست
891430	نگار	نقشینه
891614	علی	لاری
891615	محمدعلی	شیخی
892418	رستم	وطن دوست
901034	سیدعلی محمد	بلوچی
901214	سارا	شیرازی
901345	تهمینه	شیردل
902316	علیرضا	ترک

مثال) دستور زیر مشخصات دانشجویان دختر را بر می‌گرداند.

SELECT St#, StFName, StLName, Gender, Phone, Degree, Field, Address
FROM Student
WHERE Gender='زن'

SELECT * FROM Student
WHERE Gender='زن'

St#	StFName	StLName	Gender	Phone	Degree	Field	Address
891430	نگار	نقشینه	زن		کاردانی	فیزیک	اصفهان
901214	سارا	شیرازی	زن		کارشناسی	کامپیوتر	شیراز
901345	تهمینه	شیردل	زن	۰۹۱۴	کارشناسی	فیزیک	تبریز

مثال) دستور زیر نام درس‌های تخصصی ای که بیش از ۲ واحد هستند را نمایش می‌دهد.

SELECT CrName **FROM** Course
WHERE Type='تخصصی' **AND** Unit>2

عملگرهای DISTINCT و ALL

در پرس و جوهایی که با استفاده از دستور SELECT در SQL صورت می‌گیرند، سطرهای تکراری حذف نمی‌شوند، در صورتی که در بعضی از مواقع ما عدم وجود سطرهای تکراری را خواستاریم. برای اینکه سطرهای تکراری در خروجی ظاهر نشوند باید از دستور DISTINCT قبل از نام فیلد استفاده کرد.

مثال) دستور زیر رشته‌های موجود در دانشگاه را نمایش می‌دهد.

SELECT DISTINCT Field **FROM** Student

خروجی دستور فوق کامپیوتر، فیزیک و هوا-فضا خواهد شد، بدون اینکه نام رشته ها تکرار شود.

اگر به جای DISTINCT از ALL استفاده شود به این معنی است که سطرهای تکراری حذف نشوند، که به صورت پیش فرض چنین است.

عملگر BETWEEN

این عملگر برای بررسی مقدار یک فیلد در یک محدوده است و در قسمت شرط آورده می‌شود. ساختار کلی آن به صورت زیر است:

WHERE نام فیلد **[NOT]BETWEEN**

[{ASYMMETRIC | SYMMETRIC}]

حد بالا**AND**حد پایین

MS SQL Server از گزینه های ASYMMETRIC و SYMMETRICپشتیبانی نمی کند.

مثال) دستور زیر نمره و شماره دانشجویانی که نمره بین ۱۶ تا ۱۸ گرفته‌اند را مشخص می کند.

نکته: حد پایین و حد بالا می‌توانند از نوع رشته‌ای نیز باشند، که در این صورت ترتیب حروف الفبا (کد کرکترهای) بین دو حد در نظر گرفته می‌شود. همچنین از نوع زمان هم می توان استفاده کرد.

با استفاده از عملگر NOT می‌توان مواردی را یافت که خارج از محدوده باشند.

دستور	خروجی
SELECT St#, Mark **FROM**Score **WHERE** Mark **BETWEEN** 16 **AND** 18	St# | Mark 891430 | 16.00 892418 | 17.50 901214 | 16.50

با استفاده از عملگر NOT می‌توان مواردی را یافت که خارج از محدوده باشند.

مثال) دستور زیر نمره و شماره دانشجویانی که نمره ای خارج از محدوده ۱۶ تا ۱۸ گرفته‌اند را مشخص می کند.

دستور	خروجی

```
SELECT St#, Mark FROM Score
WHERE Mark NOT BETWEEN 16 AND 18
```

St#	Mark
881015	9
881015	18.25
882719	20
901034	20
901214	19
901214	20
901345	19.50

عملگر LIKE

این دستور برای جستجو در مقادیر رشته‌ای است، و زمانی استفاده می‌شود که به دنبال حروف یا کلماتی در یک رشته بگردیم. برای مثال اگر بخواهیم بدانیم که در نام شخص، کلمه "علی" وجود دارد یا خیر، یا اینکه اسامی ای که با حرف الف شروع می‌شوند کدامند از این دستور استفاده می شود. قالب کلی دستور LIKE به صورت زیر است:

WHERE نام فیلد **LIKE** 'قالب رشته' [**ESCAPE**'کرکتر گریز']

در LIKE از دو علامت استفاده می‌شود: ۱- "٪" که جایگزین هر مجموعه کرکتری می‌شود، ۲- "_" (زیر خط) که جایگزین دقیقاً یک کرکتر می‌شود.

مثال) دستور زیر مشخصات دانشجویانی که در نام آنها کلمه علی وجود دارد را بر می‌گرداند.

```
SELECT * FROM Student
WHERE StFName LIKE '%علی%'
```

حاصل جستجوی فوق جدول زیر می شود.

St#	StFName	StLName	Gender	Phone	Degree	Field	Address
891614	علی	لاری	مرد	۰۹۱۶	کاردانی	کامپیوتر	
891615	محمدعلی	شیخی	مرد	۰۹۱۶	کاردانی	کامپیوتر	اهواز
901034	سیدعلی محمد	بلوچی	مرد	۰۹۱۵	کارشناسی	کامپیوتر	
902316	علیرضا	ترک	مرد	۰۹۱۴	کارشناسی	فیزیک	

مثال) دستور زیر مشخصات دانشجویانی که اولین حرف نام آنها "س" و چهارمین حرف نامشان "الف" باشد را بر می‌گرداند.

```
SELECT *  FROM Student
WHERE StFName LIKE 'س_ _ا%'
```

حاصل جستجو جدول زیر می شود.

در بعضی موارد ممکن است علامت‌های "٪" و یا "_" خود جزء رشته‌ای باشند که قصد پیدا کردنش داریم. در چنین مواقعی از عبارت ESCAPE کمک می‌گیریم. کرکتری که بعد از عبارت ESCAPE نوشته می‌شود گویای آن است که کرکتر بعد از آن، کرکتر معمولی است، نه کرکتر عملیاتی.

St#	StFName	StLName	Gender	Phone	Degree	Field	Address
882719	سهراب	وطن دوست	مرد		کاردانی	هوا-فضا	کردستان
901214	سارا	شیرازی	زن		کارشناسی	کامپیوتر	شیراز

مثال) دستور زیر مشخصات دانشجویانی که آدرس آنها در خیابان ولیعصر شهر تهران می‌باشد را بر می‌گرداند.

```
SELECT * FROM Student
WHERE Address LIKE '%تهران_/ولیعصر%' ESCAPE'/'
```

در مثال فوق عبارتی که در فیلد آدرس جستجو می‌شود "تهران_ولیعصر" است، در واقع در اینجا کرکتر زیر خط (_) جزء رشته در نظر گرفته می شود، زیرا بلافاصله پس از کرکتر گریز تعریف شده ('/') آمده است.

توجه: دقت داشته باشید که عبارات در SQL از چپ به راست خوانده می‌شوند، یعنی در مثال فوق کرکتر بعد از '/'، زیر خط (_) می‌باشد و نه 'و'.

توجه: با استفاده از عملگر NOT قبل از LIKE می‌توان مواردی را جستجو کرد که فاقد عبارت خواسته شده باشند.

مثال) دستور زیر اسامی دانشجویانی که در نام آنها کلمه "علی" وجود ندارد را بر می گرداند.

SELECT * **FROM** Student
WHERE StFName **NOTLIKE** '%علی%'

حاصل جستجوی فوق، جدول زیر می شود.

St#	StFName	StLName	Gender	Phone	Degree	Field	Address
881015	کوروش	یزدانیان	مرد		کارشناسی	کامپیوتر	تهران
882719	سهراب	وطن دوست	مرد		کاردانی	هوا-فضا	کردستان
891430	نگار	نقشینه	زن		کاردانی	فیزیک	اصفهان
892418	رستم	وطن دوست	مرد	۰۹۱۷	کاردانی	فیزیک	بوشهر
901214	سارا	شیرازی	زن		کارشناسی	کامپیوتر	شیراز
901345	تهمینه	شیردل	زن	۰۹۱۴	کارشناسی	فیزیک	تبریز

عملگر IS NULL

این عملگر برای تشخیص تهی بودن فیلد استفاده می‌شود.

مثال) دستور زیر مشخصات دانشجویانی که آدرسشان اعلام نشده را بر می‌گرداند.

SELECT * **FROM** Student

WHERE Address **IS NULL**

حاصل جدول زیر خواهد شد.

St#	StFName	StLName	Gender	Phone	Degree	Field	Address
891614	علی	لاری	مرد	۰۹۱۶	کاردانی	کامپیوتر	
901034	سیدعلی محمد	بلوچی	مرد	۰۹۱۵	کارشناسی	کامپیوتر	
902316	علیرضا	ترک	مرد	۰۹۱۴	کارشناسی	فیزیک	

توجه: با استفاده از عملگر NOT قبل از IS NULL می‌توان فیلدهایی را پیدا کرد که تهی نیستند.

مثال) دستور زیر نام و شماره تلفن دانشجویانی که شماره تلفن خود را اعلام کرده‌اند را بر می‌گرداند.

دستور	خروجی

SELECT StFName, StLName, Phone
FROM Student
WHERE Phone **IS NOT NULL**

StFName	StLName	Phone
علی	لاری	۰۹۱۶
محمد علی	شیخی	۰۹۱۶
رستم	وطن دوست	۰۹۱۷
سید علی محمد	بلوچی	۰۹۱۵
تهمینه	شیردل	۰۹۱۴
علیرضا	ترک	۰۹۱۴

عملگر IN

با استفاده از عملگر IN می‌توان وجود یک داده را در یک مجموعه مشخص کرد. عملگر IN در قسمت شرط استفاده می‌شود.

مثال) دستور زیر مشخصات کامل جدول Score را در ترم‌های مهر (اول) بین سال‌های ۸۶ تا ۸۹ را بر می‌گرداند.

دستور	خروجی

SELECT*FROMScore
WHERESemester**IN**
('861', '871', '881','891'**)**

St#	Pr#	Cr#	Semester	Mark
۸۸۱۰۱۵	۱۰	۸۱	۸۸۱	۹
۸۸۱۰۱۵	۱۰	۸۱	۸۹۱	۱۸ ۲۵
۸۹۱۴۳۰	۱۱	۸۶	۸۹۱	۱۶
۹۰۱۲۱۴	۱۲	۸۴	۸۹۱	۲۰
۹۰۱۳۴۵	۱۱	۸۶	۸۹۱	۱۹ ۵۰

با استفاده از عملگر NOT قبل از IN می‌توان مواردی را انتخاب کرد که جزء مجموعه نباشند.

مثال) دستور زیر مشخصات کامل جدول Score را در ترم‌های به غیر از مهر (یعنی ترم بهمن و تابستان) بین سال‌های ۸۶ تا ۸۹ را بر می‌گرداند.

دستور	خروجی

SELECT* FROM Score
WHERE Semester**NOTIN**
('861', '871', '881', '891'**)**

St#	Pr#	Cr#	Semester	Mark
۸۸۲۷۱۹	۱۴	۸۹	۹۰۱	۲۰
۸۹۲۴۱۸	۱۱	۸۶	۹۰۱	۱۷ ۵۰
۹۰۱۰۳۴	۱۰	۸۱	۹۰۲	۲۰
۹۰۱۲۱۴	۱۰	۸۱	۹۰۲	۱۹
۹۰۱۲۱۴	۱۳	۸۵	۹۰۱	۱۶ ۵۰

عملگر ORDER BY (مرتب سازی)

- با استفاده از این عملگر می‌توان نتیجه ی جستجو را مرتب کرد.
- **ORDER BY** اسامی فیلدها [**ASC| DESC**]}{**NULLS FIRST| NULLS LAST**
- ASC: باعث مرتب شدن داده‌ها به صورت صعودی می‌شود.اگر نوشته نشود نیز پیش فرض صعودی است.
- DESC: باعث مرتب شدن داده‌ها به صورت نزولی می‌شود.
- NULLS FIRST: رکوردهایی که فیلد مورد نظر در آنها تهی هستند را در ابتدا نمایش می‌دهد.
- NULLS LAST: رکوردهایی که فیلد مورد نظر در آنها تهی هستند را در انتها نمایش می‌دهد.
- MS SQL Server از گزینه های NULLS FIRST و NULLS LAST پشتیبانی نمی کند و در مرتب سازی داده های تهی را به عنوان کوچکترین داده در نظر می گیرد.

مثال) دستور زیر اسامی دانشجویان مقطع کارشناسی را به صورت مرتب شده نزولی بر اساس نام خانوادگی نمایش می‌دهد.

خروجی	دستور

StFName	StLName
کوروش	یزدانیان
تهمینه	شیردل
سارا	شیرازی
علیرضا	ترک
سیدعلی محمد	بلوچی

```
SELECT StLName, StFName
FROMStudent
WHEREDegree='کارشناسی'
ORDERBYStLNameDESC
```

مثال) دستور زیر اسامی دانشجویان مقطع کاردانی را بر اساس نام خانوادگی به صورت صعودی مرتب می‌کند و در صورتی که نام خانوادگی دو نفر یکسان باشد، بر اساس نام آنها مرتب می‌کند (به صورت صعودی)، و در صورتی که فیلدهایی تهی باشند آنها را در انتها نمایش می‌دهد.

دستور	خروجی

SELECT StLName, StFName
FROM Student
WHERE Degree='کاردانی'
ORDER BY StLName **ASC NULLS LAST**,
StFName **ASC NULLS LAST**

StLName	StFName
شیخی	محمدعلی
لاری	علی
نقشینه	نگار
وطن دوست	رستم
وطن دوست	سهراب

عملگر AS (نام مستعار)

در SQL این امکان وجود دارد که نام مستعار موقتی برای فیلدها و یا جداول تعریف کنیم.

مثال) دستور زیر دو برابر نمراتی که کمتر از ۱۷ هستند را محاسبه می‌کند و نام ستون جدید را Mark2 می‌گزارد.

دستور	خروجی
SELECT Mark * 2 **AS** Mark2 **FROM** Score **WHERE** Mark < 17	**Mark2** 18 32 33

توابع آماده SQL

الف) توابع عددی

نام تابع	عملکرد	مثال
MOD(عدد, مقسوم علیه)	باقیمانده تقسیم (مودولوس)	**MOD**(25,2) → 1
FLOOR (عدد)	کف	**FLOOR**(12.6) → 12
CEIL (عدد) **CEILING** (عدد)	سقف	**CEIL**(12.6) → 13 **CEILING**(12.6) → 13
LN (عدد)	لگاریتم طبیعی	**LN**(16) → 2.772
EXP (عدد)	نمایی، $e^{\text{عدد}}$	**EXP** (3) →20.08
POWER (پایه ,توان)	توان	**POWER**(2,3) → 8
ABS (عدد)	قدر مطلق	**ABS**(-6) → 6
SQRT (عدد)	جذر	**SQRT**(25) → 5

مثال) دستور زیر شماره دانشجویی،نمره و همچنین فقط قسمت صحیح نمرات دانشجویانی که نمره آنها بین ۱۶ و ۱۸ هستند را بر می‌گرداند.

دستور	خروجی

SELECTSt#, Mark,**FLOOR (**Mark**)**
FROMScore
WHEREMark**BETWEEN** 16 **AND** 19

	Mark	St#
۱۸	۱۸ ۲۵	۸۸۱۰۱۵
۱۶	۱۶	۸۹۱۴۳۰
۱۷	۱۷ ۵۰	۸۹۲۴۱۸
۱۹	۱۹	۹۰۱۲۱۴
۱۶	۱۶ ۵۰	۹۰۱۲۱۴

ب) توابع رشته ای

۱.مشخص کردن موقعیت یک متن در متن دیگر

در MS SQL Server برای جای تابع MOD باید از عملگر ٪ استفاده کرد، بدین شکل: مقسوم علیه ٪ عدد. همچنین به جای LN باید از تابع LOG(N,Base) استفاده کرد.

POSITION(رشته مورد جستجو**IN**متن **)**

این تابع موقعیت "رشته مورد جستجو" را در "متن" می یابد.

۲.بدست آوردن طول رشته

CHAR_LENGTH(رشته کرکتری**)**

CHARACTER_LENGTH (رشته کرکتری**)**

هر دو تابع یکسان هستند و طول "رشته کرکتری" تعیین شده را مشخص می کنند.

در MS SQL Server از تابع*(رشته کرکتری) LEN* استفاده می شود.

۳. زیر رشته

این تابع از شماره کرکتری که در قسمت "شماره کرکتر شروع" از "رشته ی کرکتری" به تعداد "طول زیر رشته" بر می گرداند.

SUBSTRING(رشته کرکتری**FROM**شماره کرکتر شروع [**FOR** طول زیر**)**

در MS SQL Server ساختار تابع فوق بدین شکل می باشد:

SUBSTRING(رشته کرکتری**,**شماره کرکتر شروع**,**طول زیر رشته**)**

۴. تبدیل متن به حروف بزرگ

UPPER(رشته کرکتری**)**

این تابع کلیه حروف متن را تبدیل به حروف بزرگ می کند.

مثال:

UPPER('Computer'**)**→ COMPUTER

۵. تبدیل متن به حروف کوچک

LOWER(رشته کرکتری**)**

این تابع کلیه حروف متن را تبدیل به حروف کوچک می کند.

مثال:

LOWER('CoMpuTEr**')**→ computer

۶. حذف یک کرکتر از ابتدا یا انتهای رشته

TRIM([[{**LEADING**|**TRAILING**|**BOTH**}] [کرکتر] **FROM**] رشته کرکتری)

LEADING: حذف از ابتدای رشته (سمت چپ)

TRAILING: حذف از انتهای رشته (سمت راست)

BOTH: حذف از ابتدا و انتها

اگر کرکتری که باید حذف شود، نوشته نشود، پیش فرض فضای خالی می باشد. همچنین پیش فرض BOTH می باشد.

مثال) دستور زیر اسامی دانشجویان با حذف فضاهای خالی احتمالی در ابتدا و انتهای نام را بر می‌گرداند.

SELECTTRIM(BOTH ' 'FROM StFName**)**
FROMStudent

در MS SQL Server باید از توابع LTRIM و RTRIM استفاده کرد، هر دو تابع فقط یک پارامتر ورودی دارند که آن هم رشته کرکتری است، البته این دو تابع فقط کرکترهای فضای خالی سمت راست و سمت چپ رشته را حذف می کنند.

۷. ادغام کردن دو رشته

OVERLAY(رشته کرکتری اول**PLACING**رشته کرکتری دوم **FROM** شماره کرکتر [**FOR**به طول])

این تابع "رشته کرکتری دوم" را در موقعیت "شماره کرکتر" تعیین شده از "رشته کرکتری اول" قرار می دهد، می توان تعداد کرکتر رشته دوم که باید در رشته اول ادغام شود را نیز مشخص کرد ("به طولFOR").

مثال)

OVERLAY('This Computer'**PLACING** ' is a'**FROM** 5)→ 'This is a Computer'

MS SQL Server تابع آماده ای برای این منظور ندارد.

پ) توابع زمان

۱. استخراج کردن فیلدهای زمان

EXTRACT (داده از نوع زمان**FROM**<تعیین زمان>**)**

<تعیین زمان> : زمانی است که قصد استخراج کردن آن از "داده از نوع زمان" داریم، و یکی از موارد زیرمی تواند باشد:

YEAR, MONTH,DAY, HOUR, MINUTE,SECOND, TIMEZONE_HOUR, TIMEZONE_MINUTE

در MS SQL Server باید از تابع DATEPART (datepart , date) استفاده کرد.

۲. تاریخ جاری

CURRENT_DATE

در MS SQL Server باید از تابع ( ) GETDATE استفاده کرد.

۳. ساعت فعلی

CURRENT_TIME

در MS SQL Server باید از تابع ( ) GETDATE استفاده کرد.

۴. ساعت منطقه جغرافیایی

LOCALTIME

۵. زمان جاری

CURRENT_TIMESTAMP

۶. زمان منطقه جغرافیایی

LOCALTIMESTAMP

مثال) دستور زیر فقط ماه تاریخ فعلی سیستم را نمایش می دهد.

EXTRACT (MONTH FROM CURRENT_DATE)

مثال) دستور زیر نام و سن دانشجویان را نمایش می‌دهد (در این مثال فرض گرفته شده که ستون تاریخ تولد - BirthDate - برای دانشجویان وجود دارد).

SELECTStFName, **CURRENT_DATE** –BirthDate**AS** Age

FROMStudent

ث) توابع مجموعه ای

ساختار کلی استفاده از توابع مجموعه ای به شکل زیر است:

[(شرط**WHERE**) **FILTER**] (گروه داده‌ها [{**ALL** |**DISTINCT**}])نام تابع

با استفاده از قسمت FILTER می توان عملکرد تابع را محدود به سطرهایی کرد که شرط بعد از WHERE در آنها صادق باشد.

اسامی و عملکرد توابع مجموعه ای در جدول زیر مشخص شده اند.

نام تابع	عملکرد تابع
AVG	میانگین گروه داده‌ها را محاسبه می‌کند
MAX	بزرگ ترین داده از بین گروه داده‌ها را برمی گرداند
MIN	کوچک ترین داده از بین گروه داده‌ها را برمی گرداند
SUM	حاصل جمع گروه داده‌ها را بر می‌گرداند
EVERY	اگر همه ی داده ها TRUE باشند حاصل TRUE می شود در غیر اینصورت حاصل FALSE می شود
ANY	اگر حداقل یک داده TRUE باشد حاصل TRUE است در غیر اینصورت حاصل FALSE می شود
SOME	همانند ANY
COUNT	تعداد گروه داده‌ها را بر می‌گرداند

MS SQL Server از توابع EVERY، ANY و SOME پشتیبانی نمی کند.

مثال) دستور زیر تعداد دانشجویان پسر را بر می‌گرداند.

```
SELECTCOUNT(*)
FROMStudent
WHEREGender='مرد'
```

مثال)دستور زیر میانگین نمرات دانشجویان در ترم اول ۹۰ را بر می‌گرداند.

```
SELECTAVG(Mark)
FROMScore
WHERESemester='901'
```

عملگر GROUP BY (گروه بندی)

با استفاده از این دستور می‌توان داده‌ها را دسته بندی کرد. گاهی اوقات نیاز است که محاسبات را بر روی دسته‌ای از داده‌ها انجام داد، برای مثال دانشجویان رشته کامپیوتر، دانشجویان مقطع کارشناسی ارشد، درس‌های تخصصی و غیره. در چنین مواقعی می‌توان از دستور GROUP BY استفاده کرد.

مثال) دستور زیر تعداد دانشجویان هر رشته را مشخص می‌کند.

دستور	خروجی
SELECTCOUNT(*) ASTotal, Field **FROM**Student **GROUPBY**Field	**Total** | **Field** 5 | کامپیوتر 4 | فیزیک 1 | هوا-فضا

مثال) دستور زیر مجموع تعداد واحد هر یک از انواع درس‌ها (عمومی، پایه، تخصصی و ...) را بر می‌گرداند.

دستور	خروجی
SELECTSUM(Unit**)**, Type **FROM**Course **GROUP BY**Type	<table><tr><th></th><th>Type</th></tr><tr><td>3</td><td>اختیاری</td></tr><tr><td>4</td><td>اصلی</td></tr><tr><td>8</td><td>تخصصی</td></tr></table>

دستور شرطیHAVING در GROUP BY

دستور HAVING همانند دستور WHERE می‌باشد با این تفاوت که برای شرط‌های گروه بندی شده استفاده می‌شود.

مثال) دستور زیر تعداد دانشجویان مرد در کلیه مقاطع تحصیلی را مشخص می‌کند.

دستور	خروجی
SELECT COUNT(*),Degree **FROM**Student **GROUP BY**Degree, Gender **HAVING**Gender= 'مرد'	<table><tr><th></th><th>Degree</th></tr><tr><td>4</td><td>کاردانی</td></tr><tr><td>3</td><td>کارشناسی</td></tr></table>

مثال فوق را اینچنین نیز می توان نوشت:

SELECT COUNT(*),Degree

FROMStudent **WHERE** Gender= 'مرد'

GROUP BY Degree

نکته:فیلدهایی که در قسمت HAVING نوشته می شوند، باید در قسمت GROUP BY ذکر شده باشند.

خروجی هر دو روش کاملاً یکسان می باشد. توجه داشته باشید که نمی توان WHERE را بعد از GROUP BY نوشت.

عملگرهای مجموعه‌ای: اجتماع، اشتراکو تفاضل

نحوه ی کارکرد عملگرهای اجتماع (**UNION**)، اشتراک (**INTERSECT**) و تفاضل (**EXCEPT**) در SQL دقیقاً همانند دستورات مشابه در مجموعه‌های ریاضی می‌باشد. بعد از هر یک از این سه عملگر می‌توان از عبارت‌های ALL یا DISTINCT استفاده کرد. اگر از ALL استفاده شود به این معنی است که سطرهای تکراری ناشی از ادغام دو مجموعه در خروجی آورده شود، ولی عبارت DISTINCT از سطرهای تکراری جلوگیری می‌کند. اگر عبارتی نوشته نشود به طور پیش فرض DISTINCT می‌باشد.

مثال) دستور زیر تعداد دانشجویان دختر و پسر در کلیه مقاطع تحصیلی را مشخص می‌کند.

نکته: در عملگرهای مجموعه ای ستون هایی دو طرف عملگر باید یکسان باشند.

دستور	خروجی
SELECT COUNT(*),Degree, Gender **FROM** Student **GROUP BY**Degree, Gender**HAVING**Gender= 'زن' **UNION** **SELECT COUNT(*),**Degree, Gender **FROM**Student **GROUP BY**Degree, Gender**HAVING**Gender= 'مرد'	

	Degree	Gender
1	کاردانی	زن
2	کارشناسی	زن
4	کاردانی	مرد
3	کارشناسی	مرد

مثال) دستور زیر نام خانوادگی دانشجویان مقطع کاردانی رشته کامپیوتر را در خروجی نمایش می‌دهد.

SELECT StLName**FROM**Student**WHERE**Degree= 'کاردانی'

INTERSECT

SELECTStLName**FROM** Student **WHERE**Field= 'کامپیوتر'

خروجی آن عبارت خواهد بود از"شیخی" و"لاری".

مثال) دستور زیر اسامی دانشجویان پسری که شماره تماس آنها نامشخص است را بر می‌گرداند.

```
SELECTStFName, StLName
FROMStudent
WHEREGender= 'مرد'
EXCEPT
SELECTStFName, StLName
FROM Student
WHEREPhone IS NOT NULL
```

خروجی عبارت خواهد بود از"کوروش یزدانیان" و"سهراب وطن دوست".

جستجو در بیش از یک جدول

در بسیاری از مواقع ما نیاز داریم که داده‌ها را همزمان از چندین جدول استخراج کنیم. برای مثال برای دانستن نمرات دانشجویی به نام "داریوش" نیاز است که از دو جدول Student و Score همزمان استفاده کنیم، زیرا نام دانشجو در جدول Student است و نمرات وی در جدول Score.

در SQL برای جستجوهایی که به بیش از یک جدول نیاز دارند روش‌های مختلفی در نظر گرفته شده، که هرچند خروجی همه‌ی آنها یکسان است ولی بسته به شرایط کارایی آنها متفاوت است. در ادامه شیوه‌های مختلف را بررسی می‌کنیم.

روش اول) نوشتن نام جداول بعد از FROM

ساده ترین روش، نوشتن اسامی جداول در قسمت FROM است بدون هرگونه دستور اضافه.

مثال) دستور زیر نمرات دانشجویانی که نامشان سارا است را مشخص می‌کند.

```
SELECTMark
FROMStudent, Score
WHERESTFName='سارا' ANDStudent.St# = Score.St#
```

حاصل، ۱۹، ۲۰ و ۱۶,۵ خواهد شد.

در مثال فوق بررسی می‌شود که نام دانشجو سارا باشد و همچنین شماره دانشجویی در هر دو جدول هم یکسان باشد.

این روش همانند پیوند شرطی (از فصل جبر رابطه ای) می باشد و اگر شرط برابر بودن فیلدهای مشترک را حذف کنیم همانند ضرب دکارتی می شود. همانطور که در فصل ششم (جبر رابطه) گفته شد ضرب دکارتی بسیار هزینه بر است.

مثال) دستور زیر نمرات درس پایگاه داده ها را نمایش می دهد.

```
SELECTMarkFROM Score, Course
WHERECrName= 'پایگاه داده' ANDScore.Cr# = Course.Cr#
```

مثال) دستور زیر مشخص می کند که "سارا" در درس پایگاه داده ها چه نمره ای (نمره هایی) گرفته.

```
SELECTMarkFROMScore,Student,Course
WHEREStFName='سارا'ANDCrName= 'پایگاه داده'AND
Student.St#= Score.St# AND Course.Cr# = Score.Cr#
```

روش دوم) استفاده از دستورات پیوند جداول

دستورات پیوند جداول نیز بعد از کلمه FROM می آیند و به سه دسته تقسیم می شوند، که عبارتند از:

ضرب دکارتی جداول

جدول عامل**CROSS JOIN**جدول منبع

در ضرب دکارتی هر یک از سطرهای جدول منبع در کلیه سطرهای جدول عامل ضرب می‌شود. بنابراین نیاز به حافظه زیاد دارد و همچنین زمانبر است.

مثال) دستور زیر سطرهای جدول نمره و دانشجو را در هم ضرب می کند، البته در صورتی که دانشجو مرد باشد.

```
SELECT*  FROM Score CROSS JOIN Student
WHERE Gender = 'مرد'
```

پیوند شرطی

در این روش پیوند دو جدول بر اساس یکی از دو حالت زیر صورت می گیرد:

الف) بر حسب شرط خاص

جدول منبع [<نوع پیوند>]**JOIN** جدول عامل **ON** شرط

ب)بر حسب ستون‌های خاص

جدول منبع [<نوع پیوند>]**JOIN** جدول عامل **USING** (نام ستون ها)

در ساختار تعریف شده<نوع پیوند>یکی از مقادیر زیر می‌تواند باشد:

INNER: فقط سطرهایی از دو جدول با هم ادغام می شوند که شرط یا ستون های مشخص شده در آنها صادق باشد.

[OUTER] LEFT: علاوه بر سطرهایی که در شرط یا ستون های مشخص شده صادق است، سایر سطرهای جدول منبع را هم می آورد.

[OUTER] RIGHT: علاوه بر سطرهایی که در شرط یا ستون های مشخص شده صادق است، سایر سطرهای جدول عامل را هم می آورد.

[OUTER] FULL: علاوه بر سطرهایی که در شرط یا ستون های مشخص شده صادق است، سایر سطرهای هر دو جدول نیز آورده می شود.

پیش فرض INNER JOIN می باشد.

مثال) دستور زیر جداول نمره و درس را با استفاده از دستور INNER JOINبه هم پیوند می دهد.

```
SELECT*FROM
Course INNER JOIN Score
ON Course.Cr# = Score.Cr#
```

خروجی حاصل از دستور فوق، جدول زیر می باشد.

Cr#	CrName	Type	Unit	St#	Pr#	Cr#	Semester	Mark
۸۱	پایگاه داده	تخصصی	۳	۸۸۱۰۱۵	۱۰	۸۱	۸۸۱	۹,۰۰
۸۱	پایگاه داده	تخصصی	۳	۸۸۱۰۱۵	۱۰	۸۱	۸۹۱	۱۸,۲۵
۸۹	ارتعاشات	تخصصی	۳	۸۸۲۷۱۹	۱۴	۸۹	۹۰۱	۲۰,۰۰
۸۶	اپتیک	اصلی	۲	۸۹۱۴۳۰	۱۱	۸۶	۸۹۱	۱۶,۰۰
۸۶	اپتیک	اصلی	۲	۸۹۲۴۱۸	۱۱	۸۶	۹۰۱	۱۷,۵۰
۸۱	پایگاه داده	تخصصی	۳	۹۰۱۰۳۴	۱۰	۸۱	۹۰۲	۲۰,۰۰
۸۱	پایگاه داده	تخصصی	۳	۹۰۱۲۱۴	۱۰	۸۱	۹۰۲	۱۹,۰۰
۸۴	وب	اختیاری	۳	۹۰۱۲۱۴	۱۲	۸۴	۸۹۱	۲۰,۰۰
۸۵	مدارمنطقی	اصلی	۲	۹۰۱۲۱۴	۱۳	۸۵	۹۰۱	۱۶,۵۰
۸۶	اپتیک	اصلی	۲	۹۰۱۳۴۵	۱۱	۸۶	۸۹۱	۱۹,۵۰

مثال) در دستور زیر مجدداً جداول درس و نمره به هم پیوند داده شده اند اما این بار با استفاده از دستور LEFT OUTER JOIN .

SELECT ***FROM** Course **LEFT OUTER JOIN**Score

ONCourse.Cr#=Score.Cr#

خروجی حاصل از دستور فوق جدول زیر می باشد، در این جدول دقت داشته باشید که درس "زبان تخصصی" با اینکه در جدول نمره وجود ندارد در خروجی آمده است.

Cr#	CrName	Type	Unit	St#	Pr#	Cr#	Semester	Mark
۸۱	پایگاه داده	تخصصی	۳	۸۸۱۰۱۵	۱۰	۸۱	۸۸۱	۹,۰۰
۸۱	پایگاه داده	تخصصی	۳	۸۸۱۰۱۵	۱۰	۸۱	۸۹۱	۱۸,۲۵
۸۱	پایگاه داده	تخصصی	۳	۹۰۱۰۳۴	۱۰	۸۱	۹۰۲	۲۰,۰۰
۸۱	پایگاه داده	تخصصی	۳	۹۰۱۲۱۴	۱۰	۸۱	۹۰۲	۱۹,۰۰
۸۴	وب	اختیاری	۳	۹۰۱۲۱۴	۱۲	۸۴	۸۹۱	۲۰,۰۰
۸۵	مدارمنطقی	اصلی	۲	۹۰۱۲۱۴	۱۳	۸۵	۹۰۱	۱۶,۵۰
۸۶	اپتیک	اصلی	۲	۸۹۱۴۳۰	۱۱	۸۶	۸۹۱	۱۶,۰۰
۸۶	اپتیک	اصلی	۲	۸۹۲۴۱۸	۱۱	۸۶	۹۰۱	۱۷,۵۰
۸۶	اپتیک	اصلی	۲	۹۰۱۳۴۵	۱۱	۸۶	۸۹۱	۱۹,۵۰
۸۷	زبان تخصصی	تخصصی	۲					
۸۹	ارتعاشات	تخصصی	۳	۸۸۲۷۱۹	۱۴	۸۹	۹۰۱	۲۰,۰۰

مثال) دستور زیر جداول درس، نمره و دانشجو را به هم پیوند می دهد.

SELECTMark, Student.St#, StFName, StLName,CrName, Semester
FROM

Course**INNER JOIN** Score **ON** Course.Cr# = Score.Cr#

INNER JOIN Student**ON** Score.St# = Student.St#

خروجی حاصل از دستور فوق، جدول زیر می باشد.

Mark	St#	StFName	StLName	CrName	Semester
۹,۰۰	۸۸۱۰۱۵	کوروش	یزدانیان	پایگاه داده	۸۸۱
۱۸,۲۵	۸۸۱۰۱۵	کوروش	یزدانیان	پایگاه داده	۸۹۱
۲۰,۰۰	۸۸۲۷۱۹	سهراب	وطن دوست	ارتعاشات	۹۰۱
۱۶,۰۰	۸۹۱۴۳۰	نگار	نقشینه	اپتیک	۸۹۱
۱۷,۵۰	۸۹۲۴۱۸	رستم	وطن دوست	اپتیک	۹۰۱
۲۰,۰۰	۹۰۱۰۳۴	سیدعلی محمد	بلوچی	پایگاه داده	۹۰۲
۱۹,۰۰	۹۰۱۲۱۴	سارا	شیرازی	پایگاه داده	۹۰۲
۲۰,۰۰	۹۰۱۲۱۴	سارا	شیرازی	وب	۸۹۱
۱۶,۵۰	۹۰۱۲۱۴	سارا	شیرازی	مدارمنطقی	۹۰۱
۱۹,۵۰	۹۰۱۳۴۵	تهمینه	شیردل	اپتیک	۸۹۱

پیوند طبیعی

پیوند طبیعی دو جدول را بر اساس ستون های مشترک پیوند می دهد.

جدول منبع **NATURAL** [<نوع پیوند>] **JOIN**جدول عامل

در دستور فوق مقادیری که <نوع پیوند> می تواند داشته باشد همان موارد توضیح داده شده در پیوند شرطی می باشد.

مثال) دستور زیر جداول درس و نمره را به هم پیوند می دهد.

SELECT*FROM Course **NATURAL LEFT JOIN** Score

این پیوند در MS SQL Server پشتیبانی نمی شود.

روش سوم: با استفاده از SELECTهای تو در تو

با چند مثال این روش را بررسی می کنیم.

مثال) دستور زیر میانگین نمرات درس پایگاه داده را مشخص می‌کند.

```
SELECTAVG(Mark)FROMScore
WHERECr# = (SELECTCr# FROMCourse
WHERECrName= 'پایگاه داده')
```

حاصل ۱۶,۵۶ خواهد شد.

باید توجه داشت که نوع بازگشتی از SELECT داخلی باید با نوعی که مقایسه می‌شود یکسان باشد. در مثال فوق نوع بازگشتی و نوع مقایسه شده، هر دو کد درس (Cr#) می‌باشند.

محدودیتی در تعداد SELECTهای متداخل وجود ندارد.

مثال) دستور زیر اسامی دانشجویانی که در درس پایگاه داده با استاد مصدق در ترم دوم ۹۰ نمره ۲۰ گرفته‌اند را مشخص می‌کند.

```
SELECTStFName, StLNameFROMStudent
WHERESt#= (SELECT St# FROMScore
WHERE(Semester='۹۰۲') AND(Mark=۲۰) AND
Cr# = (SELECTCr#FROMCourse
WHERECrName='پایگاه داده') AND
(Pr#= (SELECTPr# FROM Professor
WHEREPrLName='مصدق')))
```

نتیجه پرس و جوی فوق "سید محمد علی بلوچی" خواهد شد.

مثال) دستور زیر اسامی دانشجویانی که در ترم اول ۸۹ درس پایگاه دادهداشته‌اند را نمایش می‌دهد.

```
SELECTStFName, StLNameFROMStudent
WHERESt# IN(SELECT St# FROMScore
WHERE (Semester = '۸۹۱' AND
Cr# = (SELECT Cr# FROM Course
WHERE CrName='پایگاه داده' )))
```

حاصل "کوروش یزدانیان" می شود.

عملگر EXIST

وجود حداقل یک مورد در یک جستجو را بررسی می‌کند.

EXIST (زیر جستجو**)**

اگر کاردینالیتی زیر جستجو بزرگتر از صفر باشد نتیجه صحیح است در غیر اینصورت اشتباه می‌باشد.

SELECTهای متداخل می توانند فقط از یک جدول هم استفاده کنند. به مثال زیر توجه کنید.

مثال) دستور زیر میانگین نمره درس با کد ۸۱ را می دهد. البته فقط نمره دانشجویانی محاسبه می شود که درس ۸۶ را هم گرفته باشند.

```
SELECT AVG(Mark) FROMScore
WHERECr#='81' AND
EXIST (SELECT St# FROMScoreWHERE Cr#='86')
```

در MS SQL Server باید از عبارت EXISTS استفاده کرد.

اضافه کردن داده به جدول

برای قرار دادن داده ها در جدول از دستور INSERT استفاده می شود.
مقادیری که اضافه خواهند شد، یا مقادیر ثابت هستند و یا مقادیر حاصل از یک دستور SELECT.

در ادامه هر دو روش را بررسی می کنیم.

مقادیر ثابت

در این روش به صورت مستقیم داده هایی که باید در جدول ذخیره شوند را اعلام می کنیم. برای این منظور از ساختار دستوری زیر استفاده می شود.

INSERT INTOنام جدول[(لیست ستون ها)]**VALUES**(لیست مقادیر ستون ها)

در صورتی که لیست ستون ها نوشته نشود، ترتیب ستون ها در زمان ایجاد جدول مدنظر قرار خواهد گرفت.

مثال) دستور زیر یک رکورد به جدول Course اضافه می کند.

INSERT INTO Course (Type, Cr#, Unit , CrName)

VALUES ('تخصصی', '۸۲', ۳, 'فن آوری اطلاعات')

باید دقت داشته باشید که داده ها به ترتیب در ستون ها قرار می گیرد، یعنی "تخصصی" در Type، "۸۲" در Cr# و ... .

مثال) دستور زیر نیز یک رکورد را به جدول Course اضافه می کند. ولی این بار اسامی ستون ها نوشته نشده.

INSERT INTO Course **VALUES** ('۷۱', 'ادبیات فارسی' ,'عمومی',۲)

با اجرای دستور فوق مقادیر به همان ترتیبی که در زمان ساخت جدول ستون ها را تعریف کردیم قرار خواهد گرفت. یعنی "۷۱" در Cr#، "ادبیات فارسی" در CrName، "عمومی" در Type و ۲ در Unit قرار می گیرند.

مقادیر حاصل از دستور SELECT

در این روش مقادیری که باید ذخیره شوند را به صورت مستقیم اعلام نمی کنیم، بلکه رکوردهای حاصل از یک پرس و جو را در جدول قرار می دهیم. برای مثال ممکن است که دو جدول مشابه داشته باشیم و بخواهیم که اطلاعات یک جدول را در جدول دیگر قرار دهیم. برای این کار از ساختار دستوری زیر استفاده می شود.

INSERT INTOنام جدول[(لیست ستون ها)] پرس و جو

مثال) فرض بگیرید که جدولی مشابه جدول Course با نام LCourse داشته باشیم و بخواهیم داده های جدول LCourse را در جدول Course قرار دهیم، آنگاه از دستور زیر استفاده می کنیم.

INSERT INTO Course **SELECT *FROM** LCourse

مثال) دستور زیر نام، نام خانوادگی، تلفن، آدرس و شماره دانشجویی دانشجویان مقطع دکتری را در جدول اساتید قرار می دهد.

INSERT INTO Professor(Pr#, PrFName, PrLName,Adress)
SELECT St#, StFName, StLName, Address **FROM** Student **WHERE** Degree='دکتری'

ویرایش داده‌های موجود در جداول

در بسیاری از مواقع ما نیاز داریم که داده هایی را که قبلاً با استفاده از دستور INSERT در جدول قرار داده ایم را ویرایش کنیم. برای چنین کاری باید از دستور UPDATE استفاده کرد و ساختار آن بدین شکل می باشد:

UPDATEنام جدول**SET**

,مقدار جدید۱=نام ستون۱

[,...مقدار جدید ۲ = نام ستون ۲]

[**WHERE**شرط]

مثال) دستور زیر نمره دانشجویانی که کمتر از ۱۰ است را دو برابر می کند.

UPDATEScore**SET**

Mark=Mark*2

WHERE Mark<10

مثال) دستور زیر در فیلد آدرس دانشجو با شماره "۹۰۱۰۳۴" مقدار "چابهار" را قرار می دهد.

UPDATEStudent**SET**
Address = "چابهار"
WHERESt#= '۹۰۱۰۳۴'

مثال) دستور زیر نام، نام خانوادگی، و شماره دانشجو به شماره "۸۹۱۶۱۵" را به ترتیب برابر علی محمد، شیخیان و ۸۹۱۶۵۱ قرار می دهد.

UPDATE Student**SET**
StFName = 'علی محمد' ,
StLName = 'شیخیان' ,
St# = '۸۹۱۶۵۱'
WHERESt#= '۸۹۱۶۱۵'

حذف داده‌های موجود در جداول

اگر بخواهیم برخی یا همه ی اطلاعات درون جدول را حذف کنیم از دستور DELETE باید استفاده کرد. ساختار دستور بدین شکل می باشد:

DELETE FROM نام جدول

[WHERE شرط]

اگر قسمت WHERE نوشته نشود، همه ی داده های درون جدول حذف خواهد شد. در واقع با استفاده از این قسمت، داده هایی که باید پاک شوند را محدود می کنیم.

توجه: دستور DELETE فقط داده های درون جدول را حذف می کند و جدول کماکان پابرجاست، ولی دستور DROP TABLE هم جدول و هم داده های درون آن را پاک می کند.

مثال) دستور زیر مشخصات دانشجویانی که تلفن خود را اعلام نکرده اند را حذف می کند.

DELETEFROM Student**WHERE** Phone**IS NULL**

مثال) دستور زیر مشخصات دانشجویانی که نام آنها با کلمه "علی" شروع می شود را حذف می کند.

DELETE FROM Student**WHERE** StFName**LIKE** 'علی%'

ایجاد دید

برای ایجاد دید از ساختار دستوری زیر استفاده می شود.

CREATE VIEW نام دید [**(**لیست ستون ها**)**]

AS پرس و جو

در دستور فوق اگر لیست ستون ها تعریف نشود، ستون های حاصل از پرس و جو، ستون های دید خواهند بود.

مثال) دستور زیر یک دید به نام PrScore با ترکیب جداول استاد و درس ایجاد می کند و برخی از ستون های این دو جدول را با تغییر نام در دید ایجاد شده قرار می دهد.

CREATE VIEW PrScore (Grade, Term, PrID, PrFirstName, PrLastName,StID) **AS**

SELECTMark, Semester, Professor.Pr#, PrFName, PrLName, St#

FROM Score **INNER JOIN** Professor **ON**Score.Pr# = Professor.Pr#

از دیدهایی که قبلاً ایجاد کرده ایم می توانیم برای ایجاد دید جدید استفاده کنیم. مثال بعد این موضوع را نشان می دهد.

مثال)

CREATEVIEW PrCrSt **AS**

SELECT PrLastName, PrFirstName, Grade, StFName, StLName

FROM PrScore**INNER JOIN** Student **ON** StID = St#

حال با استفاده از دید های ساخته شده فوق انجام بسیاری از پرس و جو ها ساده تر می شود. به مثال زیر دقت کنید.

مثال)فرض کنید چنین پرس و جویی داریم: اسامی دانشجویانی که از استاد مریم پاکدل نمره بیشتر از ۱۷ گرفته اند را نمایش دهد.

اگر از دید PrCrSt ساخته شده در مثال قبل استفاده کنیم، پرس و جوی ما اینچنین می شود:

```
SELECT StFName, StLName
FROM PrCrSt
WHERE(Grade > 17) AND(PrFirstName = 'مریم')AND (PrLastName = 'پاکدل')
```

اما اگر از دید استفاده نشود پرس و جوی خواسته شده را اینچنین باید نوشت:

```
SELECTStFName, StLName
FROMProfessorINNER JOIN Score ON Professor.Pr# = Score.Pr#
INNER JOIN Student ONScore.St# = Student.St#
WHERE   (Mark > 17) AND (PrFName = 'مریم') AND (PrLName = 'پاکدل')
```

با توجه به مثال های فوق، دید و ایجاد جدول با استفاده از پرس و جو بسیار شبیه به یکدیگر هستند، اما کاملاً یکسان نیستند، این تفاوت بر می گیرد به اینکه دید یک جدول مشتق است بیشتر برای جستجو در داده ها مناسب است و عملاً جدولی ایجاد نمی شود، در صورتی که ایجاد جدول با استفاده از پرس و جو یک جدول واقعی با تمام ویژگی های یک جدول ساخته می شود.

حذف دید

با استفاده از ساختار دستوری زیر می توان دیدی که قبلاً ساخته بودیم را حذف کنیم.

DROP VIEW نام دید **{CASCADE|RESTRICT}**

در صورتی که گزینه CASCADE انتخاب شود هرجا که از دید مورد نظر استفاده شده نیز حذف می شود، و اگر از RESTRICT استفاده شود نباید از دید مورد نظر در دستوراتی از قبیل: پرس و جوها، شرط ها، تریگرها و ... استفاده شده باشد، اگر چنین باشد اجازه حذف داده نمی شود.

مثال) با استفاده از دستور فوق دید PrScore ساخته شده در مثال قبلی حذف می شود.

DROP VIEW PrScore **CASCADE**

به علت استفاده از عبارت **CASCADE** سایر دیدها و جاهایی که از این دید استفاده شده نیز حذف می شوند.

اِعلان (ASSERTION)

با استفاده از اِعلان می توان از ناسازگاری داده ها جلوگیری کرد. روش های مختلفی برایَ جلوگیری از ناسازگاری داده ها وجود دارد که قبلاً آنها را مورد بررسی قرار داده ایم (در زمان ایجاد جداول، با استفاده از دامنه، با استفاده از نوع تعریفی کاربر) و حال نحوه ی انجام این کار را با استفاه از دستور اعلان بررسی می کنیم.

باید توجه داشت که همه ی قیدها را نمی توان با تعریف دامنه و نوع تعریف شده توسط کاربر فراهم کرد. به دو مثال زیر توجه کنید:

- نباید نمره درسی که دانشجو قبلاً از آن نمره ی قبولی گرفته است را مجدداً ثبت کرد.
- برای اخذ درس کارورزی دانشجو باید حداقل ۵۰ واحد را گزرانده باشد.

دو قید فوق را نمی توان با استفاده از دامنه یا نوع تعریف شده توسط کاربر فراهم کرد. برای این کار می توان از اعلان استفاده کرد.

ساختار دستوری اعلان به شکل زیر می باشد:

CREATE ASSERTION نام اعلان

CHECK (شرط**)**

[{INITIALLY DEFERRED| INITIALLY IMMEDIATE}]

هرگاه دستور تغییری در داده های جدول داده شود، اعلان فعال می شود و چک می کند که شرط تعریف شده پس از اعمال دستور هم صادق باشد، اگر چنین بود اجازه ی اجرای دستور داده می شود در غیر اینصورت از اجرای آن جلوگیری می کند. باید توجه داشت که دستور تغییر خواسته شده ممکن است چندین قسمت داشته باشد و یا اینکه بر روی چندین فیلد صورت بگیرد. INITIALLY DEFERRED مشخص می کند که شرط را پس از انجام همه ی تغییرات بررسی کند، زیرا ممکن است تغییر اول ناقض شرط باشد اما تغییرات بعدی مجدداً شرط را برقرار کند. اما INITIALLY IMMEDIATE می گوید که با انجام تک تک تغییرات شرط چک شود.پیش فرض INITIALLY IMMEDIATE است.

نحوه ی کارکرد اعلان بدین صورت است که در زمان ایجاد، چک می کند که آیا شرط اعلان شده صادق است یا خیر؟ اگر شرط برقرار نباشد اعلان ایجاد نمی شود در غیر این صورت ایجاد می شود. پس از آن برای هر درخواست تغییر بر روی فیلدهای مورد نظر، اعلان اجرا می شود تا شرط اعلان شده را بررسی کند و در صورتی که با اِعمال آن تغییر کماکان شرط برقرار باشد اجازه ی تغییر داده می شود.

MS SQL Server از این دستور پشتیبانی نمی کند.

مثال) دستور زیر از ثبت نمره درسی که دانشجو قبلاً از آن نمره ی قبولی گرفته جلوگیری می کند.

```
CREATE ASSERTIONCheckPass
CHECK (2>=(SELECT COUNT(*) FROMScore
WHEREMark>=10
GROUP BYSt#, Cr#)))
```

در دستور SELECTفوق، سطرهای جدول نمره بر اساس شماره دانشجویی و شماره درس دسته بندی می شوند، البته فقط سطرهایی مد نظر قرار می گیرند که دانشجو در آن نمره قبولی گرفته است، سپس تعداد هر دسته را برگشت می دهد، باید تعداد هر یک از دسته ها فقط ۱ باشد، اگر ۲ یا بزرگتر شود به این معنی است که دوبار نمره ی قبولی برای شخصی ثبت شده است.

دقت داشته باشید که در جدول نمره صفات کد درس، کد استاد، کد دانشجو و ترم با هم تشکیل دهنده ی کلید اصلی هستند، لذا این اطلاع که دانشجو در ترم قبل در درس X قبول شده و مجدداً در ترم بعد هم نمره ی قبولی ثبت شود جامعیت درون رابطه ای را دچار مشکل نمی کند. به عبارت دیگر کلید فقط از ثبت دو کلید اصلی یکسان جلوگیری می کند، حال اگر کد استاد یا شماره ی ترم تغییر کند دیگر کلید نمی تواند این مورد را چک کند.

مثال) با فرض اینکه برای ثبت نمره کارآموزی، دانشجو باید حداقل ۵۰ واحد را گزرانده باشد، دستور زیر این بررسی را انجام می دهد.

```
CREATE ASSERTIONApprenticeCourse
CHECK ( NOT EXIST
((SELECTSt#FROM
Score INNER JOIN Course ON Score.Cr# = Course.Cr#
WHEREMark>=10
GROUP BY St#
HAVING SUM(Unit)<50)
INTERSECT
(SELECT St#FROM Score
WHERECr#=(SELECT Cr# FROMCourse
WHERECrName='کارآموزی'))))
```

با توجه به توضیحات فوق اعلان ها باعث سربار زیاد بر روی سیستم می شوند، بنابراین در ایجاد آنها باید توجه کافی را مبذول داشت تا فقط برای شرایط اندکی آنها مورد بررسی قرار بگیرند.

با استفاده از دستور زیر می توان یک اعلان را حذف کرد:

DROP ASSERTION نام اعلان **[CASCADE | RESTRICT]**

دستورات رویه ای SQL

دستوراتی که تا به این جا بررسی شدند، دستورات غیر رویه ای بودند، به این معنی که ما فقط خواسته ی خود را اعلام می کردیم ولی رویه انجام آن را نمی گفتیم. حال در این قسمت می خواهیم با دستوراتی آشنا شویم که توسط آن ها رویه انجام کار هم مشخص می شود. دستوراتی که در این قسمت مورد بررسی قرار می گیرند بسیار شبیه دستورات متداول در زبان های برنامه نویسی است.

بنابراین SQL هم از دستورات رویه ای و هم از دستورات غیر رویه ای پشتیبانی می کند. از آنجا که دستورات رویه ای برای اکثر دانشجویان رشته کامپیوتر و فن آوری اطلاعات کاملاً شناخته شده هستند، لذا در ادامه فقط به معرفی ساختار آنها در SQL می پردازیم.

نحوه ی تعریف متغیر در SQL بدین شکل می باشد:

DECLAREاسامی متغیرهانوع **DEFAULT**[مقدار پیش فرض]

برای مقدار دهی به متغیر ها از عبارت SET به شکل زیر استفاده می شود.

SETنام متغیر=مقدار

-در MS SQL Server قبل از نام متغیرها باید علامت @ گذاشته شود، و نحوه ی تعریف متغیر در آن بدین شکل می باشد:

DECLARE @نام متغیرنوع [=مقدار پیش فرض]

دستور شرطی IF

IF شرط۱ **THEN**
    دستورات
[ELSEIF شرط۲ **THEN**
    دستورات
ELSEIF شرط۳ **THEN**
دستورات
.
.
.
ELSE
    دستورات]
END IF

باید توجه داشت که در عبارت فوق می توان در قسمت شرط از کلیه شرط هایی که بعد از عبارت WHERE در دستور SELECT استفاده می شوند استفاده کرد، بنابراین می توان شرط های بسیار پیچیده تر از شرط های زبان های برنامه نویسی رویه ای از قبیل C# و Java نوشت.
ساختار IF در MS SQL Server بدین شکل می باشد:

IF عبارت شرطی
    { دستورات }
[**ELSE**
    { دستورات }]

دستور شرطی CASE

CASE عملوند
WHEN ۱ مقادیر **THEN** ۱ دستور
WHEN ۲ مقادیر **THEN** ۲ دستور
WHEN ۳ مقادیر **THEN** ۳ دستور

.
.
.

[**ELSE** دستور]
ENDCASE

در MS SQL Server بعد از END کلمه CASE نوشته نمی شود.

دستور حلقه تکرار FOR

FOR [نام متغیر حلقه **AS**]
[نام مکان نما **CURSOR FOR**]
دستور جستجو
DO
دستورات درون حلقه
END FOR

دستور LEAVE باعث خروج از حلقه و دستور ITERATE باعث انتقال به چندتایی بعدی و رفتن به ابتدای حلقه می شود.

MS SQL Server از حلقه FOR پشتیبانی نمی کند.

دستور حلقه تکرار WHILE

WHILE شرط **DO**

دستورات

END WHILE

ساختار دستور WHILE در MS SQL Server بدین شکل می باشد:

WHILE شرط

{ دستورات | **BREAK** | **CONTINUE** }

دستور حلقه تکرار REPEAT – UNTIL

REPEAT

دستورات

UNTILشرط

END REPEAT

MS SQL Server از این حلقه پشتیبانی نمی کند.

زیر برنامه

همانطور که در زبان های برنامه نویسی ساخت یافته مرسوم است، برنامه را به چند فعالیت جزئی تر تقسیم می کنیم و به هر قسمت یک (یا چند) وظیفه را محول می کنیم. استفاده از زیر برنامه ها مزیت های زیادی دارد. استفاده از زیر برنامه در پایگاه داده ها علاوه بر مزیت های زیر برنامه در زبان های برنامه نویسی (مانند خوانایی بیشتر، اشکال زدایی راحت تر، توسعه سریعتر، استفاده چند باره از زیر برنامه به جای نوشتن چند باره کد ها و ...) دو مزیت عمده دیگر دارد که عبارتند از:

الف) استقلال دستورات کار بر روی پایگاه داده از برنامه های کاربردی (زبان میزبان). پایگاه داده ای را در نظر بگیرید که چندین برنامه کاربردی مختلف از آن استفاده می کنند و ما نیاز داریم که دستوری (مثلاً SELECT) را تغییر دهیم. در این صورت اگر از زیر برنامه ها استفاده کرده باشیم، فقط با تغییر زیر برنامه در پایگاه داده، این تغییر در همه ی برنامه ها اعمال می شود. ولی اگر از زیر برنامه استفاده نشده باشد باید تک تک برنامه ها را تغییر داد. یا فرض بگیرید که بخواهیم برنامه کاربردی (زبان میزبان) را تغییر دهیم، اگر از توابع استفاده شده باشد همه دستورات مربوط به کار با پایگاه داده باقی خواهند ماند و فقط برنامه کاربردی است که تغییر می کند، در صورتی که اگر از زیر روال استفاده نشود کلیه دستورات کار با پایگاه داده ها مجدداً باید نوشته شود.

ب) استفاده بهینه از پهنای باند شبکه هر گاه که پایگاه داده تحت شبکه استفاده می شود. ارسال نام زیر روال و پارامترهای آن حجم کمتری اشغال می کند نسبت به زمانی که یک دستور SQL به طور کامل بخواهد ارسال شود.

ج) هرگاه زیر برنامه ها ایجاد می شوند توسط DBMS کامپایل می شود، لذا در زمان اجرا نیاز نیست مرتباً کامپایل شوند، بنابراین سرعت اجرای دستورات بیشتر می شود.

د) جلوگیری از حمله

در SQL زیر برنامه ها را به دو دسته تقسیم می کنیم: ۱- زیر روال، ۲- تابع. این دو شبیه به هم هستند با این تفاوت که زیر روال مقداری بر نمی گرداند، ولی تابع بر می گرداند.

تابع

ساختار تعریف تابع بدین شکل می باشد:

CREATE FUNCTION نام تابع ([لیست پارامترها])

RETURNS <نوع بازگشتی>

دستورات تابع

نحوه ی تعریف پارامترها بدین شکل می باشد:

([{ **IN | OUT | INOUT** }] نام پارامترنوع پارامتر)

در تعریف پارامتر معنی IN، OUT و INOUT این چنین می باشد:

IN: به معنی این است که پارامتر از نوع ورودی است.

OUT: به معنی این است که پارامتر از نوع خروجی است.

INOUT: به معنی این است که پارامتر از نوع ورودی و خروجی است.

<نوع بازگشتی>هم می تواند یکی از انواع داده ای مانند INT و CHAR باشد و هم می تواند یک جدول باشد. در صورتی که جدول باشد باید به شکل زیر مشخص شود.

RETURNS TABLE (نام ستون۱نوع ستون۱ , نام ستون۲نوع ستون۲ , ...])

مثال) تابع زیر شماره دانشجویی را به عنوان پارامتر ورودی دریافت می کند، و کارنامه ی دانشجوی مورد نظر را بر می گرداند.

```
CREATE FUNCTIONGetWorkBook (StudentCode CHAR(10))
RETURNS TABLE
RETURN
(
SELECTStFName + StLName AS StudentName, Score.St#,
PrFName + PrLName AS ProffessorName,
                CrName, Semester,Mark, Type,Unit,
                (SELECT(SUM(Mark * Unit)/SUM(Unit))
                                FROMScoreINNER JOIN Course
ON Score.Cr# = Course.Cr#
                                WHERE    (Score.St# =StudentCode))AS Average
FROMCourseINNER JOINScore
ON Course.Cr# = Score.Cr#
INNER JOINStudent
ON Score.St# = Student.St#
INNER JOINProfessor
ONScore.Pr# = Professor.Pr#
WHERE(Student.St# =StudentCode)
)
```

تابع فوق را در هم می توان در زبان های میزبان (مانند C#، VB.NET و جاوا) فراخوانی کرد و هم در سایر دستورات SQL مانند دستور زیر:

```
SELECT *FROM GetWorkBook('901214')
```

مثال) تابع زیر نام یک درس را به عنوان پارامتر ورودی دریافت می کند، سپس اگر درس مورد نظر تخصصی بود، مجموع ۳ برابر نمرات آن را بر می گرداند. اگر نوع درس اصلی بود ۲ برابر و اگر اختیاری بود ۱ برابر مجموع نمرات آن درس را بر می گرداند.

```
CREATE FUNCTIONSumMarkOfCourse(CourseName VARCHAR(20))
RETURNS REAL
BEGIN
DECLARECourseTypeVARCHAR(20)
DECLARECourseCodeCHAR(5)
DECLARESumMark REAL

SETCourseType = (SELECTType   FROM Course
WHERE   (CrName =CourseName))
SETCourseCode = (SELECTCr#FROM Course
WHERE   (CrName = CourseName))
IF(CourseType='تخصصی')
SETSumMark= (SELECTSUM(Mark * 3) ASSumMark3FROMScore
                                                    WHERE    (Cr# = CourseCode)
                                                    GROUP BY Cr#)
ELSE
IF(CourseType='اصلی')
SETSumMark= (SELECTSUM(Mark * 2) ASSumMark2FROMScore
                                                    WHERE    (Cr# = CourseCode)
                                                    GROUP BY Cr#)
ELSE
IF(CourseType='اختیاری')
SETSumMark=(SELECTSUM(Mark)ASSumMarkFROMScore
                                                    WHERE    (Cr# = CourseCode)
                                                    GROUP BY Cr#)

RETURNSumMark
END
```

تابع فوق را می توان با دستورات زیر فراخوانی و مقدار آن را نمایش داد (دستور زیر با توجه به ساختار نوشتاری MS SQL Server نوشته شده):

```
DECLARE   @Temp   REAL
SET @Temp= dbo.SumMarkOfCourse('پایگاه داده')
SELECT @Temp
```

زیر روال

نحوه ی استفاده از زیر روال همانند تابع است با این تفاوت که مقداری بازگشت نمی دهد. ساختار تعریف زیر روال بدین شکل می باشد:

PROCEDUREنام زیر روال([تعریف پارامترها])

دستورات زیر روال

مثال)در این مثال زیر روالی تعریف می کنیم که نام درس را به عنوان پارامتر دریافت کند و لیست دانشجویانی که نمره آنها در آن درس از میانگین کلاس بیشتر است را نمایش دهد.

```
CREATE PROCEDURE  ListStudentAbove (CourseName VARCHAR(20)) AS
BEGIN
SELECTStFName, StLName, Semester, Mark, CrName
FROM        Course INNER JOINScore
ON Course.Cr# = Score.Cr#INNER JOINStudent
ON Score.St# = Student.St#
WHERE     (CrName = CourseName)AND
Mark>(SELECTAVG(Mark) FROM Score
WHERECr# IN(SELECTCr#FROMScore
WHERECrName=CourseName))
ORDER BYMark, StLName, StFName
END
```

برای استفاده از زیر روال فوق می توان آن را در برنامه ها فراخوانی کرد و یا با استفاده از دستوری همچون دستور زیر آن را اجرا کرد.

```
EXECUTEListStudentAbove 'پایگاه داده'
```

دستور EXECUTE برای اجرای دستورات استفاده می شود.

کنترل کننده (Trigger)

کنترل کننده ها یکی از روش های کنترل جامعیت و سازگاری داده ها می باشند. در واقع با استفاده از کنترل کننده ها می توانیم مشخص کنیم که هرگاه رویداد خاصی رخ داد، چه واکنشی باید صورت گیرد. به عبارت دیگر، کنترل کننده نوعی زیر برنامه است که با وقوع رخدادی فراخوانی می شود و دستورات درون زیر برنامه را اجرا می کند.

فرضاً اگر بخواهیم مشخصات همه ی تراکنش ها (از قبیل: تاریخ، ساعت، کاربر و ...) را ذخیره کنیم، می توان کنترل کننده هایی تعریف کرد تا با وقوع هر رخداد، مشخصات آن را در جدولی جداگانه ذخیره کند.

همچنین فرض کنید که در سیستم دانشگاه بخواهیم نمره ی یک دانشجو در درسی ثبت کنیم که قبلاً در آن درس نمره قبولی گرفته باشد، روشن است که اجازه ی ثبت چنین رکوردی نباید داده شود، می توان این کنترل را توسط کنترل کننده انجام داد.

ایجاد کنترل کننده

با استفاده از ساختار زیر می توان یک کنترل کننده را ایجاد کرد.

CREATE TRIGGER نام کنترل کننده **{BEFORE|AFTER}** <رویداد>
ON نام جدول
[**FOR EACH** {**ROW | STATEMENT**}]
[**WHEN** (شرط)]
دستور SQL

{BEFORE|AFTER} مشخص می کنند که کنترل کننده قبل از رویداد اعلام شده عمل کند یا بعد از آن.

<رویداد> یکی از سه مورد زیر می باشد:

- **INSERT**
- **DELETE**
- **UPDATE [OFاسامی ستون ها]:** می توان ستون های مد نظر را نیز بعد از کلمه OF مشخص کرد. در این صورت کنترل کننده محدود به ستون های خاصی از جدول می شود.

با نوشتن عبارت FOR EACH می توان تعیین کرد که کنترل کننده در هنگام بروز رویداد در هر سطر اجرا شود (ROW) یا اینکه فقط یکبار پس از بروز رویداد اجرا شود (STATEMENT).

پیش فرض FOR EACH STATEMENT می باشد.

با استفاده از گزینه WHEN می توان واکنش کنترل کننده را محدود به شرایط خاص کرد.

در انتهای دستور با نوشتن دستور SQL (از قبیل DELETE، INSERT، UPDATE، CREATE، DROP و ...) واکنش کنترل کننده را نسبت به رویداد مشخص می کنیم.

توجه: نحوه ی پیاده سازی دستور Trigger در DBMS ها دقیقاً از ساختار استاندارد ANSI/ISO تبعیت نمی کند و هر DBMS با کمی تغییر آن را پیاده سازی کرده است. در مثال های زیر از شکل پیاده سازی شده در MS SQL Server استفاده شده، لذا کمی با ساختار استاندارد تفاوت دارند.

مثال) کنترل کننده ی زیر هر فعالیتی که بر روی جدول نمره صورت می گیرد را در جدول جداگانه ای به نام Monitoring ذخیره می کند.

```
CREATE TRIGGERInsertScore
ON Score
AFTER INSERT
AS
INSERT INTOMonitoring (Date, Action, UserName, TableName)
        VALUES(Current_TimeStamp, 'Insert', Current_User, 'Score')
```

مثال) کنترل کننده ی زیر بررسی می کند که تعداد واحد های گزرانده شده یک دانشجو در مقطع کاردانی از ۸۰ واحد، و در کارشناسی از ۱۵۰ واحد بیشتر نشود..

```
CREATE   TRIGGER   CheckTotalUnit
ON   Score
AFTER INSERT , UPDATE
AS
IF EXISTS(
                SELECTScore.St# FROM
Score INNERJOINCourseON Score.Cr#=Course.Cr#
                                        INNER JOINStudent ON Score.St# = Student.St#
                        WHEREDegree='کاردانی' AND Mark>=10
                        GROUP BY Score.St#
                        HAVING SUM(Unit)>80
                UNION
                SELECTScore.St#FROM
Score INNER JOINCourseONScore.Cr#=Course.Cr#
                                        INNER JOINStudent ON Score.St# = Student.St#
                        WHEREDegree ='کارشناسی' AND Mark>=10
                        GROUP BY Score.St#
                        HAVING SUM(Unit)> 150
                )
BEGIN
                ROLLBACK TRANSACTION
                RETURN
END
```

در دستور فوق عبارت ROLLBACK TRANSACTION باعث لغو تکرانش می شود، در مورد تراکنش ها کمی جلوتر خواهیم پرداخت.

حذف کنترل کننده

با استفاده از دستور زیر می توان کنترل کننده ای که قبلاً ساخته شده را حذف کرد.

DROP TRIGGERنام کنترل کننده

مثال) دستور زیر کنترل کننده InsertScore را از سیستم حذف می کند.

DROP TRIGGER InsertScore

مدیریت تراکنش ها

دستوراتی که باید به صورت یک واحد منطقی (یک تراکنش) در نظر گرفته شوند را باید بلوکه کرد. برای مشخص کردن شروع تراکنش از دستور START TRANSACTION استفاده می شود. برای اعلام خاتمه موفقیت آمیز تراکنش به DBMSاز عبارت COMMIT استفاده می شود. اما اگر در اجرای دستور خطایی رخ داد از عبارت ROLLBACK استفاده خواهد شد.

برای مثال فرض بگیرید که در سیستم بانکی ما قصد انتقال پول از یک حساب به حساب دیگر داریم، که این امر از دو قسمت تشکیل شده برداشت از حساب مبدا و واریز به حساب مقصد. نکته ای که در اینجا وجود دارد این است که یا هر دو فعالیت انجام شود یا هیچکدام.

مثال) در مثال قصد داریم کلیه اطلاعات مربوط به دانشجو به شماره ی ۸۸۲۷۱۹ را از سیستم حذف کنیم. از آنجا که اطلاعات دانشجویی وی در جدول دانشجو قرار دارد و نمره های وی در جدول نمره قرار دارد، لذا باید مشخصات وی از هر دو جدول حذف شود. بنابراین یا اطلاعات باید از هر دو جدول حذف شود یا از هیچکدام، از این رو با تعریف یک تراکنش حذف از هر دو جدول را به عنوان یک فعالیت منطقی در نظر می گیریم.

در MS SQL SERVER از دستور BEGIN TRANSACTION استفاده می شود.

```
BEGIN TRANSACTION
DELETE FROM Score WHERE St#='882719'
DELETE FROM Student WHERE St#='882719'
IF ( NOT EXISTS
(SELECT St# FROM Score WHERE St#='882719'
UNION
SELECT St# FROM Student WHERE St#='882719')
    )
COMMIT TRANSACTION
ELSE
ROLLBACK TRANSACTION
```

کاربران پایگاه داده

برای مشخص کردن افرادی که می توانند از پایگاه داده استفاده کنند باید اسامی کاربران برای این منظور تعریف شود، در جلوتر خواهیم دید که می توان به هر یک از این کاربران نقش خاصی اهدا کرد و کنترل های دسترسی نیز برای آن ها تعریف کرد.

ساختار دستوری ایجاد کاربر جدید بدین شکل می باشد:

CREATE USER نام کاربر

[FOR LOGIN نام ورود به سیستم **| WITHOUT LOGIN]**

با استفاده از دستور فوق می توان یک کاربر جدید را ایجاد کرد، این کاربر دو حالت دارد یا قادر است که به سیستم وارد شود، که در این صورت باید قسمت FOR LOGIN، نوشته شود، و یا اینکه قادر به ورود به سیستم نیست که در این صورت از عبارت WITHOUT LOGIN استفاده می شود، در این صورت می تواند به عنوان مهمان وارد سیستم شود. اگر از هیچ یک از دو قسمت فوق استفاده نشود، کاربر می تواند با نام کاربری خود وارد سیستم شود (البته در صورتی که قبلاً نام ورودی هم نام کاربری وی وجود داشته باشد).

-توضیحات این قسمت با توجه به مستندات MS SQL Server نوشته شده است.

مثال) دستور زیر کاربری به نام Arash ایجاد می کند.

CREATE USER Arash

برای اینکه بتوان از قسمت FOR LOGIN استفاده کرد قبل از آن باید یک LOGIN ایجاد کرد، که نحوه ی ساختن آن بدین شکل است:

CREATE LOGINنام ورود به سیستم

WITH PASSWORD =رمز ورود به سیستم

مثال) دستورات زیر ابتدا یک نام برای ورود به سیستم به نام Admin ایجاد می کنند، سپس یک نام کاربری به نام Saraتعریف می شود.

```
CREATE LOGIN Admin
WITH PASSWORD = '123'
CREATE USER   Sara
FOR LOGIN   Admin
```

نقش های کاربران

به طور معمول در سازمان ها، افراد به گروه های مختلفی تقسیم می شوند که هر گروه نقش مشخصی دارد. برای مثال در سیستم دانشگاه یک گروه نقش دانشجو دارند با وظایف کاملاً مشابه یکدیگر، افرادی در نقش استاد، افرادی در نقش کارشناس آموزش، افرادی در نقش کارشناس امتحانات و ... .

نکته قابل توجه این است که همه ی افراد دخیل در هر یک از گروه های فوق، وظایف و اختیارت مشابه یکدیگر دارند. حال در نظر بگیرید به جای اینکه بخش نامه ای صادر شود و در آن وظیفه ی جدیدی به نقش دانشجو داده شود، به تعداد دانشجویان بخش نامه تعریف و به تک تک آنها ارسال شود، قطعاً چنین چیزی بسیار مشکل و زمانبر است.

بنابراین ما هر گروه از کاربران را در یک دسته قرار خواهیم داد و به آن نقش می گوییم. البته ممکن است که یک کاربر چند نقش در سیستم داشته باشد، برای مثال شخصی را در نظر بگیرید که در دانشگاه هم تدریس می کند (نقش استاد) و هم مدیر باشد (نقش مدیر).

تعریف نقش جدید

با استفاده از دستور زیر می توان یک نقش جدید را در سیستم تعریف کرد.

CREATE ROLEنام نقش[**WITH ADMIN**<اهدا کننده>]

با استفاده از قسمت WHIT ADMIN می توان مدیر نقش را مشخص کرد. در اینجا منظور از مدیر نقش شخص یا اشخاصی هستند که اجازه ی ویرایش و حذف نقش را در سیستم پایگاه داده داشته باشند. در دستور فوق <اهدا کننده> می‌تواند یکی از دو مقدار زیر داشته باشد:

- CURRENT_USER:کاربر جاری، همین کاربری که دارد با سیستم کار می کند.
- CURRENT_ROLE: نقش جاری، نقش همین کاربری که دارد با سیستم کار می کند.

اگر قسمت WHIT ADMIN نوشته نشود به طور پیش فرض کاربر جاری مدیر نقش می باشد.

مثال) با استفاده از دستور زیر نقشی برای دانشجویان تعریف می شود.

CREATEROLE StudentRole

-در MS SQL SERVER به جای عبارت WITH ADMIN از عبارت AUTHORIZATION استفاده می شود.

تعیین نقش کاربران

پس از تعریف نقش ها، باید کاربران را به عضویت نقش ها در آورد. برای این کار از دستور زیر استفاده می شود.

GRANTنام نقش‌ها**TO**<گیرندگان>
[**WITH ADMIN OPTION**]
[**GRANTED BY**<اهدا کننده>]

<گیرندگان> می‌تواند یکی از سه حالت زیر باشد:

- PUBLIC: همه استفاده کنندگان از پایگاه داده.
- نام کاربر: فقط به کاربر یا کاربرانی که مشخص شده مجوز اعطا می شود.
- نام نقش: به نقش یا نقش هایی که مشخص شده مجوز اعطا می شود. بنابراین کلیه ی کاربرانی که در این نقش ها هستند مجوزها را دریافت خواهند کرد.

در دستور فوق با نوشتن عبارت WITH ADMIN OPTION می توان امکانات مدیریتی نقش را نیز به گیرندگان اهدا کرد.

با اضافه کردن قسمت GRANTED BY می توان اهدا کننده نقش را نیز مشخص کرد.

مثال)دستور زیر نام کاربری Arash را به عضویت در نقش StudentRole در می آورد.

GRANTStudentRole**TO** Arash

-در MS SQL Server برای به عضویت در آوردن یک کاربر در یک نقش از دستور زیر استفاده می شود:

SP_AddRoleMember'نام نقش', 'نام کاربر'

لغو نقش کاربران

ممکن است پس از مدتی بخواهیم نقشی را که قبلاً به کاربری داده ایم پس بگیریم.
برای مثال شخص قبلاً دانشجو بوده و حال فارغ التحصیل شده، بنابراین نقش دانشجو باید از وی گرفته شود و نقش فارغ التحصیل به او داده شود.برای لغو نقش از دستور زیر استفاده می شود.

REVOKEاسامی نقش ها**FROM**<گیرندگان>
[**GRANTED BY**<اهدا کننده>]
{**CASCADE** |**RESTRICT**}

مثال) دستور زیر عضویت Arash از نقش StudentRole لغو می کند.

REVOKEStudentRole**FROM** Arash

-در MS SQL Server برای لغو عضویت کاربر از یک نقش از دستور زیر استفاده می شود:

SP_DropRoleMember 'نام نقش', 'نام کاربر'

حذف کردن نقش

اگر بخواهیم نقشی را حذف کنیم، از دستور زیر باید استفاده کرد:

DROP ROLE نام نقش

روشن است که پس از حذف نقش، نقش از تمام کاربرانی که آن را داشته اند گرفته خواهد شد.

مثال) دستور زیر نقش دانشجو را حذف می کند.

DROPROLE StudentRole