نگارش
اول

راهنمای جامع دستورات SQL

به انضمام مثال های SQL Server

این کتاب در مورد بیشتر سیستم های پایگاه داده رابطه ای معروف مثل MS Access، MS SQL Server، MySQL توضیح داده است و برای هر کدام دستورات و مثالهای مربوطه را آورده است. مثالهای ضمیمه شده مربوط به SQL Server 2008 هستند که در محیط Management Studio و با پایگاه داده آموزشی NorthWind تست شده اند.

فهرست مطالب

فصل اول

مقدمه ای بر SQL ۱

گرامر SQL ۲

عبارت SELECT در SQL ۴

عبارت SELECT DISTINC ۶

عبارت WHERE در SQL ۷

عملگرهای AND و OR ۹

کلمه کلیدی ORDER BY در SQL ۱۱

عبارت INSERT INTO در SQL ۱۳

عبارت UPDATE ۱۴

عبارت DELETE ۱۶

فصل دوم

عبارت TOP در SQL ۱۸

عملگر LIKE در SQL ۲۰

کاراکترهای جایگزین شونده SQL ۲۲

عملگر IN در SQL ۲۴

عملگر BETWEEN در SQL ۲۵

Alias در SQL ۲۷

JOIN ها در SQL ۲۸

کلمه کلیدی INNER JOIN در SQL ۲۹

کلمه کلیدی LEFT JOIN در SQL ۳۱

کلمه کلیدی RIGHT JOIN در SQL ۳۲

کلمه کلیدی FULL JOIN در SQL ۳۳

عملگر UNION در SQL ۳۵
عبارت SELECT INTO در SQL ۳۷
عبارت CREATE DATABASE در SQL ۳۹
عبارت CREATE TABLE در SQL ۳۹
محدودیت ها ۴۰
محدودیت NOT NULL ۴۰
محدودیت UNIQUE ۴۱
محدودیتPRIMARY KEY در SQL ۴۳
محدودیت FOREIGN KEY ۴۵
محدودیت CHECK ۴۷
محدودیت DEFAULT در SQL ۴۹
دستورات CREATE INDEX در SQL ۵۰
DROP INEX، DROP TABLE و DROP DATABASE در SQL ۵۱
دستور ALTER TABLE ۵۲
فیلد AUTO INCREMENT در SQL ۵۴
View ها در SQL ۵۸
توابع تاریخ در SQL ۶۰
مقدار NULL ۶۳
توابع NULL در SQL ۶۵
نوع داده ها در SQL ۶۶

فصل سوم

توابع SQL ۷۴
تابع AVG() ۷۴
تابع COUNT() در SQL ۷۶

تابع FIRST() ۷۸
تابع LAST() ۷۹
تابع MAX() ۸۰
تابع MIN() ۸۱
تابع SUM() ۸۲
دستور Group By در SQL ۸۳
عبارت Having ۸۵
تابع UCASE() ۸۶
تابع LCASE() ۸۷
تابع MID() ۸۸
تابع LEN() ۸۹
تابع ROUDN() ۹۰
تابع NOW() ۹۰
تابع FORMAT() ۹۱

ضمیمه

مرجع سریع دستورات SQL ۹۳
هاستینگ در SQL ۹۶
شما SQL را آموختید، حالا چکاری باید انجام دهید؟ ۹۷

پیشگفتار

در دهه۷۰ گروهی از شرکت آی‌بی‌ام بر روی سیستم پایگاه داده‌های سیستم **آر** کار می‌کردند و زبان SQL را به منظور عملیات و بازیابی اطلاعات ذخیره شده در سیستم آر ایجاد کردند.

زبان ساخت یافته پرس و جو (SQL که سی‌کوِل خوانده می شود) زبانی است سطح بالا مبتنی بر زبان سطح پایین و ریاضی جبر رابطه‌ای که برای ایجاد، تغییر و بازیابی داده‌ها و نیز عملیات بر روی آنها به‌کار می‌رود. زبان SQL به سمت مدل شی‌گرا - رابطه‌ای نیز پیشرفت کرده است. SQL برای کارهای ویژه و محدودی (گزارش گیری از داده‌ها در پایگاه داده‌های رابطه‌ای) طراحی شده است. بر خلاف زبانهای دستوری مثل بیسیک یا سی که برای حل مسائل طراحی شده، SQL زبانی بر پایه اعلان است.

این کتاب در مورد بیشتر سیستم های مدیریت پایگاه داده رابطه ای معروف مثل MS Access، MS SQL Server، MySQL توضیح داده است و برای هر کدام دستورات و مثالهای مربوطه را آورده است. مثالهای ضمیمه شده مربوط به SQL Server 2008 هستند که در محیط Management Studio و با پایگاه داده آموزشی NorthWind تست شده اند. پایگاه داده NorthWind را نیز می توانید از اینجا دریافت کنید.

مطالب این کتاب، ترجمه دوره آموزشی SQL از سایت W3Schools می باشد. هرچند تلاش کردیم تا حد امکان مطلب را ساده و روان بیان کنیم اما بی شک اشکالاتی در امر ترجمه این کتاب وجود دارد که از خوانندگان محترم تقاضا می کنیم با ارسال نظرات خود ما را در بهبود بخشیدن این کتاب یاری کنند.

بهار ۹۱

فصل اول - دستورات پایه

مقدمه ای بر SQL

SQL یک زبان استاندارد برای دسترسی و دستکاری پایگاه های داده است.

SQL چیست؟

- SQL مخفف Structured Query Language یا زبان پرس و جوی ساخت یافته است
- SQL به شما اجازه دسترسی و دستکاری به پایگاه های داده را می دهد
- SQL یک استاندارد ANSI (موسسه استاندارد ملی آمریکا) است

چه کارهایی SQL می تواند انجام دهد؟

- SQL یک پرس و جو را در برابر یک پایگاه داده می تواند سریع اجرا کند
- SQL می تواند داده را از یک پایگاه داده واکشی کند
- SQL می تواند رکوردهایی را در پایگاه داده درج کند
- SQL می تواند رکوردهایی را در پایگاه داده به روز کند
- SQL می تواند رکوردهای یک پایگاه داده را حذف کند
- SQL می تواند پایگاه داده های جدیدی بوجود آورد
- SQL می تواند جدولهای جدیدی را در پایگاه داده بوجود آورد
- SQL می تواند رویه های ذخیره شده در پایگاه داده بسازد
- SQL می تواند نماهایی را در پایگاه داده بوجود آورد
- SQL می تواند برای جداول، رویه های ذخیره شده و نماها مجوز تنظیم کند

SQL استاندارد هست ولی...

اگرچه SQL یک ANSI استاندارد است اما نسخه های مختلفی از زبان SQL وجود دارند.

با این حال، برای موافقت با ANSI، همه آنها دست کم از فرمانهای اصلی (مثل SELECT، UPDATE، DELETE، INSERT، WHERE) به روش مشابهی پشتیبانی می کنند.

نکته: بیشتر برنامه های پایگاه داده SQL نیز ضمیمه های اختصاصی خودشان را علاو ه بر استانداردهای SQL دارند!

استفاده از SQL در وب سایت تان

برای ساختن یک وب سایت که بعضی داده را از یک پایگاه داده نمایش دهد، به موارد زیر نیاز خواهید داشت:

- یک برنامه پایگاه داده RDBMS (مثل MS Access، SQL Server، MySQL)
- یک زبان اسکریپتی سمت سرور، مثل php یا ASP
- SQL
- HTML / CSS

RDBMS

RDBMS مخفف Relational Database Management System یا سیستم مدیریت پایگاه داده رابطه ای است.

RDBMS پایه ای برای SQL و برای همه سیستمهای پایگاه داده جدید از قبیل MS SQL Server، IBM DB2، Oracle، MySQL و Microsoft Access است.

داده در RDBMS در اشیاء پایگاه داده که جداول نامیده می شود، ذخیره می شوند.

یک جدول مجموع داده یکپارچه مرتبط است و شامل ستون ها و سطرها می شود.

گرامر SQL

جداول پایگاه داده

یک پایگاه داده اغلب شامل یک یا چند جدول است. هر جدول بوسیله یک نام (مثل "Orders" یا "Customers") شناسایی می شود. جداول شامل رکوردها (سطرها) همراه داده هستند.

در زیر مثالی از یک جدول به نام "Persons" آمده است:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

جدول بالا شامل سه رکورد (برای هر شخص) و پنج ستون (P_Id، LastName، FirstName، Address و City) است.

عبارات SQL

بیشتر عملیاتی که نیاز دارید در یک پایگاه داده انجام دهید، با عبارات SQL انجام می شوند.

جمله SQL زیر همه رکوردها را از جدول اشخاص انتخاب می کند:

```
SELECT * FROM Persons
```

در این آموزشی، همه عبارات مختلف SQL را به شما درس خواهیم داد.

به خاطر داشته باشید...

- SQL حساس به حروف کوچک و بزرگ نیست.

سمیکالون بعداز عبارات SQL ؟

بیشتر سیستم های پایگاه داده یک سمیکالون در انتهای هر جمله SQL را لازم می داند.

سمیکالون یک روش استاندارد برای جدا سازی هر عبارت SQL در سیستم های پایگاه داده است که اجازه اجرای بیش از یک عبارت SQL را در درخواست سرور را می دهد.

ما از MS Access و SQL Server 2000 استفاده می کنیم و مجبور نیستم یک سمیکالون بعد از هر عبارت SQL قرار دهیم اما بعضی برنامه های پایگاه داده شما را مجبور به استفاده آن می کنند.

DML و DDL در SQL

SQL می تواند به دو بخش تقسیم شود: زبان دستکاری داده (DML) و زبان تعریف داده (DDL).

دستورات پرس و جو و بروز کردن از بخش زبان دستکاری داده SQL:

- **SELECT** - استخراج داده از یک پایگاه داده
- **UPDATE** - بروز کردن داده در یک پایگاه داده
- **DELETE** - حذف داده از یک پایگاه داده
- **INSERT INTO** - درج یک داده جدید در یک پایگاه داده

بخش DDL از SQL به جداول پایگاه داده اجازه ساخت یا حذف را می دهد. همچنین کلیدهای فهرست را تعریف می کند، ارتباط بین جداول را تعیین می کند و محدودیت ها بین جداول را اعمال می کند. مهمترین عبارات DDL در SLQ عبارتند از:

- **CREATE DATABASE** - ساخت یک پایگاه داده جدید
- **ALTER DATABASE** - اصلاح یک پایگاه داده
- **CREATE TABLE** - ساخت یک جدول جدید
- **ALTER TABLE** - اصلاح یک جدول
- **DROP TABLE** - حذف یک جدول
- **CREATE INDEX** - ساخت یک کلید جستجو (کلید جستجو)
- **DROP INDEX** - حذف یک فهرست

عبارت SELECT در SQL

این فصل در مورد SELECT و * SELECT توضیح خواهد داد.

عبارت SELECT در SQL

عبارت SELECT برای انتخاب داده از یک پایگاه داده استفاده می شود.

نتیجه در یک جدول ذخیره می شود که جدول- نتیجه نامیده می شود.

گرامر SQL در SELECT

```
SELECT column_name(s)
FROM table_name
```

و

```
SELECT * FROM table_name
```

تذکر: SQL حساس به حروف بزرگ و کوچک نیست. SELECT مثل select است.

یک مثال SELECT در SQL

جدول"Persons":

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

اکنون می خواهیم محتوای ستون هایی با نام "LastName" و "FirstName" را از جدول انتخاب کنیم.

عبارت SELECT زیر را بکار می بریم:

```
SELECT LastName, FirstName FROM Persons
```

نتیجه اینگونه خواهد شد:

LastName	FirstName
Hansen	Ola
Svendson	Tove
Pettersen	Kari

مثال SELECT *

اکنون می خواهیم همه ستونهای جدول "Persons" را انتخاب کنیم.

عبارتSELECT زیر را بکار می بریم:

```
SELECT * FROM Persons
```

توجه: کاراکتر ستاره (*) روشی سریع برای انتخاب کل ستون ها می باشد!

نتیجه اینگونه خواهد شد:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

پیمایش در جدول - نتیجه

بیشتر سیستم های نرم افزاری پایگاه داده اجازه پیمایش با توابع برنامه نویسی را در جدول نتیجه می دهند؛ مثل Move-To-First-Record، Get-Record-Content، Move-To-Next-Record و غیره

توابع برنامه نویسی از این قبیل به عنوان قسمتی از این آموزش نیستند. برای یادگیری در مورد دسترسی به داده با فراخوانی تابع، لطفا از دروه آموزشی ADO یا دوره آموزشی PHP ما بازدید کنید.

عبارت SQL SELECT DISTINCT

این فصل در مورد جمله SELECT DISTINCT توضیح خواهد داد.

عبارت SELECT DISTINCT در SQL

برخی از ستون های یک جدول ممکن است مقدار محتوای تکراری داشته باشد. این یک مشکل نیست اما گاهی اوقات می خواهید فقط مقادیر مختلفی (متمایز) که در یک جدول هستند را لیست کنید.

کلمه کلیدی DISTINCT می تواند فقط برای برگرداندن مقادیر متمایز (متفاوت) استفاده شود.

گرامر SELECT DISTINCT در SQL

```
SELECT DISTINCT column_name(s)
FROM table_name
```

مثال SELECT DISTINCT

جدول"Persons":

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

حال می خواهیم در جدول بالا تنها مقادیر متفاوت از ستونی بنام "City" را انتخاب کنیم.

از جمله SELECT زیر استفاده می کنیم:

```
SELECT DISTINCT City FROM Persons
```

نتیجه اینگونه خواهد شد:

City
Sandnes
Stavanger

عبارت WHERE در SQL

عبارت WHERE برای فیلتر کردن رکوردها استفاده می شود.

عبارت WHERE فقط برای استخراج آن رکوردهایی که یک معیار خاصی را انجام می دهند، استفاده می شود.

گرامر WHERE در SQL

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator value
```

مثال عبارت WHERE

جدول"Persons":

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

اکنون می خواهیم فقط اشخاصی که در شهر "Sandnes" زندگی می کنند را از جدول بالا انتخاب کنیم.

از جمله SELECT زیر استفاده می کنیم:

```
SELECT * FROM Persons
WHERE City='Sandnes'
```

نتیجه اینگونه خواهد شد:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes

نقل قولها در اطراف فیلدهای متنی

SQL نقل قول های تکی در مورد مقادیر متنی بکار می برد (بیشتر سیستمهای پایگاه داده نقل قول های دوتایی می پذیرند).

اما نباید مقادیر عددی داخل نقل قول ها ضمیمه شوند.

برای مقادیر متنی:

```
This is correct:

SELECT * FROM Persons WHERE FirstName='Tove'

This is wrong:

SELECT * FROM Persons WHERE FirstName=Tove
```

برای مقادیر عددی:

```
This is correct:

SELECT * FROM Persons WHERE Year=1965

This is wrong:

SELECT * FROM Persons WHERE Year='1965'
```

عملگرهای مجاز در عبارت WHERE

عملگرهای زیر می توانند با عبارت WHERE استفاده شوند:

عملگر	توضیحات
=	مساوی
<>	نابرابر
>	بزرگتر از

<	کمتر از
>=	بزرگتر مساوی
<=	کوچکتر مساوی
BETWEEN	بین یک دامنه مشمول
LIKE	جستجوی یک الگو
IN	برای تعیین کردن مقادیر چندگانه ممکن، برای یک ستون

نکته: در بیشتر نسخه های SQL عملگر <> ممکن است به عنوان =! نوشته شود.

عملگرهای AND و OR

عملگرهای AND و OR برای فیلتر رکوردهای مبنی بر بیش از یک شرط استفاده می شوند.

عملگرهای AND و OR

عملگر AND یک رکورد را نمایش می دهد؛ اگر هر دو شرط اولی و دومی درست باشند.

عملگر OR یک رکورد را نمایش می دهد؛ اگر یکی از دو شرط اولی یا دومی درست باشد.

مثال عملگر AND

جدول "Persons":

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

اکنون می خواهیم فقط اشخاصی که نامشان "Tove" و نام خانوادگی شان "Svendson" است را انتخاب کنیم.

از جمله SELECT زیر استفاده می کنیم:

```
SELECT * FROM Persons
WHERE FirstName='Tove'
AND LastName='Svendson'
```

نتیجه اینگونه خواهد شد:

P_Id	LastName	FirstName	Address	City
2	Svendson	Tove	Borgvn 23	Sandnes

مثالی از عملگر OR

اکنون می خواهیم فقط اشخاصی که نامشان "Tove" یا "Ola" است را انتخاب کنیم.

از جمله SELECT زیر استفاده می کنیم:

```
SELECT * FROM Persons
WHERE FirstName='Tove'
OR FirstName='Ola'
```

نتیجه اینگونه خواهد شد:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes

ترکیب AND و OR

همچنین می توانید AND و OR را ترکیب کنید (از پرانتزها برای عبارات پیچیده استفاده کنید).

اکنون می خواهیم فقط اشخاصی که نام خانوادگی شان "Svendson" و نامشان "Tove" یا "Ola" است را انتخاب کنیم.

از جمله SELECT زیر استفاده می کنیم:

```
SELECT * FROM Persons WHERE
LastName='Svendson'
AND (FirstName='Tove' OR FirstName='Ola')
```

نتیجه اینگونه خواهد شد:

P_Id	LastName	FirstName	Address	City
2	Svendson	Tove	Borgvn 23	Sandnes

کلمه کلیدی ORDER BY در SQL

کلمه کلیدی ORDER BY برای مرتب کردن جدول نتیجه استفاده می شود.

کلمه کلیدی ORDER BY

کلمه کلیدی ORDER BY برای مرتب کردن جدول نتیجه توسط یک ستون مشخص، استفاده می شود.

کلمه کلیدیORDER BY به صورت پیش فرض رکوردها را صعودی مرتب می کند.

اگر می خواهید رکوردها را نزولی مرتب کنید، می توانید از کلمه کلیدی DESC استفاده کنید.

گرامر ORDER BY در SQL

```
SELECT column_name(s)
FROM table_name
ORDER BY column_name(s) ASC|DESC
```

مثال ORDER BY

جدول "Persons":

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Tom	Vingvn 23	Stavanger

حال می خواهیم همه اشخاص را در جدول بالا انتخاب کنیم؛ اما می خواهیم آنها را بوسیله نام خوانوادگی شان مرتب کنیم.

از جمله SELECT زیر استفاده می کنیم:

```
SELECT * FROM Persons
ORDER BY LastName
```

نتیجه اینگونه خواهد شد:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
4	Nilsen	Tom	Vingvn 23	Stavanger
3	Pettersen	Kari	Storgt 20	Stavanger
2	Svendson	Tove	Borgvn 23	Sandnes

مثال ORDER BY DESC

حال می خواهیم همه اشخاص را در جدول بالا انتخاب کنیم؛ اما می خواهیم انها را بوسیله نام خوانوادگی شان به صورت نزولی مرتب کنیم.

از جمله SELECT زیر استفاده می کنیم:

```
SELECT * FROM Persons
ORDER BY LastName DESC
```

نتیجه اینگونه خواهد شد:

P_Id	LastName	FirstName	Address	City
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Tom	Vingvn 23	Stavanger
1	Hansen	Ola	Timoteivn 10	Sandnes

عبارت INSERT INTO در SQL

عبارت INSERT INTO برای درج سطر جدیدی در جدول استفاده می شود.

عبارتINSERT INT در SQL

عبارت INSERT INTO برای درج سطر جدیدی در جدول استفاده می شود.

گرامر INSERT INTO در SQL

نوشتن عبارت INSERT INTO به دو شکل ممکن است.

شکل اول اسامی ستونهایی که داده در آنجا درج خواهد شد را مشخص نمی کند، فقط مقادیرشان را مشخص می کند:

```
INSERT INTO table_name
VALUES (value1, value2, value3,…)
```

شکل دوم هم اسامی ستون و هم مقادیری که درج خواهند شد را مشخص می کند.

```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

مثال INSERT INTO در SQL

جدول "Persons" زیر را داریم:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

اکنون سطر جدیدی از جدول Persons درج می کنیم.

از جمله SQL زیر استفاده می کنیم:

```
INSERT INTO Persons
VALUES (4,'Nilsen', 'Johan', 'Bakken 2', 'Stavanger')
```

جدول "Persons" اکنون شبیه این است:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger

درج داده فقط در ستون های معین

همچنین ممکن است فقط داده ها را در ستون های مشخص اضافه کنید.

عبارت SQL زیر سطرجدیدی را اضافه خواهد کرد؛ اما فقط داده را به ستون های "P_Id"، "LastName" و "FirstName" اضافه می کند.

```
INSERT INTO Persons (P_Id, LastName, FirstName)
VALUES (5, 'Tjessem', 'Jakob')
```

جدول "Persons" اکنون شبیه این است:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger
5	Tjessem	Jakob		

عبارت UPDATE

جمله UPDATE برای آپدیت کردن رکوردهای یک جدول استفاده می شود.

عبارت UPDATE

جمله UPDATE برای آپدیت کردن رکوردهای موجود یک جدول استفاده می شود.

گرامر SQL UPDATE

```
UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value
```

نکته: به عبارت WHERE در گرامر UPDATE توجه کنید. عبارت WHERE تعیین می کند که رکورد یا رکوردهایی باید به روز شوند. اگر عبارت WHERE را حذف کنید همه کوردها بروز خواهند شد!

مثال SQL UPDATE

جدول"Persons":

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger
5	Tjessem	Jakob		

اکنون می خواهیم شخص"Tjessem, Jakob" را در جدول "Persons" را آپدیت کنیم.

از جمله SQL زیر استفاده می کنیم:

```
UPDATE Persons
SET Address='Nissestien 67', City='Sandnes'
WHERE LastName='Tjessem' AND FirstName='Jakob'
```

جدول "Persons" اکنون شبیه این است:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger
5	Tjessem	Jakob	Nissestien 67	Sandnes

اخطار UPDATE در SQL

زمانیکه رکوردها را بروز می کنید، دقت کنید. اگر ما عبارت WHERE را در مثال بالا حذف کرده باشیم، مثل این:

```
UPDATE Persons
SET Address='Nissestien 67', City='Sandnes'
```

جدول "Persons" شبیه این می شود:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Nissestien 67	Sandnes
2	Svendson	Tove	Nissestien 67	Sandnes
3	Pettersen	Kari	Nissestien 67	Sandnes
4	Nilsen	Johan	Nissestien 67	Sandnes
5	Tjessem	Jakob	Nissestien 67	Sandnes

عبارت DELETE

جمله DELETE برای حذف رکوردهای یک جدول استفاده می شود.

عبارت DELETE

جمله DELETE برای حذف سطرهای یک جدول استفاده می شود .

گرامر SQL DELETE

```
DELETE FROM table_name
WHERE some_column=some_value
```

نکته: به عبارت WHERE در گرامر DELETE توجه کنید. عبارت WHERE تعیین می کند که رکوردها یا رکوردهایی باید حذف شوند. اگر شما عبارت WHERE را حذف کنید همه رکوردها حذف خواهند شد.

مثال DELETE در SQL

جدول "Persons":

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger
5	Tjessem	Jakob	Nissestien 67	Sandnes

اکنون می خواهیم شخص "Tjessem, Jakob" را از جدول "Persons" حذف کنیم.

از جمله SQL زیر استفاده می کنیم:

```
DELETE FROM Persons
WHERE LastName='Tjessem' AND FirstName='Jakob'
```

جدول "Persons" اکنون شبیه این است:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger

حذف همه سطرها

حذف کردن همه سطرها در یک جدول بدون حذف جدول ممکن است. به این معنی که ساختار، ویژگی ها و فهرست های جدول دست نخورده باقی خواهند ماند.

```
DELETE FROM table_name

or

DELETE * FROM table_name
```

نکته: وقتی رکوردها را حذف می کنید خیلی مراقب باشید. این عبارت را نمی توانید خنثی کنید!

فصل دوم - دستورات پیشرفته

عبارت TOP در SQL

عبارت TOP در SQL برای تعیین تعدادی از رکوردها برای برگرداندن، استفاده می شود .

عبارت TOP در جدول های بزرگ با هزاران رکورد می تواند بسیار مفید باشد. برگرداندن تعداد زیادی از رکوردها می تواند عملکرد را تحت فشار قرار دهد.

نکته: همه ی سیستم های پایگاه داده از عبارت TOP پشتیبانی نمی کنند.

گرامر SQL Server

```
SELECT TOP number|percent column_name(s)
FROM table_name
```

معادل SELECT TOP در Oracle و MySQL

گرامر MySQL

```
SELECT column_name(s)
FROM table_name
LIMIT number
```

مثال :

```
SELECT *
FROM Persons
LIMIT 5
```

گرامر Oracle

```
SELECT column_name(s)
FROM table_name
WHERE ROWNUM <= number
```

مثال :

```
SELECT *
FROM Persons
WHERE ROWNUM <=5
```

مثال TOP در SQL

جدول "Persons":

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Tom	Vingvn 23	Stavanger

اکنون می خواهیم فقط دو رکورد اول از جدول بالا را انتخاب کنیم .

از دستور SELECT زیر استفاده می کنیم:

```
SELECT TOP 2 * FROM Persons
```

نتیجه اینگونه خواهد شد:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes

مثال TOP PERCENT در SQL

جدول "Persons":

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Tom	Vingvn 23	Stavanger

اکنون می خواهیم فقط 50% از جدول بالا را انتخاب کنیم.

از دستور SELECT زیر استفاده می کنیم :

```
SELECT TOP 50 PERCENT * FROM Persons
```

نتیجه اینگونه خواهد شد:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes

عملگر LIKE در SQL

عملگر LIKE در یک عبارت WHERE، عبارتی است که برای جستجوی یک الگوی تعیین شده در یک ستون استفاده می شود.

عملگر LIKE

عملگر LIKE برای جستجوی یک الگوی تعیین شده در یک ستون استفاده می شود .

گرامر LIKE در SQL

```
SELECT column_name(s)
FROM table_name
WHERE column_name LIKE pattern
```

مثال عملگر LIKE

جدول "Persons":

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

اکنون می خواهیم از جدول بالا اشخاصی را که شهر زندگی آن ها با "s" شروع می شود را انتخاب کنیم.

از دستور SELECT زیر استفاده می کنیم:

```
SELECT * FROM Persons
WHERE City LIKE 's%'
```

علامت "%" می تواند برای تعریف کاراکترهای جایگزین شونده (حروف ناپیدا در الگو) قبل و بعد از الگو استفاده شود .

نتیجه اینگونه خواهد شد:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

سپس می خواهیم از جدول "persons" اشخاصی را انتخاب کنیم که شهر زندگی آنها با "s" پایان می یابد.

از دستور SELECT زیر استفاده می کنیم:

```
SELECT * FROM Persons
WHERE City LIKE '%s'
```

نتیجه اینگونه خواهد شد:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes

سپس می خواهیم از جدول "persons" اشخاصی را انتخاب کنیم که شهر زندگی آنها شامل الگوی "tav" باشد.

از دستور SELECT زیر استفاده می کنیم:

```
SELECT * FROM Persons
WHERE City LIKE '%tav%'
```

نتیجه اینگونه خواهد شد:

P_Id	LastName	FirstName	Address	City
3	Pettersen	Kari	Storgt 20	Stavanger

همچنین با استفاده از کلمه کلیدی NOT می توانیم از جدول "Persons" اشخاصی که شهر زندگی آنها شامل الگوی "tav" نیست را انتخاب کنیم.

از دستورSELECT زیر استفاده می کنیم:

```
SELECT * FROM Persons
WHERE City NOT LIKE '%tav%'
```

نتیجه اینگونه خواهد شد:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes

کاراکترهای جایگزین شونده SQL

کاراکترهای جایگزین شونده SQL، هنگام جستجوی داده در یک پایگاه داده می توانند استفاده شوند.

کاراکترهای جایگزین شونده SQL

کاراکترهای جایگزین شونده SQL، هنگام جستجوی داده در یک پایگاه داده می تواند یک یا چند کاراکتر را جایگزین کند.

کاراکترهای جایگزین شونده SQL باید همراه با عملگر LIKE استفاده شوند.

از کاراکترهای جایگزین شونده زیر می توان در SQL استفاده کرد:

کاراکترهای جایگزین شونده	تعریف
%	یک جانشین برای صفر یا کاراکترهای بیشتر
_	یک جانشین برای دقیقا یک کاراکتر
[charlist]	هر کاراکتر تنها در Charlist (لیست کاراکترها)
[^charlist] یا [!charlist]	هرکاراکتر تنها که در Charlist نباشد

مثال کاراکترهای جایگزین شونده SQL

جدول "Persons" زیر را داریم:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

استفاده از کاراکتر جایگزین شونده %

اکنون می خواهیم از جدول"persons" اشخاصی را انتخاب کنیم که شهر زندگی آنها با "sa" شروع می شود.

از دستور SELECT زیر استفاده می کنیم:

```
SELECT * FROM Persons
WHERE City LIKE 'sa%'
```

نتیجه اینگونه خواهد شد:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes

سپس می خواهیم از جدول "persons" اشخاصی را انتخاب کنیم که شهر زندگی آنها شامل الگوی "nes" باشد .

از دستور SELECT زیر استفاده می کنیم:

```
SELECT * FROM Persons
WHERE City LIKE '%nes%'
```

نتیجه اینگونه خواهد شد:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes

استفاده از کاراکتر جایگزین شونده _

اکنون می خواهیم از جدول "Persons" اشخاصی که نام آنها با هر کاراکتری شروع می شود و با "la" ادامه می یابد را انتخاب کنیم.

از دستور SELECT زیر استفاده می کنیم:

```
SELECT * FROM Persons
WHERE FirstName LIKE '_la'
```

نتیجه اینگونه خواهد شد:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes

سپس می خواهیم از جدول "persons" اشخاصی را انتخاب کنیم که نام خانوادگی آن ها با "s" شروع می شود و با هر کاراکتر، "end"، هر کاراکتر و "on" ادامه می یابد را انتخاب کنیم.

از دستور SELECT زیر استفاده می کنیم:

```
SELECT * FROM Persons
WHERE LastName LIKE 'S_end_on'
```

نتیجه اینگونه خواهد شد:

P_Id	LastName	FirstName	Address	City
2	Svendson	Tove	Borgvn 23	Sandnes

استفاده از کاراکتر جایگزین شونده [charlist]

اکنون می خواهیم از جدول "persons" اشخاصی را انتخاب کنیم که نام خانوادگی آن ها با "s"یا "b" یا "p" شروع می شود.

از دستور SELECT زیر استفاده می کنیم:

```
SELECT * FROM Persons
WHERE LastName LIKE '[bsp]%'
```

نتیجه اینگونه خواهد شد:

P_Id	LastName	FirstName	Address	City
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

سپس می خواهیم از جدول "persons" اشخاصی را انتخاب کنیم که نام خانوادگی آن ها با "s" یا "b" یا "p" شروع نمی شود .

از دستور SELECT زیر استفاده می کنیم:

```
SELECT * FROM Persons
WHERE LastName LIKE '[!bsp]%'
```

نتیجه اینگونه خواهد شد:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes

عملگر IN در SQL

عملگر IN به شما اجازه می دهد مقادیر چندگانه در عبارت WHERE تعیین کنید .

گرامر IN در SQL

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1,value2,...)
```

مثالی از عملگر IN

جدول "Persons":

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

اکنون می خواهیم از جدول بالا اشخاصی را که نام خانوادگی آن ها برابر با "Hansen" یا "pettersen" است را انتخاب کنیم.

از دستور SELECT زیر استفاده می کنیم:

```
SELECT * FROM Persons
WHERE LastName IN ('Hansen','Pettersen')
```

نتیجه اینگونه خواهد شد:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

عملگر BETWEEN در SQL

عملگر BETWEEN در یک عبارت WHERE، برای انتخاب یک دامنه از داده بین مقدار استفاده می شود.

عملگر BETWEEN

عملگر BETWEEN دامنه ای از داده ها را از بین دو مقدار انتخاب می کند. مقادیر می توانند اعداد، متن یا داده باشد.

گرامر BETWEEN در SQL

```
SELECT column_name(s)
FROM table_name
WHERE column_name
BETWEEN value1 AND value2
```

مثال عملگر BETWEEN

جدول "Persons":

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

اکنون می خواهیم از جدول بالا اشخاصی را که نام خانوادگی آن ها به صورت حروف الفبا بین "Hansen" و "pettersen" است را انتخاب کنیم.

از دستور SELECT زیر استفاده می کنیم:

```
SELECT * FROM Persons
WHERE LastName
BETWEEN 'Hansen' AND 'Pettersen'
```

نتیجه اینگونه خواهد شد:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes

تذکر: عملگر BETWEEN در پایگاه داده های مختلف به طور متفاوت عمل می کند.

در بعضی از پایگاه داده ها، اشخاصی با نام خانوادگی "Hansen" یا "Pettersen" لیست نمی شوند؛ زیرا عملگر BETWEEN فقط فیلدهایی که بین آن ها هستند (باستثنای مقادیر تست) را انتخاب می کند.

در دیگر پایگاه داده ها، اشخاصی با نام خانوادگی "Hansen" یا "pettersen" لیست می شوند، زیرا عملگر BETWEEN فیلدهایی که بین و شامل آن ها هستند را انتخاب می کند.

و در برخی دیگر، اشخاصی با نام خانوادگی "Hansen" در لیست هستند اما "pattersen" در لیست نیست (شبیه مثال بالا) ، زیرا عملگر BETWEEN فیلدهای بین مقادیر تست و اولین مقدار تست را انتخاب می کند (باستثنای مقدار آخر تست).

بنابراین: رفتار پایگاه داده خود را با عملگر BETWEEN چک کنید.

مثال ۲

برای نمایش اشخاصی که بیرون از دامنه مثال قبل هستند، از NOT BETWEEN استفاده کنید:

```
SELECT * FROM Persons
WHERE LastName
NOT BETWEEN 'Hansen' AND 'Pettersen'
```

نتیجه اینگونه خواهد شد:

P_Id	LastName	FirstName	Address	City
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Alias در SQL

با SQL ، یک نام مستعار می تواند یک جدول یا یک ستون داده شود.

Alias در SQL

می توانید یک جدول یا یک ستون را با نام دیگر، با استفاده از Alias (نام مستعار) معین کنید . این می تواند یک چیز خوب برای نام جدول پیچیده یا نام ستون طولانی باشد.

یک نام Alias هر چیزی می تواند باشد اما معمولا کوتاه است.

گرامر Alias در SQL برای جدول ها

```
SELECT column_name(s)
FROM table_name
AS alias_name
```

گرامر Alias در SQL برای ستون ها

```
SELECT column_name AS alias_name
FROM table_name
```

مثال Alias

فرض کنید یک جدول بنام "persons" و جدولی دیگر بنام "Product_Orders" داریم. Alias جدول را به ترتیب "p" و "po" معین می کنیم.

اکنون می خواهیم همه سفارشاتی که "Ola Hansen" مسئول آن هست را لیست کنیم.

زیر استفاده می کنیم:SELECT از دستور

```
SELECT po.OrderID, p.LastName, p.FirstName
FROM Persons AS p,
Product_Orders AS po
WHERE p.LastName='Hansen' AND p.FirstName='Ola'
```

عبارت SELECT مشابه بدون نام های مستعار:

```
SELECT Product_Orders.OrderID, Persons.LastName, Persons.FirstName
FROM Persons,
Product_Orders
WHERE Persons.LastName='Hansen' AND Persons.FirstName='Ola'
```

همانطور که از دو عبارت SELECT بالا می بینید، نام مستعار می تواند پرس و جو را هم برای نوشتن و هم برای خواندن آسانتر کند.

JOIN ها در SQL

JOIN ها در SQL برای پرس و جوی داده از دو یا چند جدول، مبنی بر یک رابطه بین برخی ستون ها در این جدول ها استفاده می شود.

JOIN در SQL

کلمه کلیدی JOIN برای پرس و جوی داده از دو یا چند جدول در یک عبارت SQL، مبنی بر یک رابطه بین برخی ستون ها در این جدول ها استفاده می شود.

جدول ها در یک پایگاه داده اغلب با کلیدها با یکدیگر مرتبط هستند.

کلید اصلی (Primary Key) یک ستون (یا ترکیبی از ستون ها) با یک مقدار مقدار منحصر برای هر سطر است. هر مقدار کلید اصلی باید در داخل جدول، منحصر به فرد باشد. هدف چسباندن داده ها به یکدیگر، در سراسر جدول ها، جلوگیری از تکرار همه داده ها در هر جدول است.

جدول "Persons"را ببینید:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

توجه کنید که ستون "P_Id" یک کلید اصلی در جدول "persons" است. به این معنا که دو سطر، مقدار "P_Id" یکسان نمی توانند داشته باشند. P_Id دو شخص را متمایز می نماید؛ حتی اگر آن ها نام یکسانی داشته باشند.

پس جدول"Order"را داریم:

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	1
4	24562	1
5	34764	15

توجه کنید که ستون "O-Id"یک کلید اصلی در جدول"Orders" است و ستون "p-Id" را به جدول "Persons" بدون استفاده از نام آنها ارجاع می دهد.

توجه کنید که رابطه بین دو جدول بالا، ستون"P-Id" است.

JOIN های مختلف در SQL

قبل از اینکه با مثال ها ادامه دهیم، انواع JOIN هایی که می توانید استفاده کنید و تفاوت بین آن ها را لیست می کنیم.

JOIN: سطرها را برمی گرداند، وقتی که حداقل یک تطابق در هر دو جدول داشته باشد.

LEFT JOIN: تمام سطرهای جدول چپ را برمی گرداند؛ حتی اگر نظیر آن در جدول راست نباشد.

RIGHT JOIN: تمام سطرهای جدول راست را برمی گرداند؛ حتی اگر نظیر آن در جدول چپ نباشد.

FULL JOIN: سطرها را زمانی که نظیرش در یکی از جدول ها باشد، برمی گرداند.

کلمه کلیدی INNER JOIN در SQL

کلمه کلیدی INNER JOIN در SQL

کلمه کلیدی INNER JOIN سطر ها را زمانی که حداقل یک تطابق در دو جدول وجود داشته باشد، بر می گرداند.

گرامر INNER JOIN در SQL

```
SELECT column_name(s)
FROM table_name1
INNER JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

پانوشته: INNER JOIN با JOIN یکسان است.

مثال INNER JOIN در SQL

جدول "Persons":

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

جدول "Orders":

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	1
4	24562	1
5	34764	15

اکنون می خواهیم همه اشخاصی را با هر سفارشی لیست کنیم.

از دستور SELECT زیر استفاده می کنیم:

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
INNER JOIN Orders
ON Persons.P_Id=Orders.P_Id
ORDER BY Persons.LastName
```

نتیجه اینگونه خواهد شد:

LastName	FirstName	OrderNo
Hansen	Ola	22456
Hansen	Ola	24562
Pettersen	Kari	77895
Pettersen	Kari	44678

کلمه کلیدی INNER JOIN سطر ها را زمانی که حداقل در دو جدول نظیر هستند، بر می گرداند. اگر سطرهایی در "Persons" هستند که نظیرشان در "Orders" نیست، آن سطرها لیست نمی شوند.

کلمه کلیدی LEFT JOIN در SQL

کلمه کلیدی LEFT JOIN در SQL

کلمه کلیدی LEFT JOIN تمام سطرهای جدول چپ را برمی گرداند (table_name1)؛ حتی اگر نظیرش در جدول راست (table_name2) وجود نداشته باشد.

گرامر LEFT JOIN در SQL

```
SELECT column_name(s)
FROM table_name1
LEFT JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

پانوشته: در بعضی از پایگاه داده ها LEFT JOIN، LEFT OUTER JOIN نامیده می شود.

مثال SQL LEFT JOIN

جدول "Persons":

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

جدول "Orders":

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	1
4	24562	1
5	34764	15

اکنون می خواهیم از جدول بالا همه اشخاص را با سفارشاتشان (در صورت وجود) لیست کنیم.

از دستور SELECT زیر استفاده می کنیم:

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
LEFT JOIN Orders
ON Persons.P_Id=Orders.P_Id
ORDER BY Persons.LastName
```

نتیجه اینگونه خواهد شد:

LastName	FirstName	OrderNo
Hansen	Ola	22456
Hansen	Ola	24562
Pettersen	Kari	77895
Pettersen	Kari	44678
Svendson	Tove	

کلمه کلیدی LEFT JOIN تمام سطرهای جدول چپ را برمی گرداند (persons)؛ حتی اگر نظیرش درجدول راست (orders) وجود نداشته باشد.

کلمه کلیدی RIGHT JOIN در SQL

کلمه کلیدی RIGHT JOIN در SQL

کلمه کلیدی RIGHT JOIN تمام سطرهای جدول راست را برمی گرداند (table_name2)؛ حتی اگر نظیرش درجدول چپ (table_name1) وجود نداشته باشد.

گرامر RIGHT JOIN در SQL

```
SELECT column_name(s)
FROM table_name1
RIGHT JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

پانوشته: در بعضی از پایگاه داده ها RIGHT JOIN، RIGHT OUTER JOIN نامیده می شود.

مثال RIGHT JOIN در SQL

جدول "Persons":

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

جدول "Orders" :

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	1
4	24562	1
5	34764	15

اکنون می خواهیم از جدول بالا همه سفارشات با مشخصات اشخاص (در صورت وجود) را لیست کنیم.

از دستور SELECT زیر استفاده می کنیم:

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
RIGHT JOIN Orders
ON Persons.P_Id=Orders.P_Id
ORDER BY Persons.LastName
```

نتیجه اینگونه خواهد شد:

LastName	FirstName	OrderNo
Hansen	Ola	22456
Hansen	Ola	24562
Pettersen	Kari	77895
Pettersen	Kari	44678
		34764

کلمه کلیدی RIGHT JOIN تمام سطرهای جدول راست را برمی گرداند (Orders)؛ حتی اگر نظیرش در جدول چپ (Persons) وجود نداشته باشد.

کلمه کلیدی FULL JOIN در SQL

کلمه کلیدی FULL JOIN در SQL

کلمه کلیدی FULL JOIN سطرها را زمانی که نظیرشان در یکی از جدول ها باشد، برمی گرداند .

گرامر FULL JOIN در SQL

```
SELECT column_name(s)
FROM table_name1
FULL JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

مثال FULL JOIN در SQL

جدول "Persons":

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

جدول "Orders":

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	1
4	24562	1
5	34764	15

اکنون می خواهیم همه اشخاص با سفارشاتشان و همه سفارشات با اشخاص را لیست کنیم.

از دستور SELECT زیر استفاده می کنیم:

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
FULL JOIN Orders
ON Persons.P_Id=Orders.P_Id
ORDER BY Persons.LastName
```

نتیجه اینگونه خواهد شد:

LastName	FirstName	OrderNo
Hansen	Ola	22456
Hansen	Ola	24562
Pettersen	Kari	77895

Pettersen	Kari	44678
Svendson	Tove	
		34764

کلمه کلیدی FULL JOIN همه سطرها را از جدول چپ(persons) و همه سطرها از جدول راست(orders) بر می گرداند.

اگر سطرهایی در"Persons" وجود دارد که نظیرش در"Orders" نیست، یا اگر سطرهایی در "Orders" وجود دارد که نظیرش در "Persons" نیست، آن سطرها نیز لیست می شوند.

عملگر UNION در SQL

عملگر UNION در SQL دو یا چند دستور SELECT را ترکیب می کند.

عملگر UNION در SQL

عملگر UNION برای ترکیب جدول نتیجه، از دو یا چند عبارتSELECT استفاده می شود.

توجه کنید که هر عبارت SELECT در داخل UNION باید تعداد یکسانی از ستون ها را داشته باشد. همچنین ستون ها باید همانند انواع داده های مشابه داشته باشند. همچنین ستون ها در هر عبارت مشابه باید ترتیب یکسان داشته باشند.

گرامر UNION در SQL

```
SELECT column_name(s) FROM table_name1
UNION
SELECT column_name(s) FROM table_name2
```

توجه: عملگر UNION به صورت پیش فرض فقط مقادیر متمایز را انتخاب می کند. از UNION ALL برای اجازه دادن به مقادیر تکراری استفاده کنید.

گرامر UNION ALL در SQL

```
SELECT column_name(s) FROM table_name1
UNION ALL
SELECT column_name(s) FROM table_name2
```

پانوشته: نام ستونها در جدول نتیجه UNION همیشه برابر با نام ستونها در اولین عبارت SELECT در UNION است.

مثال UNION در SQL

به جداول زیر نگاه کنید:

"Employees_Norway":

E_ID	E_Name
01	Hansen, Ola
02	Svendson, Tove
03	Svendson, Stephen
04	Pettersen, Kari

"Employees_USA":

E_ID	E_Name
01	Turner, Sally
02	Kent, Clark
03	Svendson, Stephen
04	Scott, Stephen

اکنون می خواهیم **همه کارمندهای مختلف** در Norway و USA را لیست کنیم.

از دستور SELECT زیر استفاده می کنیم:

```
SELECT E_Name FROM Employees_Norway
UNION
SELECT E_Name FROM Employees_USA
```

جدول نتیجه اینگونه خواهد شد:

E_Name
Hansen, Ola
Svendson, Tove
Svendson, Stephen
Pettersen, Kari
Turner, Sally
Kent, Clark
Scott, Stephen

توجه: این دستور نمی تواند برای لیست کردن همه کارمندهای Norway و USA استفاده شود . در مثال بالا کارمند با نام های یکسان قرار دارند و فقط یکی از آنها در لیست است. دستور UNION فقط مقادیر متمایز را انتخاب می کند.

مثال UNION ALL در SQL

اکنون می خواهیم همه کارمندهای Norway و USA را لیست کنیم.

```
SELECT E_Name FROM Employees_Norway
UNION ALL
SELECT E_Name FROM Employees_USA
```

نتیجه:

E_Name
Hansen, Ola
Svendson, Tove
Svendson, Stephen
Pettersen, Kari
Turner, Sally
Kent, Clark
Svendson, Stephen
Scott, Stephen

عبارت SELECT INTO در SQL

عبارت SELECT INTO در SQL می تواند برای ساخت کپی های پشتیبان از جدول، استفاده شود.

عبارت SELECT INTO در SQL

عبارت SELECT INTO داده ها را از یکی از جدول ها انتخاب می کند و آن را درون یک جدول متفاوت درج می کند.

عبارت SELECT INTO بیشتر اوقات برای کپی های پشتیبان از جدول استفاده می شود.

گرامر SELECT INTO در SQL

ما می توانیم همه ستون ها را درون جدول جدید انتخاب کنیم:

```
SELECT *
INTO new_table_name [IN externaldatabase]
FROM old_tablename
```

یا می توانیم فقط ستون هایی را انتخاب کنیم که می خواهیم درون جدول جدید باشند :

```
SELECT column_name(s)
INTO new_table_name [IN externaldatabase]
FROM old_tablename
```

مثال SELECT INTO در SQL

ساختن یک کپی پشتیبان - اکنون می خواهیم یک کپی دقیق از داده ها در جدول "persons" بسازیم.

از عبارت SQL زیر استفاده می کنیم:

```
SELECT *
INTO Persons_Backup
FROM Persons
```

همچنین می توانیم از عبارت IN برای کپی جدول درون پایگاه داده های دیگر، استفاده کنیم.

```
SELECT *
INTO Persons_Backup IN 'Backup.mdb'
FROM Persons
```

همچنین می توانیم فقط چند فیلد درون جدول جدید کپی کنیم:

```
SELECT LastName,FirstName
INTO Persons_Backup
FROM Persons
```

SELECT INTO در SQL با یک عبارت WHERE

همچنین می توانیم یک عبارت WHERE اضافه کنیم.

عبارت SQL زیر یک جدول "persons-backup" می‌سازد و فقط شامل اشخاصی است که در شهر "sandnes" زندگی می کنند.

```
SELECT LastName,Firstname
INTO Persons_Backup
FROM Persons
WHERE City='Sandnes'
```

SELECT INTO در SQL – جداول JOIN شده

همچنین انتخاب داده ها از بیشتر از یک جدول ممکن است.

مثال زیر یک جدول "Persons_Order_Backup" شامل داده ها از دو جدول "persons" و "orders" را می‌سازد.

```
SELECT Persons.LastName,Orders.OrderNo
INTO Persons_Order_Backup
FROM Persons
INNER JOIN Orders
ON Persons.P_Id=Orders.P_Id
```

عبارت CREATE DATABASE در SQL

عبارت CREATE DATABASE

عبارت CREATE DATABASE برای ایجاد یک پایگاه داده استفاده می شود.

گرامر CREATE DATABASE در SQL

```
CREATE DATABASE database_name
```

مثال CREATE DATABASE

اکنون می خواهیم پایگاه داده ای با نام "MY-db" ایجاد کنیم.

از عبارت CREAT DATABASE زیر استفاده می کنیم:

```
CREATE DATABASE my_db
```

جداول پایگاه داده می تواند با عبارت CREATE TABLE اضافه شوند.

عبارت CREATE TABLE در SQL

عبارت CREATE TABLE

عبارت CREATE TABLE برای ایجاد یک جدول در پایگاه داده استفاده می شود.

گرامر SQL CREATE TABLE

```
CREATE TABLE table_name
(
column_name1 data_type,
column_name2 data_type,
column_name3 data_type,
....
)
```

نوع داده مشخص می کند چه نوع داده ای در ستون می تواند نگه داشته شود. برای یک مرجع کامل از انواع داده موجود در MS Access، SQL Server و MySQL به مرجع انواع داده ما بروید.

مثال CREATE TABLE

اکنون می خواهیم یک جدول با نام "persons" ایجاد کنیم که شامل پنج ستون: P_Id و LastName و FirstName و Address و City است.

از عبارت CREATE TABLE زیر استفاده می کنیم:

```
CREATE TABLE Persons
(
P_Id int,
LastName varchar(255),
FirstName varchar(255),
Address varchar(255),
City varchar(255)
)
```

ستون P_I d از نوع int است و یک عدد را نگه می دارد.

ستون های lastName، FirstName، Address و City از نوع varchar و باطول بیشتر از ۲۵۵ کاراکتر هستند.

اکنون جدول خالی "Persons" را ببینید که اینگونه است:

P_Id	LastName	FirstName	Address	City

محدودیت ها

برای محدود کردن برخی از داده ها در یک جدول، از دستور Constraint استفاده می شود.

محدودیتها را می توان موقع ایجاد جدول (دستور ALTER TABLE) یا بعد از ایجاد جدول تعریف کنیم (دستور ALTER TABLE).

در محدودیت های زیر تمرکز خواهیم کرد :

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK
- DEFAULT

در بخش های بعد هر یک از محدودیت ها را با جزئیات توضیح می دهیم .

محدودیت NOT NULL:

به طور پیش فرض ، این عبارت در یک جدول نمی گذارد یک ستون مقدار تهی در نظر بگیرد.

محدودیت NOT NULL:

محدودیت NOT NULL باعث می شود تا فیلد مورد نظر همیشه دارای مقدار باشد. این بدان معنی است که شما نمی توانید یک رکورد جدید را بدون مقدار وارد کنید، یا یک رکورد را بدون اضافه کردن یک مقدار به این فیلد، بروز رسانی کنید.

در تکه برنامه SQL زیر ستون "P_Id" و ستون "LastName" نمی توانند خالی از مقدار باشند:

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255)
)
```

محدودیت UNIQUE:

محدودیت UNIQUE شناسایی منحصر به فردی را به هر رکورد در جدول پایگاه داده می دهد.

محدودیت های UNIQUE و PRIMARY KEY هر دو تضمینی منحصر به فرد برای یک ستون یا مجموعه ای از ستون ها را فراهم می کنند.

محدودیت PRIMARY KEY به طور خودکار دارای محدودیت UNIQUE می باشد که بر روی آن تعریف شده است.

توجه داشته باشید که می توانید محدودیت های بسیاری از UNIQUE در جدول داشته باشند، اما تنها یک محدودیت PRIMARY KEY در هر جدول می توانید داشته باشید.

محدودیت UNIQUE در ساختن جدول در SQL:

کد SQL زیر یک محدودیت UNIQUE زمانی که جدول "Persons" ساخته می شود به ستون"P_Id" نسبت می دهد:

MySQL:

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
UNIQUE (P_Id)
)
```

SQL Server / Oracle / MS Access:

```
CREATE TABLE Persons
(
P_Id int NOT NULL UNIQUE,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255)
)
```

برای نام گذاری یک محدودیت UNIQUE و یا برای معین کردن UNIQUE های موجود بر روی ستون های متعدد در یک جدول، استفاده از دستور زیر جایز است:

MySQL / SQL Server / Oracle / MS Access:

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
CONSTRAINT uc_PersonID UNIQUE (P_Id,LastName)
)
```

محدودیت UNIQUE در تغییر جدول :

برای ایجاد این محدودیت بر روی ستون "P_Id" ، از کد زیر استفاده می کنیم:

MySQL / SQL Server / Oracle / MS Access:

```
ALTER TABLE Persons
ADD UNIQUE (P_Id)
```

برای نامگذاری محدودیت UNIQUE و همچنین برای تعریف این محدودیت بر روی ستون های متعدد، استفاده از دستور زیر در SQL جایز است :

MySQL / SQL Server / Oracle / MS Access:

```
ALTER TABLE Persons
ADD CONSTRAINT uc_PersonID UNIQUE (P_Id,LastName)
```

برای حذف یک محدودیت UNIQUE

کد زیر نمونه ای برای حذف کردن محدودیت UNIQUE درSQL است:

MySQL:

```
ALTER TABLE Persons
DROP INDEX uc_PersonID
```

SQL Server / Oracle / MS Access:

```
ALTER TABLE Persons
DROP CONSTRAINT uc_PersonID
```

محدودیتPRIMARY KEY در SQL:

محدودیتPRIMARY KEY در SQL:

محدودیت کلید اصلی مسئول شناسایی منحصر به فردی هر رکورد در جدول پایگاه داده است.

کلید اصلی باید از مقادیر منحصر به فردی برخوردار باشد.

ستون کلید اصلی نمی تواند مقدار NULL داشته باشد.

هر جدول باید یک کلید اصلی داشته باشد و هر جدول می تواند فقط یک کلید اصلی داشته باشند.

محدودیت PRIMARY KEY در CREATE TABLE در SQL:

کد زیر یک محدودیت PRIMARY KEY بر روی ستون "P_Id" می سازد؛ زمانی که جدول "Persons" ایجاد می شود:

MySQL:

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
PRIMARY KEY (P_Id)
)
```

SQL Server / Oracle / MS Access:

```
CREATE TABLE Persons
(
P_Id int NOT NULL PRIMARY KEY,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255)
)
```

برای اجازه دادن به نامگذاری محدودیت کلید اصلی و برای تعریف محدودیت کلید اصلی بر روی ستون های متعدد، از دستور زیر استفاده کنید:

MySQL / SQL Server / Oracle / MS Access:

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
CONSTRAINT pk_PersonID PRIMARY KEY (P_Id,LastName)
)
```

توجه: در مثال بالا تنها یک کلید اصلی (pk_PersonID) وجود دارد. با این حال، مقدار pk_PersonID از دو ستون (P_Id و LastName) ساخته شده است.

محدودیت PRIMARY KEY در ALTER TABLE

برای ایجاد یک محدودیت کلید اصلی بر روی ستون "P_Id" زمانی که جدول ایجاد می شود، از دستور SQL زیر استفاده کنید:

MySQL / SQL Server / Oracle / MS Access:

```
ALTER TABLE Persons
ADD PRIMARY KEY (P_Id)
```

برای نامگذاری و همچنین تعریف بر روی ستون های متعدد، باید از دستور زیر استفاده کنیم:

MySQL / SQL Server / Oracle / MS Access:

```
ALTER TABLE Persons
ADD CONSTRAINT pk_PersonID PRIMARY KEY (P_Id,LastName)
```

توجه: اگر برای اضافه کردن یک کلید اصلی با استفاده از دستورات جدول را تغییر دهید، کلید اصلی ستون ها باید حتما شامل مقدار NOT NULL باشد (هنگامی که جدول برای اولین بار ایجاد شده است).

برای حذف یک محدودیت PRIMARY KEY

برای حذف کردن کلید اصلی از تکه برنامه زیر استفاده می کنیم:

MySQL:

```
ALTER TABLE Persons
DROP PRIMARY KEY
```

SQL Server / Oracle / MS Access:

```
ALTER TABLE Persons
DROP CONSTRAINT pk_PersonID
```

محدودیت FOREIGN KEY:

کلید خارجی در یک جدول به کلید اصلی در یک جدول دیگر اشاره می کند.

اجازه دهید کلید خارجی را با ذکر یک مثال توضیح دهیم. به دو جدول زیر نگاه کنید:

جدول "Persons":

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

جدول "Orders":

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	2
4	24562	1

توجه داشته باشید که ستون "P_Id" در جدول "سفارشات" به ستون "P_Id" در جدول "افراد" اشاره می کند.

ستون "P_Id" در جدول "افراد" کلید اصلی جدول "افراد" است. ستون "P_Id" در جدول "سفارشات" کلید خارجی در جدول "سفارشات" است.

محدودیت کلید خارجی(FOREIGN KEY) در واقع از وارد شدن اطلاعات نا معتبر در جدول و همچنین هر چیز که ارتباط بین دو جدول را از بین ببرد جلوگیری می کند؛ زیرا که به مقدار یکتایی در جدول اصلی اشاره دارد.

محدودیت کلید خارجی در CREATE TABLE :

کد زیر ستون "P_Id" را به عنوان کلید خارجی می سازد؛ زمانی که جدول "سفارشات" ایجاد می شود:

MySQL:

```
CREATE TABLE Orders
(
O_Id int NOT NULL,
OrderNo int NOT NULL,
P_Id int,
PRIMARY KEY (O_Id),
FOREIGN KEY (P_Id) REFERENCES Persons(P_Id)
)
```

SQL Server / Oracle / MS Access:

```
CREATE TABLE Orders
(
O_Id int NOT NULL PRIMARY KEY,
OrderNo int NOT NULL,
P_Id int FOREIGN KEY REFERENCES Persons(P_Id)
)
```

برای نامگذاری و همچنین تعریف کلید های خارجی بر روی ستون های متعدد، از دستور زیر استفاده کنید:

MySQL / SQL Server / Oracle / MS Access:

```
CREATE TABLE Orders
(
O_Id int NOT NULL,
OrderNo int NOT NULL,
P_Id int,
PRIMARY KEY (O_Id),
CONSTRAINT fk_PerOrders FOREIGN KEY (P_Id)
REFERENCES Persons(P_Id)
)
```

محدودیت FOREIGN KEY در ALTER TABLE:

برای ایجاد یک محدودیت کلید خارجی بر روی ستون "P_Id" زمانی که جدول ایجاد شده، از دستور SQL زیر استفاده کنید:

MySQL / SQL Server / Oracle / MS Access:

```
ALTER TABLE Orders
ADD FOREIGN KEY (P_Id)
REFERENCES Persons(P_Id)
```

برای نامگذاری و همچنین تعریف بر روی ستون های متعدد، از دستورات زیر استفاده کنید:

MySQL / SQL Server / Oracle / MS Access:

```
ALTER TABLE Orders
ADD CONSTRAINT fk_PerOrders
FOREIGN KEY (P_Id)
REFERENCES Persons(P_Id)
```

برای حذف یک محدودیت FOREIGN KEY

برای حذف کردن کلید خارجی از تکه برنامه زیر استفاده می کنیم:

MySQL:

```
ALTER TABLE Orders
DROP FOREIGN KEY fk_PerOrders
```

SQL Server / Oracle / MS Access:

```
ALTER TABLE Orders
DROP CONSTRAINT fk_PerOrders
```

محدودیت CHECK

محدودیت CHECK

محدودیت CHECK به منظور محدود کردن دامنه مقدار که می تواند در یک ستون قرار گیرد استفاده می شود.

اگر محدودیت CHECK را تنها بر روی یک ستون تعریف می کنید اجازه وارد کردن برخی از مقدار ها را برای آن ستون دارید.

اگر محدودیت CHECK را برای یک جدول تعریف می کنید، می توانید مقادیر را در ستون های خاصی مبنی بر مقادیر ستون های دیگر در ردیف محدود کنید.

محدودیت CHECK در CREATE TABLE

برنامه زیر محدودیت CHECK را روی ستون"P_Id" در جدول "Persons" می سازد. محدودیت مشخص شده برای ستون "P_Id" فقط شامل اعداد صحیح بزرگتر از صفر است.

MySQL:

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
CHECK (P_Id>0)
)
```

SQL Server / Oracle / MS Access:

```
CREATE TABLE Persons
(
P_Id int NOT NULL CHECK (P_Id>0),
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255)
)
```

برای محدود کردن چند ستون به طور همزمان، از دستور زیر استفاده کنید:

MySQL / SQL Server / Oracle / MS Access:

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
CONSTRAINT chk_Person CHECK (P_Id>0 AND City='Sandnes')
)
```

محدودیت CHECK در ALTER TABLE

برای ایجاد محدودیت CHECK در ستون "P_Id" زمانی که جدول از قبل ایجاد شده است، از دستور زیر استفاده کنید:

MySQL / SQL Server / Oracle / MS Access:

```
ALTER TABLE Persons
ADD CHECK (P_Id>0)
```

برای محدود کردن چند ستون به طور همزمان، از دستور زیر استفاده کنید:

MySQL / SQL Server / Oracle / MS Access:

```
ALTER TABLE Persons
ADD CONSTRAINT chk_Person CHECK (P_Id>0 AND City='Sandnes')
```

حذف کردن محدودیت CHECK

برای حذف این محدودیت باید از دستور زیر استفاده کنیم:

SQL Server / Oracle / MS Access:

```
ALTER TABLE Persons
DROP CONSTRAINT chk_Person
```

MySQL:

```
ALTER TABLE Persons
DROP CHECK chk_Person
```

محدودیت DEFAULT در SQL

محدودیت DEFAULT در SQL

محدودیت DEFAULT برای قرار دادن یک مقدار به طور پیش فرض در یک ستون، استفاده می شود.

مقدار DEFAULT به همه رکوردهای جدید اضافه می شود؛ اگر هیچ مقدار دیگری وارد نشود.

محدودیت DEFAULT در CREATE TABLE

تکه برنامه زیر محدودیت DEFAULT را بر روی ستون"City" در جدول "Persons" ایجاد می کند:

My SQL / SQL Server / Oracle / MS Access:

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255) DEFAULT 'Sandnes'
)
```

محدودیت DEFAULT همچنین می تواند برای وارد کردن مقادیر سیستم، با استفاده از توابعی مانند GETDATE() مورد استفاده قرار گیرد:

```
CREATE TABLE Orders
(
O_Id int NOT NULL,
OrderNo int NOT NULL,
P_Id int,
OrderDate date DEFAULT GETDATE()
)
```

محدودیتDEFAULT در ALTER TABLE

برای ایجاد محدودیت DEFAULT در ستون "City" هنگامی که جدول ایجاد شده، از دستور زیر استفاده کنید:

MySQL:

```
ALTER TABLE Persons
ALTER City SET DEFAULT 'SANDNES'
```

SQL Server / Oracle / MS Access:

```
ALTER TABLE Persons
ALTER COLUMN City SET DEFAULT 'SANDNES'
```

برای حذف یک محدودیت DEFAULT

به عنوان مثال برای حذف این محدودیت باید از دستور زیر استفاده کنیم:

MySQL:

```
ALTER TABLE Persons
ALTER City DROP DEFAULT
```

SQL Server / Oracle / MS Access:

```
ALTER TABLE Persons
ALTER COLUMN City DROP DEFAULT
```

دستورات CREATE INDEX در SQL

دستور CREATE INDEX برای ایجاد شاخص در جداول استفاده می شود.

شاخص ها به برنامه اجازه جستجو در پایگاه داده را برای پیدا کردن سریع اطلاعات را می دهد؛ بدون آنکه کل جدول را بگردد.

INDEX ها

شاخص می تواند برای یافتن سریعتر و موثرتر داده در یک جدول استفاده شود.

کاربران شاخص ها را نمی توانند ببینند، آنها فقط برای سرعت بخشیدن به جستجو استفاده می شوند.

توجه: بروز رسانی یک جدول با شاخص زمان بیشتری را نسبت به بروز رسانی یک جدول بدون شاخص تلف می کند (چون شاخص نیز نیاز به یک بروز رسانی دارد). بنابراین شما تنها باید شاخص را در ستون (و جدول) ایجاد کنید.

گرامر CREATE INDEX در SQL

ایجاد یک شاخص در یک جدول. مقادیر تکراری مجاز هستند:

```
CREATE INDEX index_name
ON table_name (column_name)
```

گرامر CREATE UNIQUE INDEX در SQL

ایجاد یک شاخص در یک جدول. مقادیر تکراری مجاز نیستند:

```
CREATE UNIQUE INDEX index_name
ON table_name (column_name)
```

توجه: گرامر ساخت شاخص ها در پایگاه داده های مختلف متفاوت است. بنابراین گرامر ساخت شاخص ها را در پایگاه داده خود چک کنید.

مثال CREATE INDEX

دستور زیر یک شاخص به نام "PIndex" بر روی ستون "LastName" در جدول "Persons" ایجاد می کند:

```
CREATE INDEX PIndex
ON Persons (LastName)
```

اگر می خواهید یک شاخص را بر روی ترکیبی از ستون ها ایجاد کنید باید نام ستون ها را داخل پرانتز نوشته و با کاما از هم جدا کنید:

```
CREATE INDEX PIndex
ON Persons (LastName, FirstName)
```

DROP INEX، DROP TABLE و DROP DATABASE در SQL

شاخص ها، جداول، و پایگاه داده ها براحتی می توانند با دستور DROP حذف شوند.

دستور DROP INEX

دستور DROP INDEX برای حذف یک شاخص در یک جدول استفاده می شود.

MS Access:

```
DROP INDEX index_name ON table_name
```

MS SQL Server:

```
DROP INDEX table_name.index_name
```

DB2/Oracle:

```
DROP INDEX index_name
```

MySQL:

```
ALTER TABLE table_name DROP INDEX index_name
```

دستور DROP TABLE

دستور DROP TABLE برای حذف جدول استفاده می شود.

```
DROP TABLE table_name
```

دستور DROP DATABASE

دستور DROP DATABASE برای حذف بانک اطلاعاتی استفاده می شود.

```
DROP DATABASE database_name
```

دستور TRUNCATE TABLE

و ام اگر بخواهیم فقط اطلاعات داخل جدول را پاک کنیم؛ نه خود جدول را باید چکار کرد؟

خب، از دستور TRUNCATE TABLE استفاده کنید:

```
TRUNCATE TABLE table_name
```

دستور ALTER TABLE

دستور ALTER TABLE

دستور ALTER TABLE برای اضافه کردن، حذف، یا تغییر ستون در جدول موجود استفاده می شود.

گرامر ALTER TABLE در SQL

برای اضافه کردن یک ستون در یک جدول، از دستور زیر استفاده کنید:

```
ALTER TABLE table_name
ADD column_name datatype
```

برای حذف یک ستون در یک جدول، از دستور زیر استفاده کنید (توجه داشته باشید که برخی از سیستم های پایگاه داده اجازه حذف یک ستون را نمی دهند):

```
ALTER TABLE table_name
DROP COLUMN column_name
```

برای تغییر نوع داده یک ستون در یک جدول، از گرامر زیر استفاده کنید:

```
ALTER TABLE table_name
ALTER COLUMN column_name datatype
```

مثال دستور ALTER TABLE در SQL

به جدول "Persons" نگاه کنید:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

حالا می خواهیم یک ستون به نام "DateOfBirth" را در جدول "Persons" اضافه کنیم.

از دستور SQL زیر استفاده می کنیم:

```
ALTER TABLE Persons
ADD DateOfBirth date
```

توجه کنید که در ستون جدید "DateOfBirth" نوعی از تاریخ است و برای نگه داشتن داده ای از نوع تاریخ استفاده می شود. نوع داده مشخص می کند که چه نوع، داده های در ستون می توانند وارد شوند.

جدول "Persons" اینگونه خواهد شد:

P_Id	LastName	FirstName	Address	City	DateOfBirth
1	Hansen	Ola	Timoteivn 10	Sandnes	
2	Svendson	Tove	Borgvn 23	Sandnes	
3	Pettersen	Kari	Storgt 20	Stavanger	

مثال تغییر نوع داده ها

حالا می خواهیم نوع داده های ستون "DateOfBirth" در جدول "Persons " را تغییر دهیم.

از دستور SQL زیر استفاده می کنیم:

```
ALTER TABLE Persons
ALTER COLUMN DateOfBirth year
```

توجه کنید که ستون "DateOfBirth" در حال حاضر مقدار یک سال در قالب دو رقم یا چهار رقم را در خود نگه می دارد.

مثال DROP COLUMN

در گام بعدی می خواهیم ستون "DateOfBirth" در جدول "Persons " را حذف کنیم.

از دستور SQL زیر استفاده می کنیم:

```
ALTER TABLE Persons
DROP COLUMN DateOfBirth
```

جدول "Persons" اینگونه خواهد شد:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

فیلد AUTO INCREMENT در SQL

فیلد Auto-increment اجازه می دهد یک مقدار منحصر به فرد تولید شود؛ زمانی که یک رکورد جدید در جدول وارد شود.

فیلد AUTO INCREMENT در SQL

خیلی اوقات می خواهیم مقداری را برای primary key تعریف کنیم که بتواند برای هر رکورد یک مقدار جدید را به طور خودکار تولید کند.

می خواهیم یک فیلد Auto-increment را در یک جدول ایجاد کنیم.

گرامر MySQL

دستورات زیر ستون "P_Id" را در جدول"Persons" به عنوان کلید اصلی و با خاصیت Auto-increment تعریف می کنند:

```
CREATE TABLE Persons
(
P_Id int NOT NULL AUTO_INCREMENT,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
PRIMARY KEY (P_Id)
)
```

MySQL از کلمه کلیدی AUTO_INCREMENT برای اجرای ویژگی یک auto-increment استفاده می کند.

به طور پیش فرض، مقدار شروع برای خاصیتAUTO_INCREMENT مقدار یک است، و آن را با یک گام برای هر رکورد جدید افزایش می دهد.

برای تعریف مقدار اولیه شروع ترتیب AUTO_INCREMENT ، از دستور زیر استفاده کنید:

```
ALTER TABLE Persons AUTO_INCREMENT=100
```

برای درج یک رکورد جدید به جدول "Persons"، مجبور نخواهیم شد که مقداری برای ستون "P_Id" مشخص کنیم (یک مقدار منحصر به فرد به طور خودکار اضافه خواهد شد) :

```
INSERT INTO Persons (FirstName,LastName)
VALUES ('Lars','Monsen')
```

دستور SQL فوق یک رکورد جدید به جدول "Persons" اضافه می کند. به ستون "P_Id" یک مقدار منحصر به فرد اختصاص داده شده است. ستون "FirstName" مقدار "Lars" و ستون "LastName" مقدار"Monsen"را به خود اختصاص می دهند.

گرامر SQL Server

دستورات SQL زیر ستون "P_Id" در جدول"Persons" را به عنوان کلید اصلی و با خاصیت AUTO_INCREMENT تعریف می کند :

```
CREATE TABLE Persons
(
P_Id int PRIMARY KEY IDENTITY,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255)
)
```

MS SQL Server از کلمه کلیدی IDENTITY برای انجام کار خاصیت auto-increment استفاده می کند.

به طور پیش فرض، مقدار شروع برای خاصیت IDENTITY مقدار یک است و آن را با یک گام برای هر رکورد جدید افزایش می دهد.

برای تعریف ستون "P_Id" که با مقدار ۱۰ شروع و با مقدار ۵ افزایش یابد، باید identity را به IDENTITY(10,5) تغییر دهید.

برای درج یک رکورد جدید به جدول "Persons"، مجبور نخواهیم شد که مقداری برای ستون "P_Id" وارد کنیم (یک مقدار منحصر به فرد به طور خودکار اضافه خواهد شد):

```
INSERT INTO Persons (FirstName,LastName)
VALUES ('Lars','Monsen')
```

دستور SQL فوق یک رکورد جدید به جدول "Persons" اضافه می کند. به ستون "P_Id" یک مقدار منحصر به فرد اختصاص داده شده است. ستون "FirstName" مقدار "Lars" و ستون "LastName" مقدار"Monsen"را به خود اختصاص می دهند.

گرامر Access

دستورات SQL زیر ستون "P_Id" در جدول"Persons" را به عنوان کلید اصلی و با خاصیت AUTO_INCREMENT تعریف می کند :

```
CREATE TABLE Persons
(
P_Id PRIMARY KEY AUTOINCREMENT,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255)
)
```

Access از کلمه کلیدی AUTOINCREMENT برای انجام کار خاصیت auto-increment استفاده می کند.

به طور پیش فرض، مقدار شروع برای خاصیتAUTO_INCREMENT مقدار یک است، و آن را با یک گام برای هر رکورد جدید افزایش می دهد.

برای تعریف ستون "P_Id" که با مقدار ۱۰ شروع و با مقدار ۵ افزایش یابد، باید AUTOINCREMENT را به AUTOINCREMENT(10,5) تغییر دهید.

برای درج یک رکورد جدید به جدول "Persons" ، نمی خواهد که مقداری برای ستون "P_Id" وارد کنیم (یک مقدار منحصر به فرد به طور خودکار اضافه خواهد شد):

```
INSERT INTO Persons (FirstName,LastName)
VALUES ('Lars','Monsen')
```

دستور SQL فوق یک رکورد جدید به جدول "Persons" اضافه می کند. به ستون "P_Id" یک مقدار منحصر به فرد اختصاص داده شده است. ستون "FirstName" مقدار "Lars" و ستون "LastName" مقدار"Monsen"را به خود اختصاص می دهند.

گرامر Oracle

در اوراکل، کد کمی بیشتر فریبنده تر است.

شما مجبور خواهید شد یک فیلد auto-increment با شی ترتیبی (این شی شماره ترتیبی تولید می کند) بسازید.

از گرامر CREATE SEQUENCE استفاده کنید:

```
CREATE SEQUENCE seq_person
MINVALUE 1
START WITH 1
INCREMENT BY 1
CACHE 10
```

کد بالا باعث ایجاد یک توالی با نام seq_person، که با ۱ شروع می شود و توسط ۱ افزایش خواهد کرد. مقدار آن نیز تا ۱۰، برای عملکرد بالا ذخیره گاه (Cache) تعریف کرده است. گزینه Cache مشخص می کند که چند مقدار دنباله ای برای دسترسی سریع تر، در حافظه ذخیره می شوند.

برای درج یک رکورد جدید به جدول "Persons"، باید از تابع nextval استفاده کنیم (این تابع مقدار بعدی از دنباله seq_person را بازیابی می کند):

```
INSERT INTO Persons (P_Id,FirstName,LastName)
```

```
VALUES (seq_person.nextval,'Lars','Monsen')
```

دستور SQL فوق یک رکورد جدید به جدول "Persons" اضافه می کند. به ستون "P_Id" یک مقدار منحصر به فرد اختصاص داده شده است. ستون "FirstName" مقدار "Lars" و ستون "LastName" مقدار"Monsen"را به خود اختصاص می دهند.

View ها در SQL

View یک جدول مجازی است.

این فصل نشان می دهد که چگونه یک View را ایجاد، بروز رسانی و پاک کنید.

دستور CREATE VIEW در SQL

در SQL، یک view یک جدول مجازی است که بر اساس نتیجه مجموعه ای از دستورات SQL به وجود می آید.

یک view شامل سطر و ستون، درست مثل یک جدول واقعی است. فیلد ها در یک view، همان فیلد ها از یک یا چند جدول واقعی در پایگاه داده هستند.

شما می توانید توابع SQL، WHERE و دستورات JOIN را به یک view اضافه کنید و داده را ارائه دهید؛ در صورتی که اطلاعات از یک جدول واحد باشند.

گرامر CREATE VIEW در SQL

```
CREATE VIEW view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition
```

توجه: یک view همواره اطلاعات بروز را نشان می دهد! موتور پایگاه داده داده ها را بازسازی می کند و با استفاده از دستور view's SQL ، هر بار که یک کاربر نمایش داده شد یک view است.

مثال CREATE VIEW در SQL

اگر شما پایگاه داده Northwind را دارید، می توانید ببینید که چندین view نصب شده به طور پیش فرض دارد.

نما (view) با نام "Current Product List"، تمام محصولات فعال (محصولاتی که موقوف نشده اند) از جدول "Products" لیست می کند. این view با دستورات SQL زیر ایجاد شده:

```
CREATE VIEW [Current Product List] AS
SELECT ProductID,ProductName
FROM Products
WHERE Discontinued='No'
```

می توانیم از view بالا به صورت زیر پرس و جو کنیم:

```
SELECT * FROM [Current Product List]
```

یکی دیگر از view ها در پایگاه داده Northwind بمنظور انتخاب هر محصول در جدول "Products" با یک قیمت واحد ، بالاتر از میانگین قیمت واحد است:

```
CREATE VIEW [Products Above Average Price] AS
SELECT ProductName,UnitPrice
FROM Products
WHERE UnitPrice>(SELECT AVG(UnitPrice) FROM Products)
```

می توانیم از view بالا به صورت زیر پرس و جو کنیم:

```
SELECT * FROM [Products Above Average Price]
```

یکی دیگر از view ها در پایگاه داده Northwind محاسبه کل فروش برای هر دسته بندی در سال ۱۹۹۷ است. توجه داشته باشید که این view اطلاعات خود را از view دیگری به نام "Product Sales for 1997" انتخاب می کند:

```
CREATE VIEW [Category Sales For 1997] AS
SELECT DISTINCT CategoryName,Sum(ProductSales) AS CategorySales
FROM [Product Sales for 1997]
GROUP BY CategoryName
```

می توانیم از view بالا به صورت زیر پرس و جو کنیم:

```
SELECT * FROM [Category Sales For 1997]
```

همچنین می توانیم یک شرط به پرس و جو اضافه کنیم. حالا کل فروش فقط در دسته "Beverages" را می خواهیم:

```
SELECT * FROM [Category Sales For 1997]
WHERE CategoryName='Beverages'
```

بروز رسانی VIEW

شما می توانید با استفاده از گرامر زیر یک view را بروز رسانی کنید:

گرامر CREATE OR REPLACE VIEW در SQL

```
CREATE OR REPLACE VIEW view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition
```

حالا می خواهیم ستون "Category" را به نمای "Current Product List" اضافه کنیم. این نما را با استفاده از دستورات SQL زیر بروز رسانی خواهیم کرد:

```
CREATE VIEW [Current Product List] AS
SELECT ProductID,ProductName,Category
FROM Products
WHERE Discontinued='No'
```

حذف کردن یک View در SQL

شما می توانید یک نما را با استفاده از دستور DROP VIEW حذف کنید.

گرامر DROP VIEW در SQL

```
DROP VIEW view_name
```

توابع تاریخ در SQL

تاریخ در SQL

توجه داشته باشید سخت ترین بخش در هنگام کار با تاریخ این است که اطمینان حاصل کنید که فرمت تاریخ شما زمان وارد کردن، مطابق با فرمت ستون تاریخ در پایگاه داده باشد.

تا زمانی که محتویات شامل بخش تاریخ باشد، انتظار می رود که در پرس و جو تان کار کند. اما اگر یک بخش هم درگیر شود، آن را پیچیده می کند.

قبل از صحبت کردن در مورد عوارض ناشی از پرس و جو برای تاریخ، به مهم ترین توابع داخلی در رابطه با تاریخ نگاهی بیندازید.

توابع تاریخ در MySQL

جدول زیر لیست مهم ترین توابع داخلی در رابطه با تاریخ در MySQL هستند:

تابع	توضیحات
NOW()	تاریخ و زمان فعلی را برمی گرداند.
CURDATE()	تاریخ جاری را برمی گرداند.
CURTIME()	زمان جاری را برمی گرداند.
DATE()	بخش تاریخ از یک عبارت تاریخ یا تاریخ / زمان را استخراج می کند.
EXTRACT()	تنها بخشی از یک تاریخ / زمان را بر می گرداند .
DATE_ADD()	یک بازه زمانی مشخص شده را به یک تاریخ اضافه می کند.
DATE_SUB()	یک بازه زمانی مشخص شده از تاریخ است.
DATEDIFF()	تعداد روزهای بین دو تاریخ را برمی گرداند.
DATE_FORMAT()	نمایش تاریخ / زمان داده ها در فرمت های مختلف است.

توابع تاریخ در SQL Server

جدول زیر لیستی از مهم ترین توابع تاریخ داخلی در SQL Server هستند:

تابع	توضیحات
GETDATE()	تاریخ و زمان فعلی را برمی گرداند.
DATEPART()	بخشی از یک تاریخ / زمان را برمی گرداند.
DATEADD()	یک بازه زمانی مشخص شده از تاریخ را اضافه یا کم می کند.
DATEDIFF()	زمان بین دو تاریخ را برمی گرداند.
CONVERT()	داده تاریخ / زمان در فرمت های مختلف نمایش می دهد.

انواع داده تاریخ در SQL

MySQL با انواع داده های زیر، برای ذخیره سازی یک مقدار تاریخ یا یک تاریخ / زمان در پایگاه داده می آید:

- فرمت DATE: YYYY-MM-DD
- فرمت DATETIME: YYYY-MM-DD HH:MM:SS
- فرمت TIMESTAMP: YYYY-MM-DD HH:MM:SS
- فرمت YEAR: YYYY یا YY

SQL Server با انواع داده های زیر، برای ذخیره سازی یک مقدار تاریخ یا یک تاریخ / زمان در پایگاه داده می آید:

- فرمت DATE: YYYY-MM-DD
- فرمت DATETIME: YYYY-MM-DD HH:MM:SS
- فرمت SMALLDATETIME: YYYY-MM-DD HH:MM:SS
- فرمت TIMESTAMP: یک عدد منحصر به فرد

توجه: زمانی که یک جدول جدید در پایگاه داده خود می سازید، انواع تاریخ برای یک ستون انتخاب می شوند!

برای مرور بر همه نوع داده های موجود، به مرجع کامل داده ها رجوع کنید.

کار با تاریخ در SQL

توجه داشته باشید می توانید دو تاریخ را به راحتی مقایسه کنید؛ اگر هیچ عنصر زمان مبهمی وجود نداشته باشد!

جدول "Orders" به فرض زیر است:

OrderId	ProductName	OrderDate
1	Geitost	2008-11-11
2	Camembert Pierrot	2008-11-09
3	Mozzarella di Giovanni	2008-11-11
4	Mascarpone Fabioli	2008-10-29

حالا می خواهیم از جدول بالا در ستون OrderDate رکورد "2008-11-11" را انتخاب کنیم.

از دستور SELECT زیر استفاده می کنیم:

```
SELECT * FROM Orders WHERE OrderDate='2008-11-11'
```

جدول نتیجه اینگونه خواهد شد:

OrderId	ProductName	OrderDate
1	Geitost	2008-11-11
3	Mozzarella di Giovanni	2008-11-11

حال، فرض کنیم که جدول "Orders" به این صورت باشد (به عنصر زمان در ستون "OrderDate" توجه کنید):

OrderId	ProductName	OrderDate
1	Geitost	2008-11-11 13:23:44
2	Camembert Pierrot	2008-11-09 15:45:21
3	Mozzarella di Giovanni	2008-11-11 11:12:01
4	Mascarpone Fabioli	2008-10-29 14:56:59

اگر از همان دستور SELECT استفاده کنیم:

```
SELECT * FROM Orders WHERE OrderDate='2008-11-11'
```

هیچ نتیجه ای حاصل نمی شود، این بخاطر است که پرس و جو فقط فرمت تاریخ بدون بخش زمان را جستجو می کند.

نکته: اگر می خواهید پرس و جو تان را ساده و آسان نگهداری کنید، به عنصر زمان در تاریخ تان اجازه ورود ندهید!

مقدار NULL

مقدار NULL نشان دهنده داده های ناشناخته هستند.

به طور پیش فرض، یک ستون جدول می تواند مقدار NULL را نگه دارد.

این قسمت عملگرهای IS NULL و NOT NULL را توضیح خواهد داد.

مقادیر NULL در SQL

اگر یک ستون در یک جدول اختیاری باشد، می توانیم بدون اضافه کردن یک مقدار به این ستون یک رکورد جدید ساخته یا بروز رسانی کنیم. این بدان معنی است که این فیلد با یک مقدار NULL پر خواهد شد.

مقدار NULL از بقیه مقادیر متفاوت عمل می کنند.

NULL به عنوان مقادیر ناشناخته و یا غیر قابل اجرا استفاده می شود.

نکته: مقایسه NULL و عدد ۰ ممکن نیست. آنها با هم برابر نیستند.

کار کردن با مقدار NULL

جدول "Persons":

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola		Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari		Stavanger

فرض کنید که ستون "Address" در جدول "Persons" اختیاری است. این به این معنی است که اگر هیچ مقداری برای این ستون وارد نکنید با مقدار NULL را پر می شود.

چگونه می توان مقدار NULL را آزمایش کنیم؟

ممکن نیست این کار را با عملگر های رابطه مانند = ، <، و یا < > انجام دهید.

می خواهیم از عملگر های IS NULL وIS NOT NULL استفاده کنیم.

مقدار IS NULL

چگونه می توانم تنها رکورد هایی با مقادیر NULL را در ستون "Address" انتخاب کنیم؟

می خواهیم از عملگر NULL استفاده کنیم:

```
SELECT LastName,FirstName,Address FROM Persons
WHERE Address IS NULL
```

جدول نتیجه اینگونه خواهد شد:

LastName	FirstName	Address
Hansen	Ola	
Pettersen	Kari	

نکته: همیشه از IS NULL برای جستجوی مقادیر NULL استفاده کنید.

مقدار IS NOT NULL

چگونه می توانم تنها رکوردهای با مقادیر NOT NULL را در ستون "Address" انتخاب کنم؟

می خواهیم از عملگر IS NOT NULL استفاده کنیم:

```
SELECT LastName,FirstName,Address FROM Persons
WHERE Address IS NOT NULL
```

جدول نتیجه اینگونه خواهد شد:

LastName	FirstName	Address
Svendson	Tove	Borgvn 23

در قسمت بعد به توابع ISNULL()، NVL()، IFNULL()، COALESCE() نگاه خواهیم کرد.

توابع NULL در SQL

توابع IFNULL(), NVL(), ISNULL() و COALESCE()

به جدول "Products" زیر نگاه کنید:

P_Id	ProductName	UnitPrice	UnitsInStock	UnitsOnOrder
1	Jarlsberg	10.45	16	15
2	Mascarpone	32.56	23	
3	Gorgonzola	15.67	9	20

فرض کنید که ستون "UnitsOnOrder" اختیاری است، و ممکن است حاوی مقدار NULL باشد.

از دستور SELECT زیر استفاده کنید:

```
SELECT ProductName,UnitPrice*(UnitsInStock+UnitsOnOrder)
FROM Products
```

در مثال بالا، اگر هر یک از مقادیر ستون "UnitsOnOrder" مقدار NULL باشد، نتیجه NULL است.

تابع ISNULL() موجود در Access برای مشخص کردن اینکه چگونه می خواهیم مقادیر NULL رفتار کنند، استفاده می شود.

توابع IFNULL(), NVL() و COALESCE() نیز می تواند برای رسیدن به همان نتیجه مورد استفاده قرار گیرد.

در این حالت می خواهیم مقدار NULL صفر باشد.

در زیر، اگر ستون "UnitsOnOrder" خالی باشد در محاسبه هیچ صدمه ای نمی زند، زیرا اگر مقدار NULL باشد، تابع ISNULL() صفر را بر می گرداند:

SQL Server / MS Access

```
SELECT ProductName,UnitPrice*(UnitsInStock+ISNULL(UnitsOnOrder,0))
FROM Products
```

Oracle

اوراکل تابع ISNULL() ندارد. با این حال، می توانیم برای رسیدن به نتیجه مشابه از تابع NVL() استفاده کنیم:

```
SELECT ProductName,UnitPrice*(UnitsInStock+NVL(UnitsOnOrder,0))
FROM Products
```

MySQL

MySQL یک تابع ISNULL() دارد. اما کمی متفاوت از تابعISNULL() موجود در مایکروسافت کار می کند.

در MySQL می توانیم از تابع IFNULL() استفاده کنیم :

```
SELECT ProductName,UnitPrice*(UnitsInStock+IFNULL(UnitsOnOrder,0))
FROM Products
```

و یا می توانیم از تابع COALESCE() که شبیه به این است، استفاده کنیم:

```
SELECT ProductName,UnitPrice*(UnitsInStock+COALESCE(UnitsOnOrder,0))
FROM Products
```

نوع داده ها در SQL

انواع داده ها و محدوده ها برای MySQL, Microsoft Access و SQL Server

انواع داده ها برای Microsoft Access

نوع داده	توضیحات	فضای لازم
Text	استفاده از متن و یا ترکیبی از متن و اعداد.	1 byte
Memo	برای مقادیر بزرگتر از متن استفاده می شود. تا ۶۵۵۳۶ کاراکتر ذخیره می کند. توجه: شما یک Memo را نمی توانید مرتب کنید . هر چند قابل جستجو هستند.	2 bytes
Byte	تا اعداد صحیح از ۰ تا ۲۵۵ اجازه می دهد.	4 bytes
Integer	تا اعداد صحیح بین ۳۲۷۶۸- و ۳۲۷۶۷ اجازه می دهد.	4 bytes
Long	تا تعداد کل بین ۲۱۴۷۴۸۳۶۴۸- تا ۲۱۴۷۴۸۳۶۴۷ اجازه می دهد.	8 bytes
Single	یک دقت ممیز شناور. رقم اعشار بیشتر را تحمل می کند.	8 bytes

Double	دو دقت اعشاری. رقم اعشار بیشتر را تحمل.	4 bytes
Currency	برای پول استفاده می شود. تا ۱۵ رقم کامل دلار، به علاوه ۴ رقم اعشار را نگه می دارد. نکته: شما می توانید ارز کشور خود را انتخاب کنید.	8 bytes
AutoNumber	زمینه های AutoNumber به طور خودکار به هر رکورد یک شماره اضافه می کند و با یک شروع می شود.	1 bit
Date/Time	استفاده از تاریخ و زمان.	up to 1GB
Yes/No	در زمینه های منطقی می تواند به عنوان بله / خیر، درست / غلط، و یا روشن / خاموش نمایش داده شود.در کد ، استفاده از ثابت های Yes/No(معادل ۱- و ۰). توجه: مقدار NULL در Yes/No مجاز نیست.	
Ole Object	تصاویر، صدا، ویدئو، یا BLOBs (اشیاء باینری بزرگ)را می تواند ذخیره کند.	4 byte
Hyperlink	شامل پیوندهایی به فایل های دیگر، از جمله صفحات وب.	
Lookup Wizard	یک لیست از گزینه ها است که می توانید از لیست کشویی انتخاب وارد کنید.	

انواع داده ها برای MySQL

در MySQL سه نوع داده اصلی وجود دارد: متن، شماره و تاریخ / زمان.

انواع متن:

نوع داده	توضیحات
CHAR(size)	دارای یک رشته با طول ثابت است(می تواند شامل حروف، اعداد و کاراکترهای خاص باشد). اندازه ثابت در پرانتز مشخص می شود. توانایی ذخیره سازی تا ۲۵۵ کاراکتر را دارد.
VARCHAR(size)	دارای یک رشته با طول متغیر است(می تواند شامل حروف، اعداد و کاراکترهای خاص باشد). حداکثر اندازه در پرانتز مشخص می شود. توانایی ذخیره سازی تا ۲۵۵ کاراکتر را دارد. توجه: اگر شما یک مقدار بزرگتر از ۲۵۵ قرار دهید آن را تبدیل به نوع متن می کند.
TINYTEXT	دارای یک رشته با طول حداکثر ۲۵۵ کاراکتر است.
TEXT	دارای یک رشته با طول حداکثر ۶۵۵۳۵ کاراکتر است.
BLOB	برای BLOBs(اشیاء باینری بزرگ). تا ۶۵۵۳۵ بایت از داده ها را نگه می دارد.
MEDIUMTEXT	دارای یک رشته با طول حداکثر ۱۶۷۷۷۲۱۵ کاراکتر است.

MEDIUMBLOB	برای BLOBs(اشیاء باینری بزرگ). تا ۱۶۷۷۷۲۱۵ بایت از داده ها را نگه می دارد.
LONGTEXT	تا ۱۶۷۷۷۲۱۵ بایت از داده ها نگه می دارد.
LONGBLOB	برای BLOBs(اشیاء باینری بزرگ). تا ۴۲۹۴۹۶۷۲۹۵ بایت از داده ها را نگه می دارد.
ENUM(x, y, z, etc.)	اجازه می دهد یک لیست از مقادیر ممکن را وارد کنید. شما می توانید در این لیست تا ۵۳۵،۶۵ مقدار ذخیره کنید. اگر مقدار درج شده در لیست نیست، یک مقدار خالی به جای آن قرار خواهد گرفت. توجه داشته باشید: مقادیر وارد شده در آن مرتب شده اند. مقادیر ممکن را باید با این فرمت وارد کنید : ENUM('X','Y','Z')
SET	شبیه به ENUM است البته به جز آن مجموعه ممکن است شامل بیش از ۶۴ اقلام لیست و می تواند بیش از یک انتخاب داشته باشد.

انواع شماره ها یا اعداد:

نوع داده	توضیحات
TINYINT(size)	۱۲۸- تا ۱۲۷ نرمال. از ۰ تا ۲۵۵ بدون علامت*.حداکثر تعداد ارقام ممکن است در داخل پرانتز مشخص شود.
SMALLINT(size)	۳۲۷۶۸- تا ۳۲۷۶۷ نرمال. ۰ تا ۶۵۵۳۵ بدون علامت*.حداکثر تعداد ارقام ممکن است در داخل پرانتز مشخص شود .
MEDIUMINT(size)	۸۳۸۸۶۰۸- تا ۸۳۸۸۶۰۷ عادی. ۰ تا ۱۶۷۷۷۲۱۵ بدون علامت*.حداکثر تعداد ارقام ممکن است در داخل پرانتز مشخص شود.
INT(size)	۲۱۴۷۴۸۳۶۴۸- تا ۲۱۴۷۴۸۳۶۴۷ عادی. ۰ تا ۴۲۹۴۹۶۷۲۹۵ بدون علامت*.حداکثر تعداد ارقام ممکن است در داخل پرانتز مشخص شود.
BIGINT(size)	۹۲۲۳۳۷۲۰۳۶۸۵۴۷۷۵۸۰۸ - تا ۹۲۲۳۳۷۲۰۳۶۸۵۴۷۷۵۸۰۷ عادی. ۰ تا ۱۸۴۴۶۷۴۴۰۷۳۷۰۹۵۵۱۶۱۵ بدون علامت*.حداکثر تعداد ارقام ممکن است در داخل پرانتز مشخص شود.

FLOAT(size,d)	تعداد کمی ازاعداد با نقطه اعشار شناور می آیند. حداکثر تعداد ارقام ممکن است در پارامتر size مشخص شده باشد.حداکثر تعداد ارقام را در سمت راست نقطه اعشار با پارامتر d مشخص می کنیم.
DOUBLE(size,d)	تعداد زیادی ازاعداد با نقطه اعشار شناور می آیند. حداکثر تعداد ارقام ممکن است در پارامتر size مشخص شده باشد.حداکثر تعداد ارقام را در سمت راست نقطه اعشار با پارامتر d مشخص می کنیم.
DECIMAL(size,d)	DOUBLE به عنوان یک رشته ذخیره می شود و برای یک نقطه ثابت اعشاری اجازه می دهد.حداکثر تعداد ارقام ممکن است در پارامتر size مشخص شده باشد. حداکثر تعداد ارقام را در سمت راست نقطه اعشار با پارامتر d مشخص می کنیم.

*انواع داده های صحیح گزینه اضافی به نام UNSIGNED (بدون علامت)دارند . به طور معمول، عدد صحیح را از منفی به مثبت تبدیل می کند. اضافه کردن این ویژگی محدوده را بدون علامت خواهد کرد و محدوده را با حرکت از صفر (به جای یک عدد منفی)شروع می کند.

نوع تاریخ:

نوع داده	توضیحات
DATE()	ورودی تاریخ است . با فرمت : YYYY-MM-DD توجه : این نوع داده از دامنه '1000-01-01' تا '9999-12-31' پشتیبانی می کند.
DATETIME()	*ورودی ترکیبی از تاریخ و زمان است. با فرمت :YYYY-MM-DD HH:MM:SS توجه : این نوع داده از دامنه '1000-01-01 00:00:00' تا '9999-12-31 23:59:59' پشتیبانی می کند.
TIMESTAMP()	*ورودی زمان است . تعدادی از ثانیه های بعد از مبدا تاریخ یونیکس ('1970-01-01 00:00:00' UTC) را ذخیره می کند. با فرمت : YYYY-MM-DD HH:MM:SS توجه : این نوع داده از دامنه '1970-01-01 00:00:01' UTC تا '2038-01-09 03:14:07' UTC پشتیبانی می کند.

TIME()	ورودی زمان است . با فرمت : HH:MM:SS توجه : این نوع داده از دامنه '838:59:59-' تا '838:59:59' پشتیبانی می کند.
YEAR()	ورودی یک سال در قالب دو رقمی یا چهار رقمی است. توجه: مقادیر مجاز در فرمت چهار رقمی: ۱۹۰۱ تا ۲۱۵۵. مقادیر مجاز در فرمت دو رقمی: سال ۷۰ تا ۶۹ بر گرفته از سال ۱۹۷۰ تا ۲۰۶۹.

*فرمت دو تابع DATETIME() و TIMESTAMP() شبیه به هم است ولی از نظر کارکرد بسیار با هم متفاوت هستند. در پرس و جوی INSERT و یا UPDATE،داده TIMESTAMP به طور خودکار خود را به تاریخ و زمان فعلی می چسباند. برچسب زمان فرمت های مختلفی می پذیرد مانند YYYYMMDDHHMMSS، YYMMDDHHMMSS، YYYYMMDD، یا YYMMDD.

انواع داده ها برایSQL Server

رشته های کاراکتری:

نوع داده	توضیحات	فضای لازم
char(n)	رشته کاراکتری با طول ثابت است. حداکثر ۸۰۰۰ کاراکتر است.	n
varchar(n)	رشته کاراکتری با طول متغیر. حداکثر ۸۰۰۰ کاراکتر است.	
varchar(max)	رشته کاراکتری با طول متغیر. حداکثر ۱۰۷۳۷۴۱۸۲۴ کاراکتر.	
text	رشته کاراکتری با طول متغیر. حداکثر GB۲ داده های متنی .	

رشته های یونیکد :

نوع داده	توضیحات	فضای لازم
nchar(n)	داده های یونیکد با طول ثابت است. حداکثر ۴۰۰۰ کاراکتر است.	
nvarchar(n)	داده های یونیکد با طول متغیراست. حداکثر ۴۰۰۰ کاراکتر است.	

nvarchar(max)	داده های یونیکد با طول متغیراست. حداکثر ۹۱۲،۸۷۰،۵۳۶کاراکتر است.	
ntext	داده های یونیکد با طول متغیراست. حداکثر GB۲ داده های متنی .	

نوع دودویی:

نوع داده	توضیحات	فضای لازم
bit	اعداد ۰ و ۱ و Null	
binary(n)	داده های دودویی با طول ثابت است. حداکثر ۸۰۰۰ بایت است.	
varbinary(n)	داده های دودویی با طول متغیر است. حداکثر ۸۰۰۰ بایت است.	
varbinary(max)	داده های دودویی با طول متغیر است. حداکثر GB۲	
image	داده های دودویی با طول متغیر است. حداکثر GB۲	

انواع شماره ها یا اعداد:

نوع داده	توضیحات	فضای لازم
tinyint	اعداد صحیح از ۰ تا ۲۵۵	1 byte
smallint	اعداد صحیح بین ۳۲۷۶۸- و ۳۲۷۶۷	2 bytes
int	تعداد کل اعداد بین ۲۱۴۷۴۸۳۶۴۸- تا ۲۱۴۷۴۸۳۶۴۷	4 bytes
bigint	تعداد کل اعداد بین ۹.۲۲۳.۳۷۲.۰۳۶.۸۵۴.۷۷۵.۸۰۸- و ۹.۲۲۳.۳۷۲.۰۳۶.۸۵۴.۷۷۵.۸۰۷	8 bytes
decimal(p,s)	اعداد با دقت ثابت و اعداد مقیاس. اعداد از ۱۰ ^ ۳۸ +۱- به ۱۰ ^ ۳۸ -۱. پارامتر P نشان می دهد تعداد کل حداکثر عددی که می تواند ذخیره می شود (هر دو در سمت چپ و در سمت راست نقطه اعشار). P باید مقدار ۱ تا ۳۸ داشته باشد. به طور پیش فرض ۱۸ است. پارامتر S نشان می دهد، حداکثر تعداد رقم های ذخیره شده در سمت راست نقطه اعشار است. S باید یک مقدار را از ۰ به P داشته باشد. مقدار پیش فرض ۰ است .	5-17 bytes

numeric(p,s)	اعداد با دقت ثابت و اعداد مقیاس.اعداد از ۱۰ ^ ۳۸ +۱- به ۱۰ ^ ۳۸ -۱. پارامتر P نشان می دهد تعداد کل حداکثر عددی که می تواند ذخیره می شود (هر دو در سمت چپ و در سمت راست نقطه اعشار). P باید مقدار ۱ تا ۳۸ داشته باشد. به طور پیش فرض ۱۸ است. پارامتر S نشان می دهد، حداکثر تعداد رقم های ذخیره شده در سمت راست نقطه اعشار است. S باید یک مقدار را از ۰ به P داشته باشد. مقدار پیش فرض ۰ است .	5-17 bytes
smallmoney	داده های پول از ۲۱۴۷۴۸.۳۶۴۸- تا ۲۱۴۷۴۸.۳۶۴۷	4 bytes
money	داده های پول از ۹۲۲۳۳۷۲۰۳۶۸۵۴۷۷.۵۸۰۸- به ۹۲۲۳۳۷۲۰۳۶۸۵۴۷۷.۵۸۰۷	8 bytes
float(n)	داده های عددی با دقت اعشار از 1.79E + 308- تا 1.79E + 308 پارامتر N نشان می دهد که آیا این رشته باید ۴ یا ۸ بایت را تحمل کند. float(24) دارای یک میدان ۴ بایتی و float(53) دارای یک میدان ۸ بایتی است. مقدار پیش فرض برای n عدد 53 است.	4 or 8 bytes
real	اطلاعات دقیق اعداد از شماره 3.40E + 38- تا 3.40E + 38	4 bytes

نوع تاریخ:

نوع داده	توضیحات	فضای لازم
datetime	از تاریخ ۱ ژانویه ۱۷۵۳ تا ۳۱ دسامبر، ۹۹۹۹ با دقت ۳.۳۳ میلی ثانیه است.	8 bytes
datetime2	از تاریخ ۱ژانویه ، ۰۰۰۱ تا ۳۱ دسامبر، ۹۹۹۹ با دقت ۱۰۰ نانو ثانیه است .	6-8 bytes
smalldatetime	از تاریخ ۱ ژانویه سال ۱۹۰۰ تا ۶ژوئن ، ۲۰۷۹، با دقت ۱ دقیقه است .	4 bytes
date	فقط ذخیره تاریخ. از ۱ژانویه ، ۰۰۰۱ تا ۳۱ دسامبر، ۹۹۹۹.	3 bytes
time	فقط ذخیره زمان با دقت ۱۰۰ نانو ثانیه است .	3-5 bytes
datetimeoffset	همان datetime2 است علاوه بر این از یک افست منطقه زمانی.	8-10 bytes
timestamp	یک شماره منحصر به فرد ذخیره می شود در زمانی که یک ردیف ایجاد می شود و یا با تغییر بروز رسانی می شود. مقدار timestamp زمان را بر اساس یک ساعت داخلی تنظیم می کند و به زمان واقعی مطابقت ندارد. هر جدول، ممکن است فقط یک زمان متغیر داشته باشد.	

انواع دیگر داده ها :

نوع داده	توضیحات
sql_variant	ذخیره بیشتر از ۸۰۰۰ بایت از داده ها را از انواع داده های مختلف، به جز متن، ntext، و زمان .
uniqueidentifier	ذخیره یک شناسه منحصر به فرد در سطح جهان (GUID).
xml	ذخیره XML با فرمت داده ها. حداکثر GB۲.
cursor	ذخیره مرجع با یک اشاره گر برای عملیات پایگاه داده استفاده می شود.
table	ذخیره یک نتیجه برای پردازش در آینده.

فصل سوم - توابع SQL

توابع SQL

SQL توابع داخلی زیادی برای انجام محاسبات بر روی داده دارد.

توابع جمعی SQL

توابع جمعی SQL یک مقدار تکی محاسبه شده از مقادیر یک ستون را برمی گردانند.

توابع جمعی مفید:

- AVG() - معدل را برمی گرداند
- COUNT() - تعداد سطرها را برمی گرداند
- FIRST() - اولین مقدار را برمی گرداند
- LAST() - آخرین مقدار را برمی گرداند
- MAX() - بیشترین مقدار را برمی گرداند
- MIN() - کمترین مقدار را برمی گرداند
- SUM() - مجموع را برمی گرداند

توابع اسکالر SQL

توابع اسکالر SQL یک مقدار تکی مبنی بر مقدار ورودی را برمی گردانند.

توابع اسکالر مفید:

- UCASE() - تبدیل رشته به حروف بزرگ
- LCASE() - تبدیل رشته به حروف کوچک
- MID() - استخراج کاراکترها از یک رشته متنی
- LEN() - طول یک رشته متنی را برمی گرداند
- ROUND() - یک رشته عددی را با تعداد اعشار مشخص گرد می کند
- NOW() - تاریخ و ساعت فعلی سیستم را برمی گرداند
- FORMAT() - چگونگی نمایش یک فیلد را مشخص می کند

تابع AVG()

تابع AVG() مقدار میانگین از یک ستون عددی را برمی گرداند.

گرامر AVG() در SQL

```
SELECT AVG(column_name) FROM table_name
```

مثال AVG() در SQL

جدول "Orders" زیر را داریم:

O_Id	OrderDate	OrderPrice	Customer
1	2008/11/12	1000	Hansen
2	2008/10/23	1600	Nilsen
3	2008/09/02	700	Hansen
4	2008/09/03	300	Hansen
5	2008/08/30	2000	Jensen
6	2008/10/04	100	Nilsen

اکنون می خواهیم مقدار میانگین فیلد "OrderPrice" را پیدا کنیم.

از دستور SQL زیر استفاده می کنیم:

```
SELECT AVG(OrderPrice) AS OrderAverage FROM Orders
```

نتیجه اینگونه خواهد شد:

OrderAverage
950

اکنون می خواهیم مشتریانی که (حداقل) یک مقدار "OrderPrice" بالاتر از مقدار "OrderPrice" میانگین دارند را پیدا کنیم.

از دستور SQL زیر استفاده می کنیم:

```
SELECT Customer FROM Orders
WHERE OrderPrice>(SELECT AVG(OrderPrice) FROM Orders)
```

نتیجه اینگونه خواهد شد:

Customer
Hansen
Nilsen
Jensen

تابع COUNT() در SQL

تابع COUNT() در SQL تعدادی سطر که با یک معیارهای مشخص تطبیق داده شده اند را برمی گرداند.

گرامر COUNT(column_name) در SQL

تابع COUNT(column_name) تعدادی مقادیر (مقادیر NULL شمارش نخواهد شد) از ستون مشخصی را برمی گرداند:

```
SELECT COUNT(column_name) FROM table_name
```

گرامر COUNT(*) در SQL

تابع COUNT(*) در SQL رکورد از یک جدول را برمی گرداند:

```
SELECT COUNT(*) FROM table_name
```

گرامر COUNT(DISTINCT column_name) در SQL

تابع COUNT(DISTINCT column_name) تعدادی مقادیر متمایز از ستون مشخصی را برمی گرداند:

```
SELECT COUNT(DISTINCT column_name) FROM table_name
```

توجه: COUNT(DISTINCT) در ORACLE و Microsoft SQL Server کار می کند و در Microsoft Access کار نمی کند.

مثال COUNT(DISTINCT column_name) در SQL

جدول "Orders" زیر را داریم:

O_Id	OrderDate	OrderPrice	Customer
1	2008/11/12	1000	Hansen
2	2008/10/23	1600	Nilsen
3	2008/09/02	700	Hansen
4	2008/09/03	300	Hansen
5	2008/08/30	2000	Jensen
6	2008/10/04	100	Nilsen

اکنون می خواهیم تعداد سفارشات مشتری "Nilsen" را بشماریم:

از دستور SQL زیر استفاده می کنیم:

```
SELECT COUNT(Customer) AS CustomerNilsen FROM Orders
WHERE Customer='Nilsen'
```

نتیجه دستور SQL بالا ۲ خواهد شد؛ چون مشتری Nilsen مجموعا ۲ سفارش دارد:

CustomerNilsen
2

مثال COUNT(*) در SQL

اگر از عبارت WHERE صرفنظر کنیم، مثل این:

```
SELECT COUNT(*) AS NumberOfOrders FROM Orders
```

نتیجه این گونه خواهد شد:

NumberOfOrders
6

که مجموع تعداد سطرهای جدول است.

مثال COUNT(DISTINCT column_name) در SQL

اکنون می خواهیم تعداد مشتریان غیر تکراری در جدول "Orders" را بشماریم.

از دستور SQL زیر استفاده می کنیم:

```
SELECT COUNT(DISTINCT Customer) AS NumberOfCustomers FROM Orders
```

نتیجه اینگونه خواهد شد:

NumberOfCustomers
3

که تعداد مشتریان غیر تکراری (Hansen, Nilsen و Jansen) در جدول است.

تابع FIRST()

تابع FIRST() اولین مقدار از ستون انتخاب شده را برمی گرداند.

گرامر FIRST() در SQL

```
SELECT FIRST(column_name) FROM table_name
```

مثال FIRST() در SQL

جدول "Orders" زیر را داریم:

O_Id	OrderDate	OrderPrice	Customer
1	2008/11/12	1000	Hansen
2	2008/10/23	1600	Nilsen
3	2008/09/02	700	Hansen
4	2008/09/03	300	Hansen
5	2008/08/30	2000	Jensen
6	2008/10/04	100	Nilsen

اکنون می خواهیم اولین مقدار از ستون "OrderPrice" را پیدا کنیم.

از دستور SQL زیر استفاده می کنیم:

```
SELECT FIRST(OrderPrice) AS FirstOrderPrice FROM Orders
```

نکته: راه حلی وجود دارد؛ اگر تابع FIRST() پشتیبانی نشود.

```
SELECT OrderPrice FROM Orders ORDER BY O_Id LIMIT 1
```

نتیجه اینگونه خواهد شد:

FirstOrderPrice
1000

تابع LAST()

تابع LAST() آخرین مقدار از ستون انتخاب شده را برمی گرداند.

گرامر LAST() در SQL

```
SELECT LAST(column_name) FROM table_name
```

مثال LAST() در SQL

جدول "Orders" زیر را داریم:

O_Id	OrderDate	OrderPrice	Customer
1	2008/11/12	1000	Hansen
2	2008/10/23	1600	Nilsen
3	2008/09/02	700	Hansen
4	2008/09/03	300	Hansen
5	2008/08/30	2000	Jensen
6	2008/10/04	100	Nilsen

اکنون می خواهیم آخرین مقدار از ستون "OrderPrice" را پیدا کنیم.

از دستور SQL زیر استفاده می کنیم:

```
SELECT LAST(OrderPrice) AS LastOrderPrice FROM Orders
```

نکته: راه حلی وجود دارد؛ اگر تابع LAST() پشتیبانی نشود.

```
SELECT OrderPrice FROM Orders ORDER BY O_Id DESC LIMIT 1
```

نتیجه اینگونه خواهد شد:

LastOrderPrice
100

تابع MAX()

تابع MAX() بزرگترین مقدار از ستون انتخاب شده برمی گرداند.

گرامر MAX() در SQL

```
SELECT MAX(column_name) FROM table_name
```

مثال MAX() در SQL

جدول "Orders" زیر را داریم:

O_Id	OrderDate	OrderPrice	Customer
1	2008/11/12	1000	Hansen
2	2008/10/23	1600	Nilsen
3	2008/09/02	700	Hansen
4	2008/09/03	300	Hansen
5	2008/08/30	2000	Jensen
6	2008/10/04	100	Nilsen

اکنون می خواهیم بزرگترین مقدار از ستون "OrderPrice" را پیدا کنیم.

از دستور SQL زیر استفاده می کنیم:

```
SELECT MAX(OrderPrice) AS LargestOrderPrice FROM Orders
```

نتیجه اینگونه خواهد شد:

LargestOrderPrice
2000

تابع MIN()

تابع MIN() کوچکترین مقدار از ستون انتخاب شده را برمی گرداند.

گرامر MIN() در SQL

```
SELECT MIN(column_name) FROM table_name
```

مثال MIN() در SQL

جدول "Orders" زیر را داریم:

O_Id	OrderDate	OrderPrice	Customer
1	2008/11/12	1000	Hansen
2	2008/10/23	1600	Nilsen
3	2008/09/02	700	Hansen
4	2008/09/03	300	Hansen
5	2008/08/30	2000	Jensen
6	2008/10/04	100	Nilsen

اکنون می خواهیم کوچکترین مقدار از ستون "OrderPrice" را پیدا کنیم.

از دستور SQL زیر استفاده می کنیم:

```
SELECT MIN(OrderPrice) AS SmallestOrderPrice FROM Orders
```

نتیجه اینگونه خواهد شد:

SmallestOrderPrice
100

تابع ()SUM

تابع ()SUM مجموع یک ستون عددی را برمی گرداند.

گرامر ()SUM در SQL

```
SELECT SUM(column_name) FROM table_name
```

مثال ()SUM در SQL

جدول "Orders" زیر را داریم:

O_Id	OrderDate	OrderPrice	Customer
1	2008/11/12	1000	Hansen
2	2008/10/23	1600	Nilsen
3	2008/09/02	700	Hansen
4	2008/09/03	300	Hansen
5	2008/08/30	2000	Jensen
6	2008/10/04	100	Nilsen

اکنون می خواهیم مجموع همه فیلدهای "OrderPrice" را پیدا کنیم.

از دستور SQL زیر استفاده می کنیم:

```
SELECT SUM(OrderPrice) AS OrderTotal FROM Orders
```

نتیجه اینگونه خواهد شد:

OrderTotal
5700

دستور Group By در SQL

توابع جمعی اغلب به یک دستور Group By اضافه شده نیاز دارند.

دستور Group By

دستور Group By با توابع جمعی، برای دسته بندی کردن نتیجه توسط یک یا چند ستون استفاده می شود.

گرامر Group By در SQL

```
SELECT column_name, aggregate_function(column_name)
FROM table_name
WHERE column_name operator value
GROUP BY column_name
```

مثال Group By در SQL

جدول "Orders" زیر را داریم:

O_Id	OrderDate	OrderPrice	Customer
1	2008/11/12	1000	Hansen
2	2008/10/23	1600	Nilsen
3	2008/09/02	700	Hansen
4	2008/09/03	300	Hansen
5	2008/08/30	2000	Jensen
6	2008/10/04	100	Nilsen

اکنون می خواهیم مجموع سفارشات هر مشتری را پیدا کنیم.

ما مجبور خواهیم شد از دستور Group By برای دسته بندی کردن مشتریان استفاده کنیم.

از دستور SQL زیر استفاده می کنیم:

```
SELECT Customer,SUM(OrderPrice) FROM Orders
GROUP BY Customer
```

نتیجه اینگونه خواهد شد:

Customer	SUM(OrderPrice)
Hansen	2000
Nilsen	1700
Jensen	2000

دلپسند است، مگه نه؟ ☺

بگذارید ببینیم؛ اگر از دستور Group By صرف نظر می کردیم چه اتفاقی می افتاد:

```
SELECT Customer,SUM(OrderPrice) FROM Orders
```

نتیجه اینگونه خواهد شد:

Customer	SUM(OrderPrice)
Hansen	5700
Nilsen	5700
Hansen	5700
Hansen	5700
Jensen	5700
Nilsen	5700

نتیجه بالا چیزی نیست که می خواستیم.

تشریح اینکه چرا دستور SELECT بالا نمی تواند استفاده شود: دستور SELECT بالا دو ستون مشخص شده دارد (Customer و SUM(OrderPrice)). هنگامیکه "Customer" ۶ مقدار برمی گرداند (یک مقدار برای هر در سطر جدول "Orders")، "SUM(OrderPrice)" یک مقدار واحد برمی گرداند (آن مجموع ستون "OrderPrice" است). بنابراین این نتیجه درست را به ما نخواهد داد. با این حال، دیدید که دستور Group By این مشکل را حل می کند.

Group By با بیش از یک ستون

ما می توانیم از دستور Group By برای بیش از یک ستون استفاده کنیم، مثل این:

```
SELECT Customer,OrderDate,SUM(OrderPrice) FROM Orders
GROUP BY Customer,OrderDate
```

عبارت Having

عبارت Having به SQL اضافه شد؛ چون کلمه کلیدی WHERE نمی تواند برای توابع جمعی استفاده شود.

گرامر عبارت Having

```
SELECT column_name, aggregate_function(column_name)
FROM table_name
WHERE column_name operator value
GROUP BY column_name
HAVING aggregate_function(column_name) operator value
```

مثال Having در SQL

جدول "Orders" زیر را داریم:

O_Id	OrderDate	OrderPrice	Customer
1	2008/11/12	1000	Hansen
2	2008/10/23	1600	Nilsen
3	2008/09/02	700	Hansen
4	2008/09/03	300	Hansen
5	2008/08/30	2000	Jensen
6	2008/10/04	100	Nilsen

اکنون می خواهیم هر مشتری را که جمع سفارش کمتر از ۲۰۰۰ دارد را پیدا کنیم.

از دستور SQL زیر استفاده می کنیم:

```
SELECT Customer,SUM(OrderPrice) FROM Orders
GROUP BY Customer
HAVING SUM(OrderPrice)<2000
```

نتیجه اینگونه خواهد شد:

Customer	SUM(OrderPrice)
Nilsen	1700

اکنون می خواهیم مشتریانی با نام "Hansen" یا "Jensen" که جمع سفارش بیشتر از ۱۵۰۰ دارند را پیدا کنیم.

یک عبارت WHERE معمولی به دستور SQL اضافه می کنیم:

```
SELECT Customer,SUM(OrderPrice) FROM Orders
WHERE Customer='Hansen' OR Customer='Jensen'
GROUP BY Customer
HAVING SUM(OrderPrice)>1500
```

نتیجه اینگونه خواهد شد:

Customer	SUM(OrderPrice)
Hansen	2000
Jensen	2000

تابع UCASE()

تابع UCASE() مقدار یک رشته را به حروف بزرگ تبدیل می کند.

گرامر UCASE()

```
SELECT UCASE(column_name) FROM table_name
```

گرامر برای SQL Server

```
SELECT UPPER(column_name) FROM table_name
```

مثال UCASE() در SQL

جدول "Persons" زیر را داریم:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

اکنون می خواهیم محتوای ستونهای "LastName" و "FirstName" بالا را انتخاب کنیم و ستون "LastName" را به حروف بزرگ تبدیل کنیم.

از دستور SQL زیر استفاده می کنیم:

```
SELECT UCASE(LastName) as LastName,FirstName FROM Persons
```

نتیجه اینگونه خواهد شد:

LastName	FirstName
HANSEN	Ola
SVENDSON	Tove
PETTERSEN	Kari

تابع LCASE()

تابع LCASE() مقدار یک رشته را حروف کوچک تبدیل می کند.

گرامر LCASE()

```
SELECT LCASE(column_name) FROM table_name
```

گرامر برای SQL Server

```
SELECT LOWER(column_name) FROM table_name
```

مثال تابع LCASE() در SQL

جدول "Persons" زیر را داریم:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

اکنون می خواهیم محتوای "LastName" و "FirstName" ستون های بالا را انتخاب کنیم و ستون "LastName" را به حروف کوچک تبدیل کنیم.

از دستور SQL زیر استفاده می کنیم:

```
SELECT LCASE(LastName) as LastName,FirstName FROM Persons
```

نتیجه اینگونه خواهد شد:

LastName	FirstName
hansen	Ola
svendson	Tove
pettersen	Kari

تابع MID()

تابع MID() برای استخراج کاراکترها از یک رشته متنی استفاده می شود.

گرامر تابع MID()

```
SELECT MID(column_name,start[,length]) FROM table_name
```

پارامتر	توضیحات
column_name	لازم. رشته مورد نظر برای استخراج کاراکترها
start	لازم. موقعیت شروع را مشخص کنید (شروع از ۱)
length	اختیاری. تعداد کاراکترها برای بازگشت. اگر صرفنظر شود، تابع MID() بقیه متن را برمی گرداند.

مثال MID() در SQL

جدول "Persons" زیر را داریم:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

اکنون می خواهیم چهار کاراکتر اول از ستون "City" بالا را استخراج کنیم.

از دستور SQL زیر استفاده می کنیم:

```
SELECT MID(City,1,4) as SmallCity FROM Persons
```

نتیجه اینگونه خواهد شد:

SmallCity
Sand
Sand
Stav

تابع LEN()

تابع LEN() طول مقدار یک فیلد متنی را برمی گرداند.

گرامر تابع LEN()

```
SELECT LEN(column_name) FROM table_name
```

مثال تابع LEN() در SQL

جدول "Persons" زیر را داریم:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

اکنون می خواهیم طول مقادیر در ستون "Address" بالا را انتخاب کنیم.

از دستور SQL زیر استفاده می کنیم:

```
SELECT LEN(Address) as LengthOfAddress FROM Persons
```

نتیجه اینگونه خواهد شد:

LengthOfAddress
12
9
9

تابع ROUDN()

تابع ROUDN() برای گرد کردن یک فیلد عددی به تعداد عدد اعشاری مشخص.

گرامر ROUDN() در SQL

```
SELECT ROUND(column_name,decimals) FROM table_name
```

پارامترها	توضیحات
column_name	لازم. فیلدی که باید گرد شود.
decimals	لازم. تعداد اعشار برای گرد شدن را تعیین می کند.

مثال ROUDN() در SQL

جدول "Products" زیر را داریم:

Prod_Id	ProductName	Unit	UnitPrice
1	Jarlsberg	1000 g	10.45
2	Mascarpone	1000 g	32.56
3	Gorgonzola	1000 g	15.67

اکنون می خواهیم نام محصولات و قیمت گرد شده به نزدیکترین عدد نمایش دهیم.

از دستور SQL زیر استفاده می کنیم:

```
SELECT ProductName, ROUND(UnitPrice,0) as UnitPrice FROM Products
```

نتیجه اینگونه خواهد شد:

ProductName	UnitPrice
Jarlsberg	10
Mascarpone	33
Gorgonzola	16

تابع NOW()

تابع NOW() تاریخ و زمان فعلی سیستم را برمی گرداند.

گرامر NOW() در SQL

```
SELECT NOW() FROM table_name
```

مثال NOW()

جدول "Products" زیر را داریم:

Prod_Id	ProductName	Unit	UnitPrice
1	Jarlsberg	1000 g	10.45
2	Mascarpone	1000 g	32.56
3	Gorgonzola	1000 g	15.67

اکنون می خواهیم محصولات و قیمت ها در تاریخ امروز را نمایش دهیم.

از دستور SQL زیر استفاده می کنیم:

```
SELECT ProductName, UnitPrice, Now() as PerDate FROM Products
```

نتیجه اینگونه خواهد شد:

ProductName	UnitPrice	PerDate
Jarlsberg	10.45	10/7/2008 11:25:02 AM
Mascarpone	32.56	10/7/2008 11:25:02 AM
Gorgonzola	15.67	10/7/2008 11:25:02 AM

تابع FORMAT()

تابع FORMAT() برای چگونگی نمایش فرمت یک فیلد استفاده می شود.

گرامر FORMAT() در SQL

```
SELECT FORMAT(column_name,format) FROM table_name
```

پارامتر	توضیحات
column_name	لازم. فیلدی که باید فرمت دهی شود.
format	لازم. فرمت را تعیین می کند.

مثال FORMAT() در SQL

جدول "Products" زیر را داریم:

Prod_Id	ProductName	Unit	UnitPrice
1	Jarlsberg	1000 g	10.45
2	Mascarpone	1000 g	32.56
3	Gorgonzola	1000 g	15.67

اکنون می خواهیم محصولات و قیمت در تاریخ امروز (با فرمت "YYYY-MM-DD") را نمایش دهیم.

از دستور SQL زیر استفاده می کنیم:

```
SELECT ProductName, UnitPrice, FORMAT(Now(),'YYYY-MM-DD') as PerDate FROM Products
```

نتیجه اینگونه خواهد شد:

ProductName	UnitPrice	PerDate
Jarlsberg	10.45	2008-10-07
Mascarpone	32.56	2008-10-07
Gorgonzola	15.67	2008-10-07

ضمیمه

مرجع سریع دستورات SQL

دستورات SQL	گرامر
AND / OR	SELECT column_name(s) FROM table_name WHERE condition AND|OR condition
ALTER TABLE	ALTER TABLE table_name ADD column_name datatype یا ALTER TABLE table_name DROP COLUMN column_name
AS (alias)	SELECT column_name AS column_alias FROM table_name یا SELECT column_name FROM table_name AS table_alias
BETWEEN	SELECT column_name(s) FROM table_name WHERE column_name BETWEEN value1 AND value2
CREATE DATABASE	CREATE DATABASE database_name
CREATE TABLE	CREATE TABLE table_name (column_name1 data_type, column_name2 data_type, column_name2 data_type, ...)
CREATE INDEX	CREATE INDEX index_name ON table_name (column_name) یا

	CREATE UNIQUE INDEX index_name ON table_name (column_name)
CREATE VIEW	CREATE VIEW view_name AS SELECT column_name(s) FROM table_name WHERE condition
DELETE	DELETE FROM table_name WHERE some_column=some_value یا DELETE FROM table_name (**توجه**: محتوای جدول را حذف می کند!) DELETE * FROM table_name (**توجه**: محتوای جدول را حذف می کند!)
DROP DATABASE	DROP DATABASE database_name
DROP INDEX	DROP INDEX table_name.index_name (SQL Server) DROP INDEX index_name ON table_name (MS Access) DROP INDEX index_name (DB2/Oracle) ALTER TABLE table_name DROP INDEX index_name (MySQL)
DROP TABLE	DROP TABLE table_name
GROUP BY	SELECT column_name, aggregate_function(column_name) FROM table_name WHERE column_name operator value GROUP BY column_name
HAVING	SELECT column_name, aggregate_function(column_name) FROM table_name WHERE column_name operator value GROUP BY column_name HAVING aggregate_function(column_name) operator value
IN	SELECT column_name(s) FROM table_name WHERE column_name IN (value1,value2,..)
INSERT INTO	INSERT INTO table_name VALUES (value1, value2, value3,....)

	یا INSERT INTO table_name (column1, column2, column3,...) VALUES (value1, value2, value3,....)
INNER JOIN	SELECT column_name(s) FROM table_name1 INNER JOIN table_name2 ON table_name1.column_name=table_name2.column_name
LEFT JOIN	SELECT column_name(s) FROM table_name1 LEFT JOIN table_name2 ON table_name1.column_name=table_name2.column_name
RIGHT JOIN	SELECT column_name(s) FROM table_name1 RIGHT JOIN table_name2 ON table_name1.column_name=table_name2.column_name
FULL JOIN	SELECT column_name(s) FROM table_name1 FULL JOIN table_name2 ON table_name1.column_name=table_name2.column_name
LIKE	SELECT column_name(s) FROM table_name WHERE column_name LIKE pattern
ORDER BY	SELECT column_name(s) FROM table_name ORDER BY column_name [ASC|DESC]
SELECT	SELECT column_name(s) FROM table_name
SELECT *	SELECT * FROM table_name
SELECT DISTINCT	SELECT DISTINCT column_name(s) FROM table_name
SELECT INTO	SELECT * INTO new_table_name [IN externaldatabase] FROM old_table_name یا

	SELECT column_name(s) INTO new_table_name [IN externaldatabase] FROM old_table_name
SELECT TOP	SELECT TOP number|percent column_name(s) FROM table_name
TRUNCATE TABLE	TRUNCATE TABLE table_name
UNION	SELECT column_name(s) FROM table_name1 UNION SELECT column_name(s) FROM table_name2
UNION ALL	SELECT column_name(s) FROM table_name1 UNION ALL SELECT column_name(s) FROM table_name2
UPDATE	UPDATE table_name SET column1=value, column2=value,... WHERE some_column=some_value
WHERE	SELECT column_name(s) FROM table_name WHERE column_name operator value

هاستینگ در SQL

اگر می خواهید وب سایت تان قادر به دخیره کردن و نمایش داده ها از یک پایگاه داده باشد، وب سرور تان باید به یک سیستم پایگاه داده دسترسی داشته باشد که از زبان SQL استفاده می کند.

اگر وب سرور شما توسط یک فراهم کننده سرویس اینترنت (ISP) میزبان خواهد شد؛ باید به طرح های هاستینگ SQL نگاه کنید.
رایج ترین پایگاه داده های هاستینگ MySQL، MS SQL Server و MS Access هستند.

شما می توانید پایگاه داده های SQL را در هر دو سیستم عامل ویندوز و لینوکس/یونیکس داشته باشید.

در زیر یک نمای کلی از اینکه کدام سیستم پایگاه داده روی کدام سیستم عامل اجرا می شود، آمده است:

My SQL Server

فقط بر روی سیستم عامل ویندوز اجرا می شود.

MySQL

بر روی هر دو سیستم عامل ویندوز و لینوکس/یونیکس اجرا می شود.

MS Access

فقط بر روی سیستم عامل ویندوز اجرا می شود.

شما SQL را آموختید، حالا چکاری باید انجام دهید؟

خلاصه SQL

این دوره SQL به شما آموخت که SQL زبان استاندارد کامپیوتر برای دسترسی و دستکاری سیستم های پایگاه داده است.

شما آموختید که چگونه با SQL در یک پایگاه داده پرس و جو را اجرا کنید، داده را بازیابی کنید، رکوردهای جدید درج کنید، رکوردها را حذف و بروز رسانی کنید.

شما همچنین آموختید که چگونه پایگاه های داده، جداول و فهرست ها را با SQL بسازید، و چگونه آنها را حذف کنید.

شما مهمترین توابع جمعی را در SQL آموختید.

اکنون می دانید که SQL زبان استانداردی است که با همه سیستم های پایگاه داده معروف مثل MS SQL Server، IBM DB2، Oracle، MySQL و MS Access بخوبی کار می کند.