mBÜTÜN KAYITLARI CAGIRMA KOMUTU

SELECT * FROM Customers;

SQL komutlari

SELECT - bir veritabanından verileri çıkarır

UPDATE - bir veritabanındaki verileri günceller

DELETE - bir veritabanından verileri siler

INSERT INTO - yeni verileri bir veritabanına ekler

CREATE DATABASE - yeni bir veritabanı oluşturur

ALTER DATABASE - bir veritabanını değiştirir

CREATE TABLE - yeni bir tablo oluşturur

ALTER TABLE - bir tabloyu değiştirir

DROP TABLE - bir tabloyu siler

CREATE INDEX - bir dizin oluşturur (arama tuşu)

DROP INDEX - bir dizini siler

Sütun SEÇİN Örnek

Aşağıdaki SQL ifadesi, "Müşteriler" tablosundan "MüşteriAdı" ve "Şehir" sütunlarını seçer:

SELECT CustomerName,City FROM Customers;

SQL SELECT DISTINCT İfadesi

SELECT DISTINCT deyimi yalnızca farklı (farklı) değerleri döndürmek için kullanılır.

Bir tablonun içinde, bir sütun genellikle birçok yinelenen değer içerir; ve bazen yalnızca farklı (farklı) değerleri listelemek istersiniz.

SELECT DISTINCT Country FROM Customers;

Not: Yukarıdaki örnek Firefox'ta çalışmayacaktır! COUNT (DISTINCT *sütun_adı*) Microsoft Access veritabanlarında desteklenmediğinden. Firefox, örneklerimizde Microsoft Access kullanıyor.

SELECT Count(*) AS DistinctCountries
FROM (SELECT DISTINCT Country FROM Customers);

WHERE yan tümcesi kayıtları filtrelemek için kullanılır.

WHERE yan tümcesi, yalnızca belirli bir koşulu karşılayan kayıtları ayıklamak için kullanılır.

SELECT * FROM Customers

WHERE Country='Mexico';

WHERE Maddesindeki Operatörler

Aşağıdaki operatörler WHERE yan tümcesinde kullanılabilir:

SELECT * FROM Products WHERE Price = 18; price 18 olanlari alir

SELECT * FROM Products WHERE Price > 30; price 30 dan büyük olanlari

SELECT * FROM Products WHERE Price < 30;

SELECT * FROM Products WHERE Price >= 30;

SELECT * FROM Products WHERE Price <= 30;

SELECT * FROM Products WHERE Price BETWEEN 50 AND 60; 50 ile 60 arasi

SELECT * FROM Customers WHERE City LIKE 's%'; s ile baslayan sehirler

SELECT * FROM Customers WHERE City IN ('Paris','London'); bu iki sehrin bilgilerini rufen yapiyor

SELECT * FROM Customers WHERE Country='Germany' AND City='Berlin';

SELECT * FROM Customers WHERE City='Berlin' OR City='München';

SELECT * FROM Customers WHERE NOT Country='Germany';

SELECT * FROM Customers WHERE Country='Germany' AND (City='Berlin' OR City='München');

SELECT * FROM Customers ORDER BY Country; ülkeleri a dan z ye siralar

SELECT * FROM Customers ORDER BY Country DESC; ülkeleri v- a ya dogru siralar

SELECT * FROM Customers ORDER BY Country, CustomerName; Aşağıdaki SQL deyimi, "Ülke" ve "MüşteriAdı" sütununa göre sıralanmış "Müşteriler" tablosundaki tüm müşterileri seçer. Bu, Ülkeye göre sipariş verdiği anlamına gelir, ancak bazı satırlar aynı Ülkeye sahipse, bunları MüşteriAdı'na göre sıralar:

SELECT * FROM Customers ORDER BY Country ASC, CustomerName DESC; Aşağıdaki SQL deyimi, "Müşteriler" tablosundan tüm müşterileri seçer, "Ülke" ye göre artan ve "MüşteriAdı" sütununa göre azalan sıralanır:

INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');

yukarıdaki SQL ifadesi "Müşteriler" tablosuna yeni bir kayıt ekler:

INSERT INTO Customers (CustomerName, City, Country)
VALUES ('Cardinal', 'Stavanger', 'Norway'); eklemedigimiz kisimlar null olur

SELECT CustomerName, ContactName, Address
FROM Customers
WHERE Address IS NOT NULL;

SELECT CustomerName, ContactName, Address
FROM Customers
WHERE Address IS NULL;

UPDATE Customers SET ContactName='Alfred Schmidt', City='Frankfurt'

WHERE CustomerID=1; BERLIN YERINE FRANKFURT YAZAR GÜNCELLEME KODU TABIKI NAME DE DEGISIYOR.

Bir veriyi silme kodu

DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';

EN FAZLA CAGIRABILECEGIN SAYIYI GÖSTERIR

SELECT * FROM Customers LIMIT 3;

TABLODAKI ILK YÜZDE ELIYI SECER

SELECT TOP 50 PERCENT * FROM Customers;

Tablodaki ilk 3 alman secer

SELECT TOP 3 * FROM Customers WHERE Country='Germany';

ELECT * FROM Customers WHERE Country='Germany' LIMIT 3;(MYSQL) ICIN

EN KÜCÜK VE EN BÜYÜK DEGERLERI BULMA KODU

SELECT MIN(Price) AS SmallestPrice FROM Products;

SELECT MAX(Price) AS LargestPrice FROM Products;

SELECT SUM(Quantity) FROM OrderDetails; „siparis ayrintilari“ tablosundaki „miktar“ alanlarinin toplamini verir

SELECT AVG(Price) FROM Products;tpm ürünlerin ortalama degeri

SELECT COUNT(ProductID) FROM Products; ürün sayisini verir

SELECT * FROM Customers WHERE CustomerName LIKE 'a%'; a ile basla

SELECT * FROM Customers WHERE CustomerName LIKE 'a%'; a ile bitenl

SELECT * FROM Customers WHERE CustomerName LIKE '%or%';

SELECT * FROM Customers WHERE CustomerName LIKE '_r%';2. Harf r

SELECT * FROM Customers WHERE CustomerName LIKE 'a__%';

SELECT * FROM Customers WHERE ContactName LIKE 'a%o'; a ile basla o ile biten

SELECT * FROM Customers WHERE CustomerName NOT LIKE 'a%'; a ile baslamayanlar

SQL LIKE Operatörü

LIKE operatörü, bir sütundaki belirli bir modeli aramak için bir WHERE yan tümcesinde kullanılır.

Genellikle LIKE işleci ile birlikte kullanılan iki joker karakter vardır:

% - Yüzde işareti sıfır, bir veya birden çok karakteri temsil eder

_ - Alt çizgi tek bir karakteri temsil eder

Aşağıdaki SQL ifadesi, "es" kalıbını içeren bir Şehir olan tüm müşterileri seçer:

Misal

```
SELECT * FROM Customers
WHERE City LIKE '%es%';
```

Aşağıdaki SQL deyimi, "L" ile başlayan, ardından herhangi bir karakter, ardından "n", ardından herhangi bir karakter ve ardından "on" ile başlayan bir Şehre sahip tüm müşterileri seçer:

Misal

```
SELECT * FROM Customers
WHERE City LIKE 'L_n_on';
```

[Charlist] Joker Karakterini kullanma

Aşağıdaki SQL ifadesi, "b", "s" veya "p" ile başlayan Şehri olan tüm müşterileri seçer:

Misal

```
SELECT * FROM Customers
WHERE City LIKE '[bsp]%';
```

Aşağıdaki SQL ifadesi, "a", "b" veya "c" ile başlayan Şehri olan tüm müşterileri seçer:

Misal

```
SELECT * FROM Customers
WHERE City LIKE '[a-c]%';
```

[! Charlist] Joker Karakterini kullanma

Aşağıdaki iki SQL deyimi, Şehri "b", "s" veya "p" ile başlamayan tüm müşterileri seçer:

Misal

```
SELECT * FROM Customers
WHERE City LIKE '[!bsp]%';
```

IN Operatör Örnekleri

Aşağıdaki SQL ifadesi, "Almanya", "Fransa" veya "İngiltere" de bulunan tüm müşterileri seçer:

Misal

```
SELECT * FROM Customers
WHERE Country IN ('Germany', 'France', 'UK');
```

Aşağıdaki SQL ifadesi, "Almanya", "Fransa" veya "İngiltere" konumunda OLMAYAN tüm müşterileri seçer:

Misal

```
SELECT * FROM Customers
WHERE Country NOT IN ('Germany', 'France', 'UK');
```

Aşağıdaki SQL ifadesi, tedarikçilerle aynı ülkelerden olan tüm müşterileri seçer:

Misal

```
SELECT * FROM Customers
WHERE Country IN (SELECT Country FROM Suppliers);
```

BETWEEN Örnek

Aşağıdaki SQL ifadesi, fiyatı 10 ile 20 ARASINDA olan tüm ürünleri seçer:

Misal

```
SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20;
```

ÖRNEK ARASINDA DEĞİL

Önceki örneğin aralığının dışındaki ürünleri görüntülemek için ARASINDA DEĞİL:

Misal

```
SELECT * FROM Products
WHERE Price NOT BETWEEN 10 AND 20;
```

IN Örneği ARASINDA

Aşağıdaki SQL deyimi, fiyatı 10 ile 20 ARASINDA olan tüm ürünleri seçer. Ayrıca; Kategori Kimliği 1,2 veya 3 olan ürünleri gösterme:

Misal

```
SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20
AND CategoryID NOT IN (1,2,3);
```

BETWEEN Metin Değerleri Örneği

Aşağıdaki SQL deyimi, Carnarvon Tigers ve Mozzarella di Giovanni ARASINDA ÜrünAdı olan tüm ürünleri seçer:

Misal

```
SELECT * FROM Products
WHERE ProductName BETWEEN 'Carnarvon Tigers' AND 'Mozzarella di Giovanni'
ORDER BY ProductName;
```

Metin Değerleri Örneği ARASINDA DEĞİL

Aşağıdaki SQL ifadesi, Carnarvon Tigers ve Mozzarella di Giovanni ARASINDA OLMAYAN ÜrünAdı olan tüm ürünleri seçer:

Misal

```
SELECT * FROM Products
WHERE ProductName NOT BETWEEN 'Carnarvon Tigers' AND 'Mozzarella di Giovanni'
ORDER BY ProductName;
```

TARİHLER ARASI Örnek

Aşağıdaki SQL deyimi, '01 -Temmuz-1996 've '31 -Temmuz-1996' arasında bir OrderDate olan tüm siparişleri seçer:

Misal

```
SELECT * FROM Orders
WHERE OrderDate BETWEEN #01/07/1996# AND #31/07/1996#;
```

VEYA:

Misal

```
SELECT * FROM Orders
WHERE OrderDate BETWEEN '1996-07-01' AND '1996-07-31';
```

Sütun Örneklerinin Takma Adı

Aşağıdaki SQL ifadesi, biri CustomerID sütunu ve diğeri CustomerName sütunu için olmak üzere iki takma ad oluşturur:

Misal

```
SELECT CustomerID AS ID, CustomerName AS Customer
FROM Customers;
```

Aşağıdaki SQL deyimi, biri CustomerName sütunu ve diğeri ContactName sütunu için olmak üzere iki takma ad oluşturur. **Not:** Diğer ad boşluk içeriyorsa çift tırnak işareti veya köşeli parantez gerektirir:

Misal

```
SELECT CustomerName AS Customer, ContactName AS [Contact Person]
FROM Customers;
```

Aşağıdaki SQL ifadesi, dört sütunu (Adres, Posta Kodu, Şehir ve Ülke) birleştiren "Adres" adlı bir takma ad oluşturur:

Misal

```
SELECT CustomerName, Address + ', ' + PostalCode + ' ' + City + ', ' + Country AS Address
FROM Customers
```

Not: Yukarıdaki SQL ifadesinin MySQL'de çalışması için aşağıdakileri kullanın:

```
SELECT CustomerName, CONCAT(Address,', ',PostalCode,', ',City,', ',Country) AS Address
FROM Customers;
```

"Siparişler" tablosundaki "Müşteri Kimliği" sütununun "Müşteriler" tablosundaki "Müşteri Kimliği" ne başvurduğuna dikkat edin. Yukarıdaki iki tablo arasındaki ilişki "Müşteri Kimliği" sütunudur.

Ardından, her iki tabloda da eşleşen değerlere sahip kayıtları seçen aşağıdaki SQL ifadesini (INNER JOIN içeren) oluşturabiliriz:

Misal

```
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM Orders
INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;
```

Beispiel für SQL LEFT JOIN

Die folgende SQL-Anweisung wählt alle Kunden und alle Bestellungen aus, die sie möglicherweise haben:

Beispiel

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
ORDER BY Customers.CustomerName;
```

Beispiel für SQL FULL OUTER JOIN

Die folgende SQL-Anweisung wählt alle Kunden und alle Bestellungen aus:

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;
```

Eine Auswahl aus der Ergebnismenge kann folgendermaßen aussehen:

Beispiel für SQL Self JOIN

Die folgende SQL-Anweisung stimmt mit mit Kunden überein, die aus derselben Stadt stammen:

Beispiel

```
SELECT A.CustomerName AS CustomerName1,
B.CustomerName AS CustomerName2, A.City
FROM Customers A, Customers B
WHERE A.CustomerID <> B.CustomerID
AND A.City = B.City
ORDER BY A.City;
```

SQL BİRLİĞİ Örneği

Aşağıdaki SQL ifadesi, hem "Müşteriler" hem de "Tedarikçiler" tablosundaki şehirleri (yalnızca farklı değerler) döndürür:

Misal

```
SELECT City FROM Customers
UNION
SELECT City FROM Suppliers
ORDER BY City;
```

Nicht: Bazı müşteriler veya tedarikçiler aynı şehre sahipse, ihr şehir yalnızca bir kez listelenir, çünkü UNION yalnızca farklı değerler seçer. Yinelenen değerleri de seçmek için UNION ALL kullanın!

SQL BİRLİĞİ TÜM Örneği

Die folgende SQL-Anweisung gibt die Städte (auch doppelte Werte) sowohl aus der Tabelle "Kunden" als auch aus der Tabelle "Lieferanten" zurück:

Beispiel

```
SELECT City FROM Customers
UNION ALL
SELECT City FROM Suppliers
ORDER BY City;
```

Die folgende SQL-Anweisung listet die Anzahl der Kunden in jedem Land auf:

Beispiel

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country;
```

Die folgende SQL-Anweisung listet die Anzahl der Kunden in jedem Land auf, sortiert nach hoch bis niedrig:

Beispiel

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
ORDER BY COUNT(CustomerID) DESC;
```

SQL EXISTS-Beispiele

Die folgende SQL-Anweisung gibt TRUE zurück und listet die Lieferanten mit einem Produktpreis von weniger als 20 auf:

Beispiel

```
SELECT SupplierName
FROM Suppliers
WHERE EXISTS (SELECT ProductName FROM Products WHERE Products.SupplierID =
Suppliers.supplierID AND Price < 20);
```

Die folgende SQL-Anweisung gibt TRUE zurück und listet die Lieferanten mit einem Produktpreis von 22 auf:

Beispiel

```
SELECT SupplierName
FROM Suppliers
WHERE EXISTS (SELECT ProductName FROM Products WHERE Products.SupplierID =
Suppliers.supplierID AND Price = 22);
```

Beispiel

```
SELECT ProductName
FROM Products
WHERE ProductID = ANY (SELECT ProductID FROM OrderDetails WHERE Quantity
= 10);
```

Die folgende SQL-Anweisung kopiert nur die deutschen Kunden in eine neue Tabelle:

```
SELECT * INTO CustomersGermany
FROM Customers
WHERE Country = 'Germany';
```

Die folgende SQL-Anweisung kopiert Daten aus mehr als einer Tabelle in eine neue Tabelle:

```
SELECT Customers.CustomerName, Orders.OrderID
INTO CustomersOrderBackup2017
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID;
```

İpucu: Mit SELECT INTO können Sie auch eine neue, leere Tabelle mit dem Schema einer anderen erstellen. Fügen Sie einfach eine WHERE-Klausel hinzu, die bewirkt, dass die Abfrage keine Daten zurückgibt:

```
SELECT * INTO newtable
FROM oldtable
WHERE 1 = 0;
```

INSERT INTO SELECT Syntax

Kopieren Sie alle Spalten einer Tabelle in einer anderen Tabelle:

```
INSERT INTO table2
SELECT * FROM table1
WHERE condition;
```

Kopieren Sie nur einige Spalten aus einer Tabelle in einer anderen Tabelle:

```
INSERT INTO table2 (column1, column2, column3, ...)
SELECT column1, column2, column3, ...
FROM table1
WHERE condition;
```

Beispiel

```
INSERT INTO Customers (CustomerName, ContactName, Address, City,
PostalCode, Country)
SELECT SupplierName, ContactName, Address, City,
PostalCode, Country FROM Suppliers;
```

Die gemeinsamen SQL-Anweisungen kopiert nur die deutschen Rechte in "Kunden":

Beispiel

```
INSERT INTO Customers (CustomerName, City, Country)
SELECT SupplierName, City, Country FROM Suppliers
WHERE Country='Germany';
```

SQL CASE-Beispiele

Das heißt SQL durch durchgehende Bedingungen und gibt einen Wert zurück, wenn die erste Bedingung ist:

Beispiel

```
SELECT OrderID, Quantity,
CASE
    WHEN Quantity > 30 THEN 'The quantity is greater than 30'
    WHEN Quantity = 30 THEN 'The quantity is 30'
    ELSE 'The quantity is under 30'
END AS QuantityText
FROM OrderDetails;
```

Cikti olarak eger 30 dan kücük ise karsisina the quantity is under 30, eher 30 a esit ise the quantity is 30 eher 30 dan büyük ise the quantity is greater than 30 yazacak.

Die verschiedenen SQL ordnet die Kunden nach Stadt. Wenn die Stadt keine NULL ist, bestellen Sie nach Land:

Beispiel

```
SELECT CustomerName, City, Country
FROM Customers
ORDER BY
(CASE
    WHEN City IS NULL THEN Country
    ELSE City
END);
```

War es eine gespechte Prozedur?

Eine gespeicherte Prozedur ist ein vorbereiteter SQL-Code, den Sie erhalten können, damit der Code immer wieder verwendet werden kann.

Wenn Sie auch ein SQL-Wahlrecht haben, die Sie immer wieder schreiben, speichern Sie sie als gespeicherte Prozedur und rufen Sie sie einfach auf, um sie verwendet.

Sie können Parameter auch eine gespepekte Prozedurrechte, die diepepezierten Prozedurrechte auf den gleichenen Parameterwerten können.

Syntax für gespeicherte Prozeduren

```
CREATE PROCEDURE procedure_name
AS
sql_statement
GO;
```

Unterschiedliche Sie eine gespechte Prozedur aus

```
EXEC procedure_name;
```

Demo-Datenbank

Unten finden Sie eine Auswahl aus der Tabelle "Kunden" in der Nordwind-Beispieldatenbank:

Beispiel für eine gespechte Prozedur

Die verschiedenen SQL-Anweisungen, die eine gespeicherte Prozedur mit dem Namen "SelectAllCustomers", die alle Datensätze aus der Tabelle "Kunden" auswählt:

Beispiel

```
CREATE PROCEDURE SelectAllCustomers
AS
SELECT * FROM Customers
GO;
```

Unterschiedliche Sie sterben die oben gespeckte Prozedur wie folgt aus:

Beispiel

```
EXEC SelectAllCustomers;
```

Gespekte Prozedur mit einem Parameter

Die folgenden SQL-Richtlinien werden eine gespeicherte Prozedur, mit der Kunden aus einer eigenen Stadt aus der Tabelle "Kunden" werden:

Beispiel

```
CREATE PROCEDURE SelectAllCustomers @City nvarchar(30)
AS
SELECT * FROM Customers WHERE City = @City
GO;
```

Unterschiedliche Sie sterben die oben gespeckte Prozedur wie folgt aus:

Beispiel

```
EXEC SelectAllCustomers @City = 'London';
```

Gespekte Prozedur mit genannten Parametern

Das Einrichten wird Parameter ist sehr einfach. Hören Sie einfach jeden Parameter und den Datentyp wie unbewusst durch ein Komma auf.

Die folgenden SQL-Richtlinien werden eine gespeicherte Prozedur, mit der Kunden aus einer eigenen Stadt mit einer eigenen Postleitzahl aus der Tabelle "Kunden" werden:

Beispiel

```
CREATE PROCEDURE SelectAllCustomers @City nvarchar(30), @PostalCode
nvarchar(10)
AS
SELECT * FROM Customers WHERE City = @City AND PostalCode = @PostalCode
GO;
```

Unterschiedliche Sie sterben die oben gespeckte Prozedur wie folgt aus:

Beispiel

```
EXEC SelectAllCustomers @City = 'London', @PostalCode = 'WA1 1DP';
```

SQL CREATE DATABASE İfadesi

CREATE DATABASE deyimi, yeni bir SQL veritabanı oluşturmak için kullanılır.

Sözdizimi

```
CREATE DATABASE databasename;
```

SQL DROP DATABASE İfadesi

DROP DATABASE deyimi, mevcut bir SQL veritabanını bırakmak için kullanılır.

Sözdizimi

```
DROP DATABASE databasename;
```

YEDEK VERİTABANI Örneği

Aşağıdaki SQL ifadesi, mevcut "testDB" veritabanının D diskine tam bir yedeğini oluşturur:

Misal

```
BACKUP DATABASE testDB
TO DISK = 'D:\backups\testDB.bak';
```

DİFERANSİYEL Örnek İLE YEDEKLEME

Aşağıdaki SQL deyimi, "testDB" veritabanının farklı bir yedeğini oluşturur:

Misal

```
BACKUP DATABASE testDB
TO DISK = 'D:\backups\testDB.bak'
WITH DIFFERENTIAL;
```

SQL CREATE TABLE İfadesi

CREATE TABLE deyimi, bir veritabanında yeni bir tablo oluşturmak için kullanılır.

Sözdizimi

```
CREATE TABLE table_name (
   column1 datatype,
   column2 datatype,
   column3 datatype,
  ....
```

SQL CREATE TABLE Örneği

Aşağıdaki örnek, beş sütun içeren "Kişiler" adlı bir tablo oluşturur: Kişi Kimliği, Soyadı, Ad, Adres ve Şehir:

Misal

```
CREATE TABLE Persons (
   PersonID int,
   LastName varchar(255),
   FirstName varchar(255),
```

```
    Address varchar(255),
    City varchar(255)
```

SQL TRUNCATE TABLOSU

TRUNCATE TABLE deyimi, bir tablo içindeki verileri silmek için kullanılır, ancak tablonun kendisini değil.

Sözdizimi

```
TRUNCATE TABLE table_name;
```

Aşağıdaki SQL, "Müşteriler" tablosuna bir "E-posta" sütunu ekler:

Misal

```
ALTER TABLE Customers
ADD Email varchar(255);
```

ALTER TABLE - DROP COLUMN

Bir tablodaki bir sütunu silmek için aşağıdaki sözdizimini kullanın (bazı veritabanı sistemlerinin bir sütunun silinmesine izin vermediğine dikkat edin):

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

Aşağıdaki SQL, "E-posta" sütununu "Müşteriler" tablosundan siler:

Misal

```
ALTER TABLE Customers
DROP COLUMN Email;
```

DROP COLUMN Örneği

Daha sonra, "Kişiler" tablosundaki "DateOfBirth" adlı sütunu silmek istiyoruz.

Aşağıdaki SQL ifadesini kullanıyoruz:

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

CREATE TABLE üzerinde SQL PRIMARY KEY

Aşağıdaki SQL, "Persons" tablosu oluşturulduğunda "ID" sütununda bir PRIMARY KEY oluşturur:

MySQL:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (ID)
```

ALTER TABLE üzerinde SQL PRIMARY KEY

Tablo zaten oluşturulduğunda "ID" sütununda bir PRIMARY KEY kısıtlaması oluşturmak için aşağıdaki SQL'i kullanın:

MySQL / SQL Sunucusu / Oracle / MS Erişimi:

```
ALTER TABLE Persons
ADD PRIMARY KEY (ID);
```

BİRİNCİL ANAHTAR KISITLAMASINI BIRAK

Bir PRIMARY KEY kısıtlamasını kaldırmak için aşağıdaki SQL'i kullanın:

MySQL:

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

SQL Sunucusu / Oracle / MS Erişimi:

```
ALTER TABLE Persons
DROP CONSTRAINT PK_Person;
```

MySQL için Sözdizimi

Aşağıdaki SQL ifadesi, "Personid" sütununu "Kişiler" tablosundaki bir otomatik artış birincil anahtar alanı olarak tanımlar:

```
CREATE TABLE Persons (
    Personid int NOT NULL AUTO_INCREMENT,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (Personid)
```

SQL Tarih Veri Türleri

MySQL , veritabanında bir tarih veya tarih / saat değerini depolamak için aşağıdaki veri türleriyle birlikte gelir:

TARİH - YYYY-AA-GG biçimi

DATETIME - biçim: YYYY-AA-GG SS: MI: SS

TIMESTAMP - biçim: YYYY-AA-GG SS: MI: SS

YEAR - YYYY veya YY biçimlendirin

SQL Server , veritabanında bir tarih veya tarih / saat değerini depolamak için aşağıdaki veri türleriyle birlikte gelir:

TARİH - YYYY-AA-GG biçimi

DATETIME - biçim: YYYY-AA-GG SS: MI: SS

SMALLDATETIME - biçim: YYYY-AA-GG SS: MI: SS

TIMESTAMP - biçim: benzersiz bir sayı

Not: Veritabanınızda yeni bir tablo oluşturduğunuzda bir sütun için tarih türleri seçilir!

MySQL Veri Türleri (Sürüm 8.0)

MySQL'de üç ana veri türü vardır: dizi, sayısal ve tarih ve saat.

Dize veri türleri:

Data type	Description
CHAR(size)	A FIXED length string (can contain letters, numbers, and sp characters - can be from 0 to 255. Default is 1
VARCHAR(size)	A VARIABLE length string (can contain letters, numbers, an length in characters - can be from 0 to 65535
BINARY(size)	Equal to CHAR(), but stores binary byte strings. The *size* pa
VARBINARY(size)	Equal to VARCHAR(), but stores binary byte strings. The *siz*
TINYBLOB	For BLOBs (Binary Large OBjects). Max length: 255 bytes
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT(size)	Holds a string with a maximum length of 65,535 bytes
BLOB(size)	For BLOBs (Binary Large OBjects). Holds up to 65,535 bytes
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 chara
MEDIUMBLOB	For BLOBs (Binary Large OBjects). Holds up to 16,777,215
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 ch
LONGBLOB	For BLOBs (Binary Large OBjects). Holds up to 4,294,967,29
ENUM(val1, val2, val3, ...)	A string object that can have only one value, chosen from a value is inserted that is not in the list, a blank value will b
SET(val1, val2, val3, ...)	A string object that can have 0 or more values, chosen fro

Sayısal veri türleri:

Data type	Description

BIT(*size*)	A bit-value type. The number of bits per value is specified value for *size* is 1.
TINYINT(*size*)	A very small integer. Signed range is from -128 to 127. Uns display width (which is 255)
BOOL	Zero is considered as false, nonzero values are considered
BOOLEAN	Equal to BOOL
SMALLINT(*size*)	A small integer. Signed range is from -32768 to 32767. Uns maximum display width (which is 255)
MEDIUMINT(*size*)	A medium integer. Signed range is from -8388608 to 8388 the maximum display width (which is 255)
INT(*size*)	A medium integer. Signed range is from -2147483648 to 2 specifies the maximum display width (which is 255)
INTEGER(*size*)	Equal to INT(size)
BIGINT(*size*)	A large integer. Signed range is from -9223372036854775 18446744073709551615. The *size* parameter specifies the
FLOAT(*size*, *d*)	A floating point number. The total number of digits is spec the *d* parameter. This syntax is deprecated in MySQL 8.0.1
FLOAT(*p*)	A floating point number. MySQL uses the *p* value to deter from 0 to 24, the data type becomes FLOAT(). If *p* is from 2
DOUBLE(*size*, *d*)	A normal-size floating point number. The total number of specified in the *d* parameter
DOUBLE PRECISION(*size*, *d*)	
DECIMAL(*size*, *d*)	An exact fixed-point number. The total number of digits is in the *d* parameter. The maximum number for *size* is 65. T default value for *d* is 0.
DEC(*size*, *d*)	Equal to DECIMAL(size,d)

Not: Tüm sayısal veri türlerinin fazladan bir seçeneği olabilir: İMZALANMAMIŞ veya SIFIR DOLUM. İmzalanmamış seçeneğini eklerseniz, MySQL sütun için negatif değerlere izin vermez. SIFIRFILL seçeneğini eklerseniz, MySQL otomatik olarak sütuna UNSIGNED özelliğini de ekler.

Tarih ve Saat veri türleri:

Data type	Description
DATE	A date. Format: YYYY-MM-DD. The supported range is fro
DATETIME(*fsp*)	A date and time combination. Format: YYYY-MM-DD hh:m 23:59:59'. Adding DEFAULT and ON UPDATE in the column and time
TIMESTAMP(*fsp*)	A timestamp. TIMESTAMP values are stored as the numbe YYYY-MM-DD hh:mm:ss. The supported range is from '197 initialization and updating to the current date and time ca CURRENT_TIMESTAMP in the column definition
TIME(*fsp*)	A time. Format: hh:mm:ss. The supported range is from '-8
YEAR	A year in four-digit format. Values allowed in four-digit for MySQL 8.0 does not support year in two-digit format.