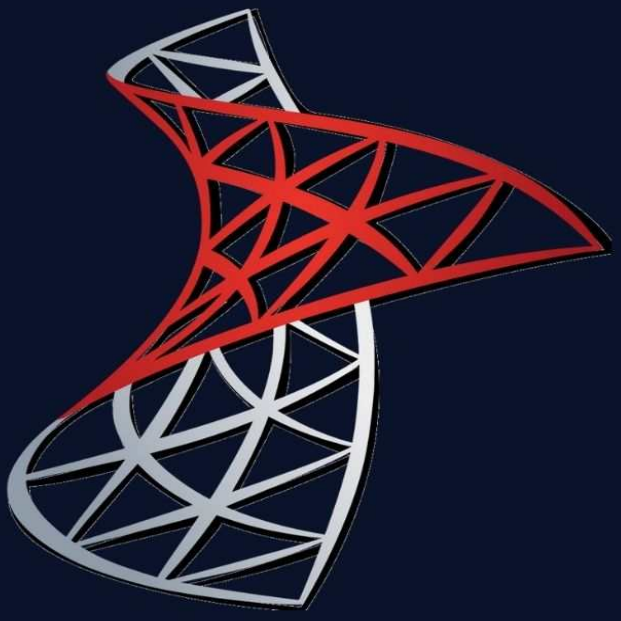
NLN

SQL

SQL GRUNDLAGEN LERNEN



2021

ALEXANDER ARONOWITZ

SQL ist eine relationale Datenbanksprache mit wenigen Befehlen und einer erkennbaren Struktur, die es Ihnen auf einfache Weise ermöglicht, auf Daten aus SQL Datenbanken, aber auch Access zuzugreifen, sie einzugeben und zu verändern. Mit Hilfe von SQL Kommandos können Sie Datenbankstrukturen definieren, aktualisieren und sogar Sicherheitsaspekte überwachen. Je nach Datenbank Hersteller wie Microsoft, Oracle oder MySQL gibt es nur geringe Unterschiede in der Sprachsyntax.

Sie suchen einen Einstieg in Microsoft SQL Server? Dieser SQL Kurs bietet eine kompakte Einführung in die Datenbankabfrage mit Hilfe von Transact-SQL. Angefangen mit einfachen Abfragen werden auch Themen wie Gruppierungen, Modifikationen, Transaktionen und heterogene Datenquellen in diesem SQL Training behandelt. Nach diesem SQL Seminar haben Sie fundierte Kenntnisse der Sprache SQL zur Datendefinition und performanten Datenabfrage.

Kapitel 1

Über SQL

SQL ist eine strukturierte Abfragesprache, die Sprache, die zum Ausführen von Vorgängen in Datenbanken verwendet wird, einschließlich Hinzufügen, Aktualisieren oder Löschen von Daten aus der Datenbank, oder zum Ändern der Datenbank selbst.

SQL-Architektur:

SQL-Anweisungen werden für Tabellen in Datenbanken ausgeführt, und diese Tabellen bestehen aus Spalten und Datensätzen. Anweisungen bestehen aus reservierten Wörtern, Variablen und Transaktionen (usw.), die bestimmen, welchen Vorgang wir ausführen möchten.

SQL deklariert das Endergebnis des Prozesses, den wir durchführen möchten, nicht die Details, wie dieser Prozess ausgeführt wird, da die Datenbankmodule für diese Details verantwortlich sind.

Das folgende Beispiel zeigt die Verwendung der SELECT-Abfrage, um zwei Spalten aus einer Tabelle mit einem alternativen Namen (AS) für eine Spalte auszuwählen, um anzugeben, welche Datensätze eine WHERE-Bedingung angewendet werden, und um sie nach ORDER BY zu sortieren:

```
SELECT name, age*365 AS AgeDays
FROM students
WHERE age > 10
ORDER BY name;
```

Dieses Verzeichnis wird entworfen, indem jedem reservierten Wort oder Wort oder Parameter in SQL eine Seite zugewiesen wird. Der Name dieser Seite ist in Kleinbuchstaben. Wenn der Ausdruck aus mehr als einem Wort besteht, das durch einen Unterstrich getrennt ist; zum Beispiel wird die SELECT-Abfrageseite auf dem SQL / select Link, Sie sind in SQL / order_by und so weiter vorhanden ...

Seine Geschichte:

Im Juni 1970 veröffentlichte der britische Wissenschaftler Edgar Cod eine wissenschaftliche Arbeit mit dem Titel "Interconnected Model of Data for Large Shared Data Banks", in der er ein Modell für die Erstellung und Verwaltung von Datenbanken vorstellte, das als Relational Database Model bekannt ist.

SELECT-Abfrage:

Eine SELECT-Abfrage wird verwendet, um Datensätze aus Tabellen abzurufen, die in der Datenbank gespeichert sind, das Ergebnis der Abfrage kann ein oder mehrere Datensätze und eine oder mehrere Spalten sein.

Allgemeine Struktur

SELECT [DISTINCT | TOP] select_list

[FROM table_source]

[WHERE search_condition]

[GROUP BY group_by_expression]

[HAVING search_condition]

[ORDER BY order_expression [ASC | DESC]];

Einige Transaktionen wie UNION, EXCEPT und INTERSECT können verwendet werden, um die Ergebnisse von Abfragen zusammenzuführen oder zu vergleichen. Die Reihenfolge der Schlüsselwörter, die in der vorherigen Formulierung verwendet wurden, sollte berücksichtigt werden, wenn sich viele in einer einzigen Abfrage befinden.

Beispiele

Im Folgenden finden Sie verschiedene Beispiele für die Verwendung der SELECT-Abfrage mit unterschiedlichen Transaktionen, die Seiten dieser Transaktionen können für weitere Informationen und Beispiele angezeigt werden.

Wählen Sie alle Spalten einer Tabelle aus.

Das *-Symbol wird verwendet, um alle Spalten aus der Tabelle abzubekommen (dies ist einer der alternativen Zeichencodes

Wildcards):

SELECT *

FROM table_source;

Wenn die vorherige Abfrage für eine Tabelle mit dem Namen riverTable ausgeführt wird (die die Namen, Längen, Steigungen und den Kontinent der Flüsse enthält), wird die folgende Tabelle angezeigt:

Serial	River	LengthKm	LengthMiles	Outflow	Continent
1	Amazon	6992	4345	Atlantic Ocean	South America
2	Nile	6853	4258	Mediterranean	Africa
3	Mississippi	6275	3902	Gulf of Mexico	North America
4	Yellow River	5464	3395	Bohai Sea	Asia
5	Congo	4700	2922	Atlantic Ocean	Africa
6	Lena	4400	2736	Laptev Sea	Asia
7	Niger	4200	2611	Gulf of Guinea	Africa
8	Volga	3645	2266	Caspian Sea	Europe
9	Salween	3060	1901	Andaman Sea	Asia

Wählen Sie einige Spalten aus, die angezeigt werden:sollen

Der Satz von Spalten, die in der Ausgabe angezeigt werden, kann bestimmt werden, indem die Namen dieser Spalten durch ein Komma getrennt werden, das in der folgenden Abfrage angezeigt wird, die die Namen und Mündungen von Flüssen und ihre Länge in km anzeigt:

```
SELECT River, Outflow, LengthKm
FROM riverTable;
```

Die vorherige Abfrage zeigt das folgende Ergebnis an, bei dem sich die Reihenfolge der Spalten von der Reihenfolge in der ursprünglichen Tabelle unterscheidet:

River	Outflow	LengthKm
Amazon	Atlantic Ocean	6992
Nile	Mediterranean	6853
Mississippi	Gulf of Mexico	6275
Yellow River	Bohai Sea	5464
Congo	Atlantic Ocean	4700
Lena	Laptev Sea	4400
Niger	Gulf of Guinea	4200
Volga	Caspian Sea	3645
Salween	Andaman Sea	3060

Datensätze in einer bestimmten Reihenfolge:auswählen

Die folgende Abfrage legt die Spalten River, LengthKm und Continent der riverTable-Tabelle fest, die im vorherigen Abschnitt angezeigt werden, und fügt der Spalte Kontinent einen alternativen Namen (die Aliasbeschriftung über den Parameter AS) hinzu, um sie als Ort anzuzeigen, und ordnet dann die Ergebnisse alphabetisch aufsteigend durch den Order BY-Ausdruck an:

```
SELECT River, LengthKm, Continent AS Place
FROM riverTable
ORDER BY Continent;
```

Das in der folgenden Tabelle dargestellte Ergebnis zeigt, wie die Spalte Kontinent als Ort angezeigt wurde:

River	LengthKm	Place
Nile	6853	Africa
Congo	4700	Africa
Niger	4200	Africa
Yellow River	5464	Asia
Lena	4400	Asia
Salween	3060	Asia
Volga	3645	Europe
Mississippi	6275	North America
Amazon	6992	South America

Bestimmen der Datensätze, für die eine bestimmte Bedingung gilt

Die folgende Abfrage wird in der RiverTable-Tabelle ausgeführt, die den Namen des Flusses angibt und seine Länge in Metern angibt (durch Angabe

der Spalte LengthKm und Multiplikation mit 1000 bis einem Meter unter Verwendung mathematischer Koeffizienten und Anzeige der Position des Kontinents aus der RiverTable-Tabelle mit einer Länge von mehr als 4000 km), und dann die gesamte alphabetische Ausgabereihenfolge durch den Namen des Kontinents anordnen, die durch den Namen des Kontinents aufsteigt). :

```
SELECT River, LengthKm * 1000 AS LengthM, Continent
FROM riverTable
WHERE LengthKm> 4000
ORDER BY Continent;
```

Das in der folgenden Tabelle angezeigte Ergebnis wird angezeigt:

River	LengthM	Continent
Nile	6853000	Africa
Congo	4700000	Africa
Niger	4200000	Africa
Yellow River	5464000	Asia
Lena	4400000	Asia
Mississippi	6275000	North America
Amazon	6992000	South America

Bestimmen Sie die Datensätze, deren Wert eines der eindeutigen Felder:ist

Fügen Sie das Schlüsselwort DISTINCT in der SELECT-Abfrage für unterschiedliche und nicht duplizierte Werte hinzu. Die folgende Abfrage identifiziert die eindeutigen Kontinentnamen aus der riverTable-Tabelle:

```
SELECT DISTINCT Continent
FROM riverTable
```

Das Ergebnis wird angezeigt:

Continent
South America
Africa
North America
Asia
Europe

Funktionen mit Tabellendaten verwenden:

Die folgende Abfrage wird in der riverTable-Tabelle ausgeführt und zeigt die kürzere Flusslänge, die längere Flusslänge und das arithmetische Mittel aller Längen mithilfe von min (), max () und avg () Funktionen in SQL an:

SELECT MIN (Längekm) AS Mindestlänge, MAX (Längekm) as Maximale Länge, AVG (Längekm) als durchschnittliche Länge

FROM riverTable;

Folgende Ergebnisse werden angezeigt:

Minimum Length	Maximum Length	Average Length
6992	3060	5065.444

Abrufen von Daten aus mehr als einer:Tabelle

Daten können aus mehr als einer Tabelle in der Datenbank abgerufen werden, indem die Namen der Tabellen angegeben werden, deren Daten im

Schlüsselwort FROM angezeigt werden sollen. Die folgende Abfrage ruft die erforderlichen Spalten ab, wie sie in der entsprechenden Tabelle vorhanden sind:

```
SELECT River, LengthKm, Mountain, HeightKm
FROM riverTable, mountainTable;
```

Wenn Sie sich mit mehr als einer Tabelle befassen, sind die Tabellen mit einem der Verknüpfungskoeffizienten verknüpft.

INSERT-Abfrage :

Die Aufgabe dieser Abfrage besteht darin, einen neuen Datensatz in der Tabelle zu erstellen, und sie hat die allgemeine Struktur:

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

Die Konsistenz zwischen dem Spaltennamen und dem zu legenden Wert muss in der Reihenfolge und in der Reihenfolge des Felddatentyps, d. h. der Kompatibilität, in der Reihenfolge, der Nummer und dem Typ erfolgen. Um allen Feldern im Datensatz Werte zuzuweisen, können die Spaltennamen für die Abfrage wie:

INSERT INTO tbl_name VALUES (value1, value2, value3, ...);

Die Reihenfolge, die Anzahl und der Typ der Werte müssen berücksichtigt werden, um spalten in der Tabelle zu entsprechen. Wenn die Abfrage wie folgt lautet:

INSERT INTO tbl DEFAULT VALUES;

Die Implementierung erstellt einen neuen Datensatz in der Tabelle, der die Standardwerte nach dem für jede Spalte angegebenen Datentyp enthält. Genauso wie MySQL das folgende INSERT-Abfrageformat unterstützt:

INSERT INTO tbl_name

SET assignment_list

Assignment_list ist eine Anzahl von Zuordnungen zu entsprechenden Feldern.

Beispiel

Wenn die folgende Tabelle für Kursteilnehmer in der Datenbank vorhanden ist:

StudentID	Name	Age	GPA
1024	Mona	25	3.68
1081	Radi	24	3.57
1012	Leen	25	2.50
1085	Sarah	26	4.00
1066	Amin	22	1.96
1056	Yusuf	23	2.87

Um einen neuen Schülerdatensatz mit der Seriennummer 1075 mit dem Namen Abed, 21 Jahre alt und 1,90 in dieser Tabelle hinzuzufügen, wird die Abfrage wie folgt:

```
INSERT INTO students (StudentID, Name, Age, GPA)
VALUES (1075, Abed, 21, 1.90);
```

Nachdem diese Abfrage ausgeführt wurde, wird die Tabelle wie:

StudentID	Name	Age	GPA
1024	Mona	25	3.68
1081	Radi	24	3.57
1012	Leen	25	2.50
1085	Sarah	26	4.00
1066	Amin	22	1.96
1056	Yusuf	23	2.87
1075	Abed	21	1.90

Fügen Sie einen Datensatz hinzu, der nicht alle Spaltendaten enthält:

Sie können einen Datensatz hinzufügen, der nicht alle Spaltendaten enthält, und dann die verbleibenden Spalten mit den Standardwerten für den in diesen Spalten angegebenen Datentyp füllen.

Die Implementierung der folgenden Abfrage für die Grundlagentabellen-Studenten:

```
INSERT INTO students (StudentID, Name, Age)
VALUES (NEWID (), Abed, 21);
```

Es wird ein Student mit dem Namen Abed im Alter von 21 und bei 0,0 hinzugefügt, da es sich um den Standardwert der GPA-Spalte handelt und ihre Seriennummer 1086 ist, da die NEWID ()-Funktion dafür verantwortlich ist, die folgende Seriennummer an die letzte Seriennummer in der Tabelle zu geben.

StudentID	Name	Age	GPA
1024	Mona	25	3.68
1081	Radi	24	3.57
1012	Leen	25	2.50
1085	Sarah	26	4.00
1066	Amin	22	1.96
1056	Yusuf	23	2.87
1086	Abed	21	0.0

Hinzufügen einer Anzahl von Datensätzen mit einer einzelnen Abfrage:

Mehrere Datensätze können in dieselbe INSERT-Abfrage eingefügt werden, indem Datensätze getrennt durch ein Komma wie in der folgenden Abfrage getrennt werden:

INSERT INTO students (StudentID, Name, Age, GPA)

VALUES (1075, Abed, 21, 1.90), (1076, Taimaa, 23, 2.75), (1077, Mohammad, 22, 1.85);

Fügen Sie der Tabelle drei Datensätze hinzu, die gelesen werden sollen:

StudentID	Name	Age	GPA
1024	Mona	25	3.68
1081	Radi	24	3.57
1012	Leen	25	2.50
1085	Sarah	26	4.00
1066	Amin	22	1.96
1056	Yusuf	23	2.87
1075	Abed	21	1.90
1076	Taimaa	23	2.75
1077	Mohammad	22	1.85

Hinzufügen von Datensätzen zur Tabelle aus einer anderen Tabelle

Sie können der Tabelle abhängig von der SELECT-Abfrage Datensätze aus einer anderen Tabelle hinzufügen.

Die folgende Tabelle ist newStudents befindet sich in der Datenbank, die die Tabelle Studenten enthält:

StudentID	Name	Age	GPA
2027	Maher	21	1.86
2051	Joud	23	3.57
2082	Kawthar	22	2.40
2075	Asmaa	25	3.68

Beim Ausführen der folgenden Abfrage:

```
INSERT INTO students (StudentID, Name, Age, GPA)
SELECT StudentID, Name, Age, GPA
```

```
FROM newStudents
WHERE GPA> 2.50;
```

Die Datensätze werden aus der neuen Tabelle "Studenten" hinzugefügt, die eine Schülerrate von mehr als 2,50 zur Tabelle "Studenten" hat, sodass die Tabelle "Studenten" wie folgt lautet:

StudentID	Name	Age	GPA
1024	Mona	25	3.68
1081	Radi	24	3.57
1012	Leen	25	2.50
1085	Sarah	26	4.00
1066	Amin	22	1.96
1056	Yusuf	23	2.87
2051	Joud	23	3.57
2075	Asmaa	25	3.68

1.6 DELETE-Abfrage:

Eine DELETE-Abfrage wird verwendet, um einen oder mehrere Datensätze zu löschen, und hat eine allgemeine Struktur:

DELETE FROM tbl_name

WHERE condition;

Geben Sie die Bedingung an, unter der die Datensätze in der WHERE-Klausel gelöscht werden sollen. Die Umsetzung der folgenden beiden Fragen:

DELETE * FROM tbl_name;

DELETE FROM tbl_name;

Löscht alle Datensätze in der Tabelle.

Beispiel

Wenn sich die Kundentabelle wie folgt in der Datenbank befindet:

CustomerID	FullName	OrderID	Bill
1156	Abd al-Salam Hadi	0291	160
1157	Ahmad Mostafa	0302	170
1158	Reem Hammad	0203	210
1159	Abd Allah Sadiq	0294	350
1160	Raghad al-Hamdan	0255	185
1161	Abd al-Razzaq Salloum	0276	165
1162	Hussam Siraj	0247	175
1163	Hiba Maktabi	0208	180
1164	Abd al-Rahman Rida	0219	170
1165	Abd al-Qader Khalil	0210	220
1166	Fateh Hammad	0211	350
1167	Abd al-Azziz Othman	0202	185
1168	Zahraa Qasem	0213	400
1169	Abd al-Wahhab Masri	0314	350
1170	Mona Saber	0215	185

Um Datensätze zu löschen, bei denen der Name des Kunden mit dem Wort "Abd" beginnt oder seine Rechnung zwischen 150 und 200 Dollar liegt, wird die Abfrage wie folgt:

```
DELETE FROM customers
WHERE FullName LIKE 'Abd%' OR Bill BETWEEN 150 AND 200;
```

Der PARAMETER LIKE gibt ein Muster für die Textzeichenfolge im Feld FullName an, das einer beliebigen Folge von Zeichen nach dem Start des Wortes Abd folgt, und der Parameter BETWEEN bestimmt den Bereich zwischen den beiden angegebenen Werten.

Nachdem Sie die Abfrage ausgeführt haben, wird die Tabelle wie:

CustomerID	FullName	OrderID	Bill
1158	Reem Hammad	0203	210
1166	Fateh Hammad	0211	350
1168	Zahraa Qasem	0213	400

Löschen Sie eine angegebene Anzahl von Datensätzen:

Sie können eine bestimmte Anzahl von Datensätzen angeben, die gelöscht werden sollen, indem Sie das SCHLÜSSELwort TOP im SQL Server-Modul und das LIMIT-Schlüsselwort in den Engines PostgreSQL, SQLite, MySQL und ROWNUM auf die gleiche Weise angeben, wie die Anzahl der Datensätze in einer SELECT-Abfrage und eine UPDATE-Abfrage anzugeben.

Wenn Sie die vorherige Abfrage ausführen, wird das Hinzufügen von TOP mit dem SQL Server-Modul wie folgt ausgeführt:

```
DELETE TOP (5) FROM customers
WHERE FullName LIKE 'Abd%' OR Bill BETWEEN 150 AND 200;
```

In PostgreSQL-, SQLite- und MySQL-Engines:

```
DELETE FROM customers
WHERE FullName LIKE 'Abd%' OR Bill BETWEEN 150 AND 200;
LIMIT 5;
```

Wenn die vorherigen drei Abfragen ausgeführt werden, werden nur 5 Datensätze gelöscht, d. h. die Werte in der Tabelle lauten wie folgt:

CustomerID	FullName	OrderID	Bill
1158	Reem Hammad	0203	210
1162	Hussam Siraj	0247	175
1163	Hiba Maktabi	0208	180
1164	Abd al-Rahman Rida	0219	170
1165	Abd al-Qader Khalil	0210	220
1166	Fateh Hammad	0211	350
1167	Abd al-Azziz Othman	0202	185
1168	Zahraa Qasem	0213	400
1169	Abd al-Wahhab Masri	0314	350
1170	Mona Saber	0215	185

Wenn die ORDER BY-Klausel in der Abfrage verwendet wird, wird die erforderliche Anzahl von Datensätzen gelöscht, nachdem sie nach der

angegebenen Spalte sortiert wurden. Die folgende Abfrage wird im SQL Server-Modul ausgeführt:

```
DELETE TOP (5) FROM customers
ORDER BY Bill;
```

Oder in PostgreSQL-, SQLite- und MySQL-Engines wie folgt:

```
DELETE FROM customers
ORDER BY Bill
LIMIT 5;
```

Löscht die ersten fünf Datensätze, nachdem sie nach Bill-Spalte in aufsteigender Reihenfolge eingeordnet wurden, und die Tabelle wird:

CustomerID	FullName	OrderID	Bill
1156	Abd al-Salam Hadi	0291	160
1157	Ahmad Mostafa	0302	170
1158	Reem Hammad	0203	210
1160	Raghad al-Hamdan	0255	185
1161	Abd al-Razzaq Salloum	0276	165
1162	Hussam Siraj	0247	175
1163	Hiba Maktabi	0208	180
1164	Abd al-Rahman Rida	0219	170
1167	Abd al-Azziz Othman	0202	185
1170	Mona Saber	0215	185

Beachten Sie, dass die Reihenfolge mit einer DELETE-Abfrage verwendet wird und daher die Datensätze der vorherigen Tabelle nicht angeordnet sind, da sie nicht das Ergebnis der Abfrage zum Abrufen der SELECT-Datensätze sind und in ihrer ursprünglichen Reihenfolge in der Tabelle verbleiben.

Löschen Sie Daten aus der Tabelle in Abhängigkeit von den Datenwerten in einer anderen Tabelle:

Wenn die folgende Tabelle in der vorherigen Datenbank vorhanden ist, die die vorherigen Debitoren enthält:

OrderID	EmployeeID	Item
0302	096	ZenFone 4 Max
0291	054	ZenFone 3 Max
0213	096	ZenFone 4 4G LTE
0210	054	VivoBook Max
0314	054	ZenFone 3 Deluxe 5.7”
0276	054	ZenFone 3 Laser (ZC551KL)

Um Datensätze aus der Tabelle der Debitoren zu löschen, die von einem bestimmten Mitarbeiter in der Tabelle "Aufträge" angefordert wurden, lautet die Abfrage wie folgt:

```
DELETE FROM customers
WHERE OrderID IN
  (SELECT OrderID
  FROM orders
  WHERE EmployeeID = 054);
```

Dies basiert auf dem Unterabfragekonzept in der IN-Parameterliste, das eine Reihe möglicher Werte für den OrderID-Feldwert angibt. Nach dem Ausführen der vorherigen Abfrage lautet die Kundentabelle wie folgt:

CustomerID	FullName	OrderID	Bill
1157	Ahmad Mostafa	0302	170
1158	Reem Hammad	0203	210
1159	Abd Allah Sadiq	0294	350
1160	Raghad al-Hamdan	0255	185
1162	Hussam Siraj	0247	175
1163	Hiba Maktabi	0208	180
1164	Abd al-Rahman Rida	0219	170
1166	Fateh Hammad	0211	350
1167	Abd al-Azziz Othman	0202	185
1168	Zahraa Qasem	0213	400
1170	Mona Saber	0215	185

Kapitel 2

Bedingungen und Aktualisierungen

2.1 UPDATE-Abfrage:

Diese Abfrage wird verwendet, um Datenwerte zu aktualisieren, die in Tabellendatensätzen gespeichert sind.

Allgemeine Struktur der Abfrage:

UPDATE tbl_name

SET col_name = [| DEFAULT], col_name = value [| DEFAULT], ...

WHERE condition

Es ist möglich, mehrere Werte in mehreren Datensätzen innerhalb derselben Abfrage zu aktualisieren, und wenn das Schlüsselwort DEFAULT verwendet wird, wird der Wert im Feld der Standardwert des darin enthaltenen Datentyps, oder wenn die Abfrage wie:

UPDATE tbl_name

SET col_name = value [| DEFAULT], col_name = value [| DEFAULT], ...

Die Implementierung ändert den Wert des angegebenen Felds in allen Datensätzen in der Tabelle, da keine Bedingung vorliegt, die bestimmt, welche Datensätze geändert werden.

Beispiel

Wenn sich die folgende Tabelle (Mitarbeiter) in der Datenbank befindet:

EmployeeID	Name	Salary	City	Work	WeeklyHours
0156	Natalie Sinno	200	Beruit	In centre	25
0157	Ahmad Rida	275	Alexandria	In centre	30
0158	Kareem al-Hamdan	210	Aleppo	In centre	25
0159	Mahdi Thabit	190	Rabat	In centre	28
0160	Rabie al-Sadi	300	Jedda	In centre	36
0161	Jaber Hammad	290	Amman	In centre	36
0162	Rawda Hussien	210	Cairo	In centre	30

Eine Erhöhung des Gehaltswerts um 10 % und der Änderung der Art der Übergestaltung der Arbeit, die über das Internet anstelle des Zentrums erfolgen soll, für jeden Mitarbeiter, der in einer der Städte lebt (Aleppo, Kairo, Rabat)

UPDATE employees

SET Salary = Salary * 1.1 , Work =’online’

WHERE City IN (Aleppo, Cairo, Rabat);

Die Tabelle mit den neuen Werten lautet wie folgt:

EmployeeID	Name	Salary	City	Work	WeeklyHours
0156	Natalie Sinno	200	Beruit	In centre	25
0157	Ahmad Rida	275	Alexandria	In centre	30
0158	Kareem al-Hamdan	231	Aleppo	online	25
0159	Mahdi Thabit	209	Rabat	online	28
0160	Rabie al-Sadi	300	Jedda	In centre	36
0161	Jaber Hammad	290	Amman	In centre	36
0162	Rawda Hussien	231	Cairo	online	30

Sprechen eines Werts zu einer angegebenen Anzahl von Datensätzen

Abhängig vom Schlüsselwort TOP im SQL Server-Modul und dem LIMIT-Schlüsselwort in den Engines PostgreSQL, SQLite, MySQL und ROWNUM im Oracle-Modul kann eine bestimmte Anzahl von Datensätzen auf die gleiche Weise zugewiesen werden, wie die Anzahl der Datensätze in einer SELECT-Abfrage und eine DELETE-Abfrage anzugeben.

Wenn Sie dieselbe Abfrage ausführen, wird das Hinzufügen von TOP mit dem SQL Server-Modul wie folgt ausgeführt:

```
UPDATE TOP (2) employees
SET Salary = Salary * 1.1, Work = 'online'
WHERE City IN (Aleppo, Cairo, Rabat);
```

In PostgreSQL-, SQLite- und MySQL-Engines:

```
UPDATE employees
SET Salary = Salary * 1.1, Work = 'online'
WHERE City IN (Aleppo, Cairo, Rabat)
LIMIT 2;
```

Im Oracle-Modul als:

```
UPDATE employees
SET Salary = Salary * 1.1, Work = 'online'
WHERE City IN (Aleppo, Cairo, Rabat) AND ROWNUM <= 2;
```

Bei der Umsetzung der drei vorherigen Abfragen wird der Wert von Gehalt und Arbeitsort in zwei Registern nur ohne Änderung des dritten, d. h. der Werte in der Tabelle wie folgt, auftreten:

EmployeeID	Name	Salary	City	Work	WeeklyHours
0156	Natalie Sinno	200	Beruit	In centre	25
0157	Ahmad Rida	275	Alexandria	In centre	30
0158	Kareem al-Hamdan	231	Aleppo	online	25
0159	Mahdi Thabit	209	Rabat	online	28
0160	Rabie al-Sadi	300	Jedda	In centre	36
0161	Jaber Hammad	290	Amman	In centre	36
0162	Rawda Hussien	210	Cairo	In centre	30

Aktualisieren der Werte in den Datensätzen in Abhängigkeit von den in einer anderen Tabelle gespeicherten Werten

Die folgende Tabelle wird in der vorherigen Datenbank mit der Tabelle employees benannt:

EmployeeID	AnnualLeave
0156	10
0157	12
0158	10
0159	11
0160	15
0161	15
0162	12

Um dem Zeitplan je nach der im Arbeitszeitplan angegebenen Anzahl von Stunden zwei zusätzliche Urlaubstage hinzuzufügen, sodass die Arbeitszeit des Mitarbeiters mehr als 25 Stunden beträgt, wird die folgende Abfrage verwendet:

```
UPDATE leaves
SET AnnualLeave = AnnualLeave + 2
FROM (SELECT EmployeeID FROM Employees WHERE 25 WeeklyHours) AS tempTable
WHERE leaves.EmployeeID = tempTable.EmployeeID;
```

Diese Abfrage hängt vom Unterabfragekonzept in der FROM-Klausel ab, das die Datensätze aus der Tabelle employees mit mehr als 25 Arbeitsstunden angibt. Die resultierende Tabelle wird mit dem Parameter AS und dem Vergleich zwischen den beiden Tabellen unter der WHERE-Klausel tempTable benannt.

2.2 WHERE-Abfrage:

Die WHERE-Klausel in der SELECT-Abfrage wird verwendet, um die Bedingungen anzugeben, die Datensätze anzeigen oder löschen, oder ihre Feldwerte entsprechend zu ändern. Sie können auch mit den DELETE- und UPDATE-Abfragen verwendet werden und basieren in erster Linie auf logischen und arithmetischen Transaktionen und einigen anderen Parametern bei der Formulierung dieser Begriffe.

Abrufen von Daten in einem einfachen Zustand

Wenn die folgende Tabelle in der Datenbank vorhanden ist:

StudentID	Name	Age	GPA
1024	Mona	25	3.68
1081	Radi	24	3.57
1012	Leen	25	2.50
1085	Sarah	26	4.00
1066	Amin	22	1.96
1056	Yusuf	23	2.87

Die folgende Abfrage gibt die Bedingung des Textzeichenfolgenwerts 'Amin' im Feld Name an:

```
SELECT *
FROM students
WHERE Name = 'Amin';
```

Das Ergebnis seiner Umsetzung ist:

StudentID	Name	Age	GPA
1066	Amin	22	1.96

Löschen des überprüften Datensatzes für eine bestimmte Bedingung

Ein oder mehr Datensatz kann gelöscht werden, wenn er eine von WHERE angegebene Bedingung erfüllt.

Durch Ausführen der folgenden Abfrage werden alle Schülerdatensätze mit einer Rate von weniger als 2,5 gelöscht, abhängig von der DELETE-Abfrage:

```
DELETE FROM students
WHERE GPA <2.5;
```

Ändern der Werte in Datensätzen gemäß einer bestimmten Bedingung

WHERE gibt die Bedingung an, unter der Datensätze, die ihre Werte ändern, mithilfe der UPDATE-Abfrage bestimmt werden, da ein oder mehrere Werte geändert werden können.

Die folgende Abfrage ändert die Alters- und Ratewerte des Datensatzes mit dem Namen Yusuf:

```
UPDATE students
SET Age = 24, GPA = 3.4
WHERE Name = 'Yusuf';
```

Beachten Sie, dass, wenn Sie in dieser Abfrage nicht WHERE angeben, die Bearbeitung alle Datensätze in der Tabelle enthält.

Verknüpfungsbedingungen mithilfe logischer Transaktionen

Logische Operatoren (AND, OR, NOT) werden verwendet, um mehr als eine Klausel zu binden oder sie innerhalb der WHERE-Klausel zu verweigern.

Die Implementierung der folgenden Abfrage bestimmt die Bedingung, dass im mittleren Feld ein Wert größer als 3,00 und im Bereich des Alters ein Wert größer als 24 Jahre vorhanden ist. Die Ergebnisse werden nach dem Mittelwert angeordnet und in der Tabelle mit der Bezeichnung AS als "Top Students" anstelle von Name bezeichnet:

SELECT Name AS Top Students, Age, GPA

FROM students

WHERE GPA> 3.00 AND age> 24

ORDER BY GPA DESC;

Die folgenden Ergebnisse werden angezeigt:

Top Students	Age	GPA
Sarah	26	4.00
Mona	25	3.68

Hier wird die Anzahl der für eine bestimmte Bedingung erreichten Datensätze angezeigt.

Um die Anzahl der Datensätze zu bestimmen, die eine bestimmte Bedingung erfüllen, abhängig von der Zählfunktion, und die angegebene gewünschte Bedingung in der WHERE-Klausel anzugeben, führen Sie die folgende Abfrage für die obige Tabelle aus (mit dem Parameter BETWEEN, der einen Bereich möglicher Werte im Feld Alter angibt):

```
SELECT count (*) AS Count
FROM students
WHERE age BETWEEN 23 AND 25;
```

Das Ergebnis wird angezeigt:

Count
4

Hilfetransaktionen in der WHERE-Anweisung

Die folgende Tabelle zeigt einige der Parameter, die in der Formulierung der Bedingung verwendet werden, die durch das Schlüsselwort WHERE angegeben wird:

Parameter // Description

= // Equivalence

The situation of inequality

</ Greater than

> // smaller than

= <// Greater than or equal to

=> // is less than or equal to

BETWEEN // located within a specified domain

LIKE // Similar to a specific pattern follows the keyword

IN // to specify several possible values for the value taken from the column

Die folgenden logischen Transaktionen können auch verwendet werden, um mehrere Bedingungen zu verknüpfen oder zu verweigern:

Parameter // Description

AND // to display the log if all conditions are met

OR // to view the record if one or more conditions are met

NOT // if the condition is met

2.3 GRUPPE BY Anfrage :

Die GROUP BY-Klausel wird in einer SELECT-Abfrage verwendet, um Ergebnisse nach einer oder mehreren Spalten der Tabelle in Gruppen zu gruppieren.

Verwenden von GROUP BY, um zu sehen, wie viele Datensätze gruppiert werden können

Lassen Sie die folgende Tabelle (Patienten) in der Datenbank vorhanden sein:

PatientID	Patient	Age	Section	Doctor
1029	Salem	76	Cardiology	Nasser
896	Hasan	17	Neurology	Abd Allah
964	Nadine	23	Maternity	Reem
1070	Rami	65	Cardiology	Fateh
1150	Abd al-Rahim	19	Neurology	Kareem
1097	Roushd	65	Critical Care	Hadi
1034	Sana	27	Maternity	Reem
963	Helen	49	Neurology	Abd Allah
1154	Ahmad	58	Critical Care	Hadi
1069	Maher	40	Neurology	Kareem
1028	Zeina	30	Maternity	Amnah
1180	Sarah	55	Cardiology	Fateh
1076	Fadi	23	Critical Care	Hadi
987	Naim	36	Neurology	Abd Allah
1156	Batoul	70	Cardiology	Fateh
1181	Rahaf	24	Maternity	Amnah
808	Aya	66	Cardiology	Nasser

Die folgende Abfrage bestimmt die Anzahl der Patienten in jedem Abschnitt basierend auf der Aggregatfunktion ():

```
SELECT Section, count (*) AS Number of patients
FROM patients
GROUP BY Section;
```

Die Ergebnisse lauten wie folgt:

Section	Number of patients
Cardiology	5
Neurology	5
Maternity	4
Critical Care	3

Und verwenden Sie zwei Felder innerhalb der GROUP BY-Klausel wie in der folgenden Abfrage:

```
SELECT Section, Doctor, count (Patient) AS Number of patients
FROM patients
GROUP BY Section, Doctor
```

Die Ergebnisse werden zuerst nach der ausgewählten Spalte, dann nach der zweiten Spalte usw. gruppiert, und die folgenden Ergebnisse werden angezeigt:

Section	Doctor	Number of patients
Cardiology	Nasser	2
Cardiology	Fateh	3
Neurology	Abd Allah	3
Neurology	Kareem	2
Maternity	Reem	2
Maternity	Amnah	2
Critical Care	Hadi	3

Zusammenstellung der für eine bestimmte Bedingung erzielten Ergebnisse:

Die folgende Abfrage bestimmt die Anzahl der Patienten pro Arzt und schließt Ärzte aus, die für weniger als 3 Patienten behandelt werden, abhängig vom Begriff HAVING, der den erforderlichen Zustand bei der Verwendung der Baugruppe bestimmt:

```
SELECT Doctor, count (Patient) AS Number of patients
FROM patients
GROUP BY Doctor
HAVING count (Patient)> 2;
```

Die Ergebnisse lauten wie folgt:

Doctor	Number of patients
Fateh	3
Abd Allah	3
Hadi	3

Gruppenergebnisse aus mehreren Tabellen:

Lassen Sie die folgende Tabelle (Ärzte) in der Datenbank (zusätzlich zu der Tabelle über den Namen der Patienten):

DoctorName	City
Nasser	Damascus
Fateh	Aleppo
Abd Allah	Aleppo
Kareem	Hama
Reem	Damascus
Amnah	Aleppo
Hadi	Damascus

Die Implementierung der folgenden Abfrage:

```
SELECT doctors.City, count (patients.Piant) AS Number of patients in city
FROM patients
LEFT JOIN doctors ON patients.Doctor = doctors.DoctorName
GROUP BY City;
```

Die Tabellen sind linkshändig verknüpft und die Ergebnisse werden nach der Stadtspalte in der Tabelle der Ärzte gruppiert. Dies führt zu der Anzahl der Patienten in jeder Stadt wie folgt:

Number of patients in city	City
7	Damascus
8	Aleppo
2	Hama

Hinweis: Spalten, die Text, ntext oder image sind, können nicht innerhalb der GROUP BY-Klausel verwendet werden.

Kapitel 3

Datenbanken

3.1 ALTER INDEX

Dieser Befehl wird verwendet, um den mit dem Befehl CREATE INDEX erstellten Index zu ändern, und verfügt über die folgenden Formeln:

```
ALTER INDEX [IF EXISTS] name RENAME TO new_name
```

Ändern Sie den Namen des Indexnamens (falls vorhanden), um new_name zu werden.

ALTER INDEX [IF EXISTS] name SET (storage_parameter = value [, ...])

So ändern Sie ggf. einen der Indexspeichertransaktionswerte.

ALTER INDEX [IF EXISTS] name RESET (storage_parameter [, ...])

Setzen Sie alle Speicherparameterwerte für diesen Index (falls vorhanden) auf die Standardwerte zurück. Im Motor

SQL müssen Sie das Objektobjekt angeben, das seinen Index so ändert, dass er zur allgemeinen Struktur im Formular wird:

```
ALTER INDEX index_name ON <object>
{
   DISABLE
   | SET (<set_index_option> [, ... n])
   | PAUSE
};
```

Das Wort DISABLE wird verwendet, um den Index zu deaktivieren, SET, um die Werte seiner Optionen anzupassen, und PAUSE, um ihn anzuhalten.

Beispiel

Um den Füllfaktorspeicherwert für Indexverteiler auf 75 festzulegen, verwenden Sie den Befehl:

```
ALTER INDEX distributors SET (fillfactor = 75);
```

Kompatibel mit

SQLServer / / SQLite / / Oracle / / PostgreSQL / MySQL

Im MySQL-Modul kann die Indexbezeichnung mit dem Befehl ALTER TABLE geändert werden. Im SQLite-Modul kann der aktuelle Index gelöscht und ein neuer Index mit dem neuen Namen erstellt werden.

3.2 ALTER TABELLE

Dieser Befehl wird verwendet, um die vordefinierte Tabellendefinition über den Befehl CREATE TABLE zu ändern, der die folgenden Formeln enthält:

```
ALTER TABLE [IF EXISTS] name
    RENAME [COLUMN] col_name TO new_col_name
```

Benennen Sie eine Spalte mit dem Namen col_name in der Tabelle um, deren Name (falls vorhanden) mit dem neuen Namen new_col_name benannt ist.

ALTER TABLE [IF EXISTS] name

    RENAME CONSTRAINT const_name TO new_const_name

Benennen Sie eine Einschränkung mit dem Namen const_name in der Tabelle um, deren Name (falls vorhanden) mit dem neuen Namen new_const_name benannt ist.

ALTER TABLE [IF EXISTS] name

    RENAME TO new_name

Benennen Sie die Tabelle, deren Name benannt ist (falls vorhanden), mit dem neuen Namen new_name um.

ALTER TABLE [IF EXISTS] name

    action [, ...]

Um eine Änderung des Zeitplans zu implementieren, unter den folgenden Änderungen:

ADD [COLUMN] [IF NOT EXISTS] column_name data_type [column_constraint [...]]

Hinzufügen einer neuen Spalte (falls nicht bereits vorhanden) als column_name und geben Sie data_type ein und fügen Sie Einschränkungen hinzu.

DROP [COLUMN] [IF EXISTS] column_name [CASCADE]

Löschen Sie eine Spalte mit dem Namen column_name (falls vorhanden), und verwenden Sie das Schlüsselwort CASCADE, wenn eine Abhängigkeit von dieser Spalte von einem anderen Element außerhalb der Tabelle besteht (z. B. im Fall des sekundären Schlüssels).

ALTER [COLUMN] column_name [SET DATA] TYPE data_type

Ändern des Spaltendatentyps column_name als data_type-Typ

ALTER [COLUMN] column_name SET DEFAULT expression

Ändern des Standardwerts für die column_name Spalte, um zum Ausdruckswert zu werden

---

ALTER [COLUMN] column_name DROP DEFAULT

Löschen des aktuellen Standardwerts für die Spalte column_name

---

ALTER [COLUMN] column_name { SET | DROP } NOT NULL

Einschränkung hinzufügen / Löschen Der NULL-Wert sollte in der spaltet column_name nicht NULL sein.

ALTER [COLUMN] column_name SET (attribute_option = value [, ...])

Ändern des Werts eines der Attribute der attribute_option-Spalte column_name zu einem Wert werden

---

ALTER [COLUMN] column_name RESET (attribute_option [, ...])

Zurücksetzen des Werts eines der Attribute der attribute_option-Spalte column_name zum Standardwert werden

---

ADD table_constraint

Hinzufügen einer Einschränkung auf Tabellenebene

Beispiele

Um der Tabelle "Studenten" eine neue Spalte hinzuzufügen, verwenden Sie die folgende Adresse:

ALTER TABLE students ADD COLUMN address varchar (50);

So ändern Sie die Anzahl der verfügbaren Zeichen:

ALTER TABLE students ALTER COLUMN address TYPE varchar (100);

Um eine Einschränkung hinzuzufügen, lautet der Wert NULL:

ALTER TABLE students ALTER COLUMN address SET NOT NULL;

So entfernen Sie diese Einschränkung:

ALTER TABLE students ALTER COLUMN address DROP NOT NULL;

Um der Tabelle "Studenten" eine Einschränkung hinzuzufügen, handelt es sich bei dieser Spalte um einen sekundären Schlüssel mit dem Namen st_fk der der Adresstabelle über die Adressspalte zugeordnet ist:

ALTER TABLE students ADD CONSTRAINT st_fk FOREIGN KEY (address) REFERENCES addresses (address);

So machen Sie diese Spalte mit dem Namen add_uniq eindeutig:

ALTER TABLE students ADD CONSTRAINT add_uniq UNIQUE (address);

So löschen Sie sie erneut:

ALTER TABLE students DROP COLUMN address;

So machen Sie die studentID-Spalte zu einem Anfangsschlüssel für die Tabelle "Studenten":

ALTER TABLE students ADD PRIMARY KEY (studentID);

So benennen Sie die Tabelle in SchoolStudents um:

ALTER TABLE students RENAME TO schoolStudents;

Kompatibel mit

SQLServer / / SQLite / / Oracle / / PostgreSQL / MySQL

3.3 INDEX ERSTELLEN

Dieser Befehl wird verwendet, um einen Index der Tabelle entsprechend der folgenden allgemeinen Struktur zu erstellen:

```
CREATE INDEX [name] ON tbl (col [, ...]);
```

Der Index ist nach der Tabelle tbl benannt, die die Spalte col enthält. Indizes sind eines der grundlegenden Konzepte in Datenbanken. Sie beschleunigen die Abfrageleistung, indem sie die Anzahl der gescannten Seiten in der Datenbank reduzieren.

Beispiel

Der folgende Befehl wird verwendet, um ein idx-Handle für die Records-Tabelle über die Titelspalte zu erstellen:

```
CREATE INDEX idx ON records (title);
```

Kompatibel mit

SQLServer / / SQLite// Oracle/ / PostgreSQL/MySQL

3.4 TABELLE ERSTELLEN

Dieser Befehl wird verwendet, um eine neue Tabelle in der Datenbank entsprechend der folgenden allgemeinen Struktur zu erstellen:

```
CREATE TABLE [IF NOT EXISTS] tbl_name ([
   {col_name data_type [column_constraint [...] | table_constraint}
    [, ...]
])
```

Die neue Tabelle wird mit dem Namen tbl_name erstellt, und ihre Spalten werden in Klammern definiert, indem die col_name Spalte, ihr data_type Datentyp angegeben und die Spalten- oder Tabellenebeneneinschränkungen angegeben werden.

Spalteneinschränkungen

Es können einige Einschränkungen für die Spalte festgelegt werden, z. B.:

NOT NULL // The field value is not NULL

DEFAULT default_expr // Specifies the default value for the field

UNIQUE // Values are not repeated between records within the same field (column)

PRIMARY KEY // to specify the column as the primary key of the table

REFERENCES reftable [(refcolumn)] // To specify the refcolum column as a secondary key with reference to the reftable table

Tabelleneinschränkungen

Einige Einschränkungen auf Tabellenebene, z. B.:

UNIQUE (column_name [, ...])

Keine Duplizieren von Werten zwischen Datensätzen innerhalb des angegebenen Feldes (Spalte)

PRIMARY KEY (column_name [, ...]]

Gibt die Spalte (oder Spalten) zwischen den Klammern als Primärschlüssel der Tabelle an.

FOREIGN KEY (column_name [, ...]) REFERENCES reftable [(refcolumn [, ...])]

Gibt die Spalte (oder Spalten) zwischen den Klammern als sekundären Schlüssel mit Verweis auf die Referenztabelle an.

Beispiel

So erstellen Sie die folgende Tabelle als Schüler:

RegDate	CourseID	E-mail	LastName	FirstName	StudentID

Das enthält 5 Spalten mit unterschiedlichen Datentypen. Die E-Mail-Adresse wird nicht zwischen mehr als einem Kursteilnehmer wiederholt, das Feld für das Registrierungsdatum ist nicht leer, das Feld StudentID ist ein Primärschlüssel, und das Feld CourseID ist ein sekundärer Schlüssel, der diese Tabelle mit einer anderen Tabelle verknüpft.

```
CREATE TABLE students (
    StudentID INT PRIMARY KEY,
    FirstName VARCHAR (20),
    LastName VARCHAR (20),
    E-mail VARCHAR (50) UNIQUE,
    CourseID INT REFERENCES courses (courseID),
    RegDate DATE NOT NULL
);
```

Kompatibel mit

SQLServer * SQLite * Oracle * PostgreSQL * MySQL

3.4 TABELLE ERSTELLEN

Dieser Befehl wird verwendet, um eine neue Tabelle in der Datenbank entsprechend der folgenden allgemeinen Struktur zu erstellen:

```
CREATE TABLE [IF NOT EXISTS] tbl_name ([
  {col_name data_type [column_constraint [...] | table_constraint}
```

[, ...]

])

Die neue Tabelle wird mit dem Namen tbl_name erstellt, und ihre Spalten werden in Klammern definiert, indem die col_name Spalte, ihr data_type Datentyp angegeben und die Spalten- oder Tabellenebeneneinschränkungen angegeben werden.

Spalteneinschränkungen

Es können einige Einschränkungen für die Spalte festgelegt werden, z. B.:

NOT NULL / / Der Feldwert ist nicht NULL

DEFAULT default_expr / Gibt den Standardwert für das Feld an

UNIQUE / Werte werden nicht zwischen Datensätzen innerhalb desselben Feldes wiederholt (Spalte)

PRIMARY KEY / um die Spalte als Primärschlüssel der Tabelle anzugeben

REFERENCES reftable [(refcolumn)] = So geben Sie die Refcolum-Spalte als sekundären Schlüssel mit Verweis auf die Reftable-Tabelle an

Tabelleneinschränkungen

Einige Einschränkungen auf Tabellenebene, z. B.:

UNIQUE (column_name [, ...])

Keine Duplizieren von Werten zwischen Datensätzen innerhalb des angegebenen Feldes (Spalte)

PRIMARY KEY (column_name [, ...]]

Gibt die Spalte (oder Spalten) zwischen den Klammern als Primärschlüssel der Tabelle an.

FOREIGN KEY (column_name [, ...]) REFERENCES reftable [(refcolumn [, ...])]

Gibt die Spalte (oder Spalten) zwischen den Klammern als sekundären Schlüssel mit Verweis auf die Referenztabelle an.

Beispiel

So erstellen Sie die folgende Tabelle als Schüler:

RegDate	CourseID	E-mail	LastName	FirstName	StudentID

Das enthält 5 Spalten mit unterschiedlichen Datentypen. Die E-Mail-Adresse wird nicht zwischen mehr als einem Kursteilnehmer wiederholt, das Feld für das Registrierungsdatum ist nicht leer, das Feld StudentID ist

ein Primärschlüssel, und das Feld CourseID ist ein sekundärer Schlüssel, der diese Tabelle mit einer anderen Tabelle verknüpft.

```
CREATE TABLE students (
    StudentID INT PRIMARY KEY,
    FirstName VARCHAR (20),
    LastName VARCHAR (20),
    E-mail VARCHAR (50) UNIQUE,
    CourseID INT REFERENCES courses (courseID),
    RegDate DATE NOT NULL
);
```

Kompatibel mit

SQLServer * SQLite * Oracle * PostgreSQL * MySQL

Herzlichen glückwunsch.. Da Sie das Buch verstanden und abgeschlossen haben, bedeutet dies, dass Sie den halben Weg im Umgang mit Datenbanken abgeschlossen haben