SQL DATE, GROUP BY & HAVING

von Mohamed Schmidt und Alexander Zierhut

`DATE` Als Datentyp

DATE:
Als Datentyp

Syntax	Format
DATE	YYYY-MM-DD
DATETIME	YYYY-MM-DD HH:MI:SS
TIMESTAMP	YYYY-MM-DD HH:MI:SS
YEAR	YYYY

*TIMESTAMP ist eindeutig (UTC) und zählt ab '1970-01-01 00:00:01' bis '2038-01-19 03:14:07' UTC.

DATE:
Als Datentyp

Beispiel:

```
ALTER TABLE rechnung ADD rechnungsdatum DATE;
```

Der Entitätstyp „rechnungsdatum” vom Datentyp „date” wird hinzugefügt.

```
INSERT INTO rechnung (rechnungsdatum)
VALUES ("2017-12-13");
```

Eine Zeile wird in die Tabelle „rechnung” eingefügt.
Dem Entitätstyp „rechnungsdatum” wird das Datum „13.12.2017” zugewiesen.

CURDATE() ≠ GETDATE()

Beispiel:

```
SELECT CURDATE();
```

CURDATE() gibt das aktuelle Datum aus
Mit CURDATE() kann gerechnet werden

```
SELECT DATE_ADD(CURDATE(), INTERVAL -1 YEAR);
```

Vom aktuellen Datum wird ein Jahr subtrahiert; das Ergebnis wird ausgegeben.

SELECT DATE

SELECT DATE

Syntax:

```
SELECT Entitätstyp FROM tabellenname WHERE DATUMSEINHEIT(datum)
BEDINGUNG;
```

Beispiel:

```
SELECT kundenID FROM rechnung WHERE YEAR(rechnungsdatum) = “2017”;
```

Aus der Tabelle „rechnung” wird jede Kundennummer desjenigen Kunden ausgegeben , der dieses Jahr bereits etwas bestellte.

DATE_ADD()

DATE_ADD

Syntax:

```
SELECT DATE_ADD(“Datum”, INTERVAL Anzahl DATUMSEINHEIT);
```

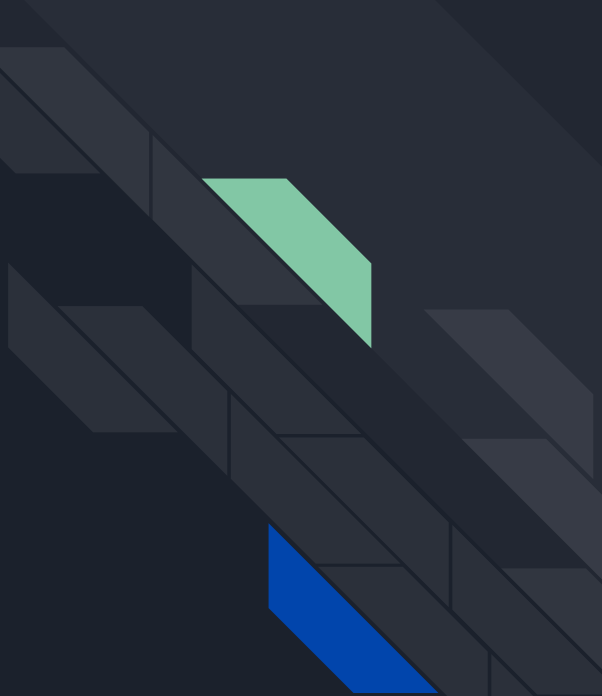
Beispiel:

```
SELECT Date_Add("2017-12-13", INTERVAL -10 DAY);
```

Vom Datum „13.12.2017” werden 10 Tage abgezogen und das Ergebnis wird ausgegeben.

DATENAME()

DAYOF...()



DATENAME

Syntax:

```
SELECT [DATUMSEINHEIT]NAME(“Datum”);
```

Die Datumseinheit wird ausgeschrieben (wenn möglich) zurückgegeben.

Beispiel:

```
SELECT DAYNAME("2017-12-13");
```

Der Wochentag wird ausgeschrieben zurückgegeben (In diesem Fall „Wednesday”).

DAYOF...

Syntax:

```
SELECT DAYOF[DATUMSEINHEIT](“Datum”);
```

Der Tag des Datums wird in der Datumseinheit zurückgegeben.

Beispiel:

```
SELECT DAYOFYEAR("2017-12-13");
```

Ausgabe: 347

DATEDIFF()

DATEDIFF

Syntax:

```
SELECT DATEDIFF('Datum1', 'Datum2');
```

Die Differenz der beiden Daten werden in Tagen ausgegeben.

Beispiel:

```
SELECT DATEDIFF('2016-12-13', '2017-12-13');
```

Ausgabe : 365

Weiter geht es mit:

`GROUP BY` & `HAVING`

GROUP BY

GROUP BY: Syntax

Syntax

```
SELECT *
FROM `Entitätstyp`
GROUP BY column;
```

GROUP BY: Anwendung

- Zeilen zusammenfassen
- Gruppen nach Attributen aufstellen
- Nutzung im Zusammenhang mit Funktionen wie. COUNT, MIN, MAX, SUM, AVG

GROUP BY: Beispiel

SQL Query zum anzeigen von Kunden nach Land

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

GROUP BY: Beispiel

Query 1

```
SELECT `Name`, Max(`Population`) FROM `country` GROUP BY `Code`
```

vs

Query 2

```
SELECT `Name`, Max(`Population`) FROM `country`
```

HAVING

HAVING: Syntax

Syntax

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

HAVING: Anwendung

- Um eine Bedingung auf nach Gruppen aufgeteilte Daten anzuwenden
- Beispiel: Nach Abteilung zusammenfassen, dann nur die Abteilungen anzeigen, mit mindestens 2 Mitarbeitern

HAVING: Beispiel

Query

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5
ORDER BY COUNT(CustomerID) DESC;
```

Nur Kundenanzahl und Land ausgeben, nach Land, sortiert von Vielen zu Wenigen. Außerdem nur Länder mit mindestens fünf Kunden.

Danke für eure Aufmerksamkeit!

von Mohamed Schmidt und Alexander Zierhut