**Mit Datums- und Zeitfunktionen arbeiten**

**Roland Grothe, Hanau; André Minhorst, Duisburg**

**In diesem Beitrag lernen Sie den Umgang mit den Datums- und Zeitfunktionen von Access und ihre Einsatzmöglichkeiten in Formularen, Berichten und Abfragen kennen. Anhand kleiner praxisbezogener Beispiele erfahren Sie, wie Datums- und Zeitinformationen mit diesen Funktionen analysiert und bearbeitet werden. Dabei lernen Sie unter anderem, wie Sie die aktuelle Zeit und das aktuelle Datum ermitteln, wie Sie Werte in ein Datumsformat konvertieren und dieses auf Gültigkeit prüfen. Auch das Extrahieren von Informationen wie Tag, Monat oder Jahr aus einer Datumsangabe kann in vielen Fällen weiterhelfen - aber sehen Sie doch einfach selbst.**

**Einführung**

Für Access sind Datums- und Zeitwerte nichts weiter als Zahlen, die in einem ganz bestimmten Format angezeigt werden. Access kann Daten und Uhrzeiten zwischen dem 1.1.100, 00:00:00 Uhr und dem 31.12.9999, 23:59:59 verwalten. Jeder Zeitpunkt in diesem Bereich wird durch einen Wert zwischen -657434,999988426 und 2958465,99998843 eindeutig definiert. Die Stellen vor dem Komma bestimmen den Tag, die Stellen danach den exakten Zeitpunkt in Stunden, Minuten und Sekunden. Diese Methode ermöglicht es, sowohl das Datum als auch die Uhrzeit in einem einzigen Feld zu speichern. Access stellt dafür den Datentyp Datum/Uhrzeit zur Verfügung.

Der Beginn der Zeitrechnung ist für Access der 30. Dezember 1899. Dieser Tag entspricht dem Datumswert 0. Alle Tage danach werden durch eine positive, alle Tage davor durch eine negative Zahl dargestellt.

Mit den Datums- und Zeitfunktionen stellt Access eine Schnittstelle zur Verfügung, welche die als 64Bit-Fließkommazahlen gespeicherten Informationen in die für uns geläufigen Einheiten zwischen Jahr und Sekunde umrechnet und bei Berechnungen diese Zeitintervalle als Eingabe erlaubt.

**Aktuelle Zeit und
aktuelles Datum**

Eine der am häufigsten eingesetzten Funktionen ist die Funktion Datum() (engl.: Date()). Sie wird meist in den Kopfzeilen von Berichten verwendet, um das Druckdatum auszugeben.

Auch wenn der Funktion keine Argumente übergeben werden können, müssen Sie dennoch das abschließende Klammernpaar eingeben. Datum() liefert das aktuelle Systemdatum als Wert des Datentyps Datum/Uhrzeit zurück.

Setzen Sie die Eigenschaft Steuerelementinhalt eines Textfeldes in einem Bericht oder Formular auf den folgenden Ausdruck, damit dort zur Laufzeit das aktuelle Datum erscheint:

=Datum()

Dieses können Sie natürlich wie jedes andere Datumsfeld nach Ihren Anforderungen formatieren.

Benötigen Sie statt des Datums die aktuelle Uhrzeit, verwenden Sie die Funktion Zeit() (engl.: Time()), um die im BIOS des Rechners gespeicherte Systemzeit zu ermitteln.

Beide Informationen zusammen erhalten Sie, wenn Sie die Funktion Jetzt() (engl.: Now())einsetzen. Diese Funktion liefert sowohl das aktuelle Datum als auch die aktuelle Uhrzeit in einem Wert des Datentyps Datum/Uhrzeit.

Wenn Sie in Ihrer Anwendung die Information, wann ein Datensatz angelegt worden ist, speichern möchten, verwenden Sie dazu die Funktion Jetzt() beziehungsweise Now(). Fügen Sie der Tabelle ein zusätzliches Feld AngelegtAm vom Felddatentyp Datum/Uhrzeit hinzu und legen Sie ein entsprechendes Textfeld im Eingabeformular an. Stellen Sie dessen Eigenschaft Standardwert auf den Ausdruck =Jetzt() und die Eigenschaft Sichtbar auf Nein ein. Dieses Feld ist für den Anwender unsichtbar, speichert aber den Zeitpunkt, wann ein Datensatz angelegt wurde, in der Datenherkunft des Formulars ab. (

**Konvertieren und Prüfen von Datumsangaben**

Die Funktion ZDate() (engl.: CDate())wandelt einen beliebigen als Datum oder Uhrzeit interpretierbaren Ausdruck in einen Wert des Datentyps Datum/Uhrzeit um.

Ob ein Ausdruck als gültig bewertet wird, ist abhängig von der Einstellung Gebietsschema in den Ländereinstellungen in der Systemsteuerung (s. Abb. 1). So liefert beispielsweise der Ausdruck

ZDate("2. Februar 2005")

in einer deutschen Umgebung ein gültiges Datum, während

ZDate("February 2, 2005")

einen Fehler zurückgibt. In der englischen Version ist es hingegen genau umgekehrt.

Dabei wird nicht nur das Format, sondern auch das Datum selbst auf seine Richtigkeit überprüft.

Die Ausdrücke

ZDate("31. September 2002")

und

ZDate("29. Februar 2002")

erzeugen immer einen Fehler, während folgender Ausdruck akzeptiert wird:

ZDate("29. Februar 2004")

Solche Eingabefehler lassen sich vermeiden, wenn der Ausdruck vorher mit der Funktion IstDatum() (engl.: IsDate()) auf seine Gültigkeit überprüft wird. Diese Funktion gibt den Bool'schen Wert Wahr zurück, wenn die an die Funktion übergebene Zeichenkette als Datum oder Uhrzeit interpretiert werden kann. Ist dies nicht der Fall, ist das Ergebnis Falsch.



**Abb. 1: Erkennt Access gültige Datums- und Zeit-Ausdrücke auf einer Maschine nicht an, könnte dies an einer fehlerhaften Einstellung des Gebietsschemas liegen.**

Mit dem Ausdruck

Bestelldatum = Wenn(IstDatum(
[Datumseingabe]);[Datumseingabe];Null)



**Abb. 2: Die Abfrage qryUmsatzProJahr ermittelt den kumulierten Gesamtumsatz jedes Jahres.**

wird dem Feld BestellDatum nur dann ein Wert zugewiesen, wenn Datumseingabe ein gültiges Datum enthält.

**Datumsteile extrahieren**

Oft ist es notwendig, nur einen bestimmten Teil eines Datums für eine Berechnung heranzuziehen. Um aus der Tabelle tblBestellungen der Beispieldatenbank beispielsweise den Jahresumsatz zu ermitteln, wird nicht das genaue Bestelldatum, sondern lediglich das Jahr, aus dem die Bestellung stammt, benötigt. Der Ausdruck

Jahr: Jahr([Bestelldatum])

in der Abfrage qryUmsatzProJahr extrahiert aus dem Feld Bestelldatum die Jahreszahl und verwendet das Ergebnis zur Gruppierung der summierten Umsatzzahlen (s. Abb. 2). Da die Funktion Jahr() (engl.: Year() die Jahreszahl als vierstelliges Ergebnis liefert, ist der Einsatz dieser Funktion in Ihren Anwendungen auch hinsichtlich der Jahrtausendwende problemlos. Die Funktionen Monat() (engl.: Month()) und Tag() (engl.: Day()) haben die gleiche Syntax und liefern den Monat (1-12) beziehungsweise den Tag (1-31) des übergebenen Datums als Ganzzahl zurück.

**Zeitteile extrahieren**

Auch aus dem Zeitteil eines Datumswertes können die einzelnen Komponenten ermittelt werden. So ermitteln Sie beispielsweise bei der Abrechnung von Telefongesprächen mit der Formel

Minutenpreis: Wenn(Stunde([Gesprächs-
beginn]) Zwischen 8 Und 17;0,39;0,19)

den Minutenpreis für ein Gespräch abhängig von der Tageszeit. Die Funktion Stunde() (engl.: Hour()) liefert einen Integerwert zwischen 0 und 23, die Funktionen Minute() (engl.: Minute()) und Sekunde() (engl.: Second()) einen Integerwert zwischen 0 und 59 zurück. Die Speicherung genauerer Zeiten, zum Beispiel von Zehntelsekunden, ist mit dem Datentyp Datum/Uhrzeit nicht möglich.

**Datums- und Zeitintervalle
extrahieren**

Zum Filtern von Datumsteilen aus einem Datum/Uhrzeit-Wert können Sie auch die Funktion DatTeil() (engl.: DatePart()) einsetzen. Diese Funktion ist wesentlich flexibler als die bereits vorgestellten Einzelfunktionen, da sich damit auch Informationen ermitteln lassen, die nicht offensichtlich aus dem Datum hervorgehen, wie beispielsweise die Kalenderwoche.

DatTeil(Intervall;Datum[;firstdayofweek
[;firstweekofyear]])

Die Funktion DatTeil() erwartet neben dem zu überprüfenden Datum das Argument Intervall, mit dem Sie genau spezifizieren, welche Datumsinformation DatTeil() zurückgeben soll. Für das Argument Intervall sind die Einstellungen aus Tab. 1 möglich.

Zwei weitere optionale Argumente bestimmen die Berechnung des Wochentages und der Kalenderwoche. Mit firstdayofweek bestimmen Sie, für welchen Wochentag der Wert 1 zurückgegeben wird (s. Tab. 2). Die Standardeinstellung ist 1 (Sonntag), oft ist aber die Einstellung 2 (Montag) einfacher zu verarbeiten, da sich dann die Wochenendtage Samstag und Sonntag beispielsweise mit folgendem Ausdruck leicht herausfiltern lassen (s. Abb. 3).

Wenn(DatTeil("w";Datum();2)>5;
"Wochenende";"Maloche")

|  |  |  |
| --- | --- | --- |
| **Wert (engl.)** | **Wert (deutsch)** | **Beschreibung** |
| yyyy | jjjj | Jahr |
| q | q | Quartal |
| m | m | Monat |
| y | j | Tag des Jahres, durchgehend gezählt ab dem 1. Januar |
| d | t | Tag |
| w | w | Wochentag |
| ww | ww | Kalenderwoche |
| h | h | Stunde |
| n | n | Minute |
| s | s | Sekunde |

**Tab. 1: Die möglichen Einstellungen für das Argument Intervall. Diese Tabelle gilt für alle Funktionen, bei denen das Argument verwendet wird.**

|  |  |
| --- | --- |
| vbUseSystem | 0 |
| vbSunday | 1 |
| vbMonday | 2 |
| vbTuesday | 3 |
| vbWednesday | 4 |
| vbThursday | 5 |
| vbFriday | 6 |
| vbSaturday | 7 |

**Tab. 2: Die möglichen Einstellungen für das Argument firstdayofweek. Diese Tabelle gilt für alle Funktionen, bei denen das Argument verwendet wird.**

|  |  |
| --- | --- |
| **Konstante** | **Wert** |
| vbUseSystem | 0 |
| vbFirstJan1 | 1 |
| vbFirstFourDays | 2 |
| VbFirstFullWeek | 3 |

**Tab. 3: Die möglichen Einstellungen für das Argument firstweekofyear. Diese Tabelle gilt für alle Funktionen, bei denen das Argument verwendet wird.**

Das Argument firstweekofyear legt fest, wie die erste Kalenderwoche ermittelt wird (s. Tab. 3). Es ist daher nur dann erforderlich, wenn Sie als Intervall ww angegeben haben und eine von der Standardeinstellung abweichende erste Kalenderwoche benötigen. In der Standardeinstellung wird die Woche als KW1 festgelegt, in die der 1. Januar fällt.



**Abb. 3: Durch die Einstellung des Arguments firstdayofweek auf den Wert 2 wird der Montag als erster Tag der Woche definiert.**

**Wochentage ermitteln**

Die Funktion Wochentag() (engl.: Weekday()) liefert das gleiche Ergebnis wie der folgende Ausdruck, erwartet aber lediglich ein Datum als Argument:

DatTeil("w";Datum;firstdayofweek)

Optional kann auch hier der erste Tag der Woche festgelegt werden, um das Ergebnis den gegebenen Anforderungen anzupassen