Übersicht aller Filter-Funktionen - Bearbeiten der Filterbedingung

> ⚠ Die nachfolgend beschriebenen Funktionen können Sie auch anwenden in den "Formeln für Bedingungen" der Regeln.

Über NEU, KOPIEREN oder ÄNDERN wechseln Sie in die Erfassungsmaske für die Filterdefinition. Jeder Filterdefinition kann eine Bezeichnung, die Filterbedingung und eine Information hinterlegt werden.

Über die Schaltfläche

? Unknown Attachment

gelangen Sie in den bereits bekannten Feldeditor, in dem über Drag & Drop die Felder eingefügt werden können.
Als Eingabe wird eine Bedingung mit dem Ergebnis "**WAHR**" oder "**FALSCH**" bzw. "**True**" oder "**False**" bzw. "**Ja**" oder "**Nein**" erwartet.

- **Boolean-Variablen** (Werte vom Typ "BOOLEAN" sind die logischen Werte WAHR oder FALSCH) können auch ohne Vergleich ausgewertet, sofern es sich um Boolean-Felder handelt, die auch manuell bearbeitet werden können.
- **Boolean-Felder**, welche nicht manuell bearbeitet werden können, sind mit einer doppelten Verneinung auf Wahr abzufragen, z.B.: **not(not (Boolean-Feld))**

Möchten Sie z. B., dass in der Historyliste nur Auslandskunden angezeigt werden, dann tragen Sie die Variable «Ans.Adr.AuslKdKz» ein.
Da es sich hierbei um eine Boolean-Variable handelt, benötigen Sie keinen Vergleichsoperator und keinen Vergleichswert.

Inhalt

- Folgende Operatoren können verwendet werden:
  - 1. Vergleichs-Operatoren:
  - 2. Verneinungs-Operator (Negation):
  - 3. Mengen-Operator:
- Folgende Funktionen können in den Filter- bzw. in den Regelbedingungen genutzt werden:
  - Text / String-Funktionen:
  - Date / Time-Funktionen:
  - Boolean (Kennzeichen)-Funktion:
  - Integer / Float (Zahlen / Fließkommazahl)-Funktionen:
  - Sonder-Funktion:
  - DBInfo-Funktion:

Folgende Operatoren können verwendet werden:

1. Vergleichs-Operatoren:

- **and**:
  - **(a) and (b)**
    - a und b müssen Wahr sein
    - («SollBetLW1» = «HabenBetLW1») and («SollBetLW2» = «HabenBetLW2»)
- **or**:
  - **(a) or (b)**
    - a oder b muss Wahr sein
    - («SollBetLW1» = «HabenBetLW1») or («SollBetLW2» = «HabenBetLW2»)
- **<>:**
  - **(a) <> (b)**
    - Wenn a sich von b unterscheidet ist das Ergebnis Wahr
    - «AdrNr» <> '10000'

2. Verneinungs-Operator (Negation):

- **not:**
  - **not (a)**
    - a muss FALSCH sein, um als Ergebnis Wahr zu erhalten
  - **not («SollBetLW1» = «HabenBetLW1»)**
    - SollBetLW1 und HabenBetLW1 müssen sich unterscheiden, um als Ergebnis Wahr zu erhalten

3. Mengen-Operator:

- **in:**
  - **(a in (b))**

- **(Wert in (Werte-Menge))**
  - Wenn a (Wert) in b (Werte-Menge) vorliegt ist das Ergebnis Wahr
  - **«AdrNr» in ('10000', '10001')**
    - Dies ist eine absolute Definition.
    - Es werden nur durch Komma Werte gefunden (durch Komma separierte Werte müssen nach jedem Komma ein Leerzeichen besitzen, damit diese ausgewertet werden können).
    - Ein Bereich kann nicht mit dem Operator "in" abgefragt werden.

> **Tipp:**
>
> Wollen Sie einen "**von - bis**" Bereich, so können Sie folgende Bedingung definieren:
>
> **Pos('100',«AdrNr»)>0**
>
> Ein mit dieser Formel definierter Filter zeigt alle Adressen an, die '100' enthalten. (Vgl. Sie auch mit der Beschreibung der Funktion **Pos()** weiter unten).

Folgende Funktionen können in den Filter- bzw. in den Regelbedingungen genutzt werden:

Text / String-Funktionen:

- **ToString([Wert])**
  - Wandelt den Wert in einen String um
- **Left([String],[Anzahl der Zeichen])**
  - **Left(a,n)**
    - Die ersten n Zeichen von dem String werden zurückgegeben
  - **Left('TEST',2)**
    - Die ersten 2 Zeichen von dem String: "TEST" werden zurückgegeben. Ergebnis: "TE"
  - **Left(«AdrNr»,3) = '100'**
    - Die ersten 3 Zeichen von der String-Variable: "AdrNr (Adressnummer)" werden mit dem Wert: "100" verglichen. Das Ergebnis ist positiv, sofern die AdrNr (Adressnummer) mit einer 100 beginnt.
- **Pos([Such-String],[String])**
  - Gibt die Position von dem gesuchten String innerhalb eines Strings zurück.
  - **Pos("c","abc")**
    - Sucht "c" innerhalb von "abc" und gibt die Position zurück. Das Ergebnis ist: "3"

Date / Time-Funktionen:

- **Date([Wert]) oder ToDate([Wert])**
  - Wandelt den Wert in ein Datum um
  - **Date('01.01.80')**
    - Der Wert "01.01.80" kann mit einem Datumsfeld verglichen werden
- **Time([Wert]) oder ToTime([Wert])**
  - Wandelt den Wert in eine Uhrzeit um
  - **Time('08:30:00')**
    - Der Wert "08:30:00" kann mit einen Zeitfeld verglichen werden
- **DateTime([Wert]) oder ToDateTime([Wert])**
  - Wandelt den Wert in ein Datum mit Uhrzeit um
  - **DateTime('01.01.80 08:30:00')**
    - Der Wert "01.01.80 08:30:00" kann mit einen DatumZeitfeld verglichen werden
- **Year([Date oder DateTime])**
  - Die Jahreszahl wird zurückgeliefert
- **Month([Date oder DateTime])**
  - Der Monat wird zurückgeliefert
- **Day([Date oder DateTime])**
  - Der Tag innerhalb des Monats wird zurückgeliefert
- **Hour([Date oder DateTime])**
  - Die Stunde wird zurückgeliefert
- **Minute([Date oder DateTime])**
  - Die Minute wird zurückgeliefert
- **Second([Date oder DateTime])**
  - Die Sekunde wird zurückgeliefert
- **GetDate()**
  - Das aktuelle Systemdatum wird zurückgeliefert
- **GetAktDate()**
  - Das aktuelle Programmdatum wird zurückgeliefert

Boolean (Kennzeichen)-Funktion:

- **ToBool([Wert])**
  - Wandelt den Wert in ein Boolean (Ja/Nein) um

Integer / Float (Zahlen / Fließkommazahl)-Funktionen:

- **ToInt([Wert])**
  - Wandelt den Wert in eine Ganzzahl um
- **ToFloat([Wert])**
  - Wandelt den Wert in eine Fließkommazahl um

Sonder-Funktion:

- **CalcUmsatz([Anzahl der letzten Monate])**
  - Berechnet den Umsatz der letzten angegebenen Monate

DBInfo-Funktion:

- **DBInfo('Name der Datentabelle', Schlüsselwert[e],'Zielfeldname')**
  - Findet in der Datentabelle "**Name der Datentabelle**" den Datensatz mit dem entsprechenden "**Schlüsselwert**" und gibt den Inhalt des Feldes "**Zielfeldname**" zurück.
  - Mit Hilfe der DB-Info-Formeln kann ein Wert aus einer Datentabelle unter Verwendung eines Primärindex abgefragt werden.
    - **Erläuterung zum Aufbau:**
      - **Name der Datentabelle**
        - Name der Tabelle, aus der ein Wert abgefragt werden soll.
      - **Schlüsselwert[e]**
        - Kann beliebigen Typs sein. Sollte jedoch zum 1. Feld des Primärindex der abzufragenden Tabelle passen.
      - **Zielfeldname**
        - Optionaler Name des Feldes, dessen Wert zurück geliefert werden soll. Falls dieser Parameter ausgelassen wird, wird dieselbe Zeichenfolge zurückgegeben mit der der Datensatz z. B. beim Löschen bezeichnet wird.
  - **Beispiel:**
    - **DBInfo('Adressen',«Vog.AdrNr»,'SteuNr')<0**
      - Dadurch wird z.B. in einer Vorgangsliste das Feld Steuernummer der Adresse abgefragt. Ist das Feld Steuernummer leer, dann werden die entsprechenden Vorgänge ausgegeben.

> **Tipp**
>
> **Um in einer Filterdefinition ein Datumsfeld mit einem exakten Tag zu vergleichen, gehen Sie wie folgt vor:**
>
> **Left(ToString(ErstDat),10) = '01.01.2003'**
>
> Diese Left-Abfrage ist nötig, da dieses Datumsfeld "ErstDat" folgend aufgebaut ist:
>
> **"TT.MM.JJJJ HH:MM:SS" z.B. "01.01.2003 12:56:43"**
>
> Würde nur «ErstDat» = '01.01.2003' eingegeben werden, wird das Programm immer Melden, dass keine Datensätze zur Verfügung stehen. Dies ist korrekt, da dieses Feld mehr als nur das Datum beinhaltet.

> **Tipp**
>
> **Wenn in einer Filterdefinition ab einem gewissen Datum gefiltert werden soll, gehen Sie wie folgt vor:**
>
> **ToDate(Left(ToString(ErstDat),10)) >= ToDate('01.07.2003')**
>
> **Weitere Beispiele:**
>
> - ToDate(Left(ToString(«Vog.AendDat»),10)) >= ToDate('03.11.2003')
> - ToDate(Left(ToString(DBInfo('Adressen', «Vog.AdrNr» , 'AendDat')),10)) <= GetDate()
> - ToDate(Left(ToString(DBInfo('Adressen', «Vog.AdrNr» , 'AendDat')),10)) <= ToDate('01.01.2003')