Dynamische Ordner

[Stand: 25.03.2019 | Programmversion: 12.00.000]

Dieses Dokument beschreibt den Aufbau und die Funktionsweise von dynamischen Ordnern in den ELO Clients.

Inhalt

1 Was sind dynamische Ordner? 2
1.1 Dynamische Ordner im ELO Java Client 3
1.2 Dynamische Ordner im ELO Web Client 3
2 Einrichtung eines dynamischen Ordners 4
2.1 !+<Kommando> 5
2.2 Abfrage über mehrere Tabellen 5
2.2.1 Beispiel 1 5
2.2.2 Beispiel 2 6
2.3 !!<Kommando> 7
2.4 !*<Kommando> 7
3 Weitere Anmerkungen 8
3.1 Performance 8
3.2 Verfügbare Spalten 8
3.3 Besonderheiten unter Oracle SQL 9
3.3.1 Beispiel 9
4 Anwendungsbeispiele 10
4.1 Neue Dokumente im Archiv der letzten 30 Tage anzeigen 10
4.2 Weitere Anwendungsbeispiele 11

1 Was sind dynamische Ordner?

Dynamische Ordner sind Ordner im ELO Archiv, deren Inhalt dynamisch erzeugt wird. Im Prinzip wird das Ergebnis einer Suche dargestellt. Diese Suche wird über eine spezielle SQL-Abfrage ausgeführt. Dabei können Sie unterschiedliche Merkmale (wie beispielsweise bestimmte Verschlagwortungsinformationen) als Suchkriterien verwenden. Jedes Mal, wenn sich im Archiv etwas ändert, was zur jeweiligen Abfrage passt, ändert sich auch der Inhalt des dynamischen Ordners.

Beachten Sie: Wenn Sie dynamische Ordner erzeugen wollen, testen Sie diese immer erst in einem kleinen Testarchiv. Ein falsch formuliertes Kommando kann Trefferlisten im Umfang "Gesamtzahl aller Dokumente" * "Gesamtzahl aller Indexzeilen aller Dokumente" erzeugen. Im Testarchiv haben Sie dann einige tausend Treffer, im Produktivarchiv schnell viele Millionen.

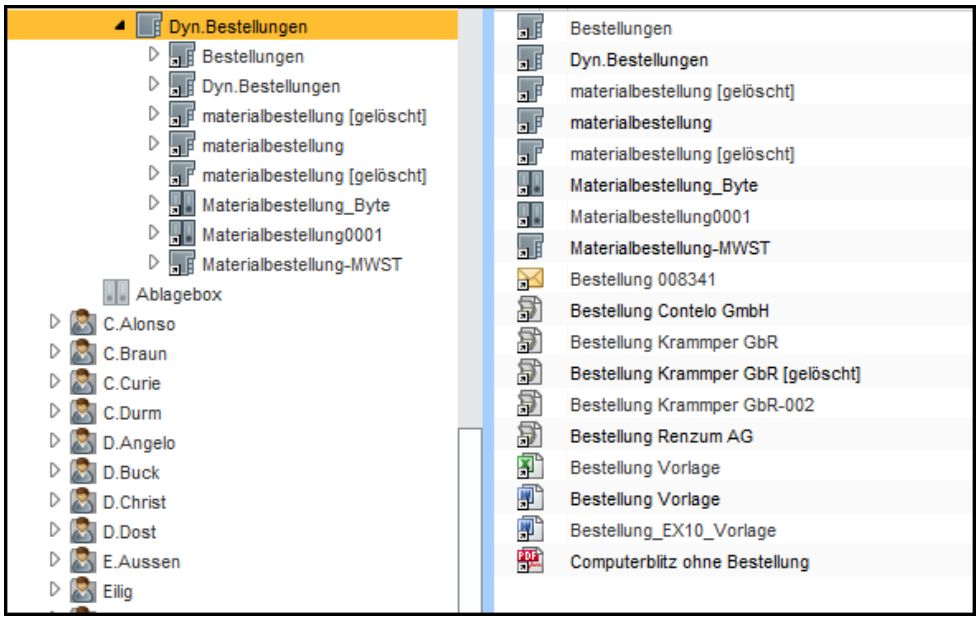


Abb. 1: Inhalt eines dynamischen Ordners, der nach 'bestellung' sucht

Beachten Sie: Falls Sie einen dynamischen Ordner in einem Produktivarchiv einsetzen wollen, sollen Sie sich auf jeden Fall Gedanken über die Performance machen. Insbesondere wenn es einen regelmäßigen Zugriff auf diesen Ordner geben soll, ist es notwendig, dass der Zugriff keine Full-Table-Scans auslöst. Hierzu sollten Sie das erzeugte SQL-Statement analysieren (Bspw. im SQL Server Management Studio).

1.1 Dynamische Ordner im ELO Java Client

Der ELO Java Client bietet die Möglichkeit eine Suchanfrage als dynamischen Ordner abzuspeichern.

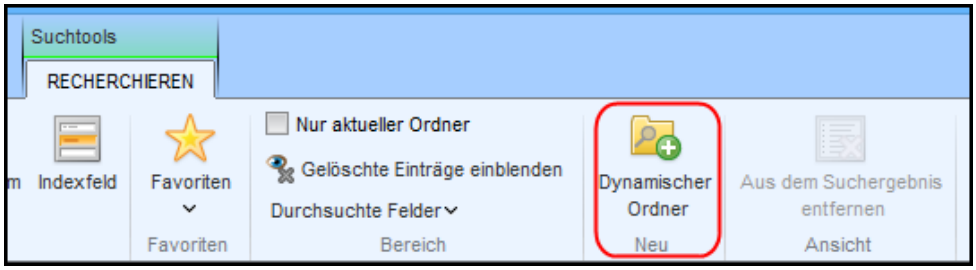


Abb. 2: Funktion 'Dynamische Ordner' im ELO Java Client

Die Funktion finden Sie unter *Multifunktionsleiste* > *Recherchieren* > *Dynamischer Ordner*. Auf diese Funktion geht dieses Dokument nicht näher ein. Weitere Informationen finden Sie im ELO Java Client-Handbuch.

1.2 Dynamische Ordner im ELO Web Client

Der ELO Web Client bietet ebenfalls die Möglichkeit eine Suchanfrage als dynamischen Ordner abzuspeichern.

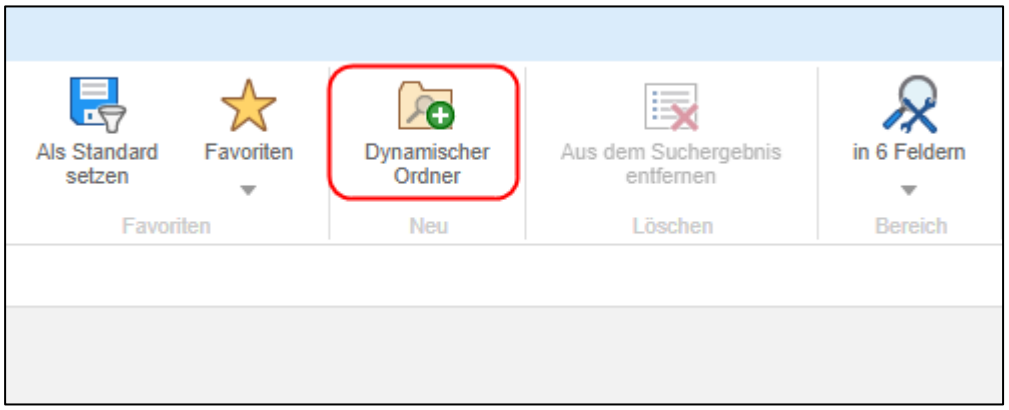


Abb. 3: Funktion 'Dynamischer Order' im ELO Web Client

Die Funktion finden Sie unter *Multifunktionsleiste* > *Suche* > *Dynamischer Ordner*. Auf diese Funktion geht dieses Dokument nicht näher ein. Weitere Informationen finden Sie im ELO Web Client-Handbuch.

2 Einrichtung eines dynamischen Ordners

Ein dynamischer Ordner wird durch einen speziellen Eintrag im Zusatztext definiert. Dieser Text muss ganz am Anfang des Feldes stehen und es darf auch kein weiterer Text vorhanden sein. Damit die Definition nicht irrtümlich verändert werden kann, sollte der Ordner nur für Administratoren im Schreibzugriff vorliegen, für alle anderen reicht ein Lesezugriff. Durch diese Einschränkung können Sie auch vermeiden, dass dort manuell Untereinträge angelegt werden, da diese nicht angezeigt würden und somit „verschwunden“ wären.

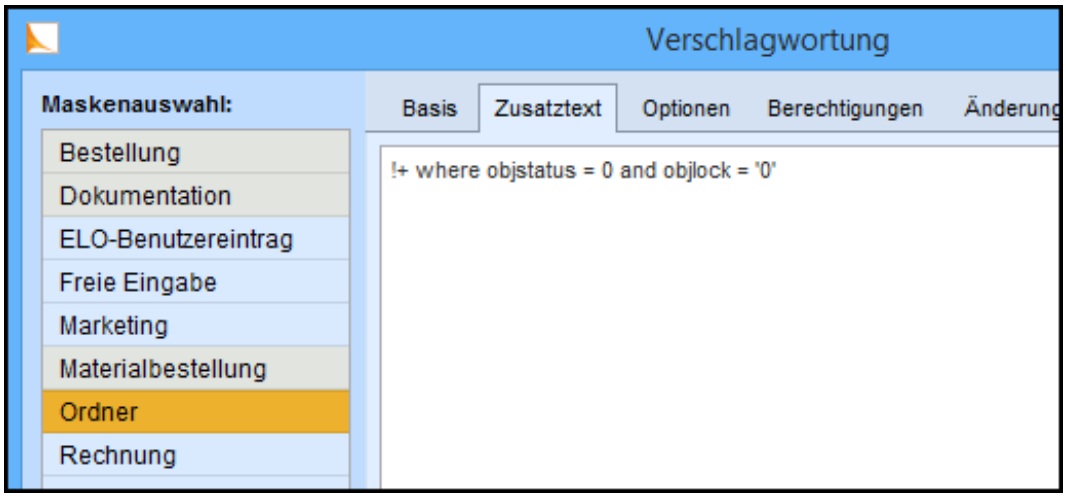


Abb. 4: Beispiel eines Statements für einen dynamischen Ordner

2.1 !+<Kommando>

Das Kommando *!+* ersetzt das SQL-Statement `SELECT * FROM` für die Datenbanktabelle *dbo.objekte*. Sie können nach dem Kommando *!+* den kompletten WHERE-Anteil der Abfrage festlegen.

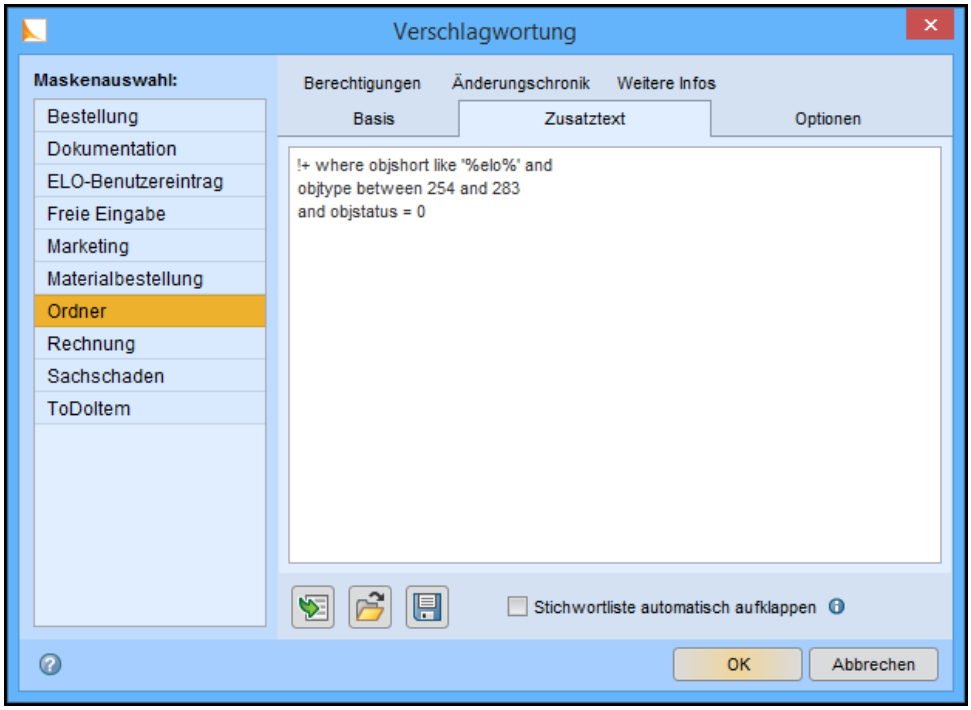


Abb. 5: Beispiel für ein SQL-Statement mit dem Kommando '!+'

Achten Sie bei der Formulierung Ihrer WHERE-Klausel darauf, dass Sie eine Einschränkung auf die gewünschten Objekttypen vornehmen (falls diese nicht ohnehin durch die weitere Sucheinschränkung festgelegt wird). Gelöschte Einträge, also alle Einträge mit einem objstatus ungleich 0, sollten Sie ausschließen.

2.2 Abfrage über mehrere Tabellen

Zusätzlich zur Objekte-Tabelle können Sie weitere Tabellen in die Abfrage aufnehmen. Dafür müssen Sie die weiteren Tabellen mit führendem Komma eintragen.

2.2.1 Beispiel 1

In diesem Beispiel wird die Abfrage auf die Verschlagwortungsmaske *EMail* (hier durch `objmask = 2`) eingeschränkt. Weiterhin wird nur nach Dokumenten gesucht, die im Indexfeld *Von* (ELOOUTL1) einen bestimmten Namen (hier: *Sorglos*) enthalten.

```
!+ , objkeys WHERE objid = parentid
AND objmask = 2
AND (okeyname LIKE 'ELOOUTL1' AND okeydata LIKE '%sorglos')
```

2.2.2 Beispiel 2

Das folgende Beispiel führt eine Abfrage durch, in der nach zwei unterschiedlichen Indexzeilen sortiert werden soll. Aus diesem Grund muss die Indextabelle objkeys auch doppelt aufgenommen werden.

```
!+ , objkeys ok1, objkeys ok2 WHERE
objid=ok1.parentid AND objid=ok2.parentid AND
ok1.okeyno=0 AND ok2.okeyno=1 AND
objtype=254 AND objmask=5 AND
ok1.okeydata LIKE 'p1%'
ORDER BY ok2.okeydata, ok1.okeydata
```

Erläuterung zu den einzelnen Zeilen:

Die Tabelle *objkeys* wird doppelt aufgenommen unter den Namen *ok1* und *ok2*. Beachten Sie das führende Komma. Es ist unbedingt notwendig. Wenn Sie es vergessen, erzeugen Sie auf der SQL-Ebene einen Syntaxfehler. Der Zielordner bleibt dann leer.

Die Basisdatentabelle *objekte* wird über den eindeutigen Schlüssel *objid* mit den beiden Indextabellen verbunden.

Es sollen nicht beliebige Zeilen, sondern zwei ganz bestimmte Indexzeilen betrachtet werden (die internen Zeilennummern *0* und *1* entsprechen der ersten und zweiten Indexzeile)

Es sollen nur Dokumente vom Maskentyp *5* gefunden werden, die im ersten Indexfeld mit dem Wert *p1* anfangen.

Die Dokumente werden dann nach dem zweiten Indexfeld sortiert. Alle identischen Einträge werden nach der ersten Indexzeile sortiert.



Beachten Sie: Das Beispiel sucht nach irgendeinem Indexfeld, dessen Inhalt mit *p1* beginnt, d.h. es wird auch dann ein Treffer erzeugt, wenn ein *p1* in einem unsichtbaren Indexfeld steht. Der Benutzer wundert sich dann, da er kein passendes Indexfeld sieht. In einem Produktivsystem sollten Sie das Indexfeld konkret benennen.



Beachten Sie: Der Teil `objid=parentid` ist in jedem Fall notwendig. Dieser verbindet die Basisdatentabelle mit der Indexzeilentabelle. Wenn Sie diesen weglassen, bekommen Sie eine riesige Treffertabelle mit lauter unsinnigen Einträgen.

2.3 !!<Kommando>

Dies ist eine veraltete Form für die dynamischen Ordner und wird von aktuellen ELO Versionen nicht mehr unterstützt.

2.4 !*<Kommando>

Dies ist eine veraltete Form für die dynamischen Ordner und wird von aktuellen ELO Versionen nicht mehr unterstützt.

3 Weitere Anmerkungen

3.1 Performance

Beachten Sie bitte, dass jedes Aufblättern eines dynamischen Ordners eine Suche initiiert. Bei kleinen Datenbanken (<50.000 Einträge) muss man sich wenig Gedanken um Performance Gesichtspunkte machen, hier optimiert der SQL-Server alle Probleme weg.

Sobald die Datenbank aber recht groß wird, kann man durch eine unglückliche Suchanfrage eine massive Datenbanklast erzeugen. Das führt dann auf allen Clients zu einer geringeren Systemleistung. Hier sollte man folgende Punkte beachten:

Steht dem Selektionskriterium ein geeigneter Index zur Verfügung? Wenn nicht, kann einer angelegt werden? Wenn beide Fragen mit "Nein" beantwortet werden, dann sollten Sie auf den dynamischen Ordner verzichten. Ein Full-Table-Scan auf einer großen Datenbank kann durchaus eine halbe Stunde dauern, solange wird kein Benutzer warten wollen.

Verwendet der SQL-Server den Index überhaupt? Manchmal hat der Optimizer eine andere Vorstellung davon, wie eine Anfrage abgearbeitet werden soll als der Administrator. Wenn der SQL-Server einen ungünstigen Index auswählt, dann kann das auch zu unerwartet schlechten Antwortzeiten führen.

Bleibt die Größe der Trefferliste in einem vernünftigen Rahmen? Wenn Sie alle Dokumente aus einem Monat sammeln, dann mag das auf dem Testsystem noch gehen. Wenn dann aber im Produktivsystem 50.000 Dokumente in einem Ordner angezeigt werden sollen, führt das zu schlechten Ergebnissen.

3.2 Verfügbare Spalten

Unter den Basisdaten stehen Ihnen folgende Spalten zur Verfügung:

Spaltenname	Inhalt
objtype	Art des Eintrags, Level1=1, Level2=2 ... Ordner=253, Dokument=254
objshort	Kurzbezeichnung
objidate	Ablagedatum im numerischen Format (Anzahl der Minuten seit dem 31.12.1899)
objxdate	Dokumentendatum im numerischen Format
objkind	Farbe
objmask	Dokumententyp

objuser	Ersteller des Dokuments
objstatus	0: nicht gelöscht, alle anderen Werte kennzeichnen gelöschte Einträge.
objdeldate	Verfallsdatum in numerischer Form

Die Tabelle *objkeys* besitzt folgende Einträge:

Spaltenname	Inhalt
parentid	Interne eindeutige ELO Nummer des Eintrags, mit *objid* aus der Basistabelle *objekte* verbunden.
okeyno	Nummer der Indexfelder, beginnend mit 0. Ab dem Indexfeld 50 gibt es eine Reihe von unsichtbaren Indexfeldern.
okeyname	Gruppenname des Indexfeldes. Wenn Sie Dokumententypunabhängig suchen wollen, dann sollten Sie den Gruppenname statt der Feldnummer zur Selektion verwenden.
okeydata	Inhalt des Indexfeldes.
okeyudata	(nur unter Oracle) Inhalt des Indexfeldes in Großschreibweise

3.3 Besonderheiten unter Oracle SQL

Unter Oracle SQL gibt es eine Reihe von Besonderheiten, die Sie beachten müssen. Ansonsten erhalten Sie unvollständige Suchergebnisse oder sogar Syntaxfehler.

Oracle SQL unterscheidet zwischen Groß- und Kleinschreibweise. Wenn Sie dort nach "ELO" suchen und in der Datenbank steht "Elo", dann wird Oracle den Eintrag nicht finden. Bei der Suche in Indexzeilen steht Ihnen deshalb das Feld *okeyudata* zur Verfügung.

Die Tabellennamen müssen den Archivnamen vorangestellt bekommen, die beiden Teile werden durch einen Punkt getrennt.

3.3.1 Beispiel

In der Anfrage muss `okeydata ok` unter Oracle SQL als `archiv1.okeydata ok1` eingetragen werden.

4 Anwendungsbeispiele

4.1 Neue Dokumente im Archiv der letzten 30 Tage anzeigen

Tragen Sie die SQL-Abfrage wie gewohnt im Zusatztextfeld des Ordners ein, der als dynamischer Ordner konfiguriert werden soll.

```
!+ WHERE objtype>=254 AND objstatus=0 AND DATEADD(mi, objidate, '18991230')
>= DATEADD(day, -30, SYSUTCDATETIME())
```

Die einzelnen Bestandteile der Abfrage:

- **objtype >=254**: Beschränkt die Abfrage auf Dokumente.
- **objstatus=0**: Beschränkt die Abfrage auf Dokumente, die nicht gelöscht sind.
- **DATEADD(mi, objidate, '18991230')**: Sucht nach dem Ablagedatum (objidate) in Minuten (mi) seit dem Referenzdatum (30.12.1899).

Information: Hier wurde das ISO-Datum gewählt, um mögliche Umrechnungsprobleme zu umgehen. Meist funktioniert auch die Angabe im Format TT.MM.JJJJ.

- **>=**: Die Operatoren grenzen die Suche ein auf Dokumente deren Ablagedatum größer (=jünger) oder gleich dem aktuellen Datum -30 Tage ist.
- **DATEADD(day, -30, SYSUTCDATETIME())**: Hier wird das Systemdatum im UTC-Format (SYSUTCDATETIME()) in Tagen (day) ausgelesen. Davon werden 30 Tage abgezogen.

Information: Bis zur SQL-Server-Version 2005 war die Funktion *SYSUTCDATETIME()* nicht verfügbar. Stattdessen muss die Funktion *GEDATE()* verwendet werden.

4.2 Weitere Anwendungsbeispiele

Aufgabe	Eintrag im *Zusatztext*
Alle Dokumente mit der Verschlagwortungsmaske *Freie Eingabe*. Die Ergebnisliste wird absteigend (DESC für „descending“) nach dem Dokumentendatum (objxdate) sortiert.	`!+ WHERE objmask=0 AND objtype>=254 AND objstatus=0 ORDER BY objxdate DESC`
Alle Ordner mit der Verschlagwortungsmaske *Freie Eingabe*. bsteigend nach dem Dokumentendatum sortiert.	`!+ WHERE objmask=0 AND objtype<254 AND objstatus=0 ORDER BY objxdate DESC`
Alle Objekte mit einer bestimmten Farbe (objkind).	`!+ WHERE objkind = 12 ORDER BY objxdate DESC`
Dokumente eines bestimmten Dokumentenpfades mit einem Ablagedatum innerhalb eines bestimmten Zeitraums A-B (in Minuten seit dem 30.13. 1899).	`!+ WHERE objpath =3 AND objidate BETWEEN 60587305 AND 60587308`
Alle Objekte, die in der Kurzbezeichnung den Text "rechnung" enthalten.	`!+ WHERE objshort LIKE '%rechnung%'`
Alle Dokumente mit "ELO“ und “xc" in der Kurzbezeichnung (absteigend sortiert nach Ablagedatum).	`!+ WHERE objtype>=254 AND objshort LIKE '%ELO%' AND objshort LIKE '%xc%' AND objstatus=0 ORDER BY objidate DESC`
Achtung: Alle Objekte mit "Jeder" Vollzugriff – dieser dynamische Ordner sollte im Idealfall immer leer sein.	`!+ WHERE objacl='75PYJA' AND objstatus=0`
Alle ausgecheckten/gesperrten Ordner und Dokumente anzeigen (sortiert nach dem Bearbeiter).	`!+ WHERE (objlock <> - 1) ORDER BY objuser`
Gesamten Inhalt der Chaosablage im Archiv suchen (für die Weiterverarbeitung, z. B. auf die Suchansicht legen und dann in das Archiv überführen)	`!+WHERE objparent = 0`