

SQL und Reguläre Ausdrücke (Regular Expressions) - Teil 1

Überblick

Mit Hilfe von skalaren Funktionen und dem Prädikat LIKE können alphanumerische Strings mit SQL gescannt werden. Sofern jedoch Texte nach komplexeren Regeln, z.B. 3 Buchstaben und 2 Zahlen jeweils zwischen 3 und 5, durchsucht werden müssen, kommt die zuvor erwähnte Technik schnell an ihre Grenzen. In vielen Programmiersprachen werden für solche Textscans Reguläre Ausdrücke (Regular Expressions) verwendet. Bei den regulären Ausdrücken handelt es sich um eine standardisierte Syntax mit der komplexe Suchkriterien beschrieben werden können. In Release 7.2 TR2 wurde die DB2 for i um einige skalare Funktionen und Prädikate erweitert, die die Verarbeitung von regulären Ausdrücken erlauben. In diesem Artikel werden die neuen Funktionen kurz vorgestellt und ein kurzer Überblick über die Syntax, die zur Definition von regulären Ausdrücken verwendet wird, gegeben.

Bevor wir uns jedoch mit den neuen Funktionen, die in der DB2 for i implementiert wurden beschäftigen, zunächst einige Grundlagen:

Was sind Reguläre Ausdrücke?

Unter regulären Ausdrücken (Regular Expressions, REGEXP, REGEX) versteht man verallgemeinerte Suchmuster, mit deren Hilfe Texte gescannt und gefiltert werden können.

Die Implementierung der regulären Ausdrücke in der DB2 for i basiert auf den International Components for Unicode (ICU) APIs (Application Programming Interfaces). ICU stellt für die Verarbeitung von Unicode-Texten eine Reihe von C/C++ und Java Bibliotheken bereit. Diese Funktionen und Methoden können auf annähernd jeder Plattform, auf der C/C++ und/oder JAVA verwendet werden können, eingesetzt werden, also auch auf der IBM i. Die ICU APIs wurden als Open Source für den Einsatz sowohl in freier, open source als auch kommerzieller Software herausgegeben.

Da diese APIs für den Einsatz in SQL auf der DB2 for i in neuen skalaren Funktionen und Prädikaten implementiert wurden, gelten für die regulären Ausdrücke die gleichen Regeln, wie für C++ oder JAVA. Beispiele von regulären Ausdrücken (z.B. zum Prüfen von IP-Adressen), die im Internet nur für C++, JAVA, Javascript, PHP etc. gefunden werden können, können in den SQL-Funktionen und Prädikaten verwendet werden.

Funktionen und Prädikate in DB2 for i

Für die Verwendung von Regulären Ausdrücken in Verbindung mit SQL wurden in der DB2 for i, die folgenden Prädikate und Funktionen integriert:

- **REGEXP_LIKE** Bei REGEXP_LIKE handelt es sich um ein Prädikat, das ähnlich wie das Prädikat LIKE in den WHERE-Bedingungen angegeben werden muss.
Der reguläre Ausdruck, nach dem eine alphanumerische Spalte sowohl im Single Byte Character Set als auch im Double Byte Character Set mit fixer oder variabler Länge durchsucht werden soll, muss als zweiter Parameter angegeben werden.
Sowohl der zu durchsuchende Text, als auch der reguläre Ausdruck werden implizit in einen DBCLOB (Double Byte Character Large Object)-String mit UTF-16 CCSID (Character Set Id) konvertiert.
- **REGEXP_COUNT** Mit Hilfe der skalaren Funktion REGEXP_COUNT kann ermittelt werden wie häufig eine Zeichenkombination, die dem regulären Ausdruck entspricht, innerhalb eines Strings vorkommt.
Die Funktion gibt die Anzahl der Vorkommen zurück

- **REGEXP_INSTR** Mit der skalaren Funktion REGEXP_INSTR kann die Position des ersten oder auch eines beliebigen Vorkommens einer Zeichenkombination, die dem regulären Ausdrucks entspricht, ermittelt werden.
Die Funktion gibt die Position des Vorkommens innerhalb des Strings zurück. Sofern kein Vorkommen gefunden werden kann wird der Wert 0 zurückgegeben.
- **REGEXP_SUBSTR** Mit der skalaren Funktion REGEX_SUBSTR kann ein Text-Bereich, der dem regulären Ausdruck, sowie dem angegebenen Vorkommen entspricht, aus einem vorgegebenen Text extrahiert werden.
Die Funktion gibt den entsprechenden Text-Bereich aus.
- **REGEXP_REPLACE** Mit der skalaren Funktion REGEXP_REPLACE können Text-Bereiche, die dem angegebenen regulären Ausdruck entsprechen, durch einen vorgegebenen String ersetzt oder aus dem String entfernt werden.
Die Funktion gibt den korrigierten Text aus.

Mit Hilfe des Prädikats REGEXP_LIKE und den Skalaren REGEXP-Funktionen können beliebige Texte, sowohl im Single-Byte-Character Set als auch Double Byte Character Set gescannt werden. Sofern der Text oder der reguläre Ausdruck nicht als UTF-16 DBCLOB (Double Byte Large Object) angegeben wurden, erfolgt automatisch eine implizite Konvertierung in dieses Format.

Die komplexen Suchmuster, also die regulären Ausdrücke, müssen bei den neuen Funktionen und Prädikaten angegeben werden. Die Syntax für die regulären Ausdrücke unterliegt einem komplexen Regelwerk.

Reguläre Ausdrücke – Syntax

Zur Definition von regulären Ausdrücken werden Buchstaben, Zahlen, sowie eine Reihe von Sonderzeichen (Meta-Zeichen) verwendet. Mit Hilfe dieser Meta-Zeichen, deren Anordnung und Kombination mit anderen Zeichen, werden komplexe Suchmuster erstellt, die als Parameter in den skalaren Funktionen und Prädikaten angegeben werden können.

Bei der einfachsten Version der regulären Ausdrücke, können die Zeichen, nach denen ein Text gefiltert werden soll der Reihe nach aufgelistet werden. Wird der reguläre Ausdruck 'ABC' verwendet, wird ein Text nach der Zeichenfolge ABC (in Großbuchstaben) durchsucht. Das Suchergebnis entspricht einer Suche mit Hilfe des SQL Prädikates LIKE mit führendem und folgendem Prozent (→ LIKE '%ABC%').

Die meisten Zeichen können direkt in einem regulären Ausdruck angegeben werden. Einige Sonderzeichen (Metazeichen), haben jedoch eine besondere Bedeutung für die Definition der Suchstruktur. Sofern nach diesen Zeichen in einem Text gesucht werden soll, müssen diese Zeichen durch einen vorangestellten Backslash maskiert werden.

Die folgenden Meta-Zeichen werden zur Definition der regulären Ausdrücke verwendet.

- **** **Backslash:** Maskierung von Sonderzeichen mit besonderer Bedeutung.
- **.** **Punkt:** Einzelnes beliebiges Zeichen
ABC.EFG ABCDEFG ist ebenso gültig wie ABC?EFG oder XXABCXEFHG
- ***** **Stern:** Beliebige Anzahl des vorhergehenden Zeichens. Wird der Stern in Verbindung mit einem vorangestellten Punkt angegeben, bedeutet dies beliebig viele beliebige Zeichen.
A*B* Findet sowohl AB, als auch ABBBBB,
A.*B.* als auch AvielTextBandererText

- **?** **Fragezeichen:** Kein oder genau ein Vorkommen des vorangestellten Zeichens an der entsprechenden Position
123?456 Sowohl 123456 als auch 12456 werden gefunden
- **+** **Pluszeichen:** Der Text muss mindestens 1 Vorkommen des vorhergehenden Zeichens oder Elements enthalten
A+BCDE+ Sowohl ABCDE als auch ABCDEE als auch AAAABCDEEEE werden gefunden
- **[]** **Brackets / eckige Klammern:** Beschreiben eine Zeichenklasse.
Die eckigen Klammern umschließen eine Liste oder einen Bereich von Zeichen (oder beides). Nur ein Text, der alle Zeichen enthält oder nur aus einem Teil der angegebenen Zeichen besteht, wird gefunden.
Die Reihenfolge in der die Zeichen angegeben wurden spielt dabei keine Rolle.
[AaBbCc] Texte, die nur aus den Buchstaben A, B, C in Groß- oder Kleinschreibung bestehen werden gefunden.
A ist ebenso gültig wie cC, bBcc oder AAABBBCCC
Anstatt die Zeichen einzeln aufzulisten, ist es auch möglich Bereiche anzugeben. Dabei werden der untere und der obere Wert durch ein Minuszeichen getrennt in den eckigen Klammern angegeben, z.B. [A-Z].
[0-9] .Texte, die nur aus Ziffern bestehen werden gefunden.
0 und 9876 sind ebenso gültig wie 111111
Mehrere Bereiche können innerhalb der gleichen eckigen Klammer angegeben werden. Wenn der String z.B. nur aus Großbuchstaben von A bis F und Ziffern von 0-9 bestehen darf, kann dies in der Form [[A-F][0-9]] angegeben werden. Die Angabe der inneren Klammern ist jedoch nicht zwingend erforderlich, d.h. [A-F0-9] liefert das gleiche Ergebnis.
- **{ }** **Curly Brackets / Geschwungene Klammern:** Innerhalb der geschwungenen Klammern, wird die zulässige Anzahl des Vorkommens des vorangegangenen Zeichens oder Ausdrucks angegeben.
{n} String muss genau n Vorkommen enthalten
[ABC]{3} findet AAA und CBC, jedoch nicht AA
{n,} String muss mindestens n Vorkommen enthalten, darf jedoch soviel wie mögliche Vorkommen enthalten
[A-C]{2,}[X-Z]{3,} findet, AAZZZ, ABCABCXYZ und ABZZZZZZ
jedoch weder AZZZ noch ABCABCXZ
{n,m} String muss mindestens n und darf höchstens m Vorkommen enthalten.
[A-C]{2,3} findet ABZ, CC und CBBXYZ jedoch weder A noch BBBX
[A-Z]{3}[1-3]{4} Der reguläre Ausdruck muss aus 3 Buchstaben und 4 Ziffern zwischen 1 und 3 zusammengesetzt sein.
z.B. AAA1111 oder ZXY1313
- **()** **Parenthesis / Runde Klammern:** Mit Hilfe von runden Klammern können längere Ausdrücke zu einer Einheit zusammengefasst werden.

A(na)*s Findet sowohl Anas, als auch Ananas, als auch Ananananas, da durch die Angabe des Sterns beliebig viele Wiederholungen des Ausdrucks (na) erlaubt sind.

- | **Pipe / Senkrechter Strich:** Durch den senkrechten Strich wird eine ODER Verknüpfung von Alternativen definiert.

[Ää]|Ae|ae Findet alle Texte, die entweder Ä, ä, Ae oder ae beinhalten, z.B. Käse, Baer, Ärger, Aenderung

- ^ **Caret / Dach:** Wird das Zeichen am Anfang des regulären Ausdrucks angegeben, bedeutet dies, dass der gefundene String mit der Übereinstimmung beginnen muss.

^[0-3]{2}[A-C] String muss mit 2 Ziffern jeweils zwischen 0 und 3 beginnen. Bei dem 4. Zeichen muss es sich um einen Buchstaben zwischen A und C handeln.

Wird das Zeichen innerhalb einer Zeichengruppe angegeben, stellt dies die Negation dar, also alle Zeichen außer den in der Zeichengruppe angegebenen.

^[^0-9] Die gefundenen Texte enthalten keine Ziffern.

- \$ **Dollar-Zeichen:** Wird das Dollar-Kennzeichen am Ende des regulären Ausdrucks angegeben, muss der gefundene Text mit der Übereinstimmung enden.

[A-F]{3}\$ Die letzten 3 Zeichen des Strings müssen ein Buchstabe zwischen A und F sein.

123FFF wird ebenso gefunden, wie ABC oder XXXXXXEEEE.

Das Dach und das Dollarzeichen können innerhalb des gleichen Ausdrucks angegeben werden, wodurch die zulässige Länge des gefundenen Textes festgelegt werden kann.

^[1-3]{1,2}[A-F]{2}\$ Der gesuchte String muss aus einer oder zwei Ziffern zwischen 1 und 3 gefolgt von zwei Buchstaben zwischen A und F bestehen.

11AA wird ebenso gefunden wie 1FB.

Daneben gibt es einige vordefinierte Zeichenklassen, die direkt in dem regulären Ausdruck verwendet werden können, wodurch die Syntax des Ausdrucks etwas vereinfacht werden kann:

- **\d** String besteht nur aus Ziffern (0-9), entspricht [0-9]
- **\D** String enthält Zeichen, bei denen es sich nicht um Ziffern handelt, entspricht [^0-9]
- **\w** String besteht nur aus Buchstaben in Groß- oder Kleinschrift, also A-Z oder a-z, Ziffern (0-9) und/oder Unterstrich (Underscore _), entspricht [a-zA-Z0-9_]
- **\W** String enthält Zeichen bei denen es sich weder um Buchstaben, noch Ziffern, noch den Unterstrich (Underscore) handelt).
- **\s** Alle Whitespace Zeichen

Leider ist es im Rahmen dieses Artikels nur möglich einen kurzen Überblick über die Syntax von regulären Ausdrücken zu geben.

Im nächsten Artikel wird gezeigt, wie diese regulären Ausdrücke in Verbindung mit dem neuen Prädikat REGEXP_LIKE zur Text-Suche verwendet werden können.

Bis dahin schon mal viel Spaß bei der Konstruktion von regulären Ausdrücken.