**Регулярные выражения, усовершенствующие SQL-предложения**

**Регулярные выражения в Oracle Database 10g - это мощный инструмент для манипулирования текстовыми данными.**

Новая возможность Oracle Database 10g значительно увеличивает способность поиска и манипулирования символьными данными. Эта возможность, регулярные выражения, это транскрипция для описания образца текста. Она давно имеется во многих языках программирования и множестве утилит для UNIX.

Регулярные выражения в Oracle реализованы в виде SQL-функций и оператора выражения WHERE. Тем, кто еще ничего не знает о регулярных выражениях, эта статья может дать общее понятие об этой новой и весьма мощной, но пока кажущейся непонятной, возможности. Читатели, которые уже познакомились с регулярными выражениями, могут пополнить знания о том, как применить эту функциональность в контексте языка Oracle SQL.

**Что такое регулярное выражение?**

Регулярное выражение содержит один и более символов и/или метасимволов. В самом простом виде регулярное выражение может состоять только из символов, например, регулярное выражение cat. Оно читается как буква c, за которой следует буква a и t, и этому шаблону соответствуют такие строки, как cat, location и catalog. Метасимволы обеспечивают алгоритмы обработки в Oracle символов, из которых состоит регулярное выражение. Когда значение различных метасимволов будет понятным, вы увидите, как удобны регулярные выражения для выделения и замены каких-либо текстовых данных.

Проверка данных, поиск дубликатов, обнаружение лишних пробелов или разбор строки - это некоторые из многих примеров использования регулярных выражений. Их можно применять для проверки формата телефонного номера, zip-кода, email-адресов, номеров социального обеспечения, IP-адресов, имен файлов и директорий и так далее. Кроме того, можно искать комбинации, например, HTML-тегов, чисел, дат, и другое, которые соответствуют какому-либо шаблону в тексте, и заменять их другим набором символов.

**Использование регулярных выражений в Oracle Database 10g**

Чтобы воспользоваться возможностями регулярных выражений, можно применить функции REGEXP\_INSTR, REGEXP\_SUBSTR и REGEXP\_REPLACE и новый оператор Oracle SQL - REGEXP\_LIKE. Вы увидите, как эта новая функциональность поддерживает существующий оператор LIKE и функции INSTR, SUBSTR и REPLACE. Они действительно похожи на существующие оператор и функции, однако теперь предоставляются мощные возможности сопоставления с шаблоном. Искомые данные могут быть простой строкой или текстом большого объема, хранимым в символьном столбце базы данных. Регулярные выражения позволяют искать, заменять и проверять данные способом, о котором ранее и не мечтали, с высокой степенью гибкости.

**Примеры регулярных выражений**

Перед использованием новой функциональности необходимо понять значение некоторых метасимволов. Период (.) в регулярном выражении соответствует любому символу (за исключением перехода на новую строку). Например, регулярное выражение a.b соответствует строке, содержащей букву a, за которой следует один любой символ (за исключением перехода на новую строку), за которым следует буква b. Строки axb, xaybx и abba подходят, так как содержат этот шаблон. Если требуется точное соответствие трехсимвольной строке, в которой строка начинается с a и заканчивается b, регулярное выражение необходимо привязать. Метасимвол вставки (^) обозначает начало строки, а доллар ($) - конец строки (см. Таблицу 1). Поэтому регулярному выражению ^a.b$ соответствуют строки aab, abb или axb. Чтобы сравнить этот метод со знакомым сопоставлением с шаблоном оператора LIKE, можно использовать такой шаблон как a\_b, где подчеркивание (\_) - это любой одиночный символ.

По умолчанию отдельный символ или список символов регулярного выражения сопоставляются один раз. Чтобы найти несколько вхождений символа регулярного выражения, применяется квантификатор, называемый также оператором повтора. Если требуется соответствие строке, которая начинается с буквы a и заканчивается буквой b, регулярное выражение выглядит следующим образом: ^a.\*b$. Метасимвол \* повторяет предыдущее соответствие любому метасимволу (.) ноль, один и более раз. Эквивалентный шаблон оператора LIKE - это a%b, где (%) обозначает ноль, одно и более вхождений любых символов.

В Таблице 2 показан полный список операторов повтора. Заметьте, что в ней содержатся специфичные варианты повтора, которые допускают больше гибкости, чем существующие групповые символы оператора LIKE. Если выражение заключить в скобки, в результате чего образуется подвыражение, то подвыражение может повторяться заданное число раз. Например, регулярному выражению b(an)\*a соответствует ba, bana, banana, yourbananasplit, и так далее.

Регулярные выражения Oracle поддерживают символьные классы POSIX (Portable Operating System Interface), которые перечислены в Таблице 3. Это значит, что можно искать совершенно особые типы символов. Представьте написание условия с оператором LIKE, которое ищет только символы, не являющиеся буквами - получающееся выражение WHERE легко становится очень сложным.

Символьный класс POSIX должен входить в список символов, обозначаемый квадратными скобками ([]). Например, регулярное выражение [[:lower:]] соответствует одному символу в нижнем регистре, а [[:lower:]]{5} - пяти последовательным символам в нижнем регистре.

Кроме символьных классов POSIX, в список символов можно включать отдельные символы. Например, регулярное выражение ^ab[cd]ef$ соответствует строкам abcef и abdef. Сопоставляются как c, так и d.

Большинство метасимволов списка понимаются как литеры, за исключением вставки (^) и дефиса (-). Регулярные выражения могут казаться сложными, так как одни и те же метасимволы имеют несколько значений в зависимости от контекста. ^ как раз один из таких метасимволов. Если он - первый символ в списке символов, то означает не вхождение в список. Поэтому, [^[:digit:]] ищет соответствие, состоящее только из любых нецифровых символов, в то время как ^[[:digit:]] ищет соответствие, которое начинается с цифры. Дефис (-) обозначает диапазон; регулярное выражение [a-m] соответствует любым буквам от a до m. Однако он является литерой дефис, если находится в начале списка символов, например [-afg].

Один из предыдущих примеров показывал использование скобок для создания подвыражения; они позволяют ввести альтернативы, разделенные вертикальной чертой (/) - метасимволом альтернативы.

Например, регулярное выражение t(a/e/i)n допускает три возможных альтернативных символа между буквами t и n. Этому шаблону соответствуют слова tan, ten, tin и Pakistan, но не teen, mountain или tune. С другой стороны, регулярное выражение t(a/e/i)n можно описать как список символов, например t[aei]n. Эти метасимволы сгруппированы в Таблице 4. Хотя метасимволов значительно больше, этот краткий список важен для понимания регулярных выражений, используемых в настоящей статье.

**Оператор REGEXP\_LIKE**

Когда в базе данных Oracle встречается оператор REGEXP\_LIKE, он знакомит с регулярными выражениями. В таблице 5 показан синтаксис REGEXP\_LIKE.

Следующий SQL-запрос с выражением WHERE демонстрирует оператор REGEXP\_LIKE, который ищет столбец ZIP со значениями, которые удовлетворяют регулярному выражению [^[:digit:]]. Отбираться будут те строки таблицы ZIPCODE, в которых значение столбца ZIP содержит любой символ, не являющийся цифрой.

SELECT zip

FROM zipcode

WHERE REGEXP\_LIKE(zip, "[^[:digit:]]");

ZIP

-----

ab123

123xy

007ab

abcxy

Этот пример регулярного выражения состоит только из метасимволов, а точнее, из символьного класса POSIX digit, ограниченного двоеточиями и квадратными скобками. Второй набор скобок (как в [^[:digit:]]) ограничивает список символьных классов. Как пояснялось выше, это необходимо, так как символьные классы POSIX могут использоваться только для формирования списка символов.

**Функция REGEXP\_INSTR**

Эта функция возвращает начальную позицию образца, поэтому она работает примерно так, как уже знакомая функция INSTR. Синтаксис новой функции REGEXP\_INSTR показан в Таблице 6. Основное различие между этими двумя функциями заключается в том, что REGEXP\_INSTR позволяет указать шаблон вместо конкретной строки поиска; обеспечивая, таким образом, большее многообразие. Следующий пример использует функцию REGEXP\_INSTR, которая возвращает начальную позицию пятицифрового zip-кода в строке Joe Smith, 10045 Berry Lane, San Joseph, CA 91234. Если регулярное выражение выглядит как [[:digit:]]{5}, то вместо zip-кода возвратится начальная позиция номера дома, так как 10045 - это первое вхождение из пяти последовательных цифр. Поэтому выражение необходимо привязать к концу строки метасимволом $, тогда функция отобразит начальную позицию zip-кода вместо набора цифр, соответствующих номеру дома.

SELECT REGEXP\_INSTR("Joe Smith, 10045 Berry Lane, San Joseph, CA 91234",

"[[:digit:]]{5}$")

AS rx\_instr

FROM dual;

RX\_INSTR

----------

45

**Более сложные шаблоны**

Давайте усложним шаблон zip-кода предыдущего примера, чтобы найти дополнительные четыре цифры. Теперь он может выглядеть так: [[:digit:]]{5}(-[[:digit:]]{4})?$. Если исходная строка заканчивается пятицифровым zip-кодом или 5-цифровым + 4 zip-кодом, то надо будет показать начальную позицию шаблона.

SELECT REGEXP\_INSTR(

"Joe Smith, 10045 Berry Lane, San Joseph, CA 91234-1234",

" [[:digit:]]{5}(-[[:digit:]]{4})?$")

AS starts\_at

FROM dual;

STARTS\_AT

----------

44

В этом примере в скобки заключено подвыражение (-[[:digit:]]{4}), которое может повторяться ноль и более раз, что показывает оператор повтора? С другой стороны, если попытаться использовать традиционные SQL-функции для получения того же результата, то формулировка будет сложна даже для знатока SQL. Чтобы лучше объяснить различные компоненты этого примера регулярного выражения, в Таблице 7 представлено описание отдельных литер и метасимволов.

**Функция REGEXP\_SUBSTR**

Функция REGEXP\_SUBSTR, очень похожа на функцию SUBSTR и возвращает часть строки. В Таблице 8 показан синтаксис новой функции. В следующем примере возвращается строка, которая соответствует шаблону [^,]\*. Регулярное выражение ищет запятую, за которой следует пробел; затем ноль и более символов, не являющихся запятыми, что показывает [^,]\*; и затем ищет другую запятую. Шаблон будет выглядеть примерно как строка значений, разделенных запятыми.

SELECT REGEXP\_SUBSTR("first field, second field , third field",

", [^,]\*,")

FROM dual;

REGEXP\_SUBSTR("FIR

------------------

, second field ,

**Функция REGEXP\_REPLACE**

Сначала рассмотрим традиционную SQL-функцию REPLACE, которая заменяет одну строку другой. Предположим, что в тексте имеются дополнительные пробелы, а вам хотелось бы заменить их одним пробелом. В функции REPLACE необходимо перечислить ровно столько пробелов, сколько надо заменить. Однако дополнительные пробелы могут располагаться не только в одном месте. Следующий пример имеет три пробела между Joe и Smith. Параметр функции REPLACE показывает, что два пробела следует заменить одним пробелом. В этом случае результат имеет на один дополнительный пробел меньше, в то время как в исходной строке между Joe и Smith было три пробела.

SELECT REPLACE("Joe Smith"," ", " ")

AS replace

FROM dual;

REPLACE

---------

Joe Smith

Функция REGEXP\_REPLACE выполняет замены на порядок дальше; она описана в Таблице 9. Следующий запрос заменяет два и более любых пробелов на один пробел. Подвыражение ( ) содержит один пробел, который может повторяться два и более раз, что показывает {2,}.

SELECT REGEXP\_REPLACE("Joe Smith",

"( ){2,}", " ")

AS RX\_REPLACE

FROM dual;

RX\_REPLACE

----------

Joe Smith

**Ссылки**

Полезная возможность регулярных выражений - это способность запоминать подвыражение для дальнейшего использования; она называется также ссылочность (описана в Таблице 10). Она позволяет выполнять сложные замены, такие как перемещение образца на новую позицию или нахождение повторяющегося слова или буквы. Соответствующая подвыражению часть сохраняется во временном буфере. Буферы нумеруются слева направо, а доступ к ним осуществляется через цифра, где цифра, это число от 1 до 9, которое соответствует порядковому номеру подвыражения, обозначенному скобками.

Следующий пример показывает, как имя Ellen Hildi Smith преобразовывается в Smith, Ellen Hildi с помощью ссылок на отдельные подвыражения по номеру.

SELECT REGEXP\_REPLACE(

"Ellen Hildi Smith",

"(.\*) (.\*) (.\*)", "3, 1 2")

FROM dual;

REGEXP\_REPLACE("EL

------------------

Smith, Ellen Hildi

В этом SQL-предложении есть три отдельных подвыражения, заключенных в скобки. Каждое отдельное подвыражение содержит метасимвол соответствия любому символу (.), за которым следует метасимвол \*, показывающий, что любой символ (за исключением перехода на новую строку) должен появиться ноль и более раз. Каждое подвыражение разделяет пробел, который также должен быть сопоставлен. Скобки обозначают подвыражения для фиксации значений, на которые можно ссылаться через цифра. Первое подвыражение связывается с 1, второе 2 и так далее. Эти ссылки используются в последнем параметре этой функции (3, 1 2), которая возвращает измененную подстроку, преобразовав ее в требуемый формат (включая запятую и пробелы). В Таблице 11 подробно описываются отдельные элементы этого регулярного выражения.

Ссылки полезны для замены, форматирования и нахождения подстроки, и могут применяться для поиска расположенных рядом значений. Следующий пример показывает использование функции REGEXP\_SUBSTR для поиска дубликатов символьно-цифровых значений, разделенных пробелом. Полученный результат показывает слова, которые продублированы.

SELECT REGEXP\_SUBSTR(

"The final test is is the implementation",

"([[:alnum:]]+)([[:space:]]+)1") AS substr

FROM dual;

SUBSTR

------

is is

**Дополнительный параметр сопоставления**

Вы, возможно, заметили, что оператор и функции регулярного выражения могут содержать дополнительный параметр сопоставления. Этот параметр управляет учетом регистра, переходом на новую строку и многострочными вхождениями повторов.

**Практические приложения регулярных выражений**

Регулярные выражения используются не только в запросах, но также в любом месте, где допускаются SQL-операторы и функции, например, в PL/SQL. Можно написать триггеры, в которых регулярные выражения полезно использовать для проверки, генерации или выделения значений.

Следующий пример показывает, как оператор REGEXP\_LIKE можно применять в check-ограничении на столбец для контроля данных. Он проверяет корректность формата номера социального обеспечения при вставке или изменении записи. Для такого ограничения столбца допустим номер социального обеспечения в формате 123-45-6789 или 123456789. Корректные данные должны начинаться с трех цифр, за которыми следует дефис, две или более цифр, дефис и, наконец, другие четыре цифры. Альтернативное выражение допускает только девять последовательных цифр. Варианты разделяются вертикальной чертой (/).

ALTER TABLE students

ADD CONSTRAINT stud\_ssn\_ck CHECK

(REGEXP\_LIKE(ssn,

"^([[:digit:]]{3}-[[:digit:]]{2}-[[:digit:]]{4}/[[:digit:]]{9})$"));

Символы перед или после не допускаются, что показывают ^ и $. Необходимо убедиться, что регулярное выражение не разбито на несколько строк и не содержит лишних пробелов, если только не требуется, чтобы они были частью шаблона и поэтому сопоставлялись. В Таблице 12 объясняются отдельные элементы этого примера регулярного выражения.

Подробнее

Смотрите на странице the Oracle Database 10g:

[/products/database/oracle10g/index.html](http://www.findnews.ru/click?url=http%3a%2f%2fotn%2eoracle%2ecom%2fproducts%2fdatabase%2foracle10g%2findex%2ehtml" \t "_blank)

**Сравнение регулярных выражений с существующей функциональностью**

Регулярные выражения имеют некоторые преимущества перед обычным оператором LIKE и функциями INSTR, SUBSTR и REPLACE. Эти традиционные SQL-функции не имеют возможности сопоставления с шаблоном. Только оператор LIKE может выполнять символьное сопоставление с помощью символов группировки % и \_, однако LIKE не поддерживает повторы выражения, сложное чередование, диапазоны символов, списки символов, символьные классы POSIX, и др. А новые функции с регулярными выражениями позволяют найти еще и дубликаты, и выполнить перестановку. Примеры, показанные в этой статье, позволяют кратко заглянуть в мир регулярных выражений и понять, как применять их в приложениях.

**Полезное добавление к инструментарию**

Регулярные выражения очень полезны, так как помогают решить сложные задачи. Некоторые возможности регулярных выражений трудно воспроизвести с помощью традиционных SQL-функций. Когда будет изучено создание основных конструкций этого несколько сложного языка, регулярные выражения станут необходимой частью инструмента в контексте не только SQL, но и других языков программирования. Хотя тестирование и допущение ошибок иногда необходимо для получения индивидуальных навыков написания шаблонов, элегантность и мощность регулярных выражений бесспорна.

Алиса Ришет ( [ar280@yahoo.com](http://www.findnews.ru/click?url=http%3a%2f%2fwww%2einterface%2eru%2fmailto%3aar280%40yahoo%2ecom) ) работает в Database Application Development и Design track в Columbia University"s Computer Technology and Application Program. Она автор [Oracle SQL Interactive Workbook 2nd edition](http://www.findnews.ru/click?url=http%3a%2f%2fwww%2eamazon%2ecom%2fexec%2fobidos%2ftg%2fdetail%2f-%2f0131002775%2fqid%3d1060735468%2fsr%3d1-2%2fref%3dsr_1_2%2f002-8332797-9003230%3fv%3dglance%26s%3dbooks) (Prentice Hall, 2002) и готовящейся в публикации [Oracle SQL by Example](http://www.findnews.ru/click?url=http%3a%2f%2fwww%2eamazon%2ecom%2fexec%2fobidos%2ftg%2fdetail%2f-%2f0131451316%2fqid%3d1060735383%2fsr%3d1-2%2fref%3dsr_1_2%2f002-8332797-9003230%3fv%3dglance%26s%3dbooks) (Prentice Hall, 2003). Ришет - архитектор баз данных, DBA и руководитель проекта с 15-летним стажем для Fortune 100 компаний и работает с Oracle, начиная с 5-ой версии.

**Таблица 1: Метасимволы привязки**

|  |  |
| --- | --- |
| Метасимвол | Описание |
| ^ | Привязать выражение к началу строки |
| $ | Привязать выражение к концу строки |

**Таблица 2: Квантификаторы и операторы повтора**

|  |  |
| --- | --- |
| Квантификатор | Описание |
| \* | Встречается 0 и более раз |
| ? | Встречается 0 или 1 раз |
| + | Встречается 1 и более раз |
| {m} | Встречается ровно m раз |
| {m,} | Встречается по крайней мере m раз |
| {m, n} | Встречается по крайней мере m раз, но не более n раз |

**Таблица 3: Предопределенные символьные классы POSIX**

|  |  |
| --- | --- |
| Класс символов | Описание |
| [:alpha:] | Буквы |
| [:lower:] | Буквы в нижнем регистре |
| [:upper:] | Буквы в верхнем регистре |
| [:digit:] | Цифры |
| [:alnum:] | Буквы и цифры |
| [:space:] | Пробелы (не печатаемые символы), такие как перевод каретки, новая строка, вертикальная табуляция и подача страницы |
| [:punct:] | Знаки препинания |
| [:cntrl:] | Управляющие символы (не печатаемые) |
| [:print:] | Печатаемые символы |

**Таблица 4: Альтернативное сопоставление и группировка выражений**

|  |  |  |
| --- | --- | --- |
| Метасимвол | | Описание |
| / | Альтернатива | Разделяет альтернативные варианты, часто используется с оператором группировки () |
| ( ) | Группа | Группирует подвыражения для альтернативы, квантификатора или ссылочности (см. раздел "Ссылки") |
| [char] | Список символов | Обозначает список символов; большинство метасимволов в списке символов представляют собой литеры, за исключением символьных классов и метасимволов ^ и - |

**Таблица 5: Оператор REGEXP\_LIKE**

|  |  |
| --- | --- |
| Синтаксис | Описание |
| REGEXP\_LIKE(исходная\_строка, шаблон[, параметр\_сопоставления]) | "исходная\_строка" поддерживает символьные типы данных (CHAR, VARCHAR2, CLOB, NCHAR, NVARCHAR2 и NCLOB, но не LONG). Параметр "шаблон" - это другое название регулярного выражения. "параметр\_сопоставления" позволяет использовать дополнительные параметры, такие как символ перехода на новую строку, многострочное форматирование и обеспечение управления учетом регистра. |

**Таблица 6: Функция REGEXP\_INSTR**

|  |  |
| --- | --- |
| Синтаксис | Описание |
| REGEXP\_INSTR(исходная\_строка, шаблон  [, начальная\_позиция  [, вхождение  [, опция\_возврата  [, параметр\_сопоставления]]]]) | Эта функция ищет по шаблону и возвращает первую позицию. Дополнительно можно указать параметр "начальная\_позиция", с которой должен начинаться поиск. Параметр "вхождение" по умолчанию имеет значение 1, если пользователь не укажет поиск последовательных вхождений. Значение по умолчанию для параметра "опция\_возврата" - это 0, тогда возвратится начальная позиция шаблона; при значении 1 возвращается позиция символа, следующего за шаблоном. |

**Таблица 7: Описание выражения для 5-цифрового + 4 Zip-кода**

|  |  |
| --- | --- |
| Синтаксис | Описание |
|  | Пробел, который должен быть сопоставлен |
| [:digit:] | Числовой класс POSIX |
| ] | Конец списка символов |
| {5} | Повторять список символов ровно пять раз |
| ( | Начало подвыражения |
| - | Литера дефис, так как он не является метасимволом диапазона внутри списка символов |
| [ | Начало списка символов |
| [:digit:] | Класс POSIX [:digit:] |
| [ | Начало списка символов |
| ] | Конец списка символов |
| {4} | Повторять список символов ровно четыре раза |
| ) | Закрывающая скобка - конец подвыражения |
| ? | Квантификатор ? относится к групповому подвыражению 0 или 1 раз, в результате чего 4-цифровой код становится дополнительным |
| $ | Метасимвол привязки, чтобы показать конец строки |

**Таблица 8: Функция REGEXP\_SUBSTR**

|  |  |
| --- | --- |
| Синтаксис | Описание |
| REGEXP\_SUBSTR(исходная\_строка, шаблон  [, позиция [, вхождение  [,параметр\_сопоставления]]]) | Функция REGEXP\_SUBSTR возвращает подстроку, которая соответствует шаблону. |

**Таблица 9: Функция REGEXP\_REPLACE**

|  |  |
| --- | --- |
| Синтаксис | Описание |
| REGEXP\_REPLACE(исходная\_строка, шаблон  [, строка\_замены [, позиция  [,вхождение, [параметр\_сопоставления]]]]) | Эта функция заменяет подстроку, соответствующую шаблону на заданную "строку\_замены", используя сложные операции поиска-и-замены. |

**Таблица 10: Метасимвол ссылки**

|  |  |  |
| --- | --- | --- |
| Метасимвол | | Описание |
| digit | Обратная косая черта | За ней следует цифра от 1 до 9, обратная косая черта связана с предыдущим сопоставлением с соответствующим номером заключенного в скобки подвыражения.  (Заметьте: Обратная косая черта может иметь другое значение в регулярном выражении; в зависимости от контекста она может означать также символ Escape |

**Таблица 11: Описание регулярного выражения замены-по-шаблону**

|  |  |
| --- | --- |
| Элемент регулярного выражения | Описание |
| ( | Начало первого подвыражения |
| . | Соответствие одному любому символу, за исключением перехода на новую строку |
| \* | Оператор повтора соответствует предыдущему метасимволу . от 0 до n раз |
| ) | Конец первого подвыражения; результат сопоставления сохраняется в 1  (В этом примере, Ellen.) |
|  | Пробел, который должен быть сопоставлен |
| ( | Начало второго подвыражения |
| . | Сопоставление с одним любым символом, за исключением перехода на новую строку |
| \* | Оператор повтора соответствует предыдущему метасимволу . от 0 до n раз |
| ) | Конец второго подвыражения; результат сопоставления сохраняется в 2  (В этом примере, Hildi.) |
|  | Пробел |
| ( | Начало третьего подвыражения |
| . | Сопоставление с одним любым символом, за исключением перехода на новую строку |
| \* | Оператор повтора соответствует предыдущему метасимволу . от 0 до n раз |
| ) | Конец третьего подвыражения; результат сопоставления сохраняется в 3  (В этом примере, Smith.) |

**Таблица 12: Описание регулярного выражения для номера социального обеспечения**

|  |  |
| --- | --- |
| Элемент регулярного выражения | Описание |
| ^ | Начало строки символов (Регулярное выражение не может иметь лидирующих символов перед сопоставлением.) |
| ( | Начало подвыражения и список альтернатив, разделенных метасимволом / |
| [ | Начало списка символов |
| [:digit:] | Числовой класс POSIX |
| ] | Конец списка символов |
| {3} | Повтор списка символов ровно три раза |
| - | Дефис |
| [ | Начало списка символов |
| [:digit:] | Числовой класс POSIX |
| ] | Конец списка символов |
| {2} | Повтор списка символов ровно два раза |
| - | Другой дефис |
| [ | Начало списка символов |
| [:digit:] | Числовой класс POSIX |
| ] | Конец списка символов |
| {4} | Повтор списка символов ровно четыре раза |
| / | Метасимвол альтернативы; конец первого варианта и начало следующего альтернативного выражения |
| [ | Начало списка символов |
| [:digit:] | Числовой класс POSIX |
| ] | Конец списка символов |
| {9} | Повтор списка символов ровно девять раз |
| ) | Закрывающая скобка, закрывает группу подвыражения, используемого как альтернатива |
| $ | Метасимвол привязки, показывает конец строки; дополнительные символы после шаблона не допускаются |