Of course! In SQL, REGEXP (or RLIKE) is a powerful operator used for complex pattern matching in strings, going far beyond the capabilities of the LIKE operator.

You can think of LIKE as using a blunt instrument (% for any string, \_ for any character), while REGEXP is like having a full set of surgical tools for text. It allows you to define precise rules for what constitutes a match.

The exact syntax can vary slightly between database systems (like MySQL, PostgreSQL, etc.), but the core concepts are the same. The examples below use **MySQL syntax**.

### Core Regex Building Blocks (Metacharacters)

Before the examples, here are the essential symbols you'll use to build your patterns.

|  |  |  |  |
| --- | --- | --- | --- |
| Symbol | Description | Example Pattern | What it Matches |
| . | Any single character | c.t | cat, cot, c@t |
| ^ | Start of the string | ^A | Apple, but not Banana |
| $ | End of the string | com$ | test.com, but not company |
| \* | Zero or more of the preceding character | go\*l | gl, gol, gooool |
| + | One or more of the preceding character | go+l | gol, gooool, but not gl |
| ? | Zero or one of the preceding character | colou?r | color, colour |
| [] | Any single character within the brackets | gr[ae]y | gray, grey |
| [^] | Any single character **not** in the brackets | [^aeiou] | Any consonant |
| (|) | (|) | Acts as an "OR" operator | (Cat|Dog) |
| {n} | Exactly n occurrences | \d{3} | Exactly 3 digits, like 123 |
| {n,m} | Between n and m occurrences | \w{5,8} | 5 to 8 word characters |
| \d | Any digit ([0-9]) | \d | 0, 1, 2, ... 9 |
| \w | Any "word" character ([a-zA-Z0-9\_]) | \w | a, B, 7, \_ |

### Comprehensive Examples

Let's imagine we have a customers table:

+----+---------------+---------------------------+

| id | name | email |

+----+---------------+---------------------------+

| 1 | Jon Doe | jon.doe@example.com |

| 2 | Jane Smith | jsmith@work.io |

| 3 | Ana | ana1995@webmail.org |

| 4 | johnathan | J-money@server-01.com |

+----+---------------+---------------------------+

### 🎯 Example 1: Finding Names with Specific Vowels

**Goal:** Find all customers whose first name is "Jon" or "John". The o? makes the 'h' optional.

SQL

SELECT name, email  
FROM customers  
WHERE name REGEXP '^Joh?n';

Result:

Jon Doe

johnathan

**Explanation:**

* ^ ensures the pattern starts at the beginning of the name.
* Joh matches the literal characters "Joh".
* n? matches the character 'n' **zero or one** time.
* This pattern matches both "Jon" and "John". Notice it also matches "johnathan" because the name *starts with* that pattern.

### 🔍 Example 2: Validating Email Domains

**Goal:** Find customers who have a .com or .org email address.

SQL

SELECT name, email  
FROM customers  
WHERE email REGEXP '\\.(com|org)$';

Result:

Jon Doe (jon.doe@example.com)

Ana (ana1995@webmail.org)

johnathan (J-money@server-01.com)

**Explanation:**

* \\. The backslash \ is an escape character. Since . is a special symbol (any character), \\. tells the engine to match a literal period.
* (com|org) The parentheses group the options, and the pipe | acts as an **OR**. It looks for either "com" or "org".
* $ ensures this pattern is at the very **end** of the string.

### 📝 Example 3: Finding Usernames with Numbers

**Goal:** Find customers whose email prefix (the part before the @) contains at least one number.

SQL

SELECT name, email  
FROM customers  
WHERE email REGEXP '^[a-zA-Z.\_-]+\\d+[a-zA-Z0-9.\_-]\*@';

Result:

Ana (ana1995@webmail.org)

**Explanation:**

* ^ starts the search at the beginning of the email string.
* [a-zA-Z.\_-]+ matches one or more letters, dots, underscores, or hyphens.
* \\d+ is the key part: it requires **one or more digits**.
* [a-zA-Z0-9.\_-]\* allows for more valid characters after the number.
* @ matches the literal "@" symbol.

### Dialect Differences

It's crucial to know which operator your specific SQL database uses.

|  |  |  |
| --- | --- | --- |
| Database | Operator(s) | Case-Sensitive |
| **MySQL** | REGEXP, RLIKE | REGEXP BINARY |
| **PostgreSQL** | ~, ~\*, !~, !~\* | ~ (yes), ~\* (no) |
| **Oracle** | REGEXP\_LIKE() (function) | Controllable with a parameter |
| **SQL Server** | *(No built-in operator, requires CLR integration)* | N/A |