

How to make your complex SET FILTER expression work on the PostgreSQL server

Background

SET FILTER expressions in Xbase++ can be of a simple or a complex nature. A simple filter is the comparison of a column/field with a constant or another column/field of the table, like "age>10". A complex expression uses runtime functions, such as DTos(), SubStr() or AT() to convert/compare the value of the column/field. An example is "Year(updated) == 2021".

Because FILTER expressions in the ISAM table become a WHERE <clause> in the SQL world using the Xbase++ ISAM Emulation for the PostgreSQL server, the expression is executed on the server and not on the client side. This simple fact implicates that only expressions which can be executed by the PostgreSQL server are valid in FILTER expressions. For this reason, the Xbase++ ISAM emulation rewrites your filter expressions to conform to the SQL syntax in terms of operations, alias names and so on. However, functions used in your FILTER expressions still need be executed by the PostgreSQL server.

Solution

To solve this problem, two steps need to be taken.

1. Use grep or the findstr utility to identify all your filter expressions, then identify all the Xbase++ functions you need.
2. Implement stored functions on the PostgreSQL server which implement the required functionality

Important

Functions such as RecNo() or RecCount() require the execution context. Because of this, unfortunately, there is no way to implement these functions properly. However, the following table shows how to adapt your expressions to work on the SQL server and Xbase++ side.

Function	Use instead
RecNo()	__record
Deleted()	__deleted

Examples

In the following, example implementations are shown for Date(), Len() and AllTrim():

```
SQL:
create or replace function date( indata date ) RETURNS text
language plpgsql
as $$
begin
    RETURN CAST( indata AS text);
end;$$
```

SQL:

```
create or replace function alltrim( anydata text ) RETURNS text
language plpgsql
as $$
begin
    RETURN TRIM(BOTH FROM anydata);
end;$$
```

SQL:

```
create or replace function len( indata text ) RETURNS int
language plpgsql
as $$
begin
    RETURN char_length(indata);
end;$$
```