

SQL function return-type: TABLE vs SETOF records

Asked 9 years, 7 months ago Modified 3 years, 11 months ago Viewed 22k times



What's the difference between a function that returns `TABLE` vs `SETOF` records , all else equal.

38



```
CREATE FUNCTION events_by_type_1(text) RETURNS TABLE(id bigint, name text) AS
$$
    SELECT id, name FROM events WHERE type = $1;
$$ LANGUAGE SQL STABLE;
```



```
CREATE FUNCTION events_by_type_2(text) RETURNS SETOF record AS $$
    SELECT id, name FROM events WHERE type = $1;
$$ LANGUAGE SQL STABLE;
```

These functions seem to return the same results. See this [SQLFiddle](#).

[sql](#) [postgresql](#) [return-type](#) [sql-function](#) [sql-types](#)

Share Improve this question Follow

asked Mar 15, 2014 at 12:43



[ma11hew28](#)

122k

117

451

651

2 Answers

Sorted by: Highest score (default)



When returning `SETOF` record the output columns are not typed and not named. Thus this form can't be used directly in a `FROM` clause as if it was a subquery or a table.

30

That is, when issuing:



```
SELECT * from events_by_type_2('social');
```



we get this error:



```
ERROR: a column definition list is required for functions returning "record"
```

It can be "casted" into the correct column types by the SQL caller though. This form does work:

```
SELECT * from events_by_type_2('social') as (id bigint, name text);
```

and results in:

```

id |      name
---+-----
 1 | Dance Party
 2 | Happy Hour
...

```

For this reason `SETOF` record is considered less practical. It should be used only when the column types of the results are not known in advance.

Share Improve this answer

edited Oct 31, 2019 at 22:22

answered Mar 15, 2014 at 13:02

Follow



ma11hew28

122k 117 451 651



Daniel Vérité

58.4k 15 129 156

- 3 @MattDiPasquale: I'm not aware of any performance difference. However in both cases if using these functions in more complex queries, you might be interested in [How to avoid multiple function evals with the \(func\(\)\).* syntax in an SQL query?](#) – Daniel Vérité Mar 15, 2014 at 15:33



17



This answer is only to remember alternative **context where `TABLE` and `SETOF` are equivalent**.

As @a_horse_with_no_name pointed, it is not a `RETURNS SETOF "unknown record"`, is a defined one.

In this example, the types `table` and `setof` are equivalent,

```

CREATE TYPE footype AS (score int, term text);

CREATE FUNCTION foo() RETURNS SETOF footype AS $$
  SELECT * FROM ( VALUES (1,'hello!'), (2,'Bye') ) t;
$$ language SQL immutable;

CREATE FUNCTION foo_tab() RETURNS TABLE (score int, term text) AS $$
  SELECT * FROM ( VALUES (1,'hello!'), (2,'Bye') ) t;
$$ language SQL immutable;

SELECT * FROM foo();      -- works fine!
SELECT * FROM foo_tab();  -- works fine and is equivalent.

```

The `RETURNS SETOF` have the **advantage of reuse type** (see `footype`), that is impossible with `RETURNS TABLE`.

Share Improve this answer


edited Mar 14, 2016 at 15:37

community wiki

Follow

4 revs

Peter Krauss

2 returns setof footype is something different then RETURNS SETOF record because now it's no longer an "unknown" type. The columns of footype are well defined. – user330315 Mar 14, 2016 at 15:06 

Thank you, @PeterKrauss. :-) – [ma11hew28](#) Oct 31, 2019 at 22:28
