Sql Recursive Loop Check

SqlRecursiveCheckTask

`SqlRecursiveCheckTask` helps you find **loops** in data where one item points to another — for example, parent-child or dependency relationships.

It checks for **cycles**, like when A → B → C → A. These can break trees, graphs, or dependency systems.

What You Give It

You can give it data in two ways:

1. **Query Builder or Eloquent Builder**

Pass a query where you select exactly two fields:

- `from_id`: the starting point
- `to_id`: where it points to

Example:

```
Model::select('id as from_id', 'parent_id as to_id')
```

2. **Array or Collection**

You can pass an array or a Laravel collection of pairs:

- As named keys:

```
[
    ['from_id' => 1, 'to_id' => 2],
    ['from_id' => 2, 'to_id' => 3],
    ['from_id' => 3, 'to_id' => 1],
]
```

- Or just plain arrays:

```
[
    [1, 2],
    [2, 3],
    [3, 1],
]
```

What It Does

It runs a recursive SQL query to find **all loops** — meaning chains where something eventually points back to itself.

This is useful if you want to prevent circular references that could cause bugs or infinite loops in your system.

What It Returns

It returns a list of found loops. Each loop is shown from the perspective of each node inside the loop.

Output format (same as `DB::select`):

```
[
{
    "root_id": 3,
        "path": "{5,8,3}"
    },
    {
    "root_id": 5,
        "path": "{8,3,5}"
    },
    {
    "root_id": 8,
        "path": "{3,5,8}"
    },
    ...
```

```
]
```

Each item means:

- `root_id`: the node where the search started
- `path`: the full loop path it found (as a string like a PostgreSQL array)

Revision #2
Created 2025-06-22 12:50:07 UTC by Admin
Updated 2025-06-22 13:23:03 UTC by Admin