608. Tree Node

July 10, 2017 | 17.1K views

Average Rating: 4.78 (9 votes)

Given a table `tree`, **id** is identifier of the tree node and **p_id** is its parent node's **id**.

```
+----+------+
| id | p_id |
+----+------+
| 1  | null |
| 2  | 1    |
| 3  | 1    |
| 4  | 2    |
| 5  | 2    |
+----+------+
```

Each node in the tree can be one of three types:

- Leaf: if the node is a leaf node.
- Root: if the node is the root of the tree.
- Inner: If the node is neither a leaf node nor a root node.

Write a query to print the node id and the type of the node. Sort your output by the node id. The result for the above sample is:

```
+----+------+
| id | Type |
+----+------+
| 1  | Root |
| 2  | Inner|
| 3  | Leaf |
| 4  | Leaf |
| 5  | Leaf |
+----+------+
```

Explanation

- Node '1' is root node, because its parent node is NULL and it has child node '2' and '3'.
- Node '2' is inner node, because it has parent node '1' and child node '4' and '5'.
- Node '3', '4' and '5' is Leaf node, because they have parent node and they don't have child node.

- And here is the image of the sample tree as below:

```
          1
        /   \
      2       3
    /   \
  4       5
```

Note

If there is only one node on the tree, you only need to output its root attributes.

Solution

Approach I: Using `UNION` [Accepted]

Intuition

We can print the node type by judging every record by its definition in this table. *Root: it does not have a parent node at all* Inner: it is the parent node of some nodes, and it has a not NULL parent itself. * Leaf: rest of the cases other than above two

Algorithm

By transiting the node type definition, we can have the following code.

For the root node, it does not have a parent.

```
SELECT
    id, 'Root' AS Type
FROM
    tree
WHERE
    p_id IS NULL
```

For the leaf nodes, they do not have any children, and it has a parent.

```
SELECT
    id, 'Leaf' AS Type
FROM
    tree
WHERE
    id NOT IN (SELECT DISTINCT
            p_id
        FROM
            tree
        WHERE
            p_id IS NOT NULL)
        AND p_id IS NOT NULL
```

For the inner nodes, they have have some children and a parent.

```
SELECT
    id, 'Inner' AS Type
FROM
    tree
WHERE
    id IN (SELECT DISTINCT
            p_id
        FROM
            tree
        WHERE
            p_id IS NOT NULL)
        AND p_id IS NOT NULL
```

So, one solution to the problem is to combine these cases together using `UNION`.

MySQL

```
SELECT
    id, 'Root' AS Type
FROM
    tree
WHERE
    p_id IS NULL

UNION

SELECT
    id, 'Leaf' AS Type
FROM
    tree
WHERE
    id NOT IN (SELECT DISTINCT
            p_id
        FROM
            tree
        WHERE
            p_id IS NOT NULL)
        AND p_id IS NOT NULL

UNION

SELECT
    id, 'Inner' AS Type
FROM
    tree
WHERE
    id IN (SELECT DISTINCT
            p_id
        FROM
            tree
        WHERE
            p_id IS NOT NULL)
        AND p_id IS NOT NULL
ORDER BY id;
```

Approach II: Using flow control statement `CASE` [Accepted]

Algorithm

The idea is similar with the above solution but the code is simpler by utilizing the flow control statements, which is effective to output differently based on different input values. In this case, we can use `CASE` statement.

MySQL

```
SELECT
    id AS `Id`,
    CASE
        WHEN tree.id = (SELECT atree.id FROM tree atree WHERE atree.p_id IS NULL)
          THEN 'Root'
        WHEN tree.id IN (SELECT atree.p_id FROM tree atree)
          THEN 'Inner'
        ELSE 'Leaf'
    END AS Type
FROM
    tree
ORDER BY `Id`
;
```

> MySQL provides different flow control statements besides `CASE`. You can try to rewrite the slution above using `IF` flow control statement.

Approach III: Using `IF` function [Accepted]

Algorithm

Also, we can use a single `IF` function instead of the complex flow control statements.

MySQL

```
SELECT
    atree.id,
    IF(ISNULL(atree.p_id),
        'Root',
        IF(atree.id IN (SELECT p_id FROM tree), 'Inner','Leaf')) Type
FROM
    tree atree
ORDER BY atree.id
```

> Note: This solution was inspired by @richarddia

Rate this article: ★ ★ ★ ★ ★

Previous Next

Comments: 28 Sort By

Type comment here... (Markdown is supported)

Preview Post

Mr-Bin ★ 123 January 3, 2019 11:08 PM Report

My clean solution:

```
select id,
    case when p_id is null then 'Root'
        when id in (select p_id from tree) then 'Inner'
```

Read More

29 Share Reply

SHOW 5 REPLIES

abhi25 ★ 58 July 19, 2018 8:24 AM

```
SELECT DISTINCT a.id, CASE
    WHEN a.p_id IS NULL THEN 'Root'
    WHEN b.id IS NULL THEN 'Leaf'
    ELSE 'Inner'
```

Read More

8 Share Reply

olivia612 ★ 78 October 8, 2018 7:55 PM

I think my solution is way better.

```
select id,
(case
 when p_id is null then 'Root'
```

Read More

7 Share Reply

ztj ★ 15 August 3, 2018 10:36 AM

select id, (case when p_id is null then "root" when id in (select p_id from tree) then "inner" else "leaf" end) as type
from tree

4 Share Reply

ashokraj ★ 7 April 6, 2019 3:45 AM Report

```
SELECT
    id
    , CASE
```

Read More

2 Share Reply

SHOW 1 REPLY

Pallavi0779 ★ 2 February 23, 2019 9:05 AM

select Id , case
when p_id is null then 'Root'
when id in (select p_id from tree) then 'Inner'
else 'Leaf'
end as Type

Read More

2 Share Reply

XIONGCODE ★ 14 March 6, 2018 5:26 AM

Thanks for sharing your solution. I found the LEFT JOIN and IF combination is easier for me to understand:

SELECT DISTINCT a.id AS id, IF(a.p_id IS NULL, 'Root', IF(b.id IS NULL, 'Leaf', 'Inner')) AS Type
FROM tree a LEFT JOIN tree b

Read More

1 Share Reply

ntnmathur ★ 1 December 29, 2017 11:38 AM

How about this:
select
distinct
parent.id,
CASE WHEN parent.p_id IS NULL THEN 'Root'

Read More

1 Share Reply

ethanai ★ 1 June 27, 2019 2:52 AM

```
SELECT     DISTINCT t.id,
           CASE
               WHEN t.p_id IS NULL THEN 'Root'
               WHEN i.id IS NOT NULL THEN 'Inner'
```

Read More

0 Share Reply

Arpita151 ★ 0 March 31, 2019 2:37 AM Report

Write your MySQL query statement below

Read More

0 Share Reply

‹ 1 2 3 ›