



# Models for Hierarchical Data with SQL and PHP

Bill Karwin, Percona Inc.

# Me

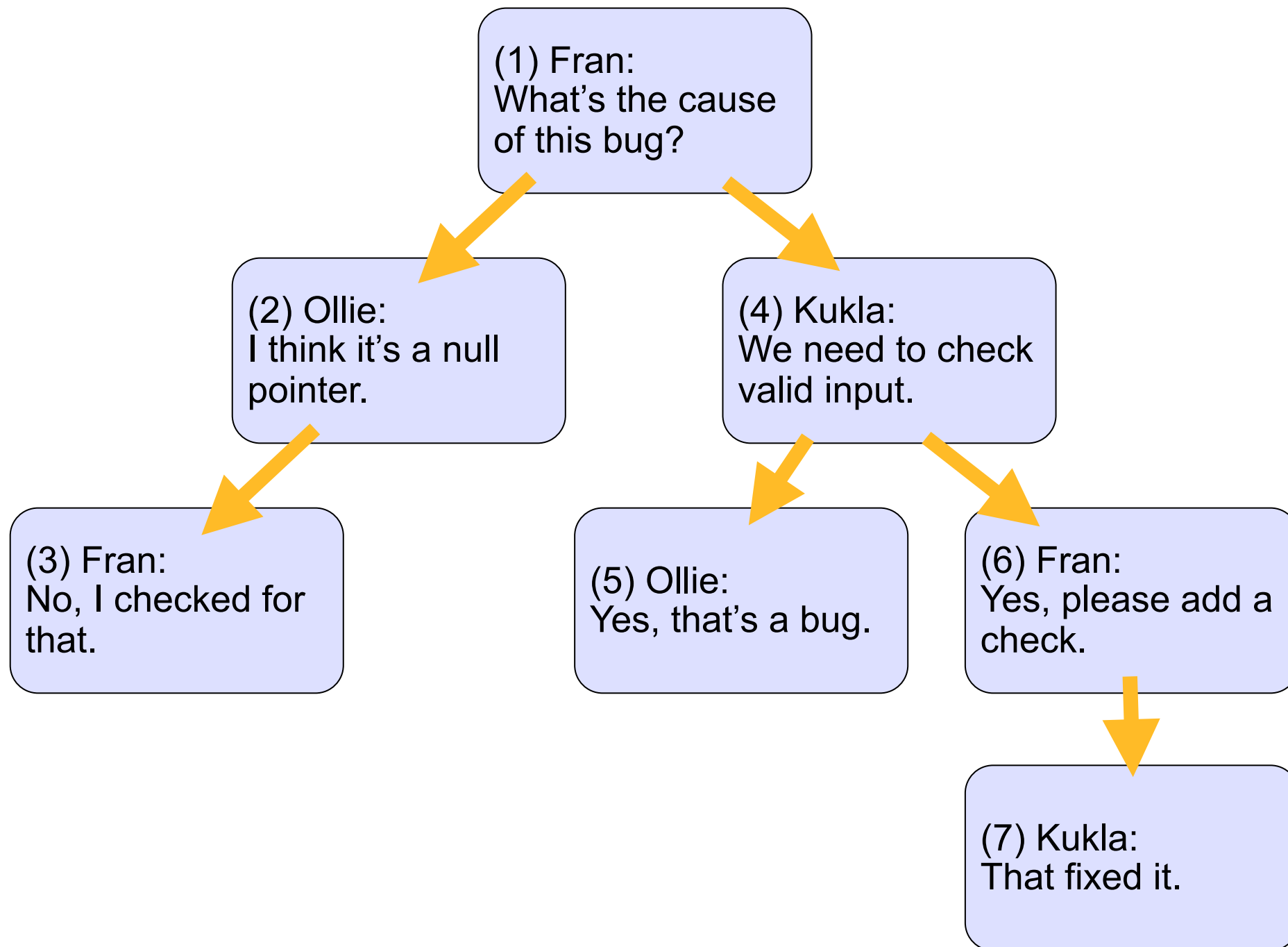
- Software developer
- C, Java, Perl, PHP, Ruby
- SQL maven
- MySQL Consultant at Percona
- Author of *SQL Antipatterns: Avoiding the Pitfalls of Database Programming*



# Problem

- Store & query hierarchical data
  - Categories/subcategories
  - Bill of materials
  - Threaded discussions

# Example: Bug Report Comments



# Solutions

- Adjacency list
- Path enumeration
- Nested sets
- Closure table

# Adjacency List

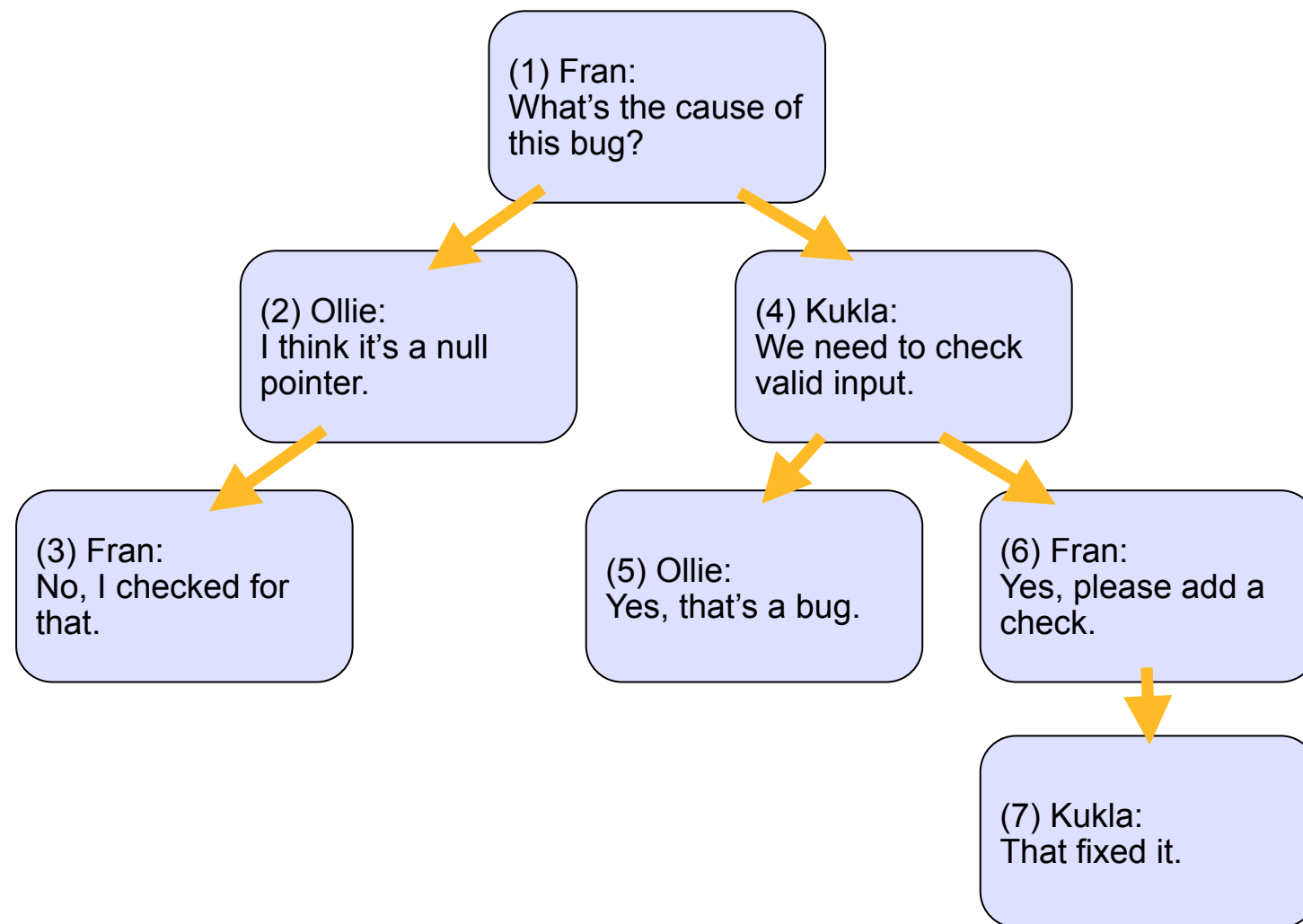
# Adjacency List

- Naive solution nearly everyone uses
- Each entry knows its immediate parent

comment_id	parent_id	author	comment
1	NULL	Fran	What's the cause of this bug?
2	1	Ollie	I think it's a null pointer.
3	2	Fran	No, I checked for that.
4	1	Kukla	We need to check valid input.
5	4	Ollie	Yes, that's a bug.
6	4	Fran	Yes, please add a check
7	6	Kukla	That fixed it.

# Insert a New Node

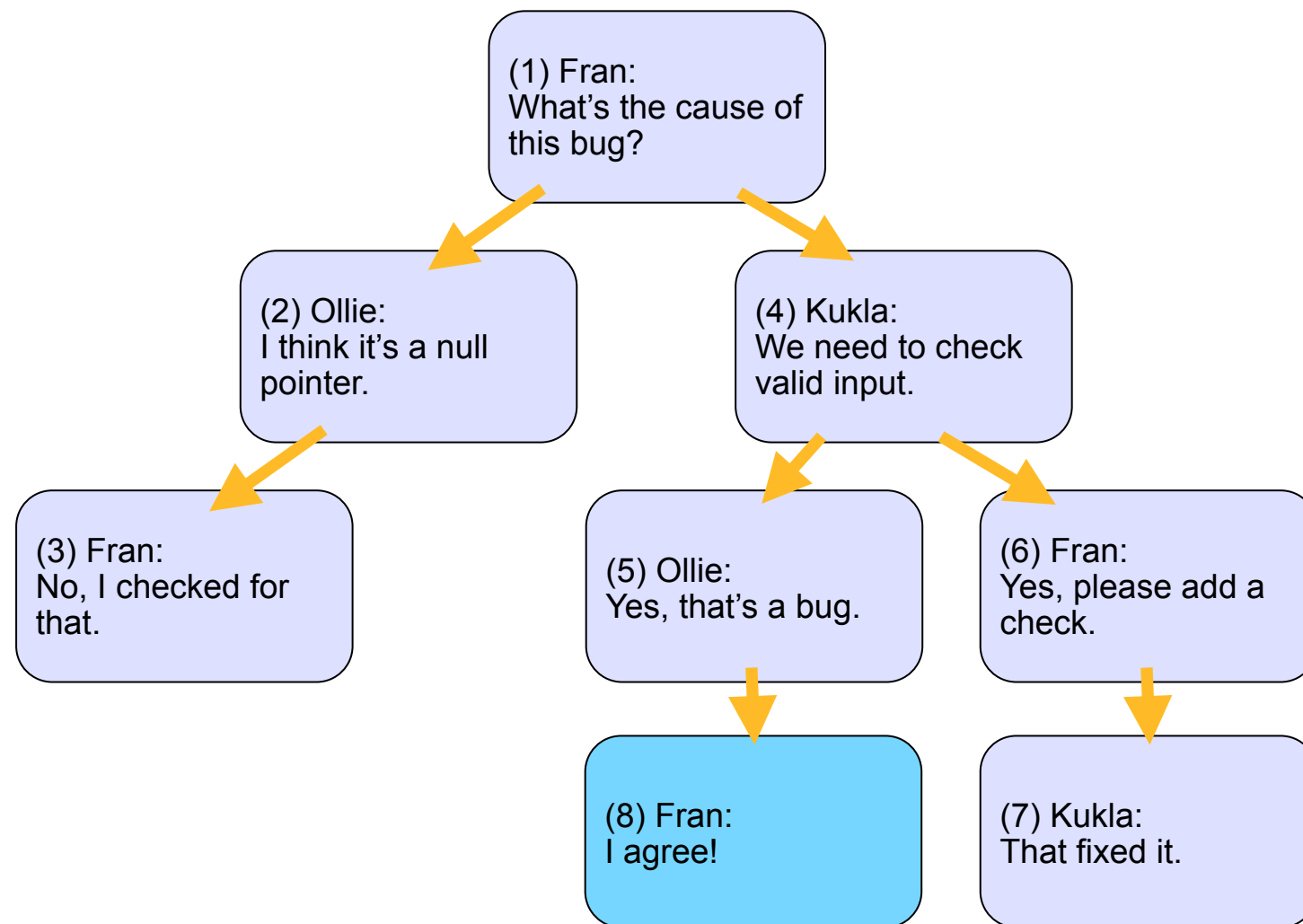
INSERT INTO Comments (parent\_id, author, comment)  
VALUES (5, 'Fran', 'I agree!');





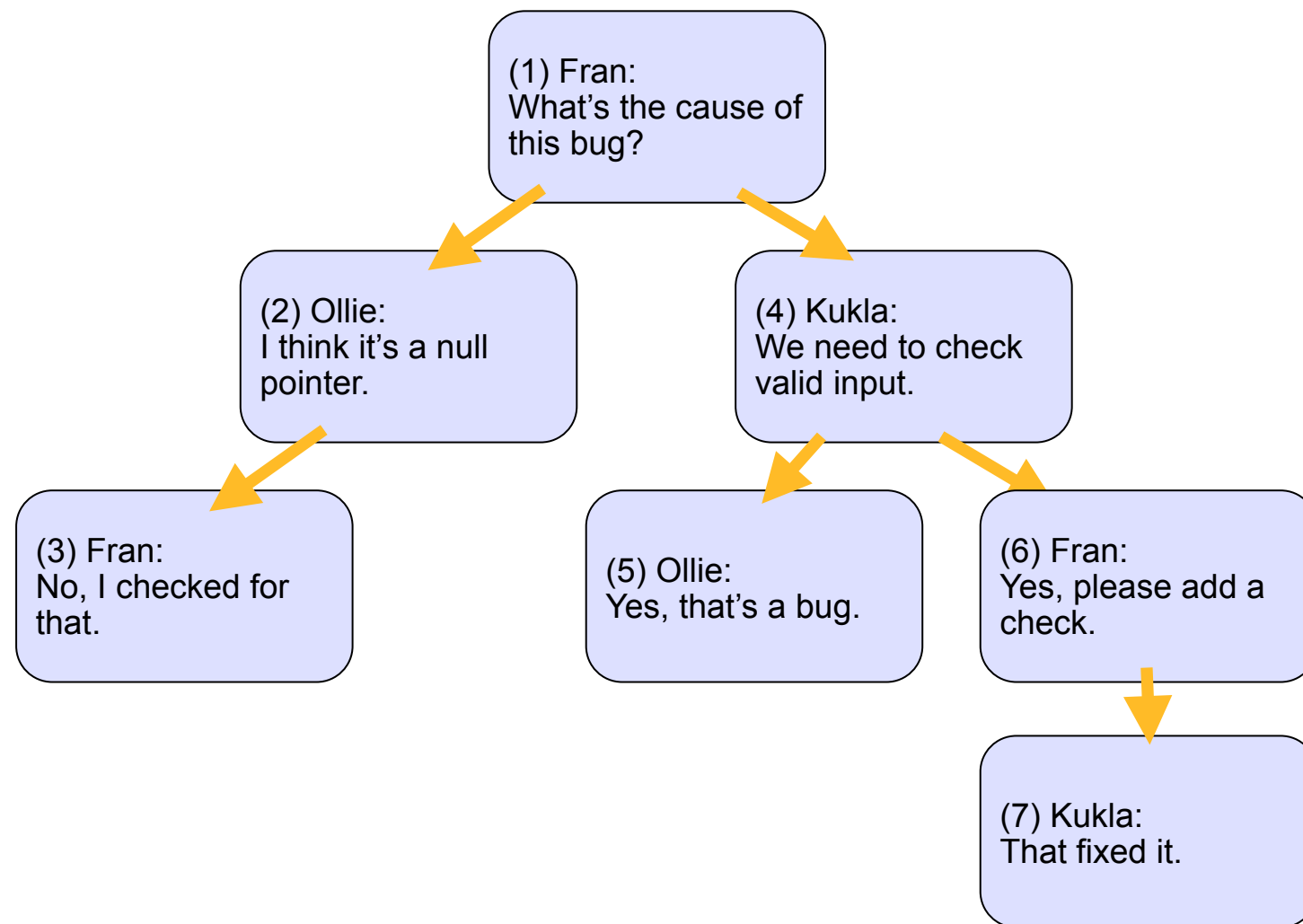
# Insert a New Node

INSERT INTO Comments (parent\_id, author, comment)  
VALUES (5, 'Fran', 'I agree!');



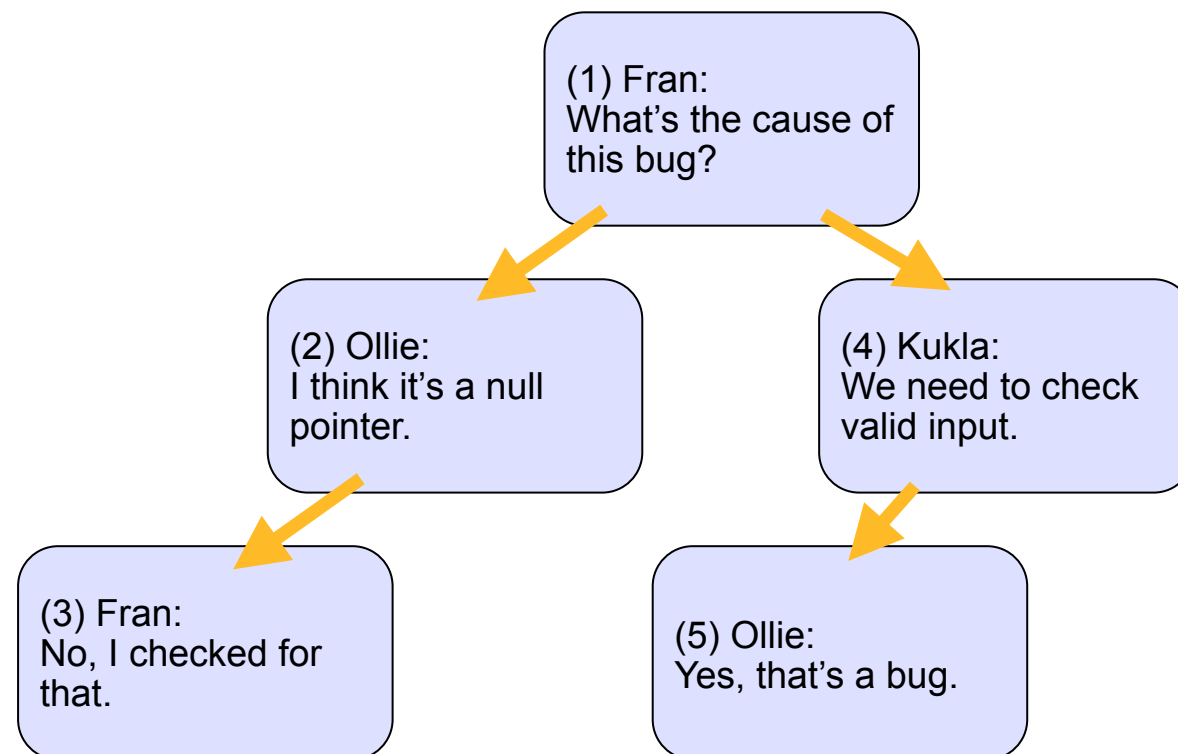
# Move a Node or Subtree

UPDATE Comments SET parent\_id = 3  
WHERE comment\_id = 6;



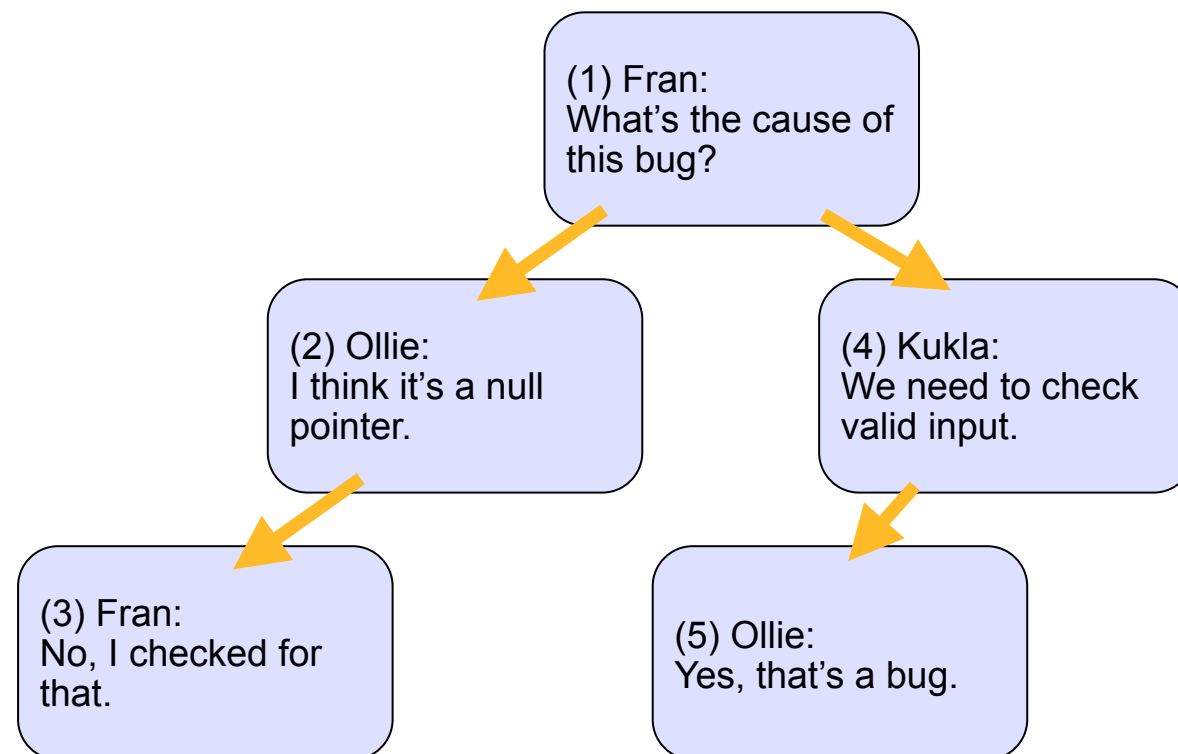
# Move a Node or Subtree

UPDATE Comments SET parent\_id = 3  
WHERE comment\_id = 6;



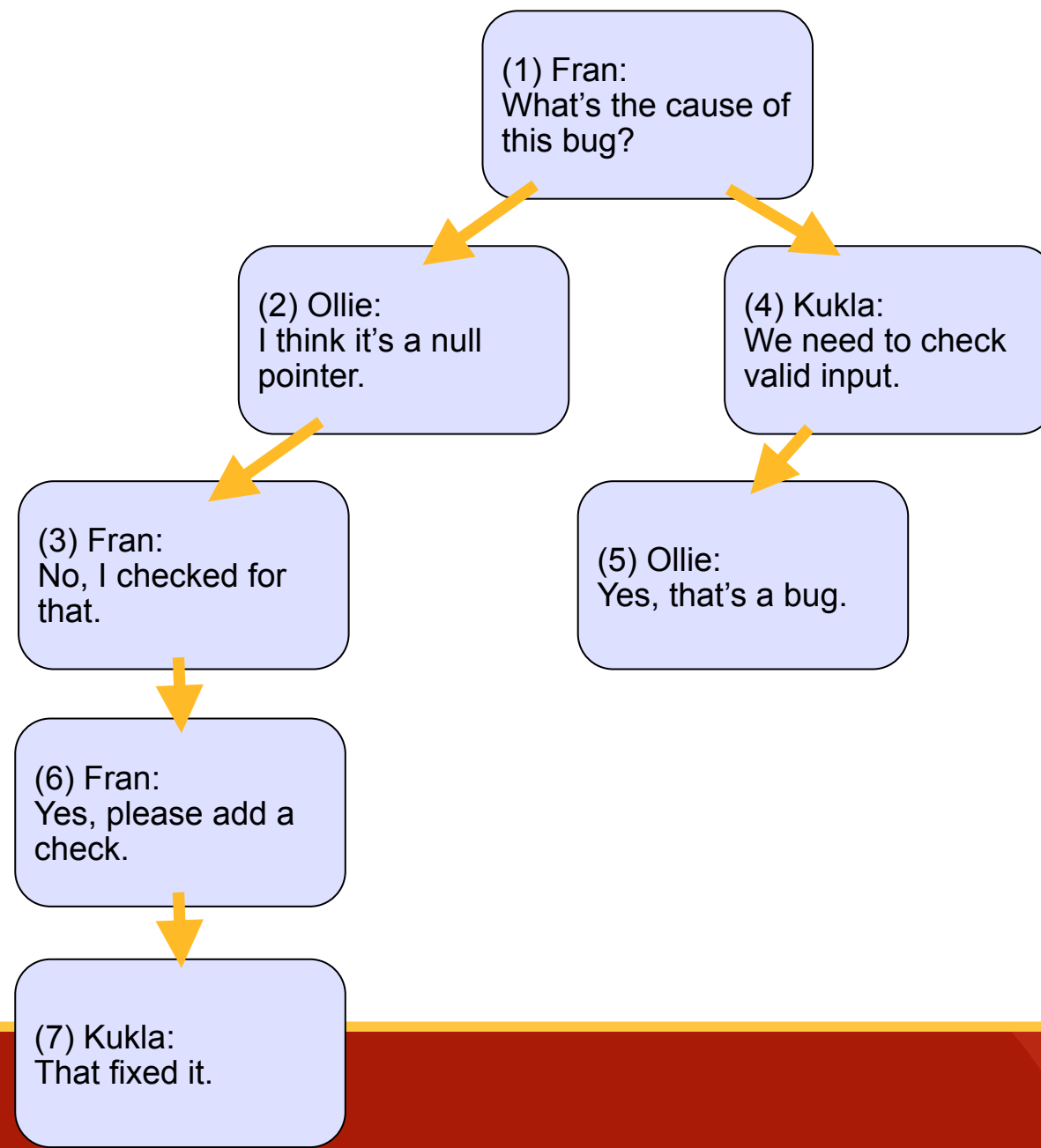
# Move a Node or Subtree

UPDATE Comments SET parent\_id = 3  
WHERE comment\_id = 6;



# Move a Node or Subtree

UPDATE Comments SET parent\_id = 3  
WHERE comment\_id = 6;



# Query Immediate Child/Parent

- Query a node's children:

```
SELECT * FROM Comments c1  
LEFT JOIN Comments c2  
ON (c2.parent_id = c1.comment_id);
```

- Query a node's parent:

```
SELECT * FROM Comments c1  
JOIN Comments c2  
ON (c1.parent_id = c2.comment_id);
```

# Can't Handle Deep Trees

```
SELECT * FROM Comments c1
  LEFT JOIN Comments c2 ON (c2.parent_id = c1.comment_id)
  LEFT JOIN Comments c3 ON (c3.parent_id = c2.comment_id)
  LEFT JOIN Comments c4 ON (c4.parent_id = c3.comment_id)
  LEFT JOIN Comments c5 ON (c5.parent_id = c4.comment_id)
  LEFT JOIN Comments c6 ON (c6.parent_id = c5.comment_id)
  LEFT JOIN Comments c7 ON (c7.parent_id = c6.comment_id)
  LEFT JOIN Comments c8 ON (c8.parent_id = c7.comment_id)
  LEFT JOIN Comments c9 ON (c9.parent_id = c8.comment_id)
  LEFT JOIN Comments c10 ON (c10.parent_id = c9.comment_id)
  . . .
```

# Can't Handle Deep Trees

```
SELECT * FROM Comments c1
  LEFT JOIN Comments c2 ON (c2.parent_id = c1.comment_id)
  LEFT JOIN Comments c3 ON (c3.parent_id = c2.comment_id)
  LEFT JOIN Comments c4 ON (c4.parent_id = c3.comment_id)
  LEFT JOIN Comments c5 ON (c5.parent_id = c4.comment_id)
  LEFT JOIN Comments c6 ON (c6.parent_id = c5.comment_id)
  LEFT JOIN Comments c7 ON (c7.parent_id = c6.comment_id)
  LEFT JOIN Comments c8 ON (c8.parent_id = c7.comment_id)
  LEFT JOIN Comments c9 ON (c9.parent_id = c8.comment_id)
  LEFT JOIN Comments c10 ON (c10.parent_id = c9.comment_id)
```

...

*it still doesn't support  
unlimited depth!*



# SQL-99 recursive syntax

```
WITH [RECURSIVE] CommentTree
    (comment_id, bug_id, parent_id, author, comment, depth)
AS (
    SELECT *, 0 AS depth FROM Comments
    WHERE parent_id IS NULL
    UNION ALL
    SELECT c.*, ct.depth+1 AS depth FROM CommentTree ct
    JOIN Comments c ON (ct.comment_id = c.parent_id)
)
SELECT * FROM CommentTree WHERE bug_id = 1234;
```



*PostgreSQL, Oracle 11g,  
IBM DB2, Microsoft SQL  
Server, Apache Derby*



*MySQL, SQLite, Informix,  
Firebird, etc.*

# Path Enumeration

# Path Enumeration


- Store chain of ancestors in each node

comment_id	path	author	comment
1	1/	Fran	What's the cause of this bug?
2	1/2/	Ollie	I think it's a null pointer.
3	1/2/3/	Fran	No, I checked for that.
4	1/4/	Kukla	We need to check valid input.
5	1/4/5/	Ollie	Yes, that's a bug.
6	1/4/6/	Fran	Yes, please add a check
7	1/4/6/7/	Kukla	That fixed it.

# Path Enumeration

- Store chain of ancestors in each node

*good for  
breadcrumbs*



comment_id	path	author	comment
1	1/	Fran	What's the cause of this bug?
2	1/2/	Ollie	I think it's a null pointer.
3	1/2/3/	Fran	No, I checked for that.
4	1/4/	Kukla	We need to check valid input.
5	1/4/5/	Ollie	Yes, that's a bug.
6	1/4/6/	Fran	Yes, please add a check
7	1/4/6/7/	Kukla	That fixed it.

# Query Ancestors and Subtrees

- Query ancestors of comment #7:

```
SELECT * FROM Comments  
WHERE '1/4/6/7/' LIKE path || '%';
```

- Query descendants of comment #4:

```
SELECT * FROM Comments  
WHERE path LIKE '1/4/%';
```

# Add a New Child of #7

```
INSERT INTO Comments (author, comment)
VALUES ('Ollie', 'Good job!');
SELECT path FROM Comments
WHERE comment_id = 7;
UPDATE Comments
SET path = $parent_path || LAST_INSERT_ID() || '/'
WHERE comment_id = LAST_INSERT_ID();
```

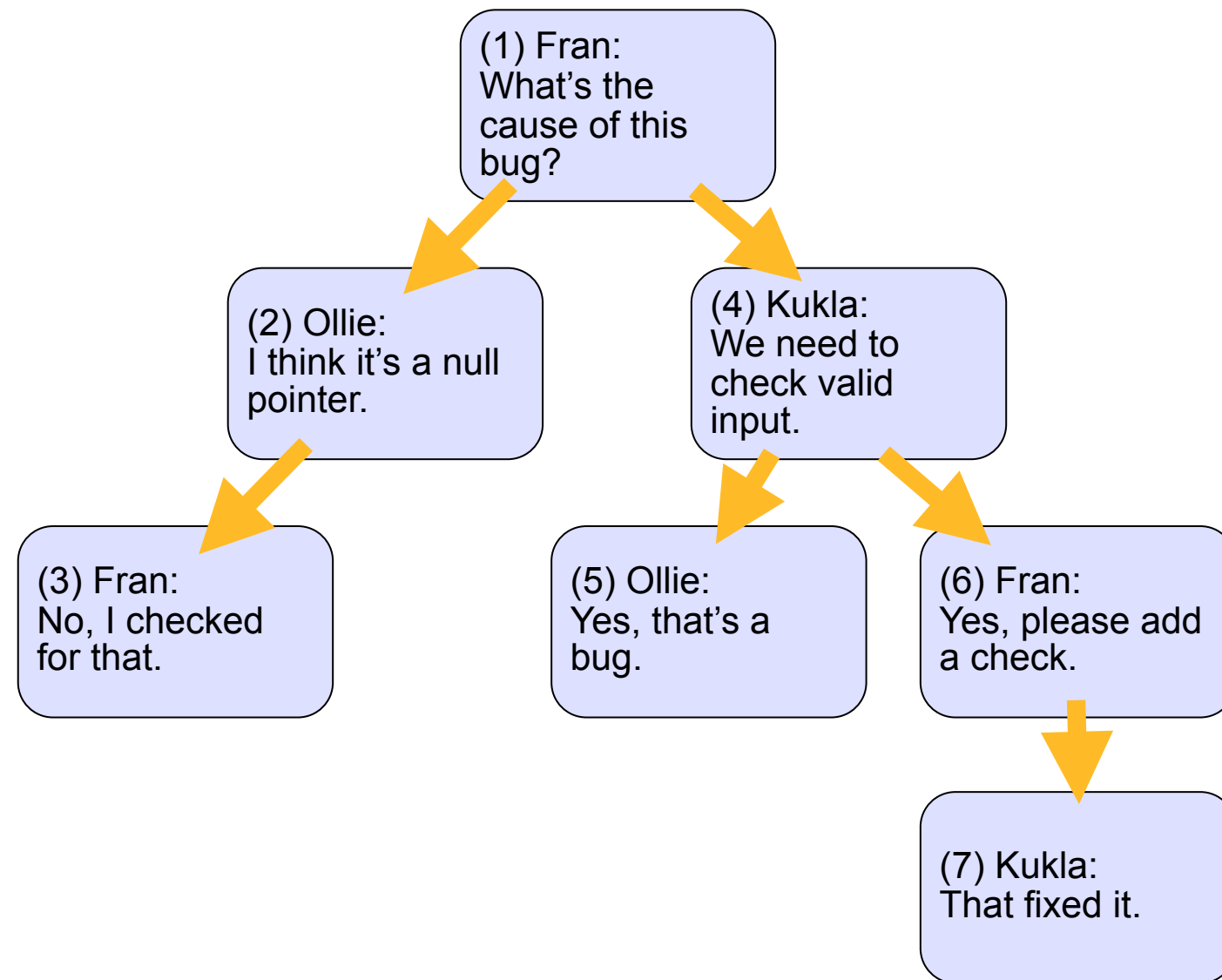
# Nested Sets

# Nested Sets

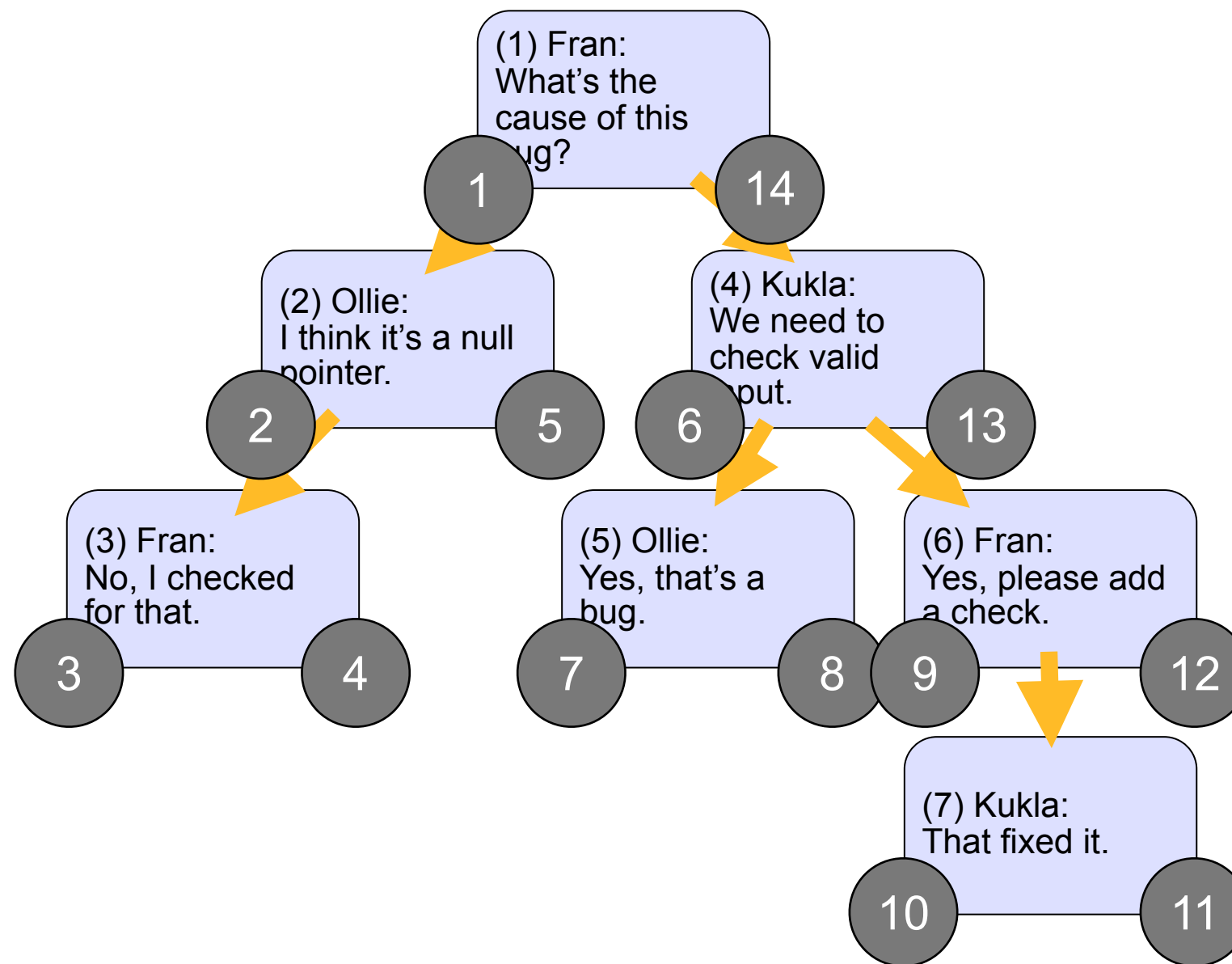
- Each comment encodes its descendants using two numbers:
  - A comment's **left** number is **less than** all numbers used by the comment's descendants.
  - A comment's **right** number is **greater than** all numbers used by the comment's descendants.
  - A comment's numbers are **between** all numbers used by the comment's ancestors.



# What Does This Look Like?



# What Does This Look Like?



# What Does This Look Like?

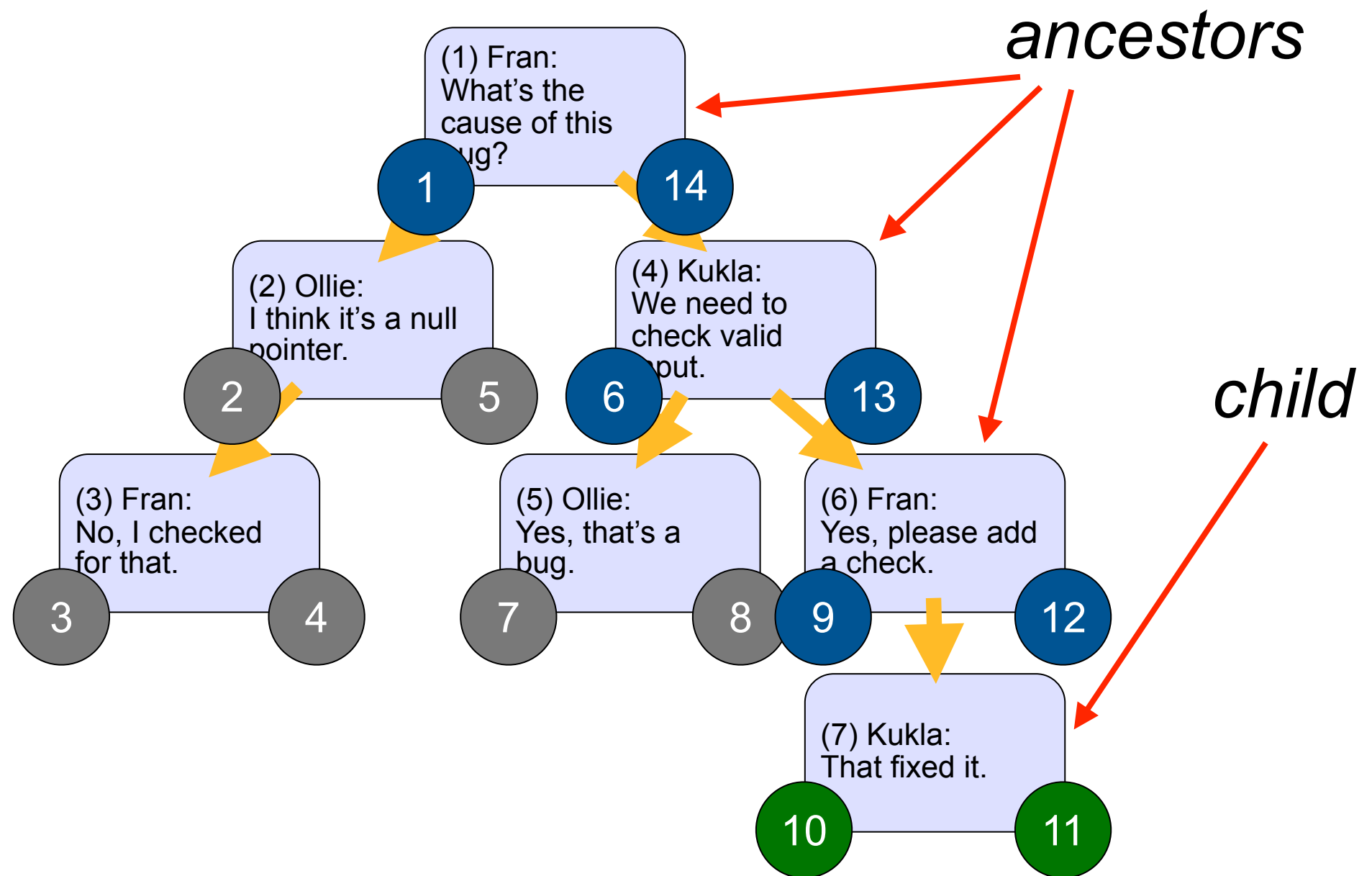
comment_id	nsleft	nsright	author	comment
1	1	14	Fran	What's the cause of this bug?
2	2	5	Ollie	I think it's a null pointer.
3	3	4	Fran	No, I checked for that.
4	6	13	Kukla	We need to check valid input.
5	7	8	Ollie	Yes, that's a bug.
6	9	12	Fran	Yes, please add a check
7	10	11	Kukla	That fixed it.

# What Does This Look Like?

comment_id	nsleft	nsright	author	comment
1	1	14	Fran	What's the cause of this bug?
2	2	5	Ollie	I think it's a null pointer.
3	3	4	Fran	No, I checked for that.
4	6	13	Kukla	We need to check valid input.
5	7	8	Ollie	Yes, that's a bug.
6	9	12	Fran	Yes, please add a check
7	10	11	Kukla	That fixed it.

*these are not  
foreign keys*

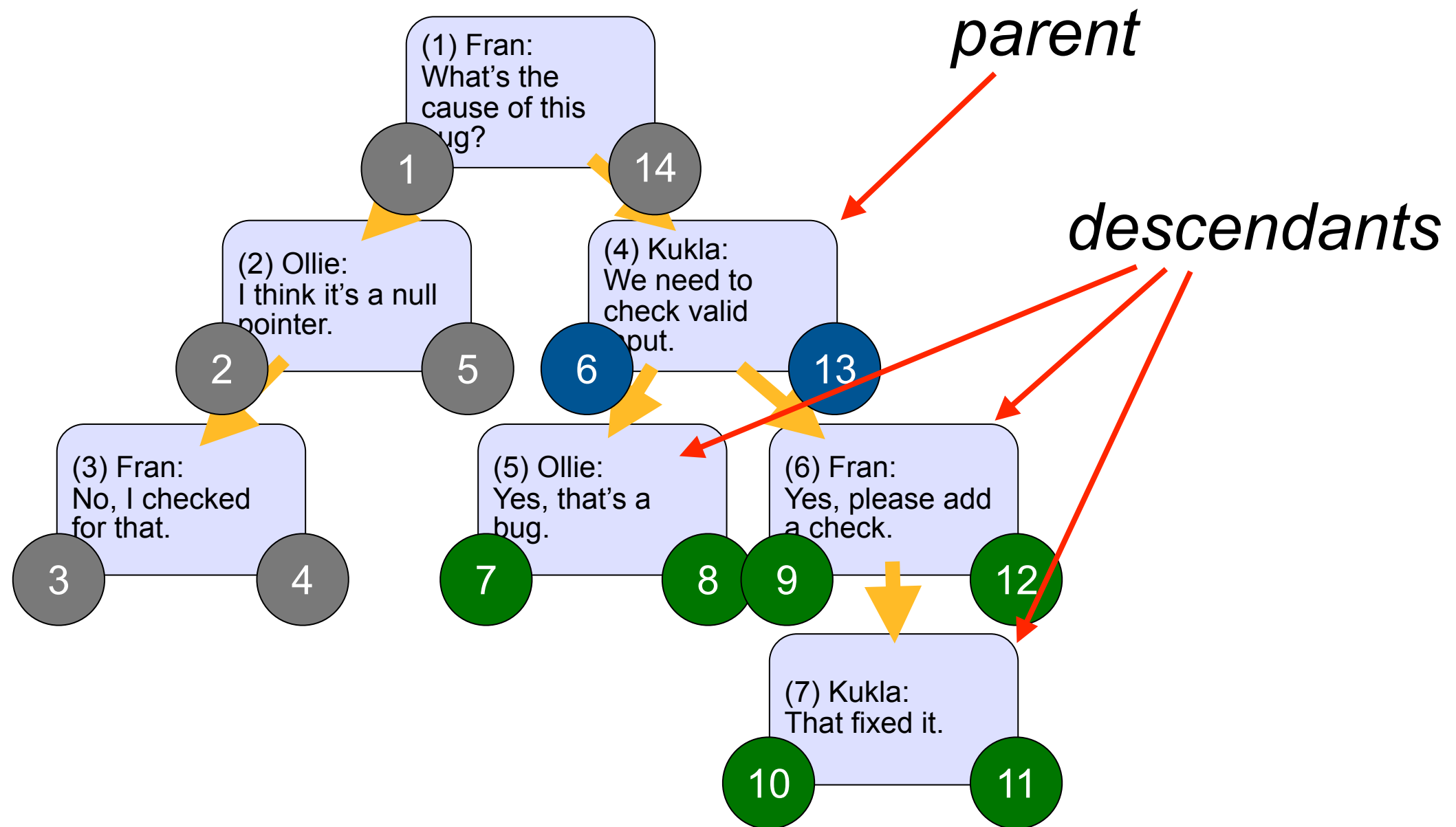
# Query Ancestors of #7



# Query Ancestors of #7

```
SELECT * FROM Comments child  
  JOIN Comments ancestor ON child.nsleft  
    BETWEEN ancestor.nsleft AND ancestor.nsright  
WHERE child.comment_id = 7;
```

# Query Subtree Under #4

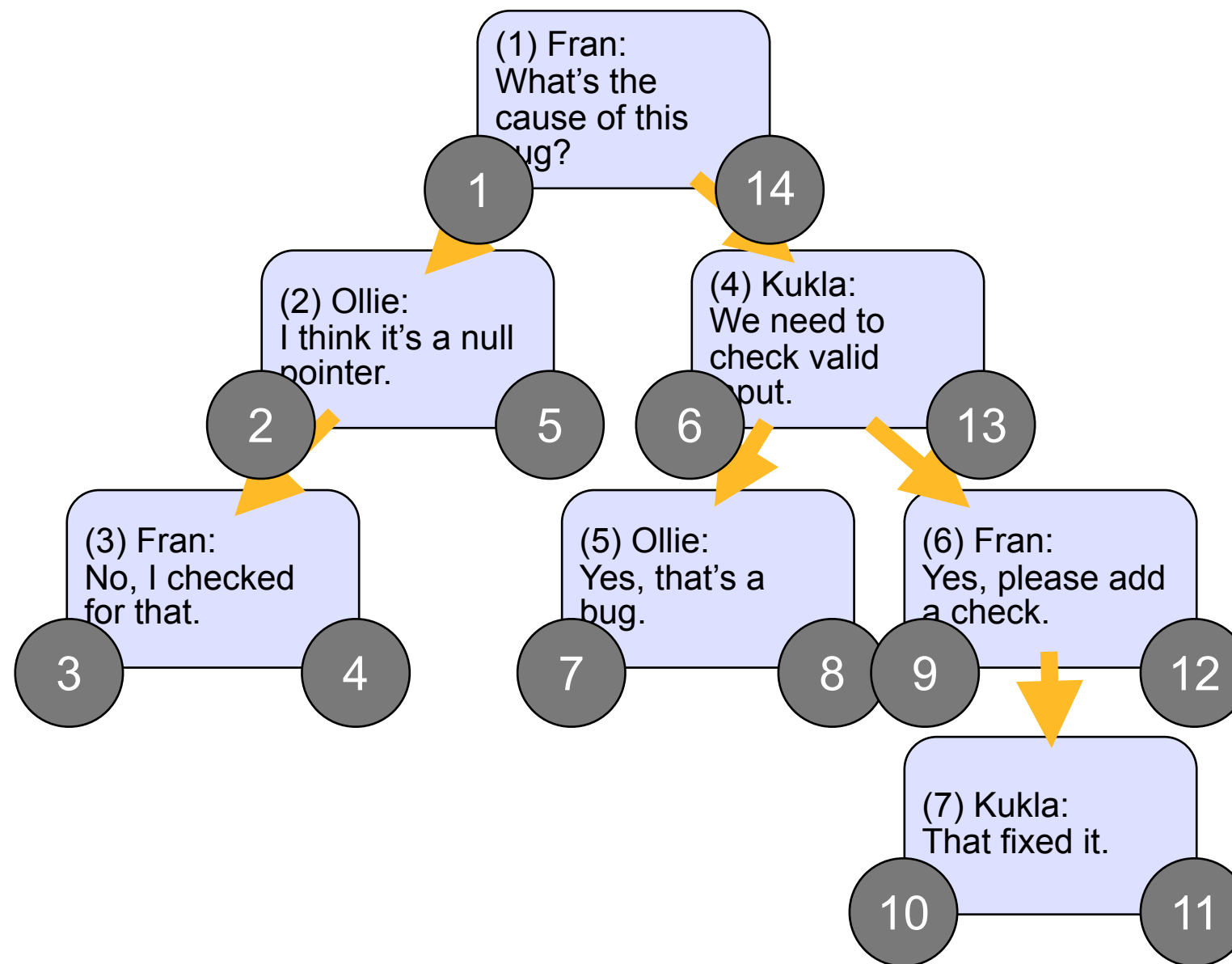


# Query Subtree Under #4

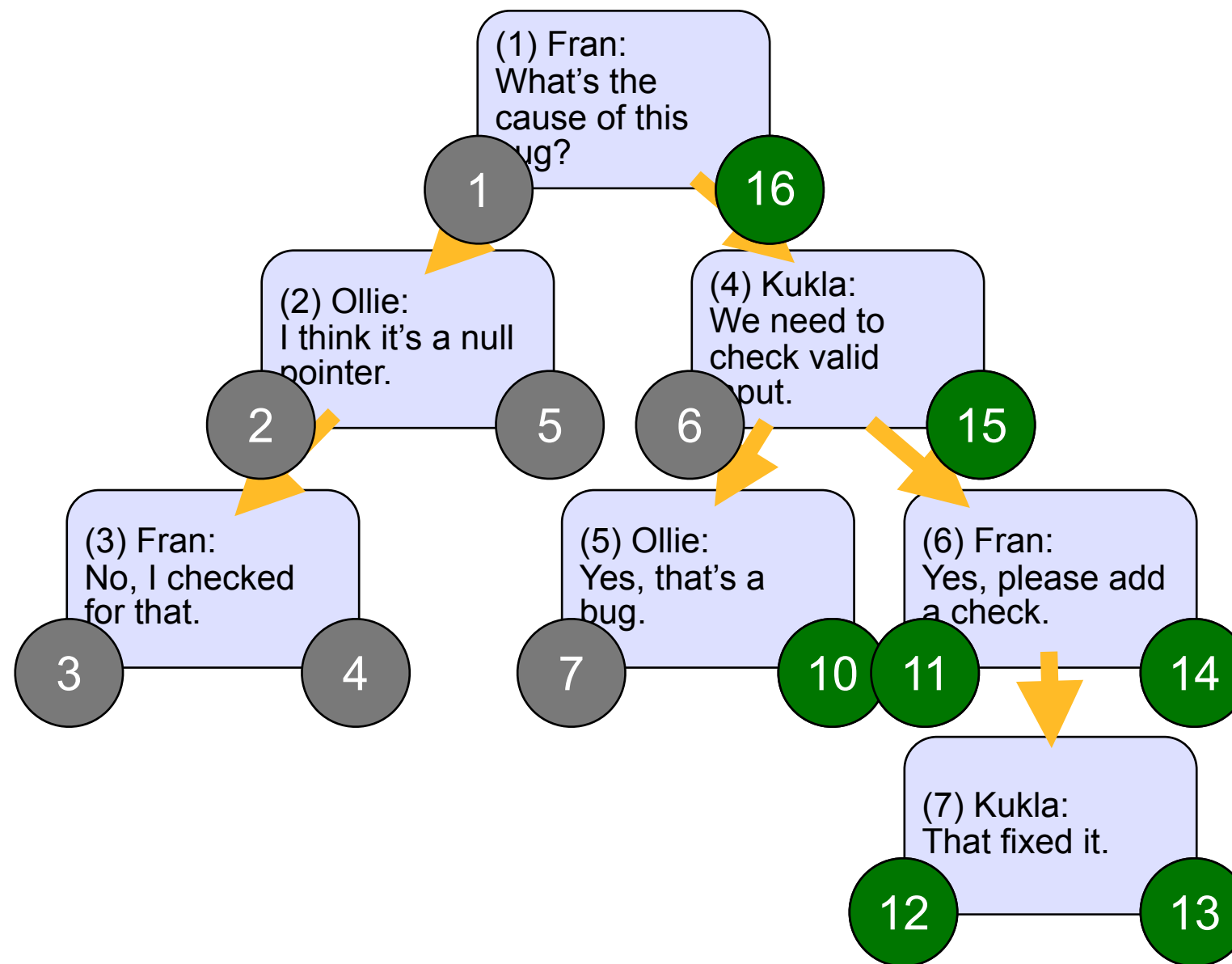
```
SELECT * FROM Comments parent
  JOIN Comments descendant ON descendant.nsleft
    BETWEEN parent.nsleft AND parent.nsright
 WHERE parent.comment_id = 4;
```



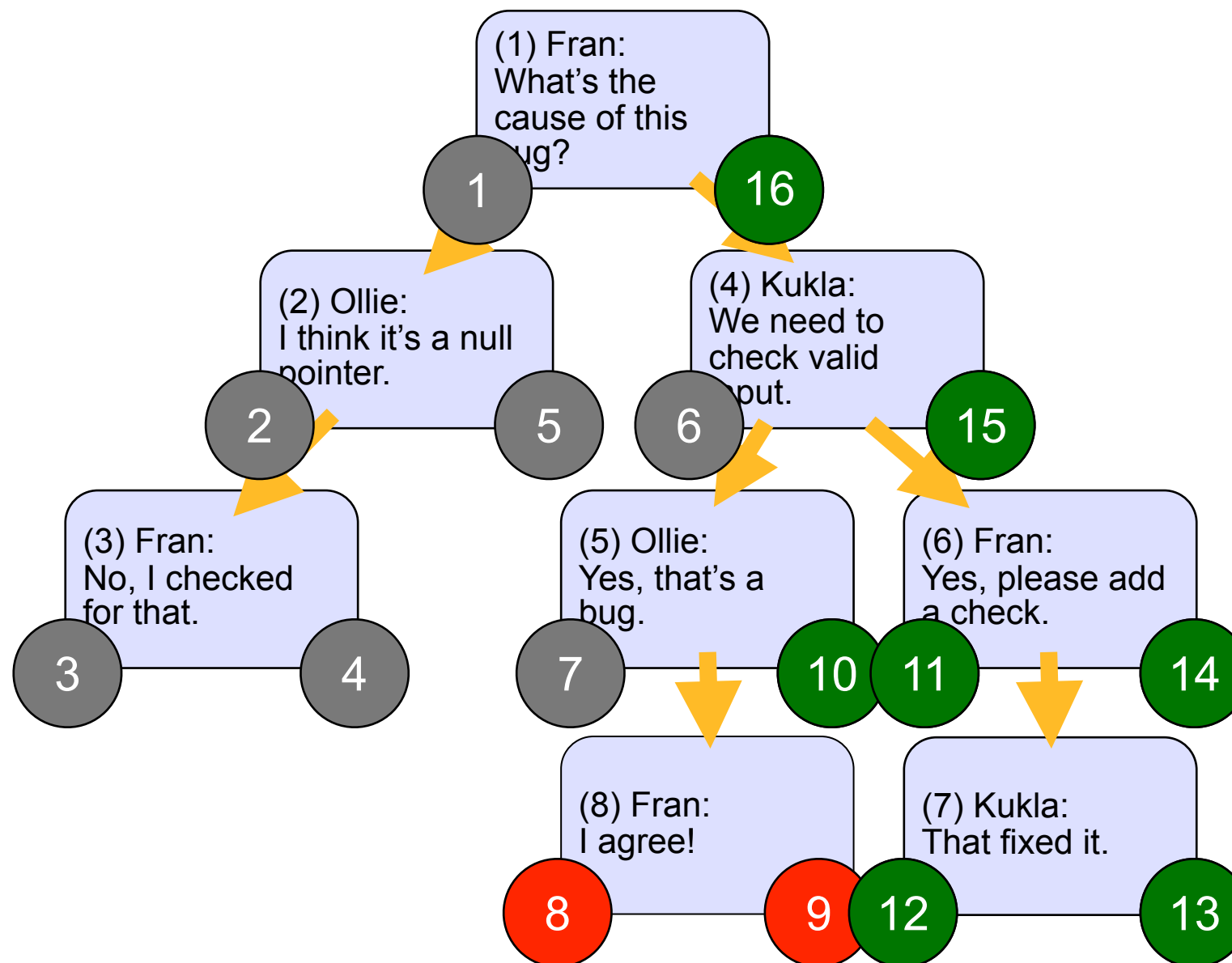
# Insert New Child of #5



# Insert New Child of #5



# Insert New Child of #5



# Insert New Child of #5

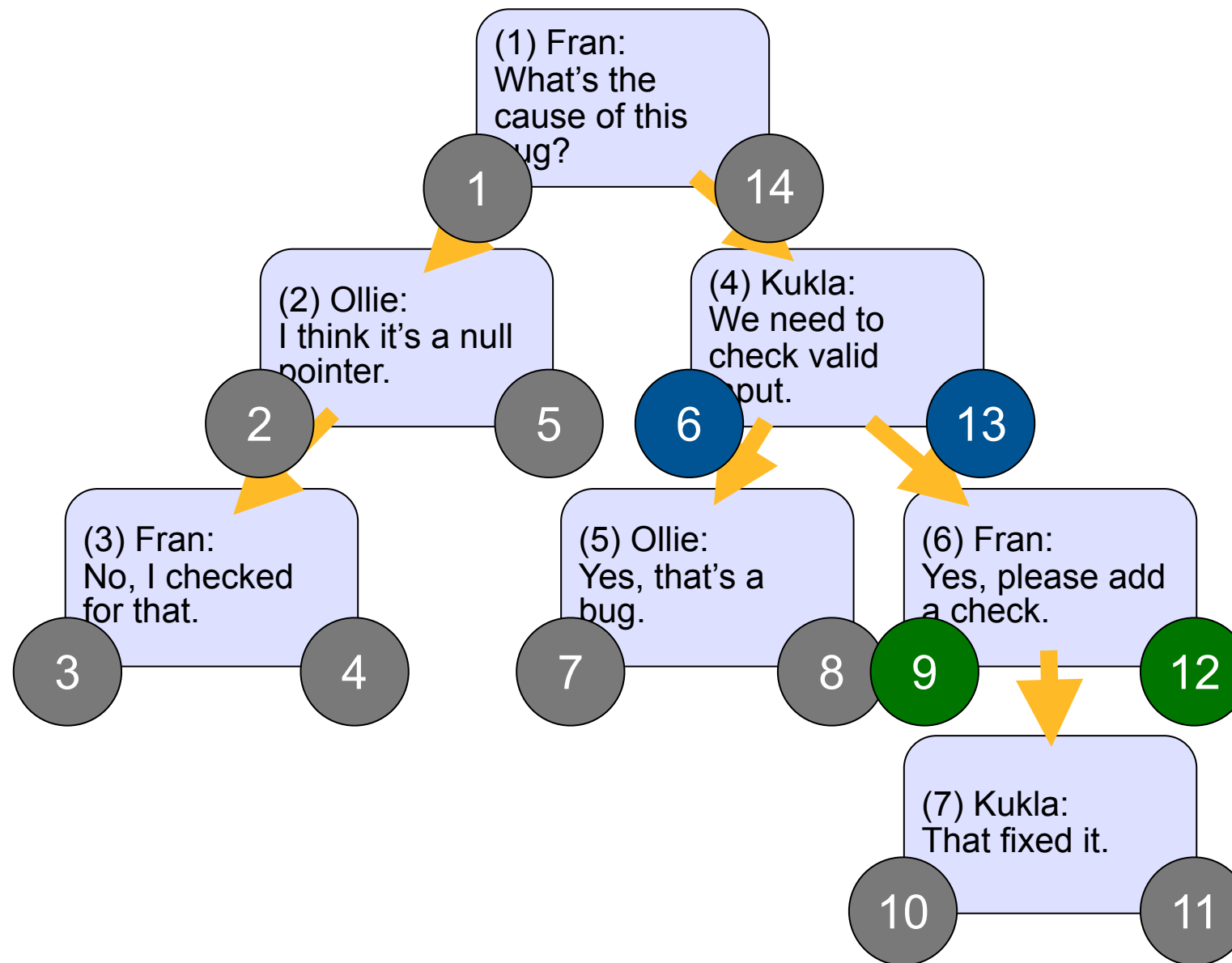
UPDATE Comments

```
SET    nsleft = CASE WHEN nsleft >= 8 THEN nsleft+2  
      ELSE nsleft END,  
      nsright = nsright+2  
WHERE nsright >= 7;
```

```
INSERT INTO Comments (nsleft, nsright, author, comment)  
VALUES (8, 9, 'Fran', 'I agree!');
```

- Recalculate ***left*** values for all nodes to the right of the new child. Recalculate ***right*** values for all nodes above and to the right.

# Query Immediate Parent of #6



# Query Immediate Parent of #6

- Parent of #6 is an ancestor who has no descendant who is also an ancestor of #6.

```
SELECT parent.* FROM Comments AS c
  JOIN Comments AS parent
    ON (c.nsleft BETWEEN parent.nsleft AND parent.nsright)
  LEFT OUTER JOIN Comments AS in_between
    ON (c.nsleft BETWEEN in_between.nsleft AND in_between.nsright
        AND in_between.nsleft BETWEEN parent.nsleft AND parent.nsright)
 WHERE c.comment_id = 6 AND in_between.comment_id IS NULL;
```

# Query Immediate Parent of #6

- Parent of #6 is an ancestor who has no descendant who is also an ancestor of #6.

```
SELECT parent.* FROM Comments AS c
  JOIN Comments AS parent
    ON (c.nsleft BETWEEN parent.nsleft AND parent.nsright)
  LEFT OUTER JOIN Comments AS in_between
    ON (c.nsleft BETWEEN in_between.nsleft AND in_between.nsright
        AND in_between.nsleft BETWEEN parent.nsleft AND parent.nsright)
 WHERE c.comment_id = 6 AND in_between.comment_id IS NULL;
```

*querying immediate child  
is a similar problem*

# Closure Table



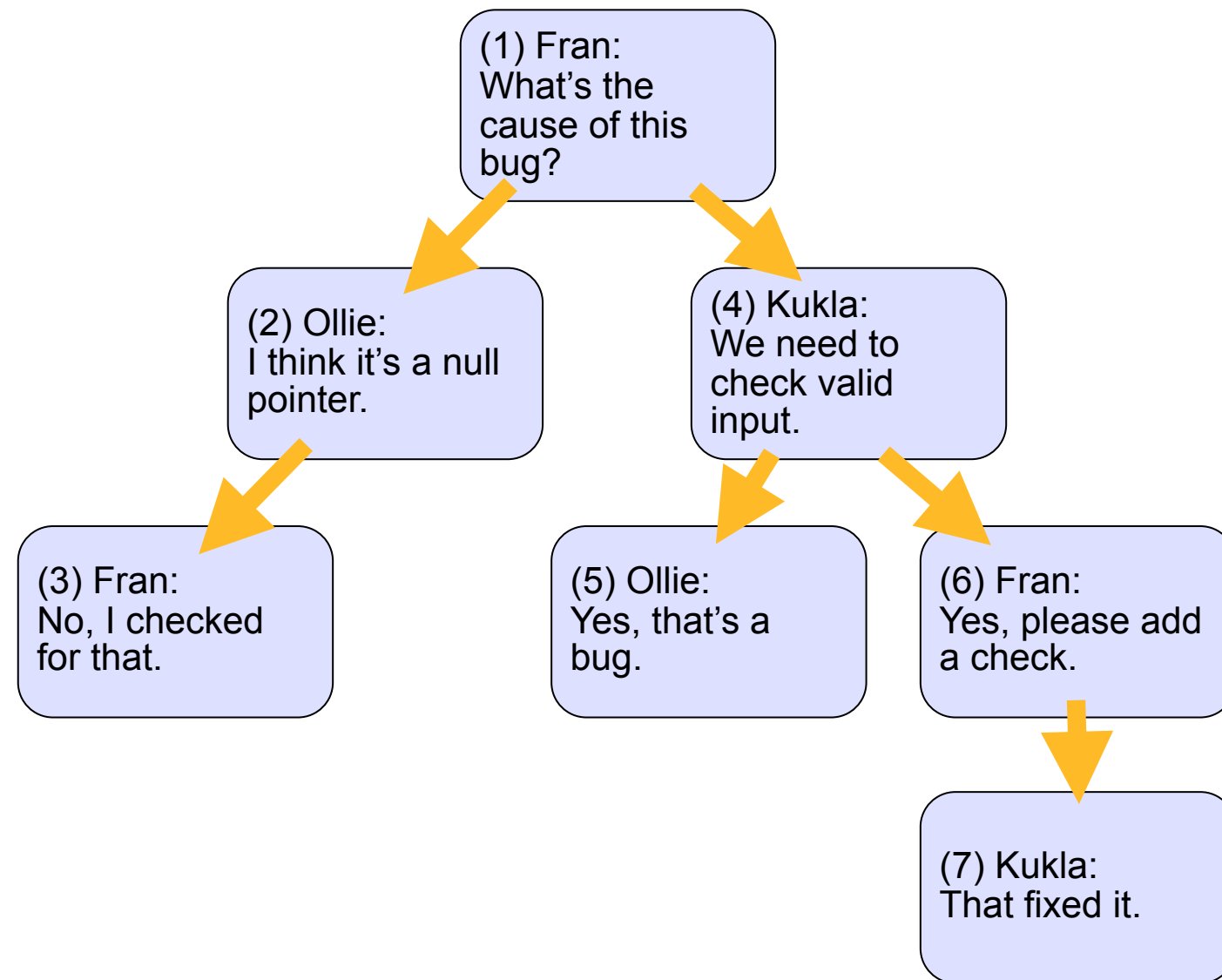
# Closure Table

```
CREATE TABLE TreePaths (  
    ancestor      INT NOT NULL,  
    descendant    INT NOT NULL,  
    PRIMARY KEY (ancestor, descendant),  
    FOREIGN KEY(ancestor)  
        REFERENCES Comments(comment_id),  
    FOREIGN KEY(descendant)  
        REFERENCES Comments(comment_id)  
);
```

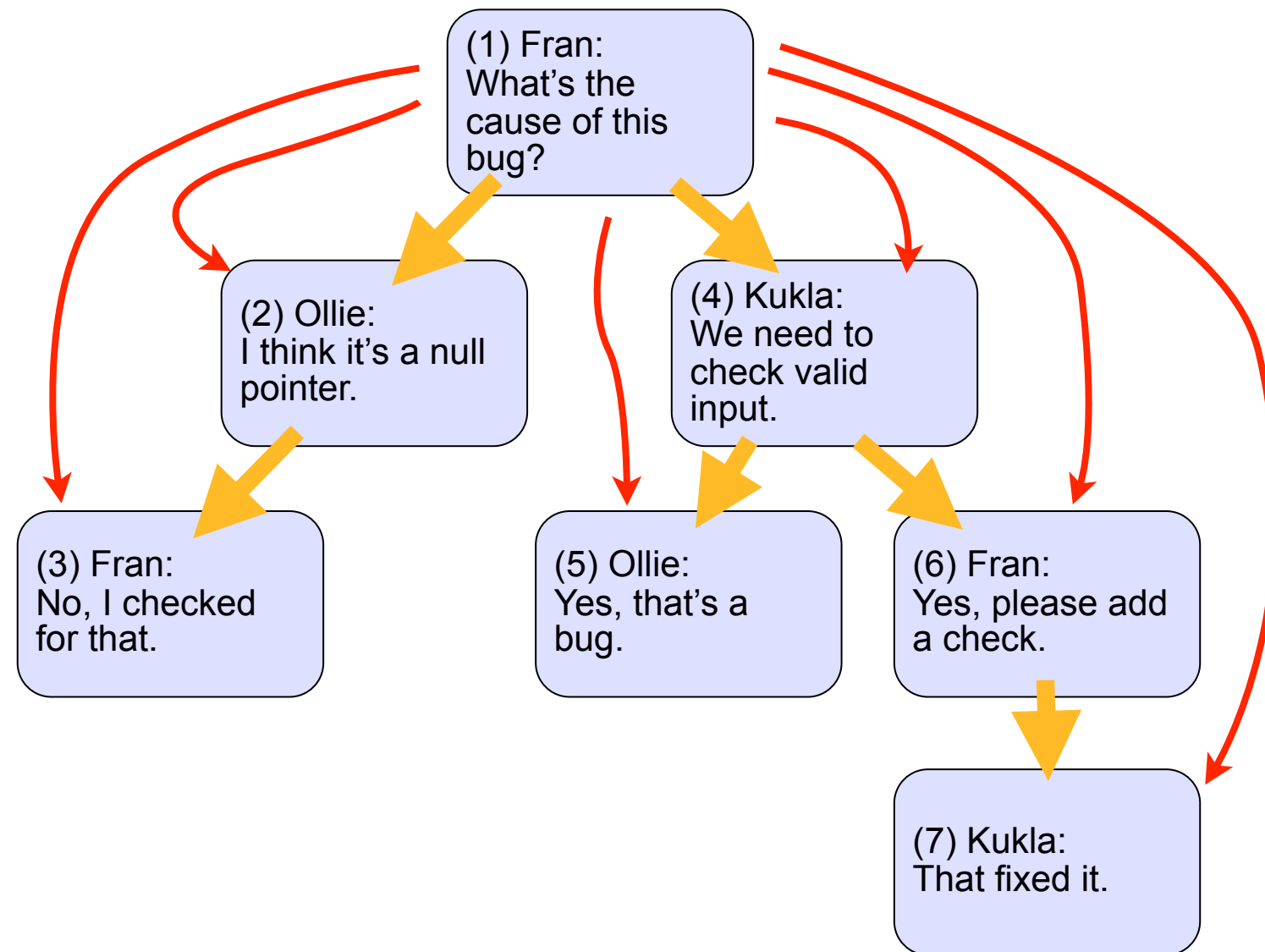
# Closure Table

- Many-to-many table
- Stores every path from each node to each of its descendants
- A node even connects to itself

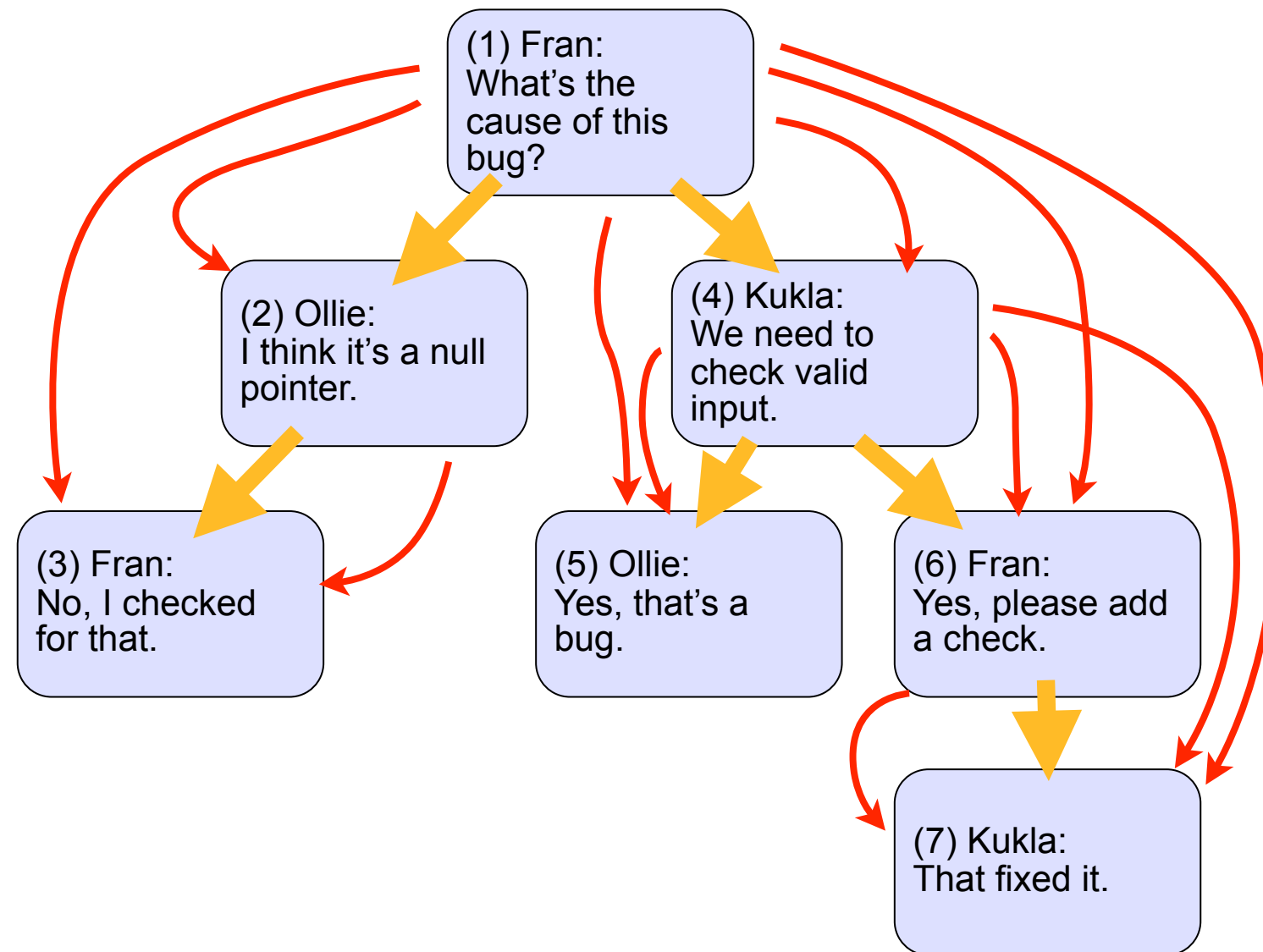
# Closure Table illustration



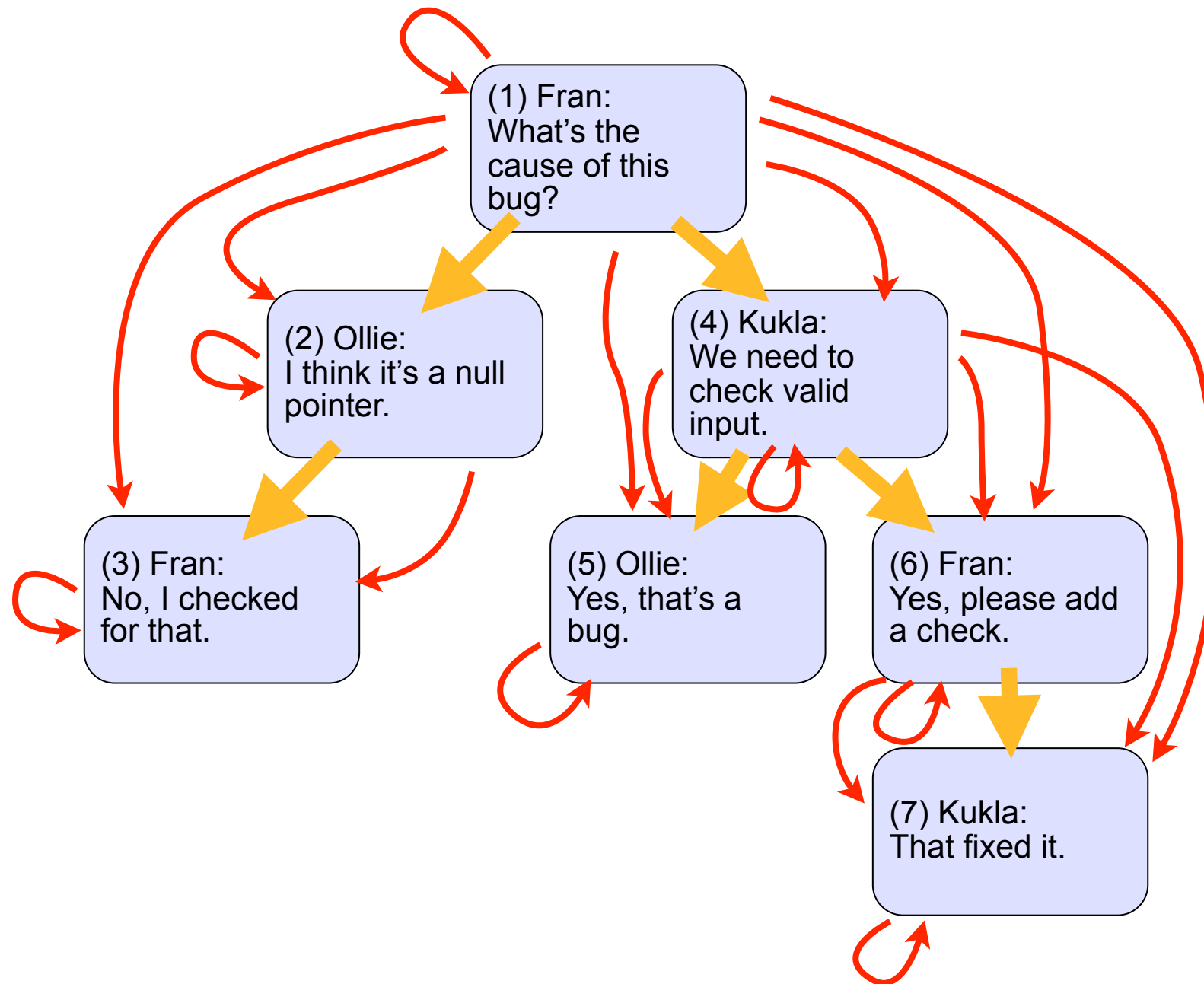
# Closure Table illustration



# Closure Table illustration



# Closure Table illustration



# What Does This Look Like?

comment_id	author	comment
1	Fran	What's the cause of this bug?
2	Ollie	I think it's a null pointer.
3	Fran	No, I checked for that.
4	Kukla	We need to check valid input.
5	Ollie	Yes, that's a bug.
6	Fran	Yes, please add a check
7	Kukla	That fixed it.

*requires  $O(n^2)$  rows*

ancestor	descendant
1	1
1	2
1	3
1	4
1	5
1	6
1	7
2	2
2	3
3	3
4	4
4	5
4	6
4	7
5	5
6	6
6	7
7	7

# What Does This Look Like?

comment_id	author	comment
1	Fran	What's the cause of this bug?
2	Ollie	I think it's a null pointer.
3	Fran	No, I checked for that.
4	Kukla	We need to check valid input.
5	Ollie	Yes, that's a bug.
6	Fran	Yes, please add a check
7	Kukla	That fixed it.

*requires  $O(n^2)$  rows*  
*(but far fewer in practice)*

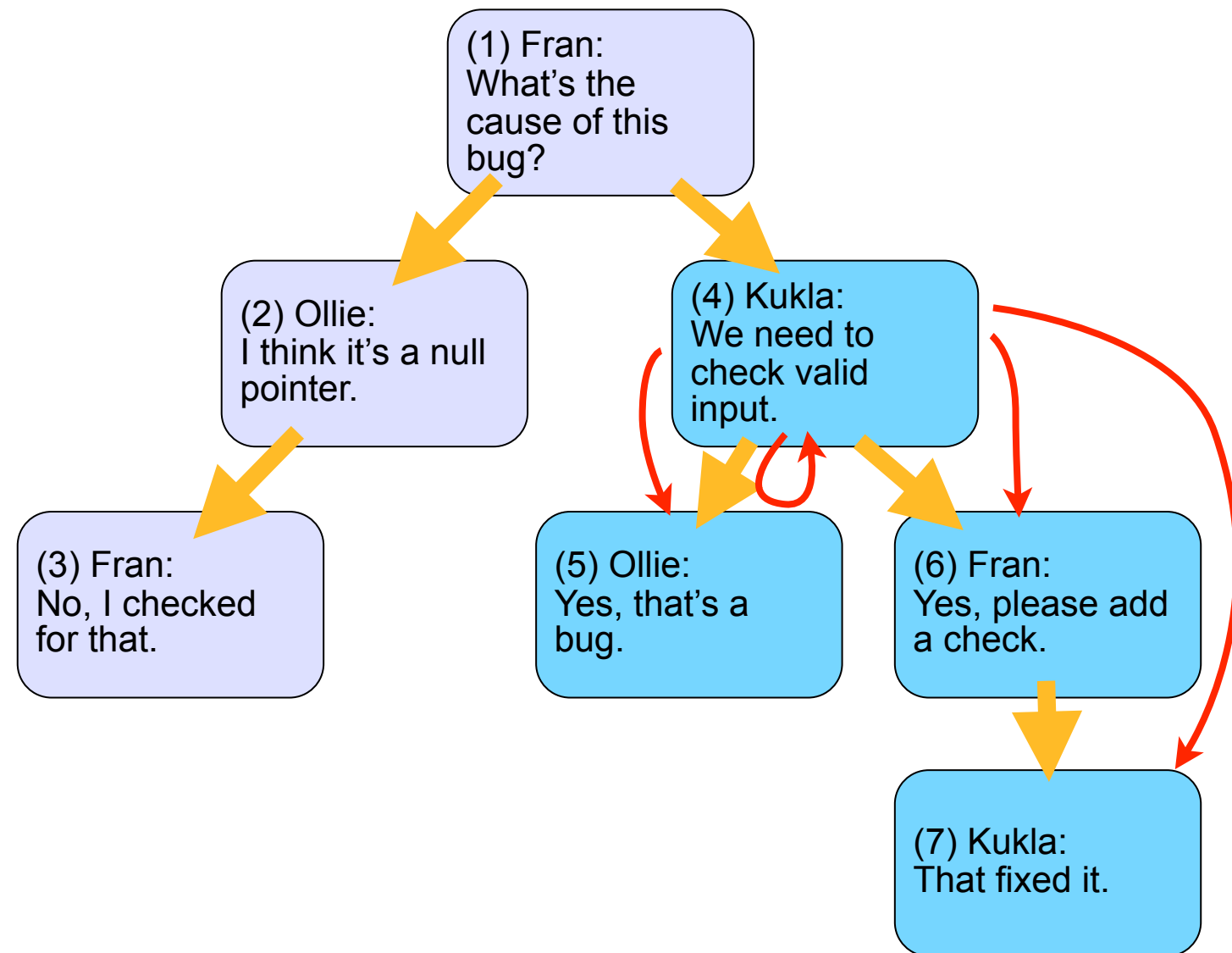
ancestor	descendant
1	1
1	2
1	3
1	4
1	5
1	6
1	7
2	2
2	3
3	3
4	4
4	5
4	6
4	7
5	5
6	6
6	7
7	7



# Query Descendants of #4

```
SELECT c.* FROM Comments c
JOIN TreePaths t
  ON (c.comment_id = t.descendant)
WHERE t.ancestor = 4;
```

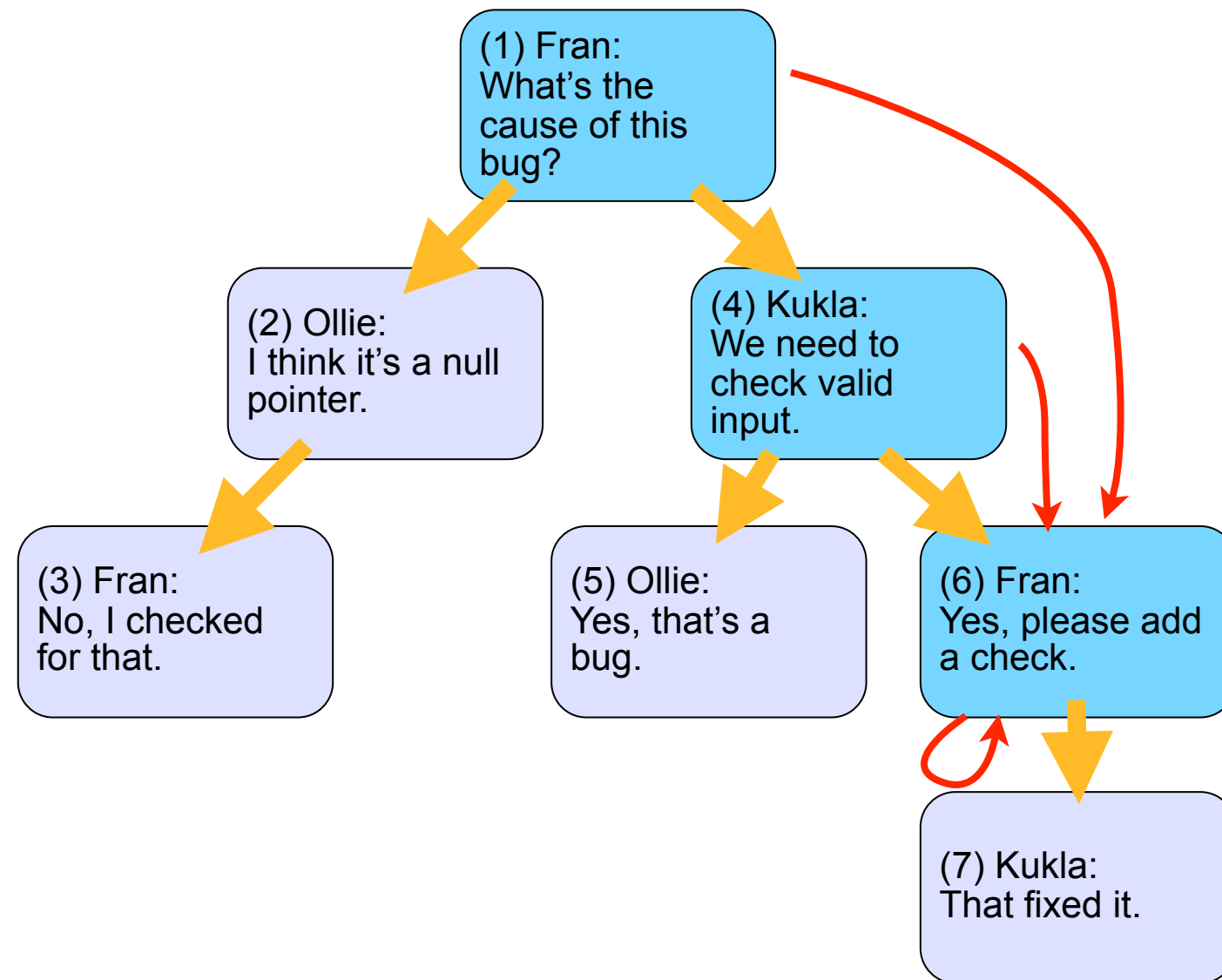
# Paths Starting from #4



# Query Ancestors of #6

```
SELECT c.* FROM Comments c
JOIN TreePaths t
  ON (c.comment_id = t.ancestor)
WHERE t.descendant = 6;
```

# Paths Terminating at #6

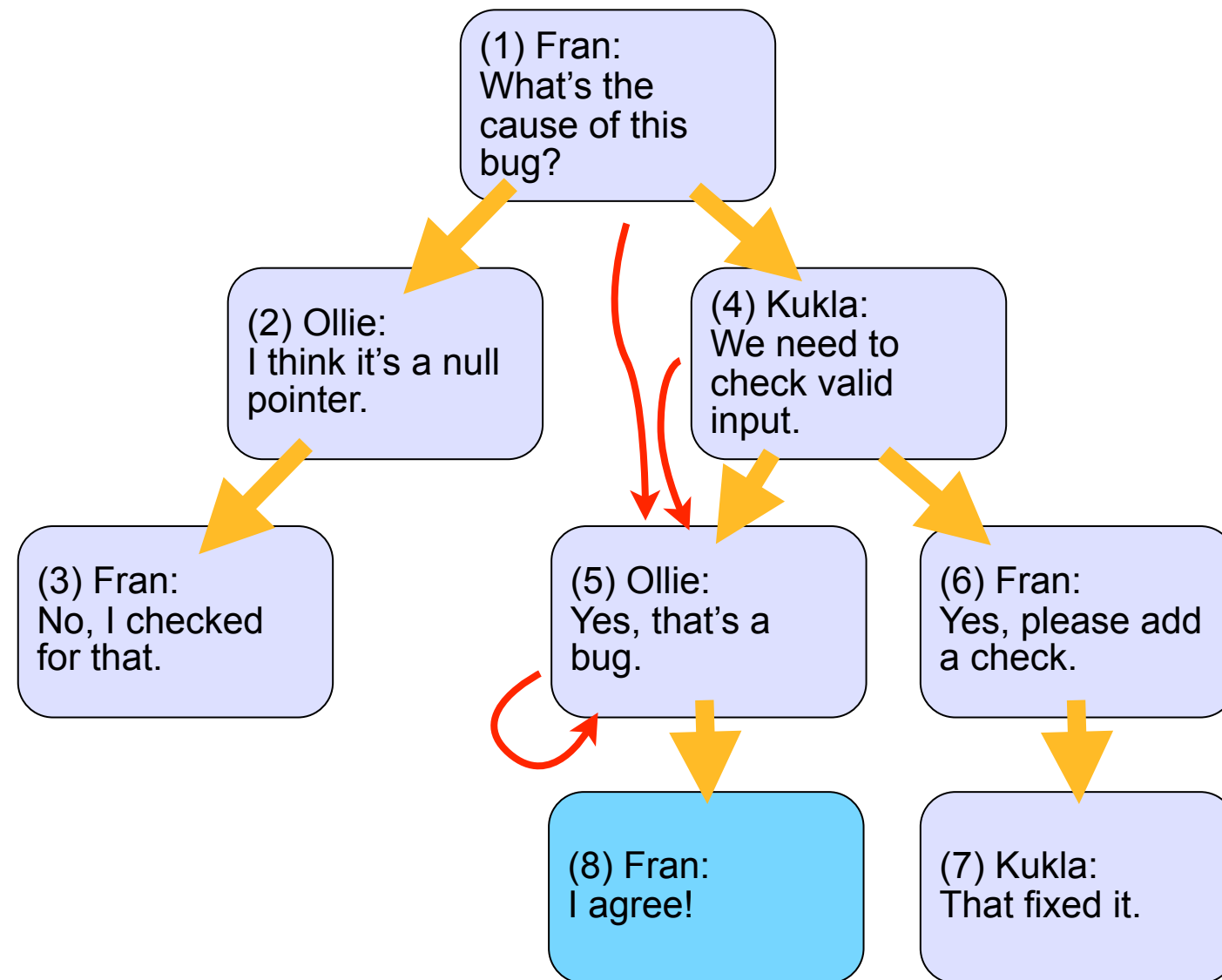


# Insert New Child of #5

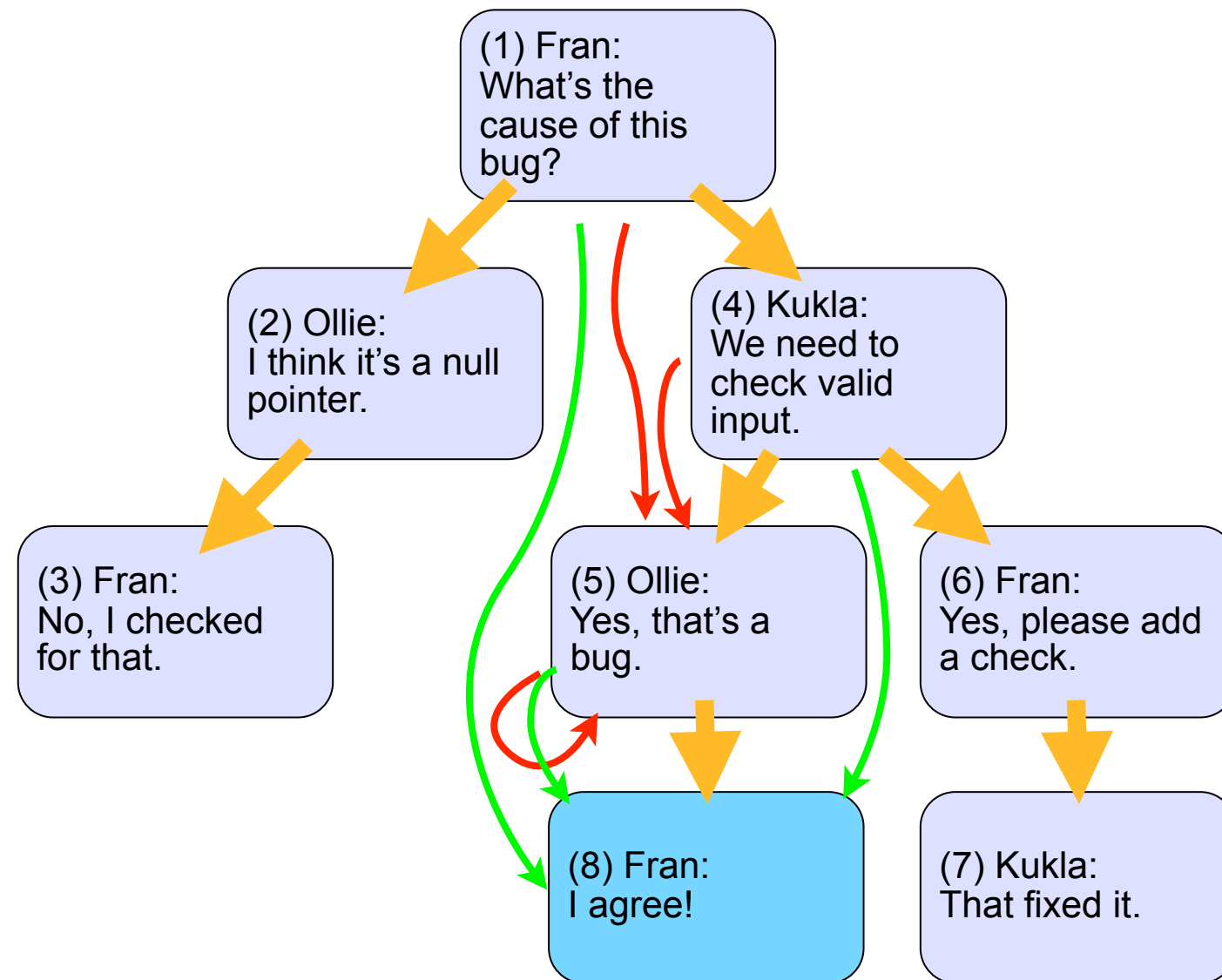
```
INSERT INTO Comments  
VALUES (8, 'Fran', 'I agree!');
```

```
INSERT INTO TreePaths (ancestor, descendant)  
SELECT ancestor, 8 FROM TreePaths  
WHERE descendant = 5  
UNION ALL SELECT 8, 8;
```

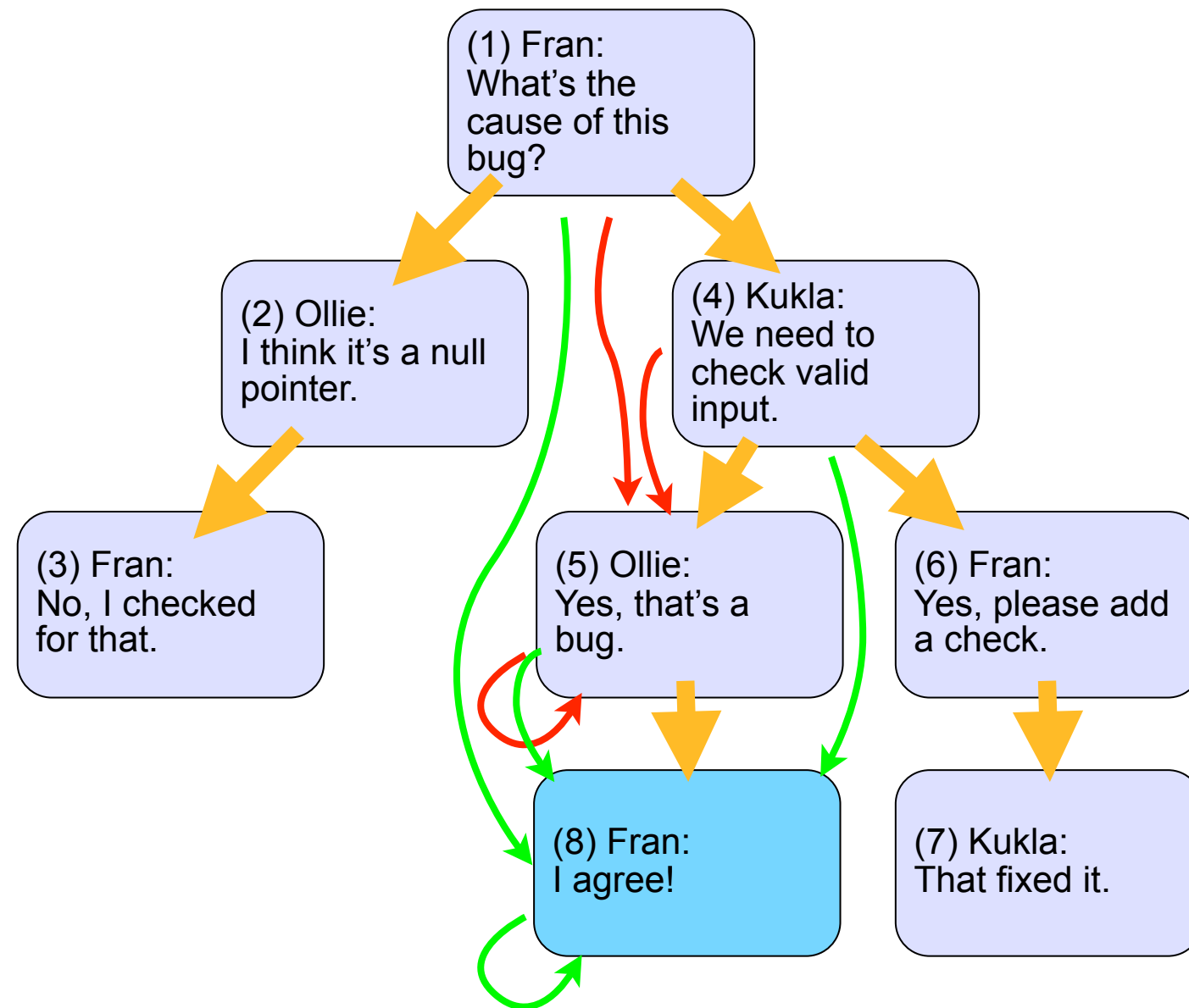
# Copy Paths from Parent



# Copy Paths from Parent



# Copy Paths from Parent

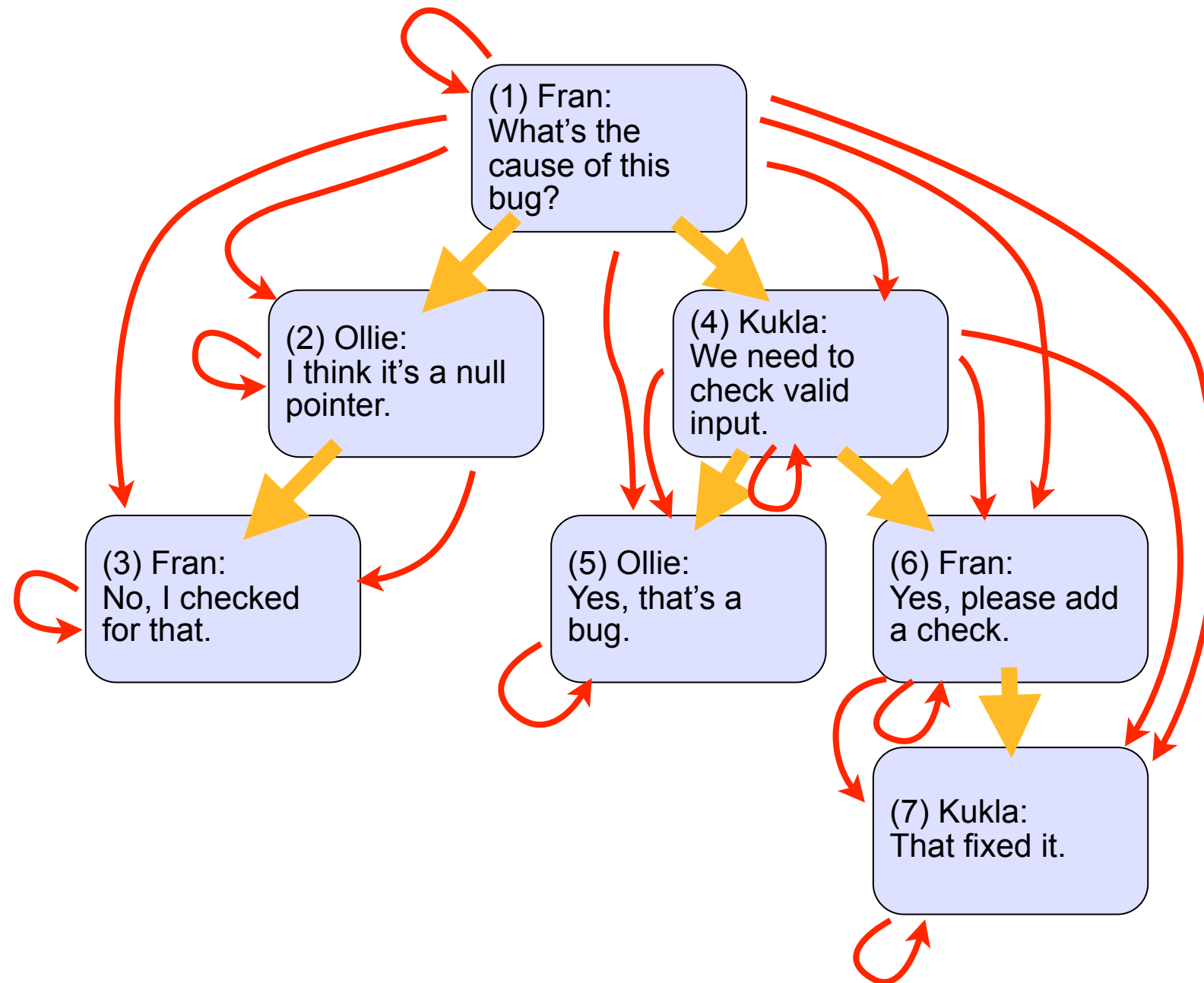




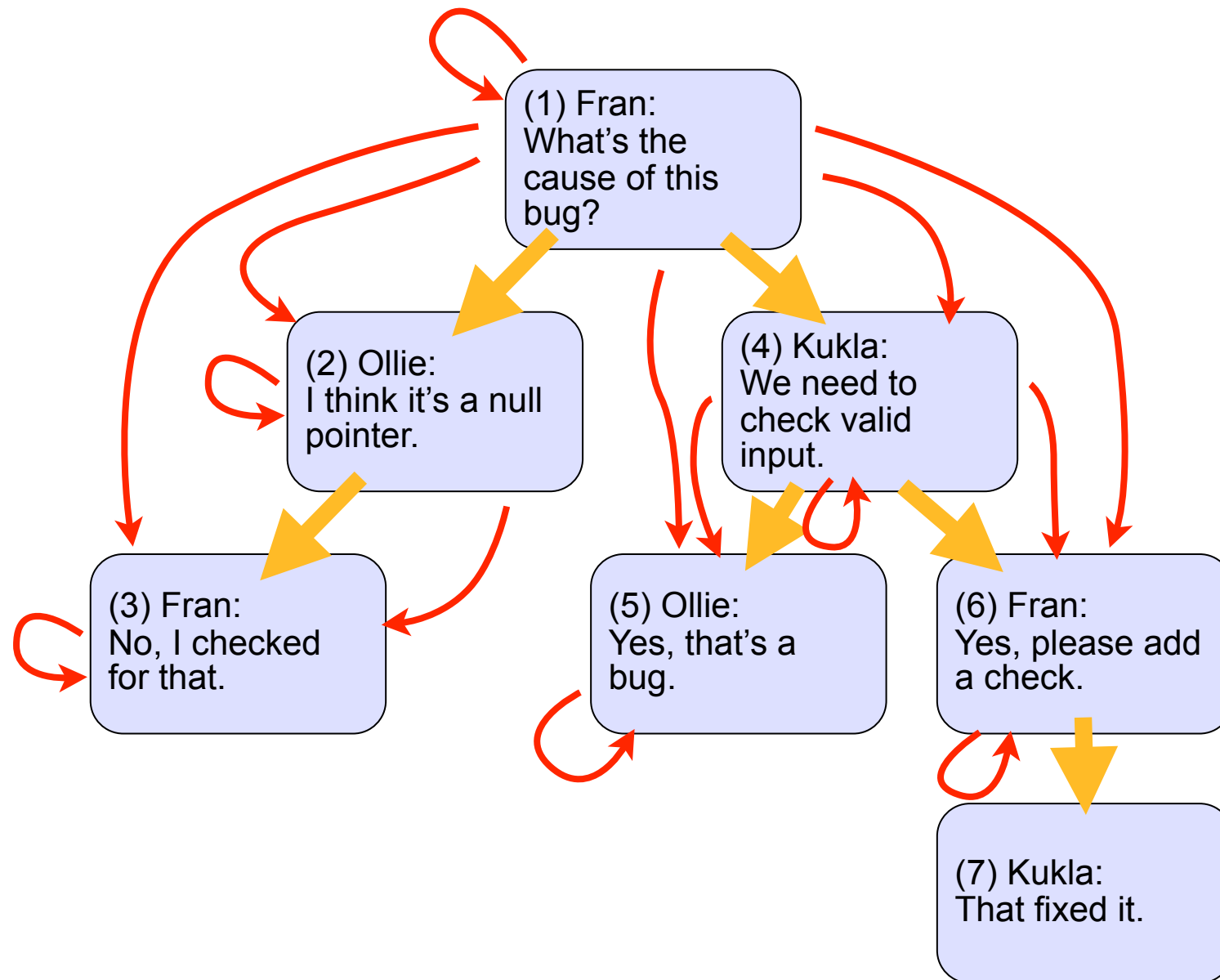
# Delete Child #7

```
DELETE FROM TreePaths  
WHERE descendant = 7;
```

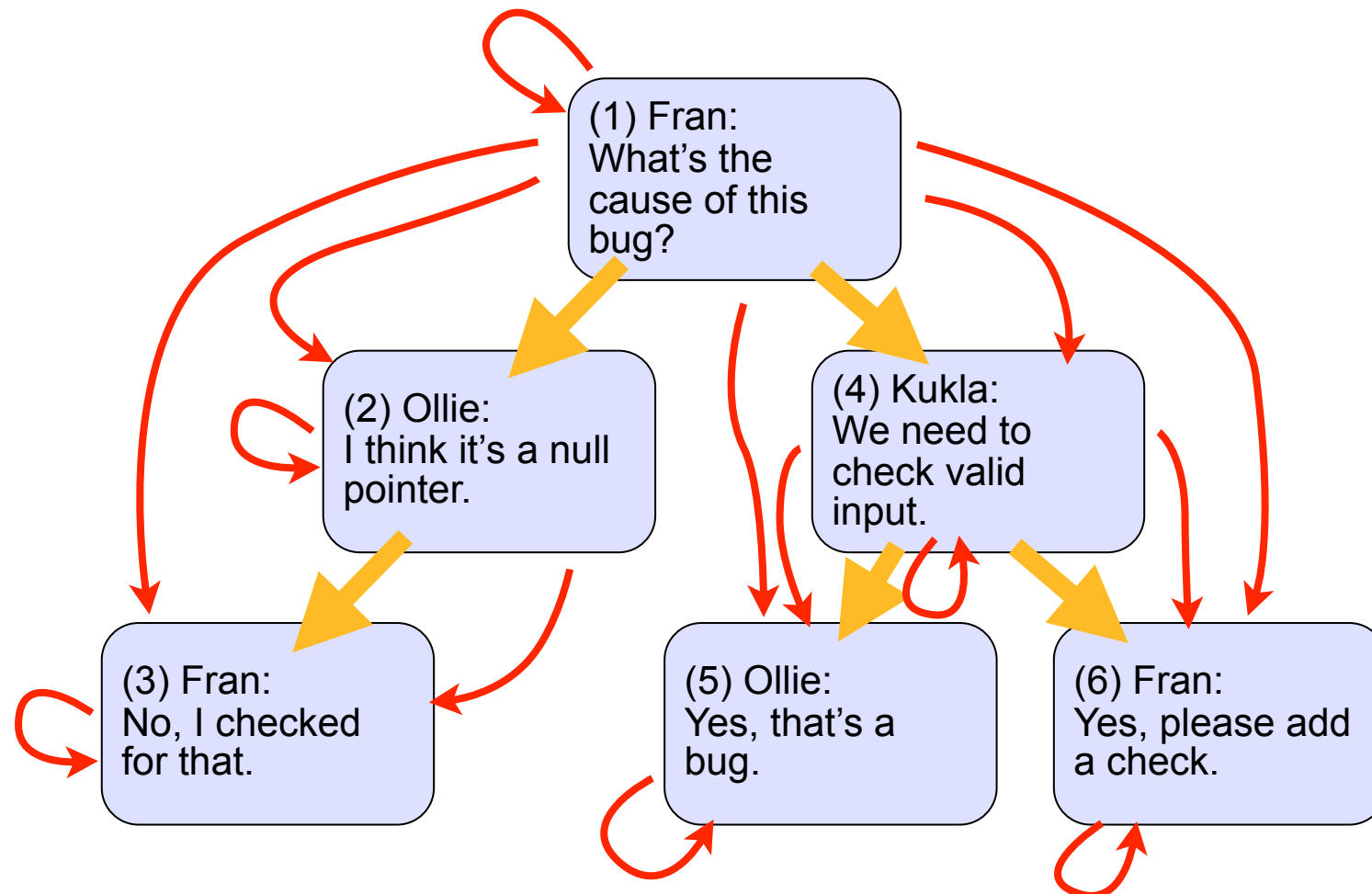
# Delete Paths Terminating at #7



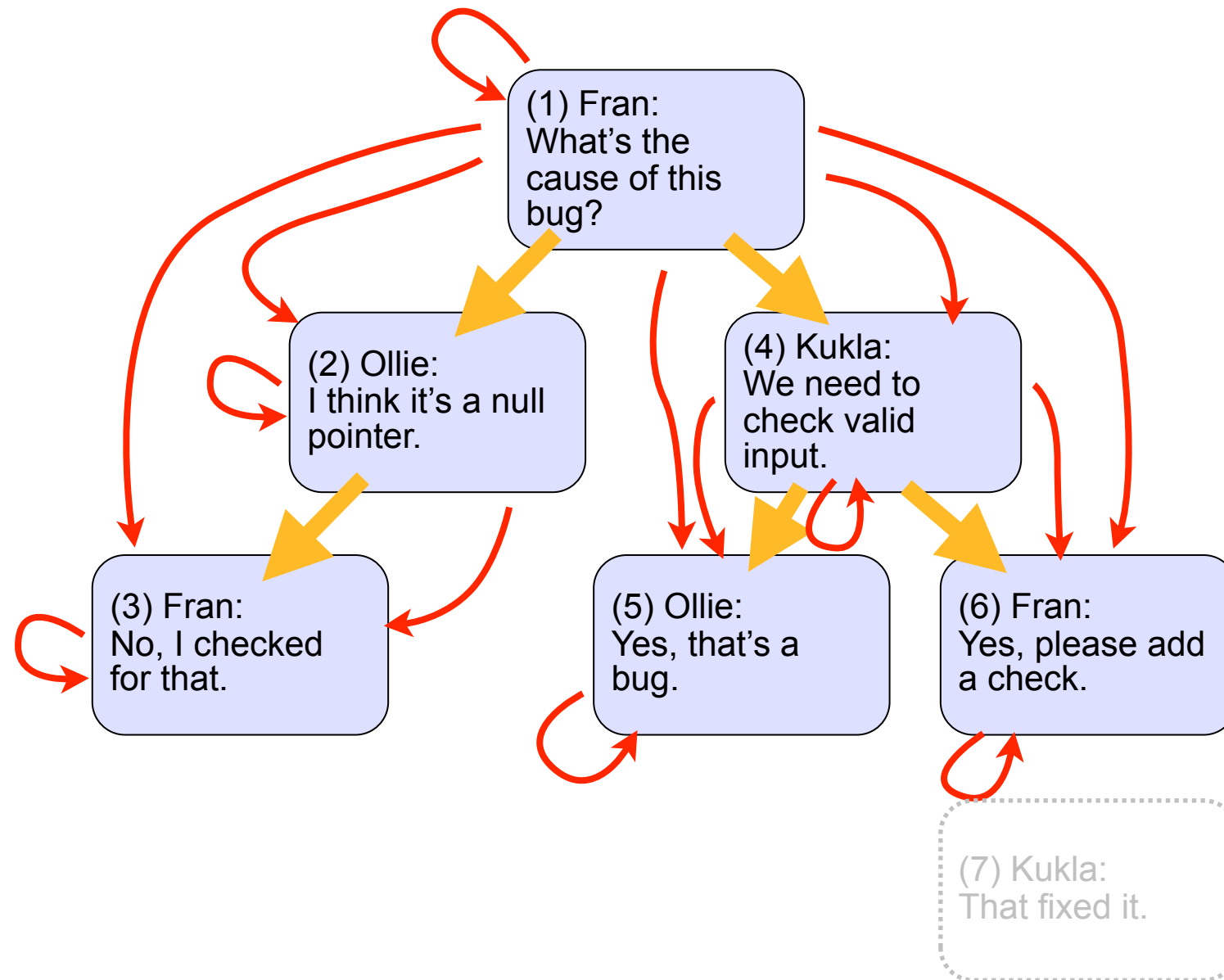
# Delete Paths Terminating at #7



# Delete Paths Terminating at #7



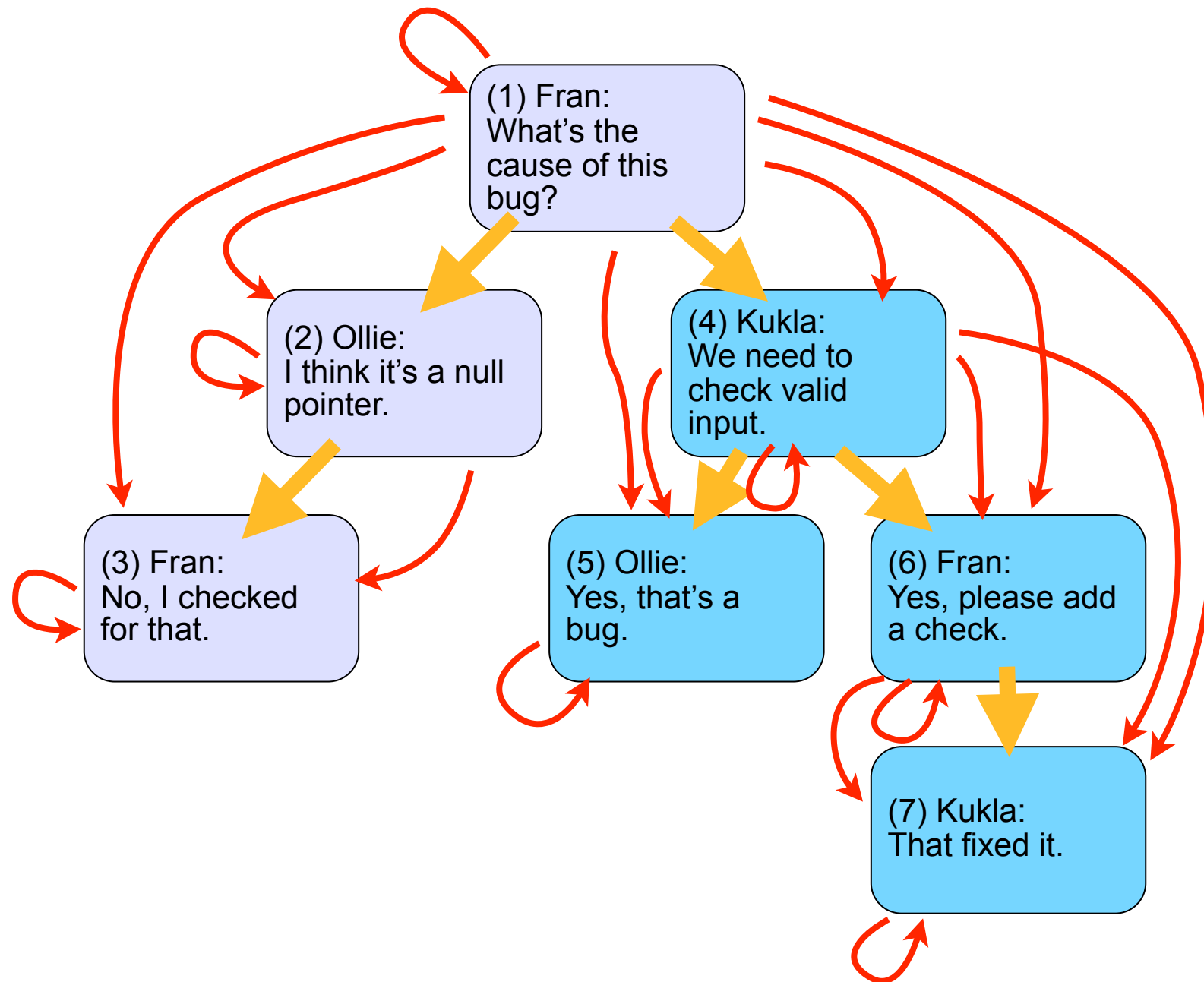
# Delete Paths Terminating at #7



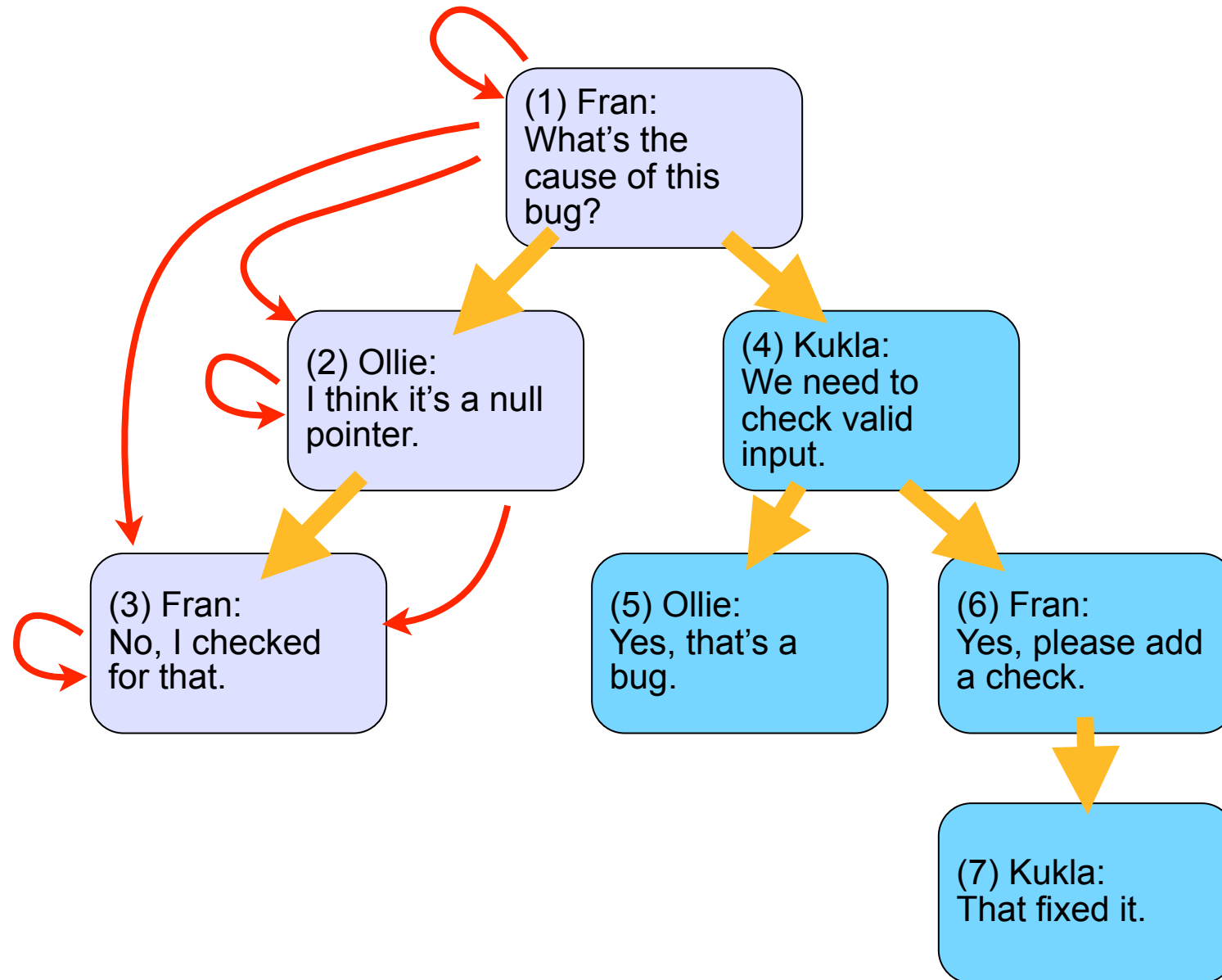
# Delete Subtree Under #4

```
DELETE FROM TreePaths  
WHERE descendant IN  
  (SELECT descendant FROM TreePaths  
   WHERE ancestor = 4);
```

# Delete Any Paths Under #4

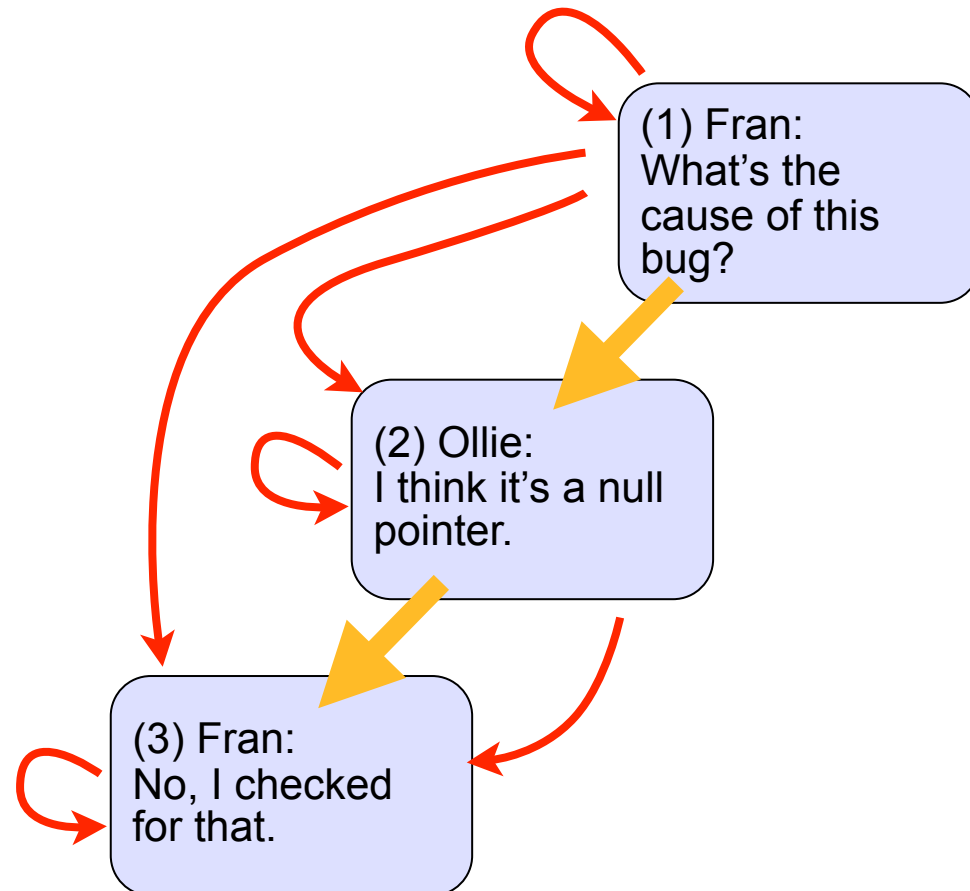


# Delete Any Paths Under #4

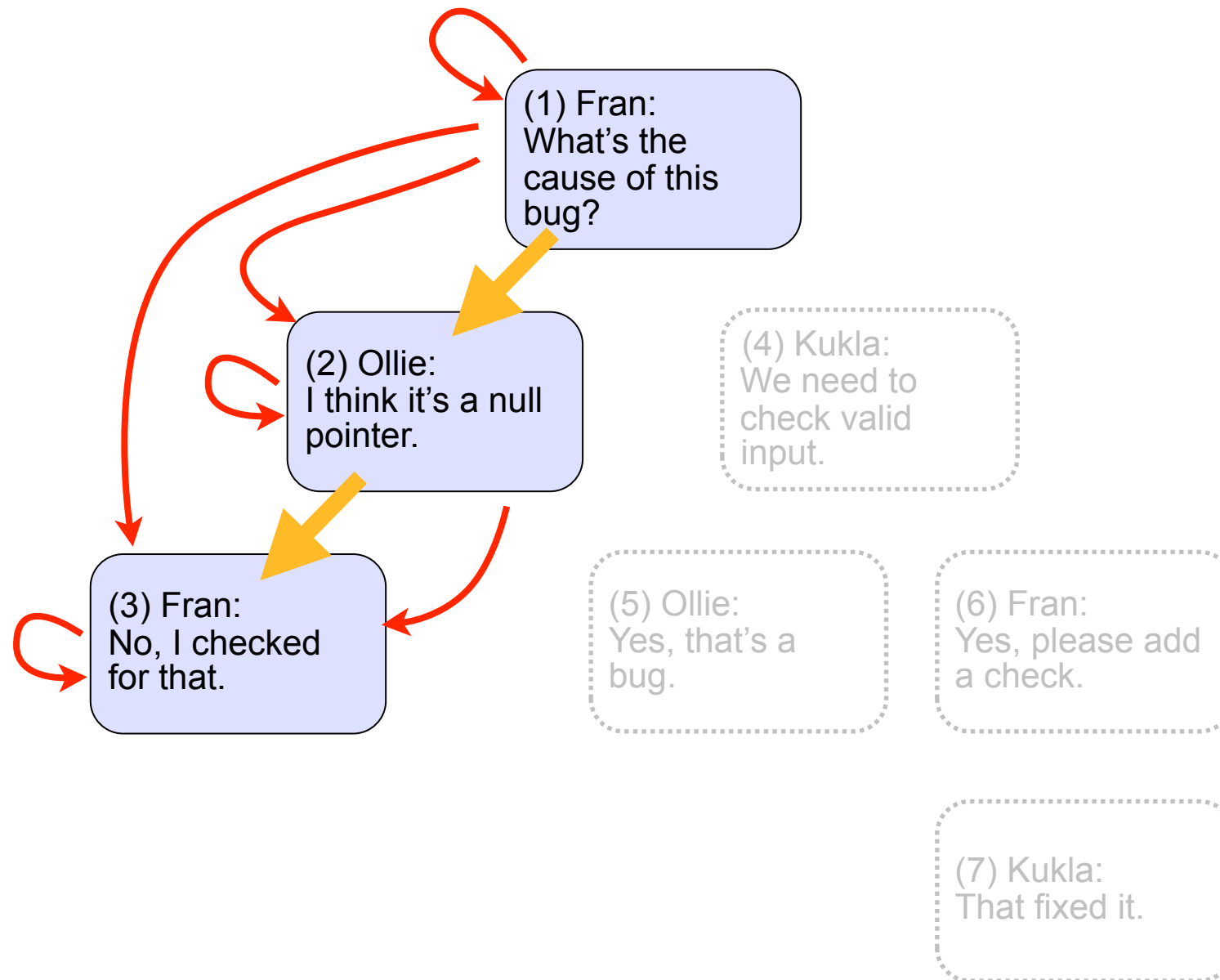




# Delete Any Paths Under #4



# Delete Any Paths Under #4



# Path Length

- Add a *length* column
- MAX(length) is depth of tree
- Makes it easier to query immediate parent or child:

```
SELECT c.*  
FROM Comments c  
JOIN TreePaths t  
  ON (c.comment_id = t.descendant)  
WHERE t.ancestor = 4  
      AND t.length = 1;
```

ancestor	descendant	length
1	1	0
1	2	1
1	3	2
1	4	1
1	5	2
1	6	2
1	7	3
2	2	0
2	3	1
3	3	0
4	4	0
4	5	1
4	6	1
4	7	2
5	5	0
6	6	0
6	7	1
7	7	0

# Path Length

- Add a *length* column
- MAX(length) is depth of tree
- Makes it easier to query immediate parent or child:

```
SELECT c.*  
FROM Comments c  
JOIN TreePaths t  
  ON (c.comment_id = t.descendant)  
WHERE t.ancestor = 4  
      AND t.length = 1;
```

ancestor	descendant	length
1	1	0
1	2	1
1	3	2
1	4	1
1	5	2
1	6	2
1	7	3
2	2	0
2	3	1
3	3	0
4	4	0
4	5	1
4	6	1
4	7	2
5	5	0
6	6	0
6	7	1
7	7	0

# Choosing the Right Design

Design	Tables	Query Child	Query Subtree	Delete Node	Insert Node	Move Subtree	Referential Integrity
Adjacency List	1	Easy	Hard	Easy	Easy	Easy	Yes
Path Enumeration	1	Hard	Easy	Easy	Easy	Easy	No
Nested Sets	1	Hard	Easy	Hard	Hard	Hard	No
Closure Table	2	Easy	Easy	Easy	Easy	Easy	Yes

# PHP Demo of Closure Table

# Hierarchical Test Data

- Integrated Taxonomic Information System
  - <http://itis.gov/>
  - Free authoritative taxonomic information on plants, animals, fungi, microbes
  - 518,756 scientific names (as of Feb 2011)

# California Poppy

Kingdom: *Plantae*  
Division: *Tracheobionta*  
Class: *Magnoliophyta*  
Order: *Magnoliopsida*  
*unranked*: *Magnoliidae*  
*unranked*: *Papaverales*  
Family: *Papaveraceae*  
Genus: *Eschscholzia*  
Species: *Eschscholzia californica*





# California Poppy

Kingdom: *Plantae*  
Division: *Tracheobionta*  
Class: *Magnoliophyta*  
Order: *Magnoliopsida*  
*unranked*: *Magnoliidae*  
*unranked*: *Papaverales*  
Family: *Papaveraceae*  
Genus: *Eschscholzia*  
Species: *Eschscholzia californica*

*id=18956*



# California Poppy: ITIS Entry

```
SELECT * FROM Hierarchy  
WHERE hierarchy_string LIKE '%-18956';
```

hierarchy_string
202422-564824-18061-18063-18064-18879-18880-18954-18956

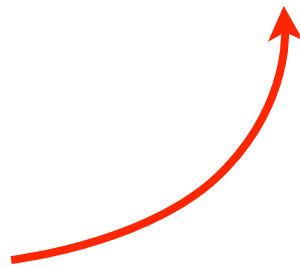
# California Poppy: ITIS Entry

```
SELECT * FROM Hierarchy  
WHERE hierarchy_string LIKE '%-18956';
```

hierarchy_string
202422-564824-18061-18063-18064-18879-18880-18954-18956

*ITIS data uses  
path enumeration*

*...but I converted  
it to closure table*



# Hierarchical Data Classes

```
abstract class ZendX_Db_Table_TreeTable
    extends Zend_Db_Table_Abstract
{
    public function fetchTreeByRoot($rootId, $expand)
    public function fetchBreadcrumbs($leafId)
}
```

# Hierarchical Data Classes

```
class ZendX_Db_Table_Row_TreeRow
    extends Zend_Db_Table_Row_Abstract
{
    public function addChildRow($childRow)
    public function getChildren()
}

class ZendX_Db_Table_Rowset_TreeRowset
    extends Zend_Db_Table_Rowset_Abstract
{
    public function append($row)
}
```

# Using TreeTable

```
class ItisTable extends ZendX_Db_Table_TreeTable
{
    protected $_name = "longnames";
    protected $_closureName = "treepaths";
}

$itis = new ItisTable();
```

# Breadcrumbs

```
$breadcrumbs = $itis->fetchBreadcrumbs(18956);  
foreach ($breadcrumbs as $crumb) {  
    print $crumb->completename . " > ";  
}
```

Plantae > Tracheobionta > Magnoliophyta > Magnoliopsida >  
Magnoliidae > Papaverales > Papaveraceae > Eschscholzia >  
[Eschscholzia californica](#) >

# Breadcrumbs SQL

```
SELECT a.* FROM longnames AS a  
  INNER JOIN treepaths AS c ON a.tsn = c.a  
WHERE (c.d = 18956)  
ORDER BY c.l DESC
```



# How Does it Perform?

- Query profile = 0.0006 sec
- MySQL EXPLAIN:

table	type	key	ref	rows	extra
c	ref	tree_dl	const	9	Using where; Using index
a	eq_ref	primary	c.a	1	

# Dump Tree

```
$tree = $itis->fetchTreeByRoot(18880); // Papaveraceae  
print_tree($tree);
```

```
function print_tree($tree, $prefix = "")  
{  
    print "{$prefix} {$tree->completename}\n";  
    foreach ($tree->getChildren() as $child) {  
        print_tree($child, "{$prefix} ");  
    }  
}
```

# Dump Tree Result

## Papaveraceae

### Platystigma

Platystigma linearis

### Glaucium

Glaucium corniculatum

Glaucium flavum

### Chelidonium

Chelidonium majus

### Bocconia

Bocconia frutescens

### Stylophorum

Stylophorum diphyllum

### Stylomecon

Stylomecon heterophylla

### Canbya

Canbya aurea

Canbya candida

### Chlidonium

Chlidonium majus

## Romneya

Romneya coulteri

Romneya trichocalyx

### Dendromecon

Dendromecon harfordii

Dendromecon rigida

### Eschscholzia

[Eschscholzia californica](#)

Eschscholzia glyptosperma

Eschscholzia hypecoides

Eschscholzia lemmonii

Eschscholzia lobbii

Eschscholzia minutiflora

Eschscholzia parishii

Eschscholzia ramosa

Eschscholzia rhombipetala

Eschscholzia caespitosa

etc...

# Dump Tree SQL

```
SELECT d.*, p.a AS _parent
FROM treepaths AS c
INNER JOIN longnames AS d ON c.d = d.tsn
LEFT JOIN treepaths AS p ON p.d = d.tsn
    AND p.a IN (202422, 564824, 18053, 18020)
    AND p.l = 1
WHERE (c.a = 202422)
    AND (p.a IS NOT NULL OR d.tsn = 202422)
ORDER BY c.l, d.complete_name;
```

# Dump Tree SQL

*show children  
of these nodes*



```
SELECT d.*, p.a AS _parent
FROM treepaths AS c
INNER JOIN longnames AS d ON c.d = d.tsn
LEFT JOIN treepaths AS p ON p.d = d.tsn
    AND p.a IN (202422, 564824, 18053, 18020)
    AND p.l = 1
WHERE (c.a = 202422)
    AND (p.a IS NOT NULL OR d.tsn = 202422)
ORDER BY c.l, d.completename;
```

# How Does it Perform?

- Query profile = 0.20 sec on Macbook Pro
- MySQL EXPLAIN:

table	type	key	ref	rows	extra
c	ref	tree_adl	const	114240	Using index; Using temporary; Using filesort
d	eq_ref	primary	c.d	1	
p	ref	tree_dl	c.d, const	1	Using where; Using index

# SHOW CREATE TABLE

```
CREATE TABLE `treepaths` (  
  `a` int(11) NOT NULL DEFAULT '0',  
  `d` int(11) NOT NULL DEFAULT '0',  
  `l` tinyint(3) unsigned NOT NULL DEFAULT '0',  
  PRIMARY KEY (`a`,`d`),  
  KEY `tree_adl` (`a`,`d`,`l`),  
  KEY `tree_dl` (`d`,`l`),  
  CONSTRAINT FOREIGN KEY (`a`)  
    REFERENCES `longnames` (`tsn`),  
  CONSTRAINT FOREIGN KEY (`d`)  
    REFERENCES `longnames` (`tsn`)  
) ENGINE=InnoDB
```

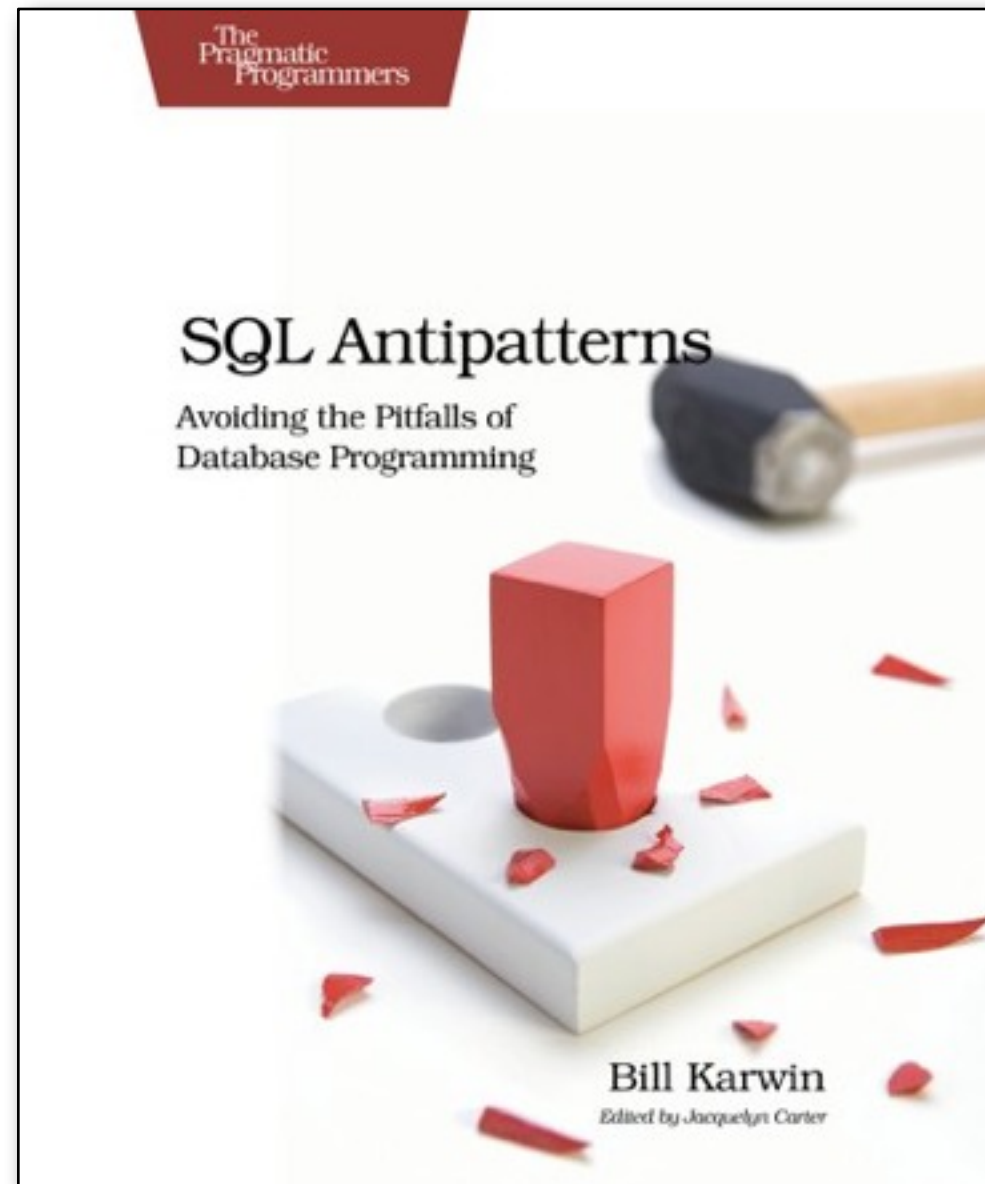
# SHOW TABLE STATUS

Name:	treepaths
Engine:	InnoDB
Version:	10
Row_format:	Compact
Rows:	4600439
Avg_row_length:	62
Data_length:	288276480
Max_data_length:	0
Index_length:	273137664
Data_free:	7340032



# Demo Time!

# SQL Antipatterns



<http://www.pragprog.com/titles/bksqla/>

# License and Copyright

Copyright 2010-2013 Bill Karwin

[www.slideshare.net/billkarwin](http://www.slideshare.net/billkarwin)

Released under a Creative Commons 3.0 License:  
<http://creativecommons.org/licenses/by-nc-nd/3.0/>

You are free to share - to copy, distribute and transmit this work, under the following conditions:



**Attribution.**

You must attribute this work to Bill Karwin.

**Noncommercial.**

You may not use this work for commercial purposes.

**No Derivative Works.**

You may not alter, transform, or build upon this work.