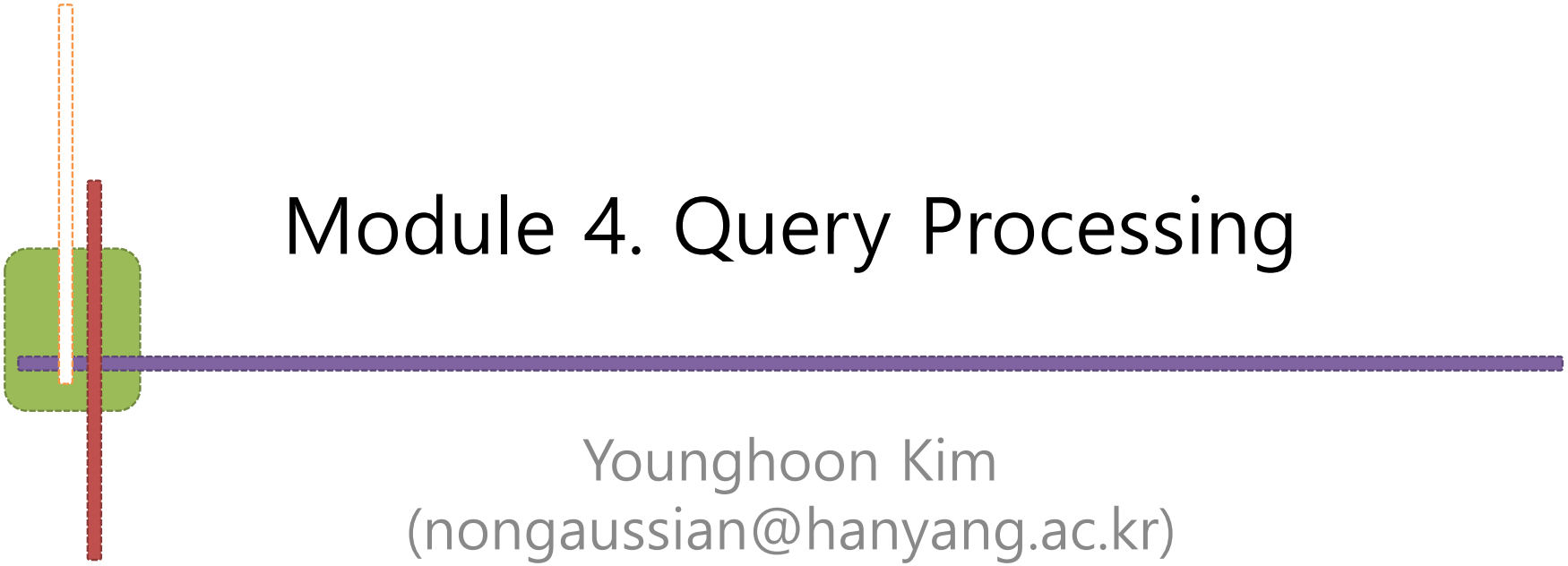


Module 4. Query Processing

Younghoon Kim
(nongaussian@hanyang.ac.kr)

Goal

- Given
  - A query string of String (note that all terms are replaced with term ids in integers)
    - E.g.,
      - 124 223
      - "483 293 1040 2381"
      - "382 294" 2391
- Goal
  - Implement modules
    - Parsing a query string
    - Joining posting lists (intersection with / without positions)
- Return
  - An array of doc IDs
    - Note that the result to be returned finally consists of doc IDs only even if input is a phrase query

Code Template

- We provide a package of
  - A maven project created in Eclipse
- It contains
  - Template codes (`edu.hanyang.submit.TinySEQueryProcess.java`)
  - TinySE framework (`lib/tinyse-0.0.1-SNAPSHOT.jar`) ← to be updated on every stage
    - Interface files (e.g., QueryProcess)
    - API to access posting lists (e.g., DocumentCursor, PositionCursor)
    - Indexer and query processer codes which will complete a search engine by connecting your submissions

To Use Code Template

- Complete *edu.hanyang.submit.TinySEQueryProcess*
- By implementing the interface:

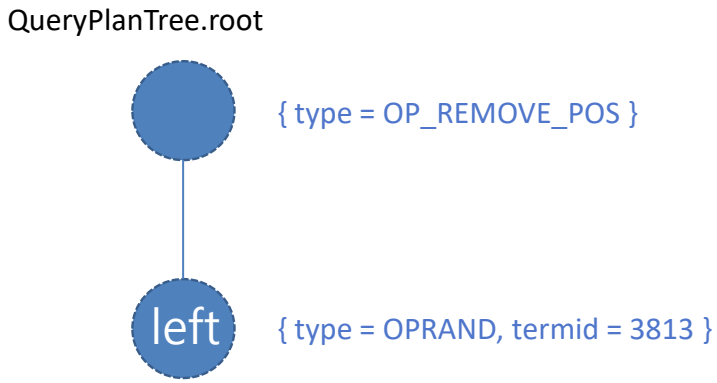
```
package edu.hanyang.submit;

import java.io.IOException;
import edu.hanyang.indexer.DocumentCursor;
import edu.hanyang.indexer.IntermediateList;
import edu.hanyang.indexer.QueryPlanTree;


public class TinySEQueryProcess {
	public void op_and_wo_pos (DocumentCursor op1, DocumentCursor op2, IntermediateList out) throws IOException
	{
	}
	public void op_and_w_pos (DocumentCursor op1, DocumentCursor op2, int shift, IntermediateList out) throws
	IOException {
	}
	public QueryPlanTree parse_query(String query, StatAPI stat) throws IOException {
		//stat.get_pages(termid);
		//stat.get_doc_count(termid);
		//stat.get_min_docid(termid);
		//stat.get_max_docid(termid);
		return null;
	}
}
```

QueryPlanTree

```
public class QueryPlanTree {

        public enum NODE_TYPE { OPRAND, OP_AND, OP_SHIFTED_AND, OP_REMOVE_POS }

        public class QueryPlanNode {
                public NODE_TYPE type;
                public QueryPlanNode left;
                public QueryPlanNode right;
                public int shift;           /* shift has a value when type = OP_SHIFTED_AND
                                            constraint on the positions of terms;
                                            right term's pos - left term's pos = shift */
                public int termid;          /* termid has a value when type = OPRAND */
        }

        public QueryPlanNode root = null;
}
```

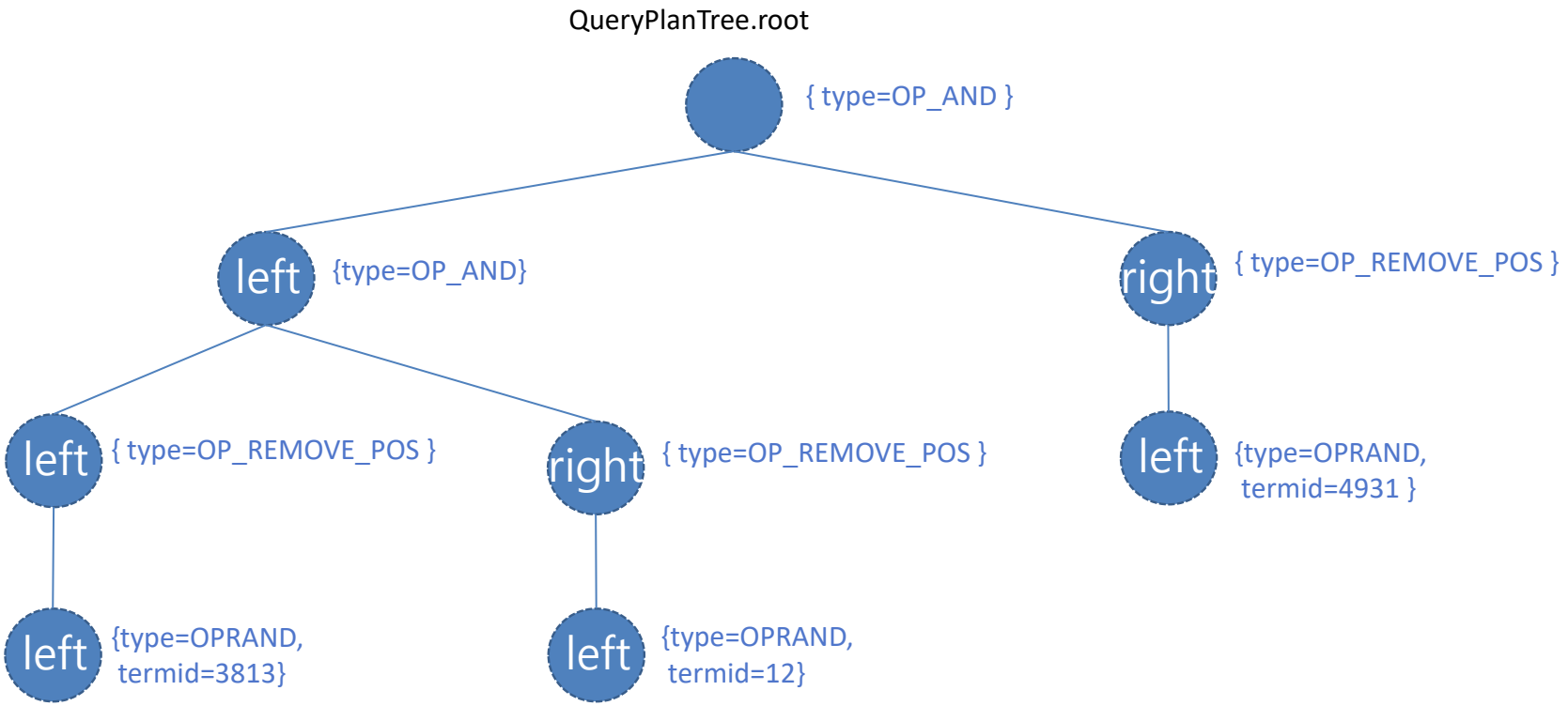
QueryPlanTree

- The method parse_query is required to return a query plan tree (i.e., an instance of QueryPlanTree class)
  - Refer to edu.hanyang.indexer.QueryPlanTree.java
  - Parse the input query string and generate a tree such as
    - E.g., hanyang (=3813)



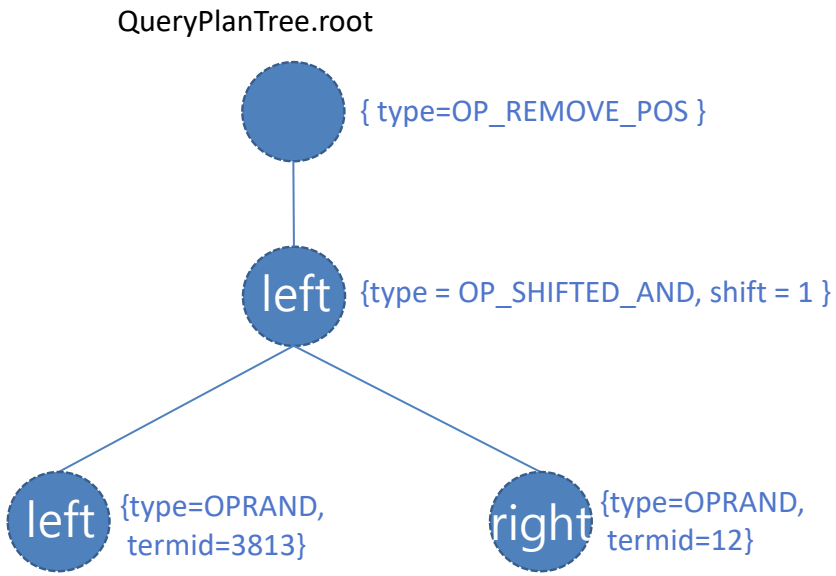
QueryPlanTree

- E.g., hanyang university erica (=3813 12 4931)



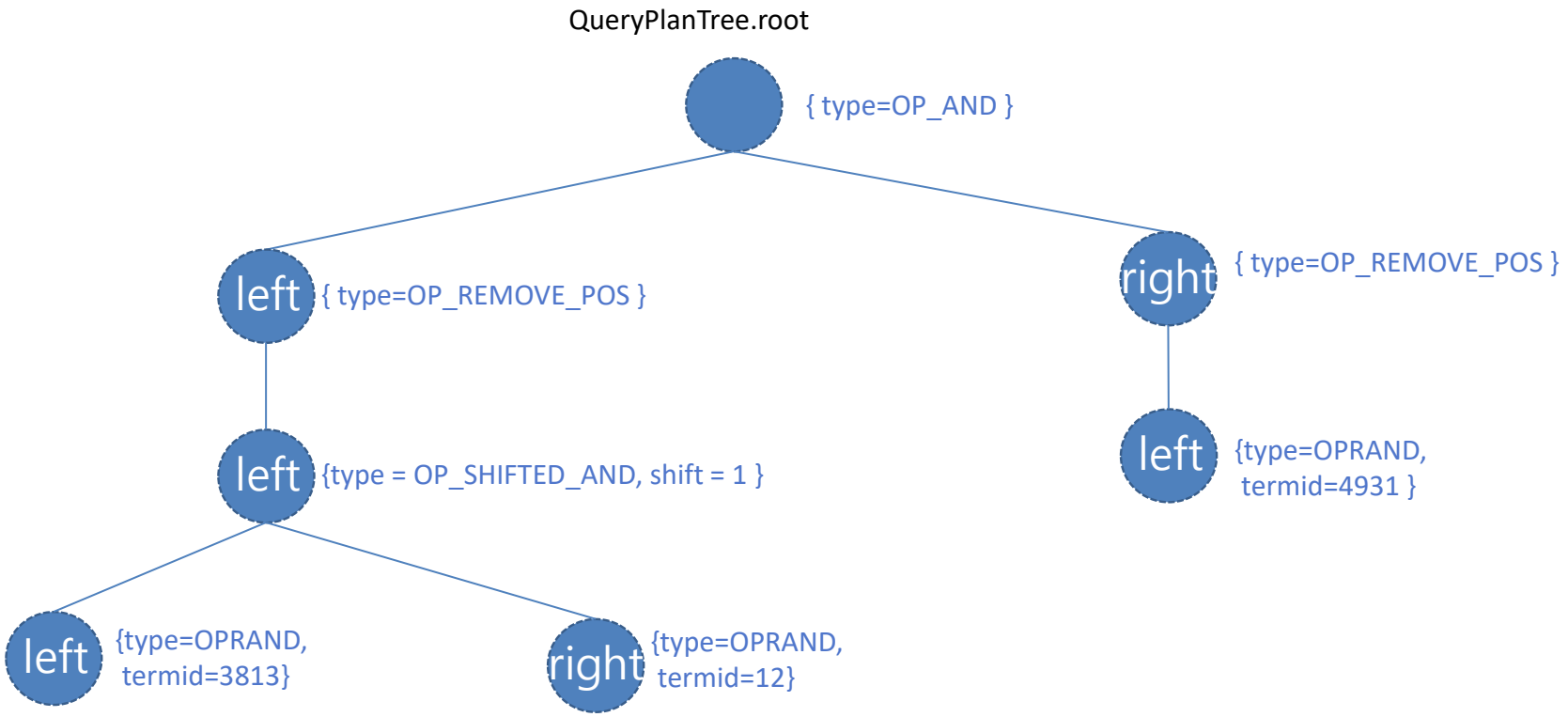
QueryPlanTree

- E.g., "hanyang university" (="3813 12")



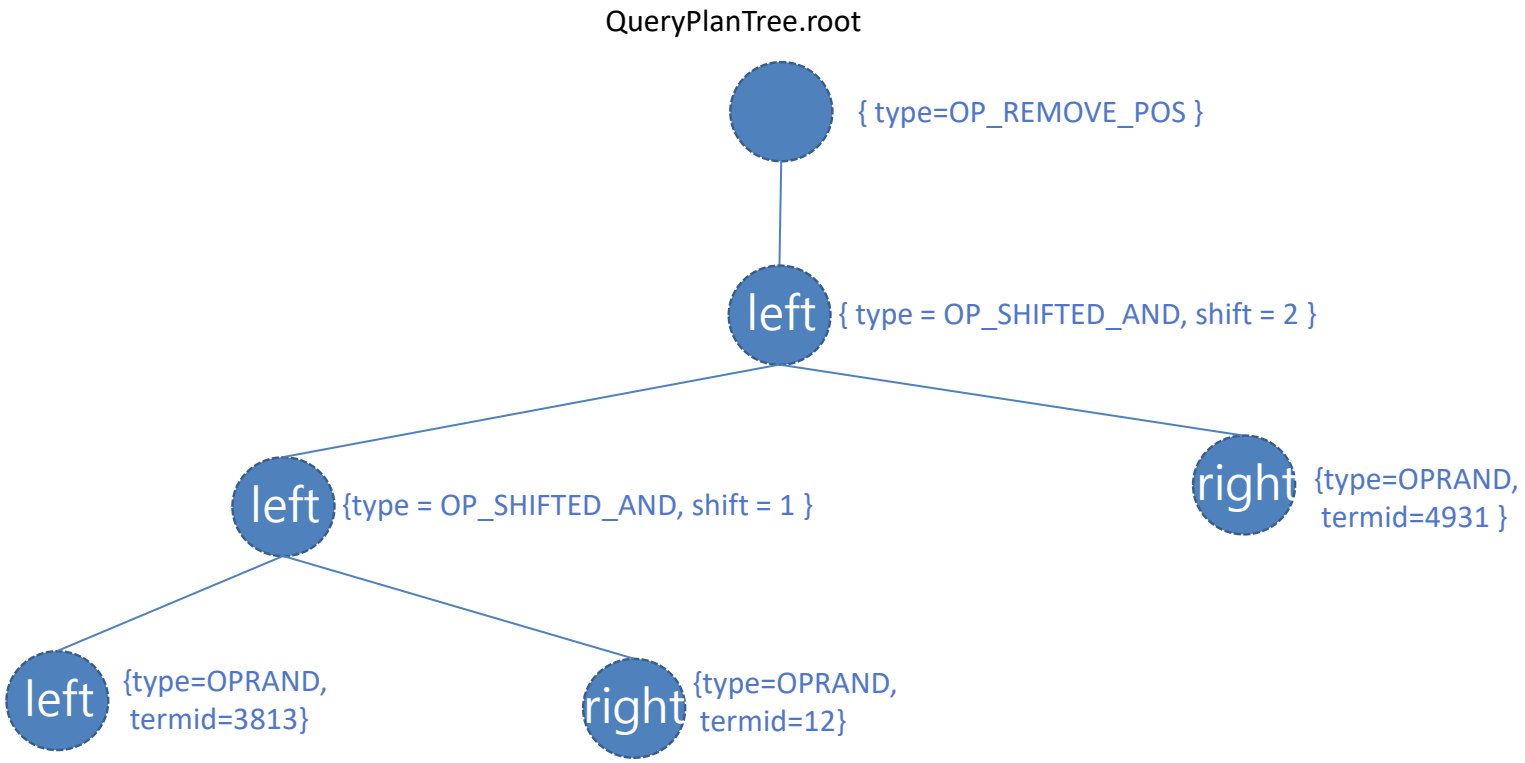
QueryPlanTree

- E.g., "hanyang university" erica = (="3813 12" 4931)



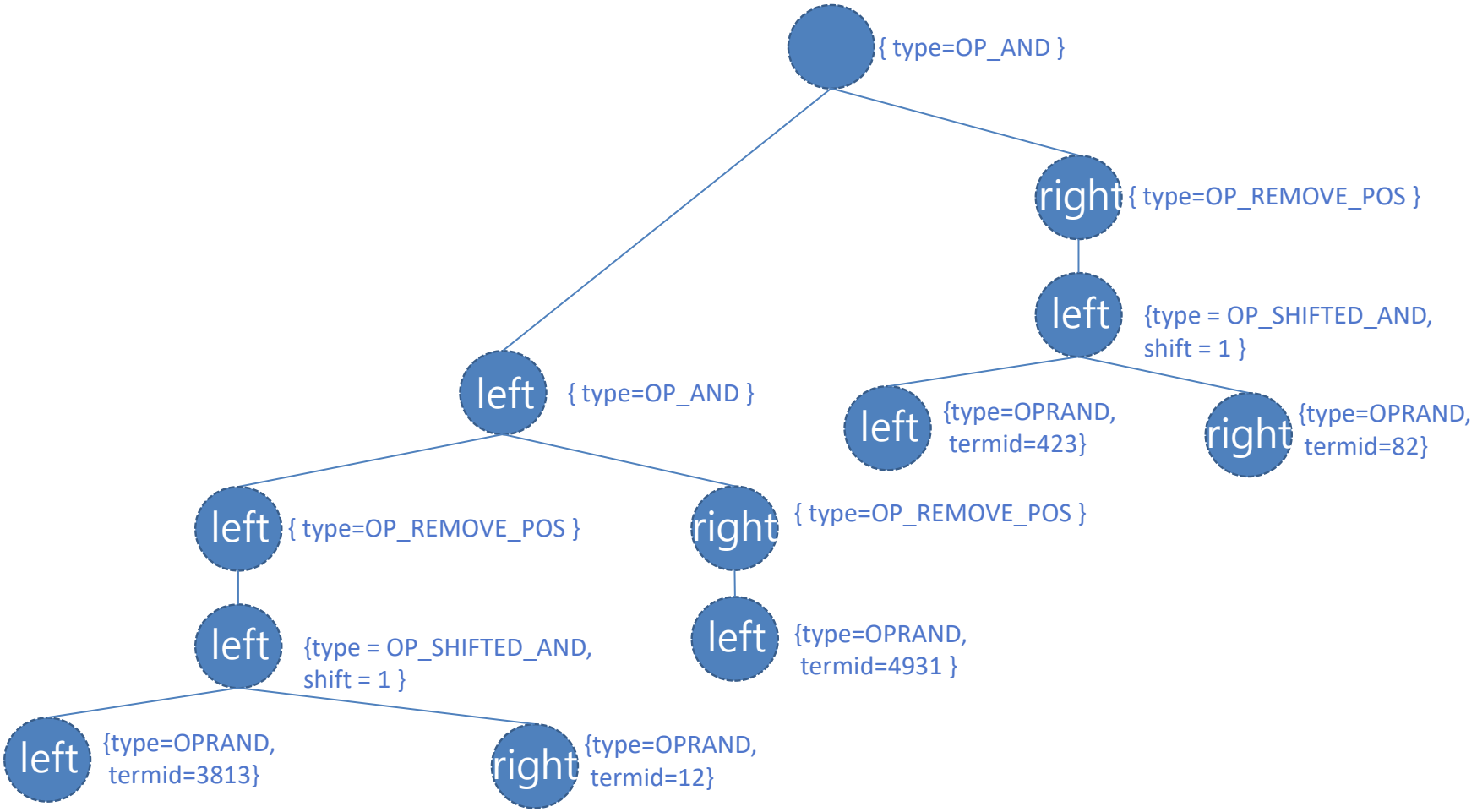
QueryPlanTree

- E.g., "hanyang university erica" = (="3813 12 4931")



QueryPlanTree

- E.g., "hanyang university" erica “ansan kyungki” (="3813 12" 4931 “423 82”)



Parsing A Query String

in_phase = off

- 123 493 “493 349”

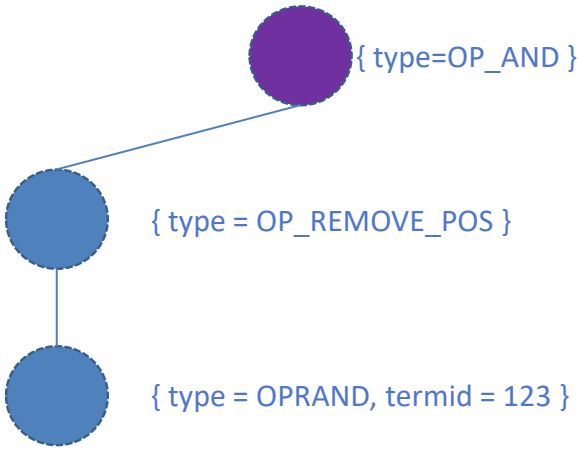
{ type = OP_REMOVE_POS }

{ type = OPRAND, termid = 123 }

Parsing A Query String

in_phase = off

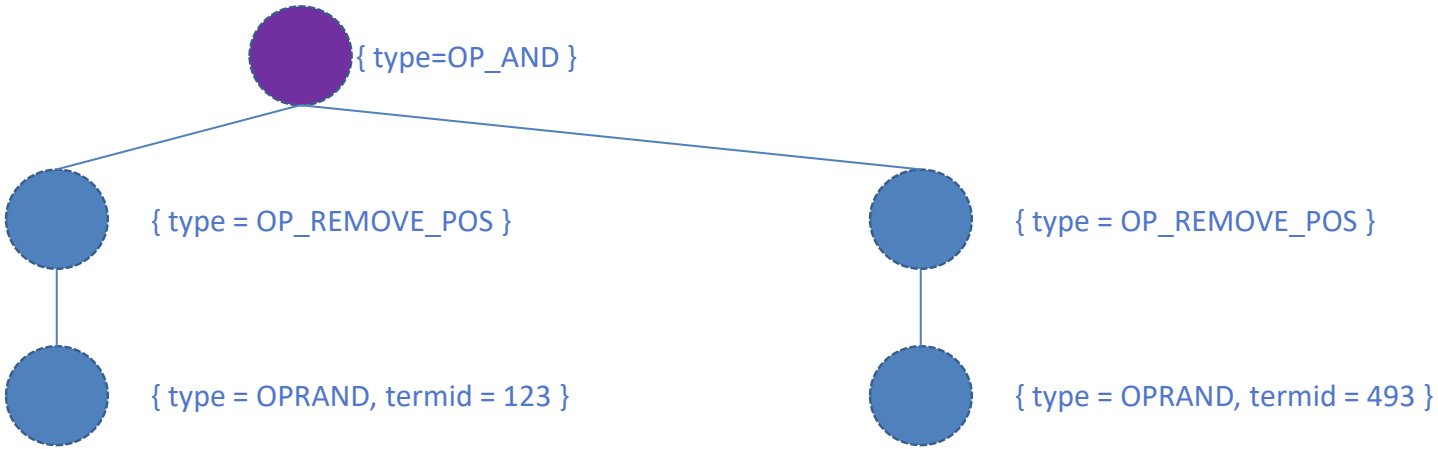
- 123 493 “493 349”



Parsing A Query String

in_phase = off

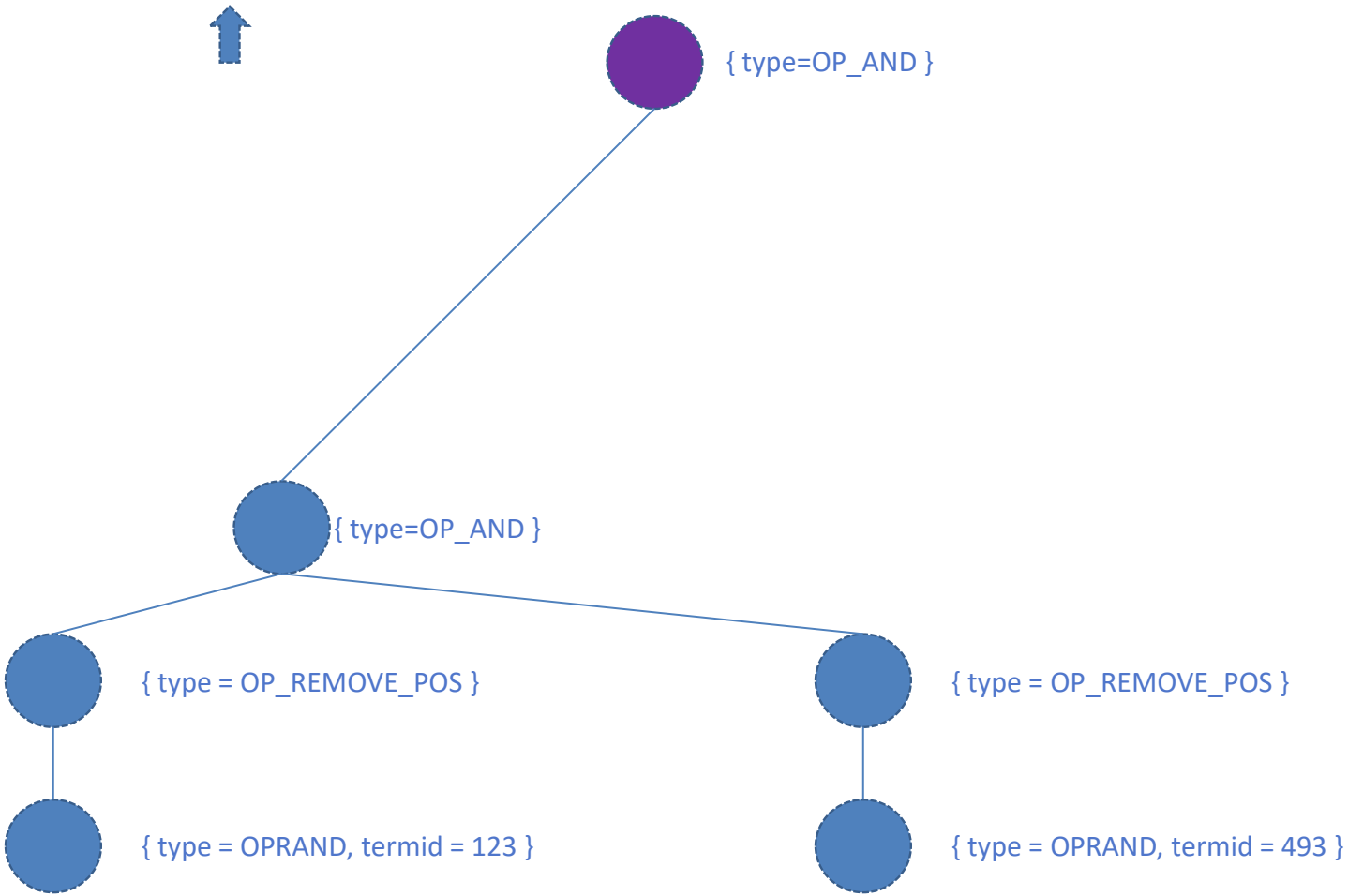
- 123 493 “493 349”



Parsing A Query String

in_phase = off

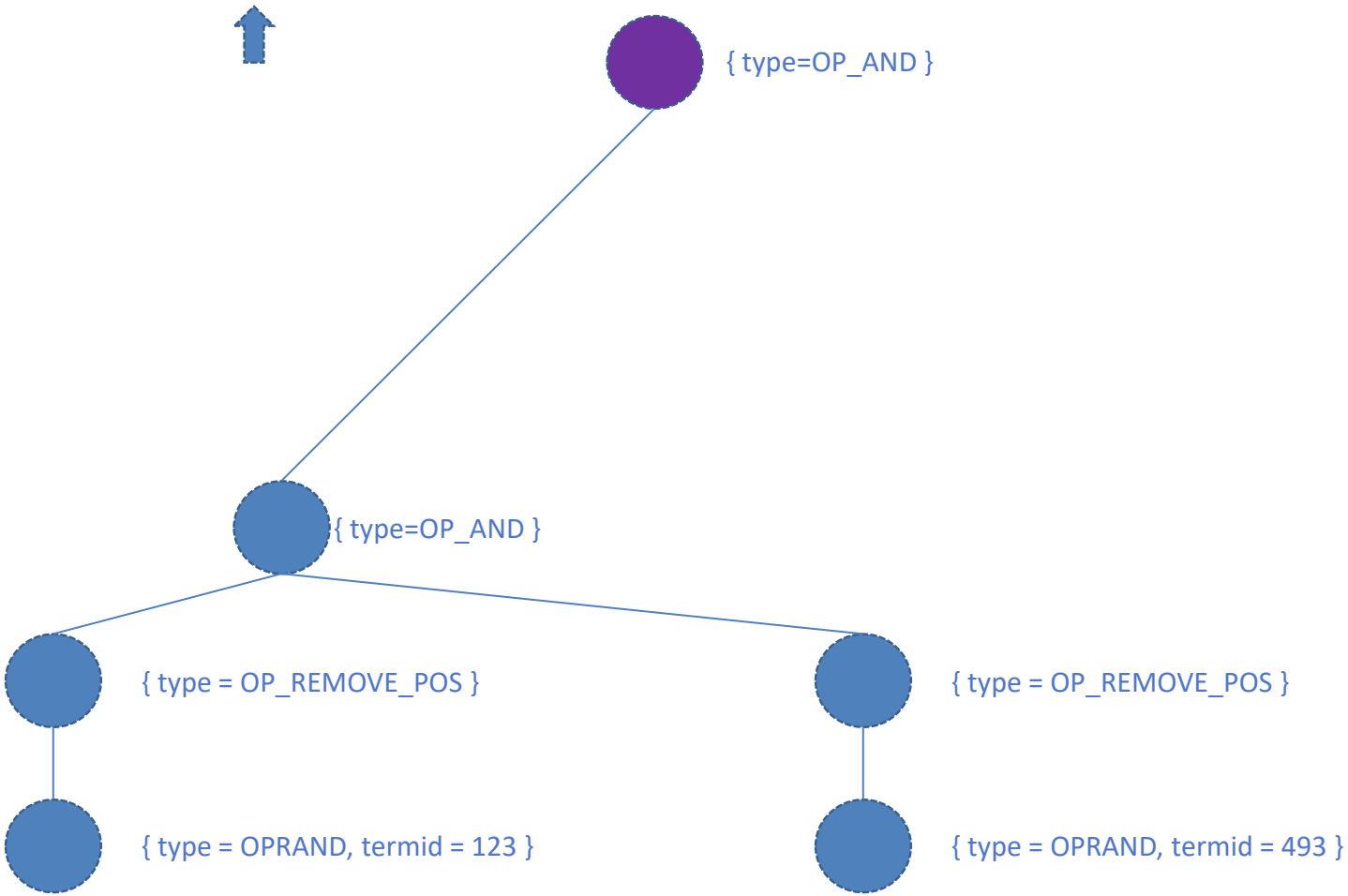
- 123 493 “493 349”



Parsing A Query String

in_phase = on

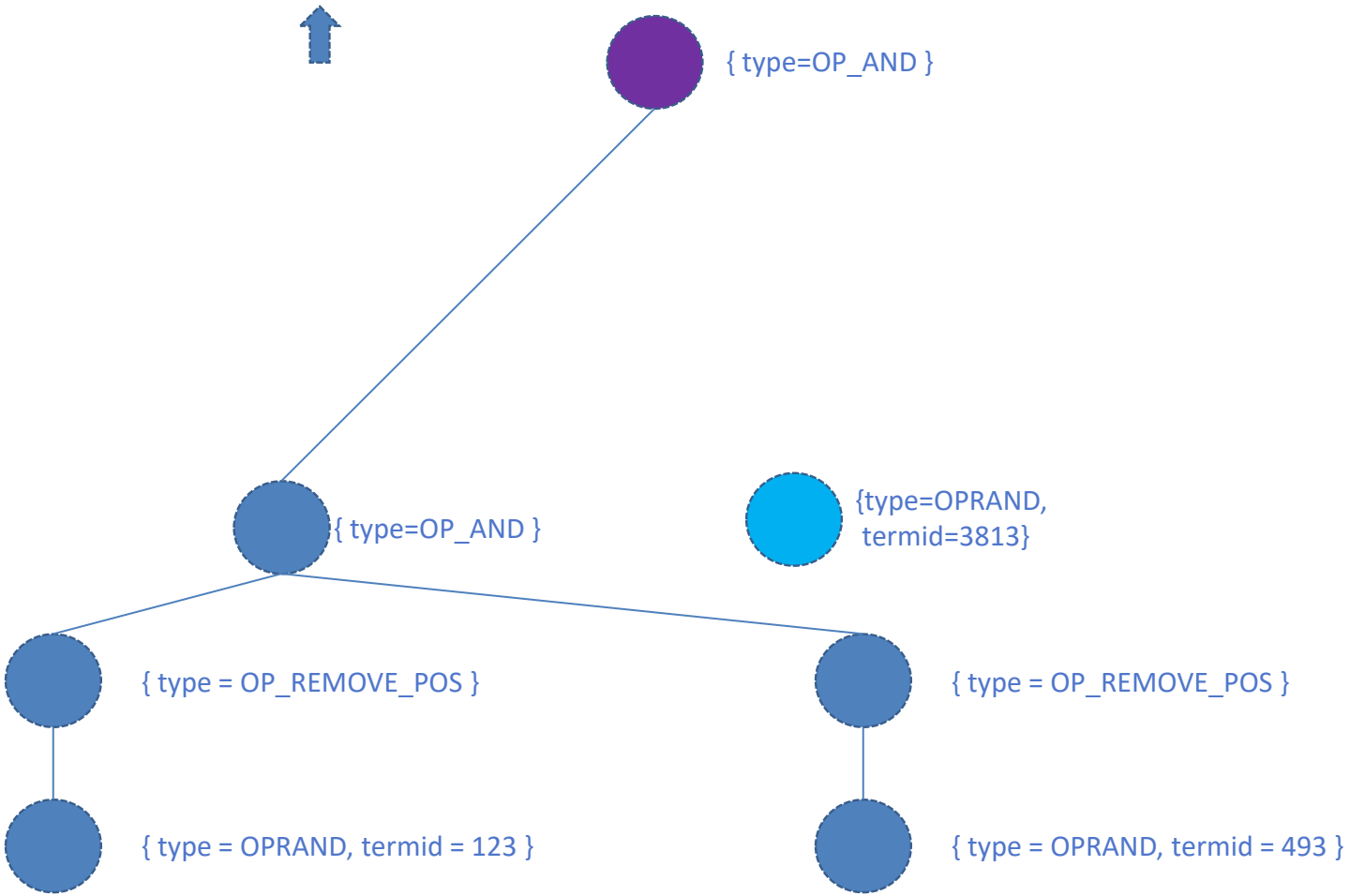
- 123 493 “493 349”



Parsing A Query String

in_phase = on

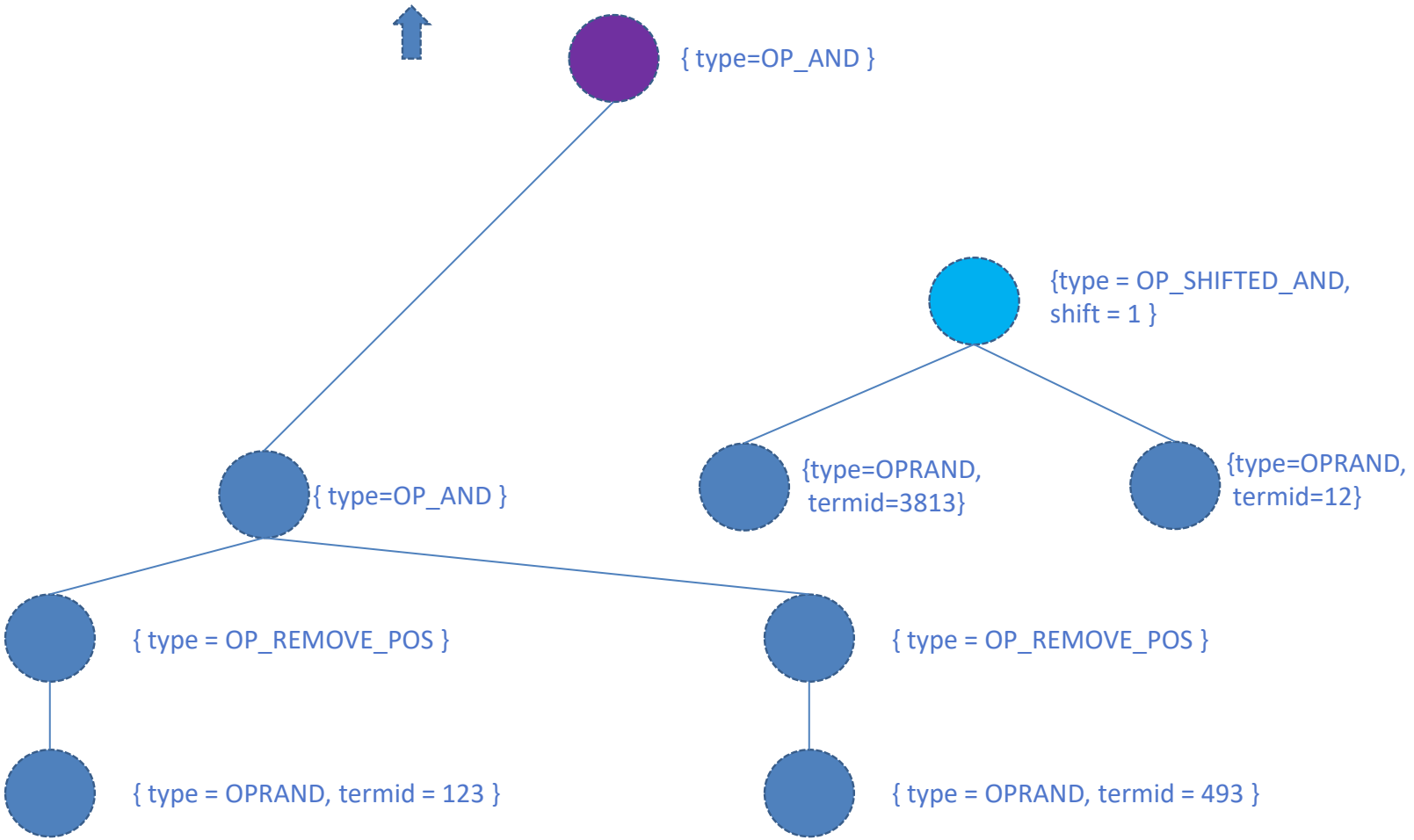
- 123 493 “493 349”



Parsing A Query String

in_phase = on

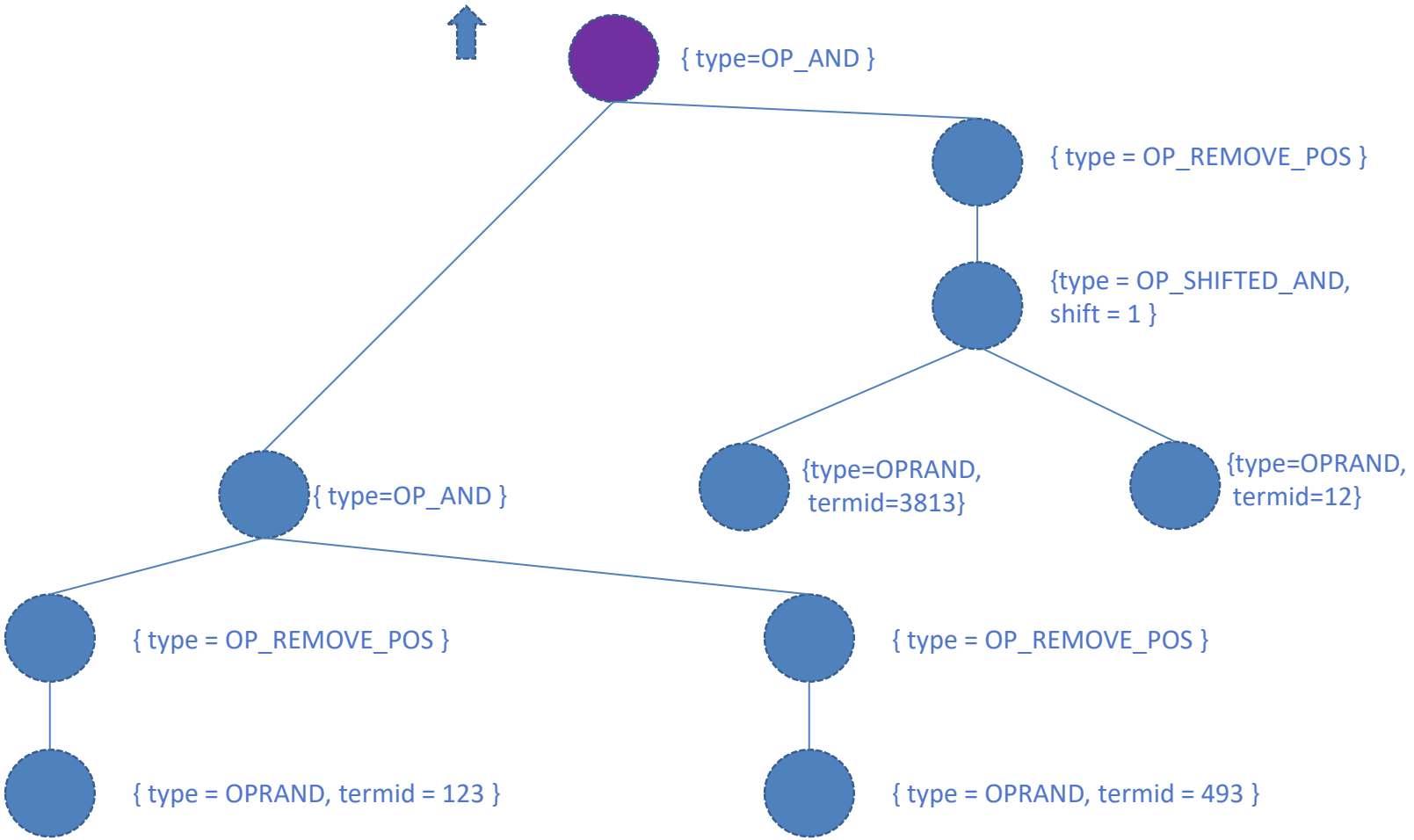
- 123 493 “493 349”



Parsing A Query String

in_phase = off

- 123 493 “493 349”



Revisit: TinySEQueryProcess

```
package edu.hanyang.submit;

import java.io.IOException;
import edu.hanyang.indexer.DocumentCursor;
import edu.hanyang.indexer.IntermediateList;
import edu.hanyang.indexer.QueryPlanTree;


public class TinySEQueryProcess {
	public void op_and_wo_pos (DocumentCursor op1, DocumentCursor op2, IntermediateList out) throws
	IOException {
	}
	public void op_and_w_pos (DocumentCursor op1, DocumentCursor op2, int shift, IntermediatePositionalList out)
	throws IOException {
	}
	public QueryPlanTree parse_query(String query) throws IOException {
		return null;
	}
}
```

Recall: Intersection with Positional Indexes

and_with_positions (p_1, p_2, d)

- answer ← <>
- while p_1 is not null and p_2 is not null
  - if docID(p_1) < docID(p_2)
    - p_1 ← next(p_1)
  - else if docID(p_1) > docID(p_2)
    - p_2 ← next(p_2)
  - else
    - q_1 ← init(p_1), q_2 ← init(p_2)
    - While q_1 is not null and q_2 is not null
      - » If Pos(q_1) + d < Pos(q_2) then q_1 ← next(q_1)
      - » Else if Pos(q_1) + d > Pos(q_2) then q_2 ← next(q_2)
      - » Else
        - add(answer, docID(p_1), Pos(q_1))
        - q_1 ← next(q_1)
        - q_2 ← next(q_2)

DocumentCursor

- Refer to edu.hanyang.indexer.DocumentCursor of the framework package

```
package edu.hanyang.indexer;

import java.io.IOException;

public abstract class DocumentCursor {
        public abstract boolean is_eol() throws IOException;
        public abstract int get_docid() throws IOException;
        public abstract void go_next() throws IOException;
        public abstract PositionCursor get_position_cursor () throws IOException;
        public abstract int get_doc_count() throws Exception;
        public abstract int get_min_docid() throws Exception;
        public abstract int get_max_docid() throws Exception;
}
```

PositionCursor

- Refer to edu.hanyang.indexer.PositionCursor of the framework package

```
package edu.hanyang.indexer;


import java.io.IOException;

public abstract class PositionCursor {
	public abstract boolean is_eol() throws IOException;
	public abstract int get_pos() throws IOException;
	public abstract void go_next() throws IOException;
	public abstract int get_term_count() throws Exception;
}
```

Revisit: TinySEQueryProcess

```
package edu.hanyang.submit;

import java.io.IOException;
import edu.hanyang.indexer.DocumentCursor;
import edu.hanyang.indexer.IntermediateList;
import edu.hanyang.indexer.QueryPlanTree;


public class TinySEQueryProcess {
        public void op_and_wo_pos (DocumentCursor op1, DocumentCursor op2, IntermediateList out) throws
        IOException {
        }
        public void op_and_w_pos (DocumentCursor op1, DocumentCursor op2, int shift, IntermediatePositionalList out)
        throws IOException {
        }
        public QueryPlanTree parse_query(String query) throws IOException {
                return null;
        }
}
```

IntermediatePositionalList

- Refer to edu.hanyang.indexer.IntermediatePositionalList of the framework package

```
package edu.hanyang.utils;

public abstract class IntermediatePositionalList {

        public abstract void put_docid_and_pos(int docid, int pos);

}
```

Recall: Intersection with Non-positional Indexes

- **and_without_positions** (p_1, p_2)
  - answer ← <>
  - while p_1 is not null and p_2 is not null
    - if docID(p_1) < docID(p_2)
      - p_1 ← next(p_1)
    - else if docID(p_1) > docID(p_2)
      - p_2 ← next(p_2)
    - else
      - Add(answer, docID(p_1), Pos(q_1))
      - q_1 ← next(q_1)
      - q_2 ← next(q_2)

IntermediateList

- Refer to of the framework package

```
package edu.hanyang.indexer;

public abstract class IntermediateList {

        public abstract void put_docid(int docid);

}
```

Test Setting

- Query string
  - 10,000 queries randomly generated by selecting sentences from indexed documents
- Evaluation
  - Average running time