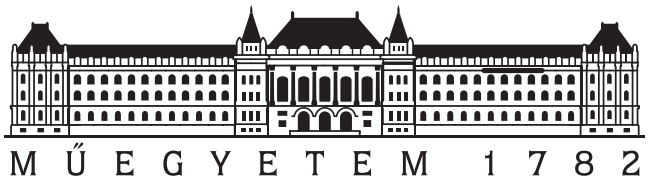
tresorit

Graph-based analysis of JavaScript repositories

Adam Lippai

Web team lead of **Tresorit**,
the encrypted cloud storage company

ingraph



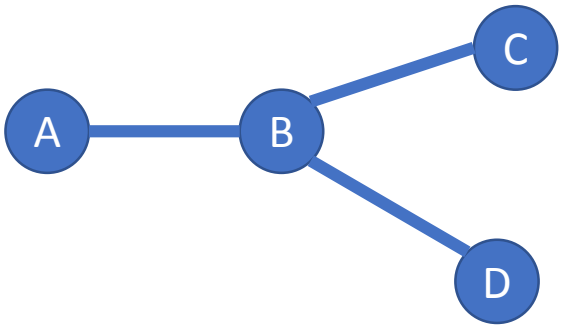
Graph database engine for incremental evaluation of **openCypher** queries

https://github.com/szarnyasg

Similar to Neo4j + incremental evaluation

openCypher – pattern matching

```
(user)-[:KNOWS]-(friend)-[:KNOWS]-(foaf)
```

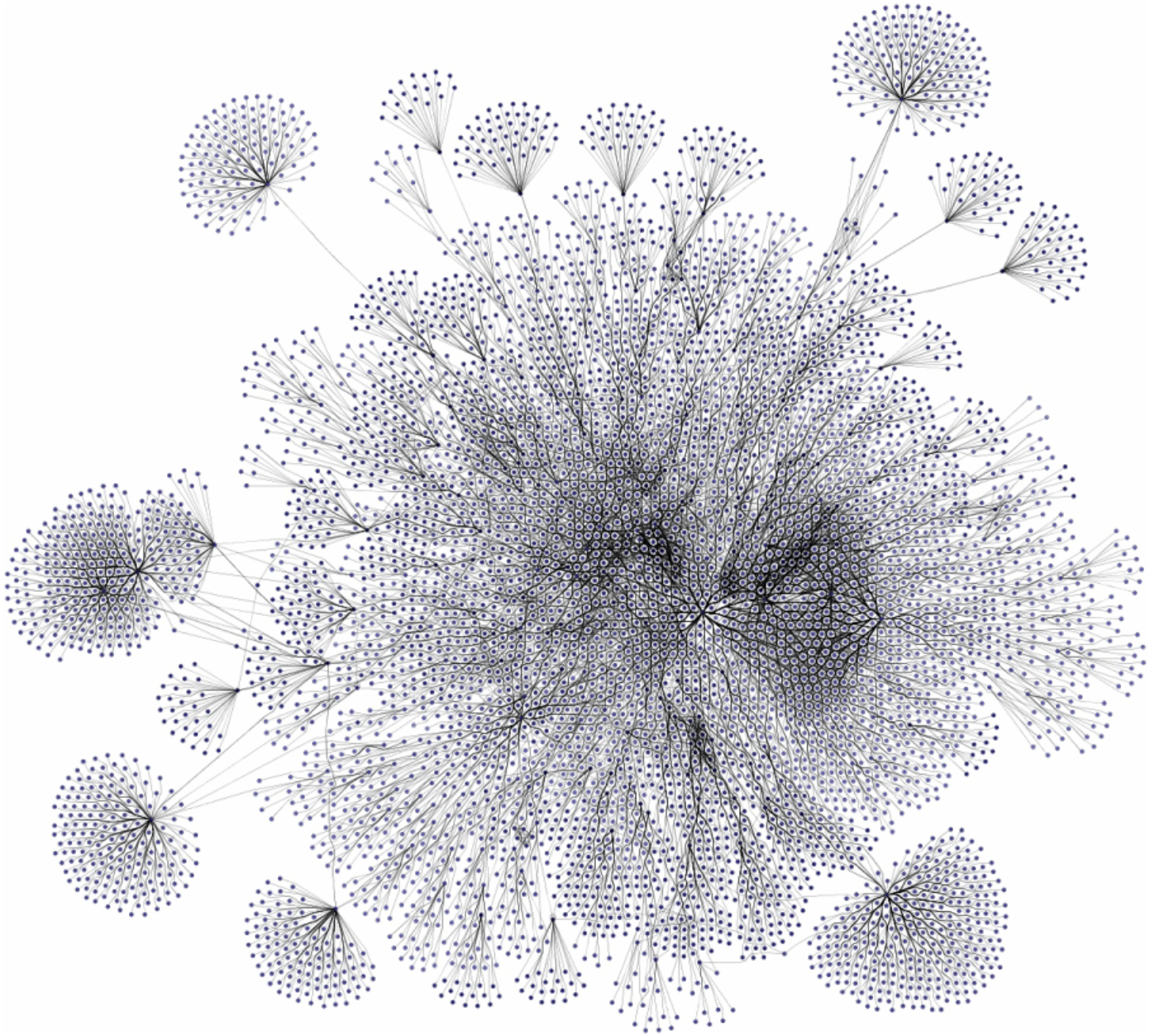


user	friend	foaf
A	B	C
A	B	D

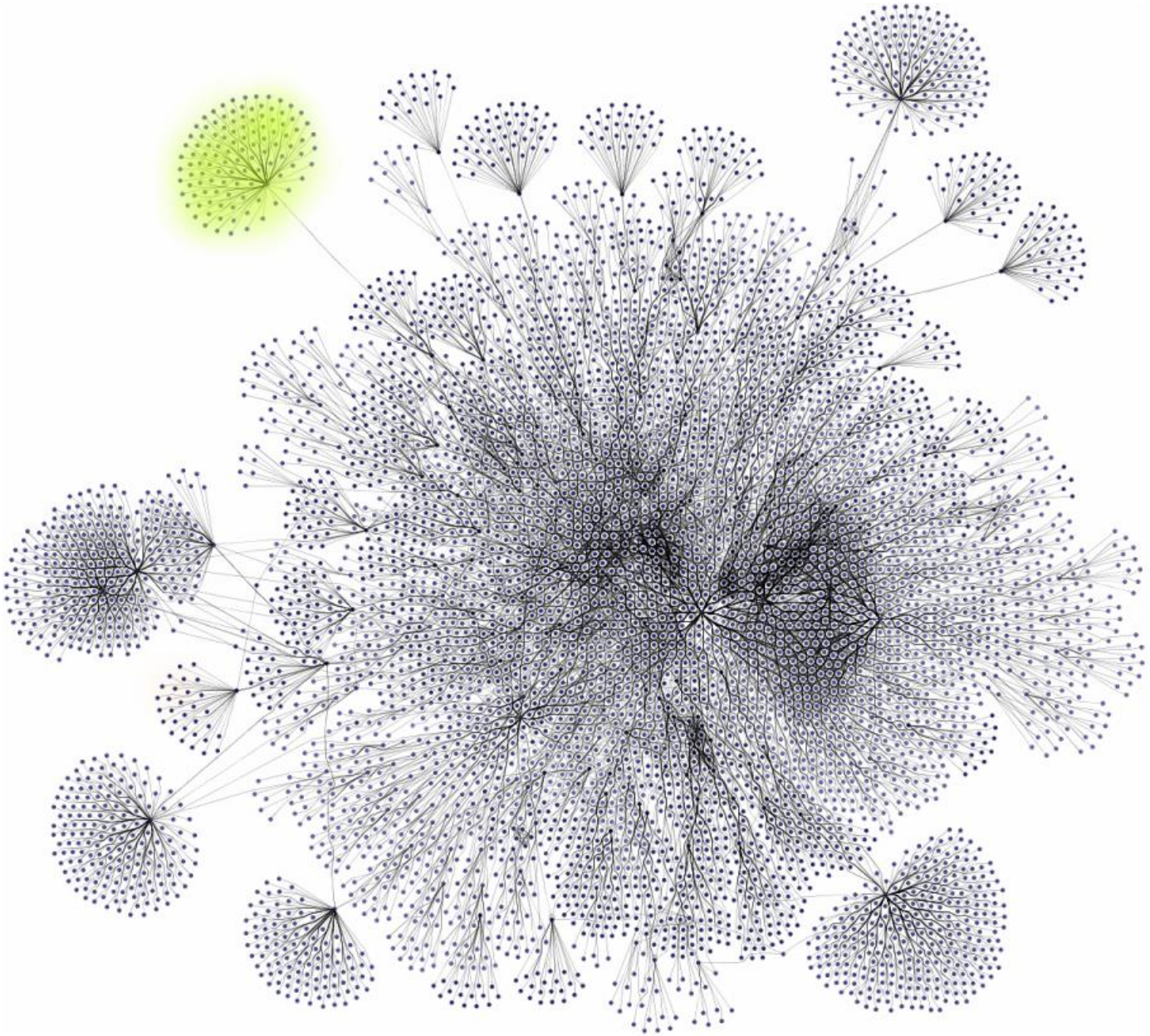
Incremental evaluation

1. A v B v C
2. A v B v C v D

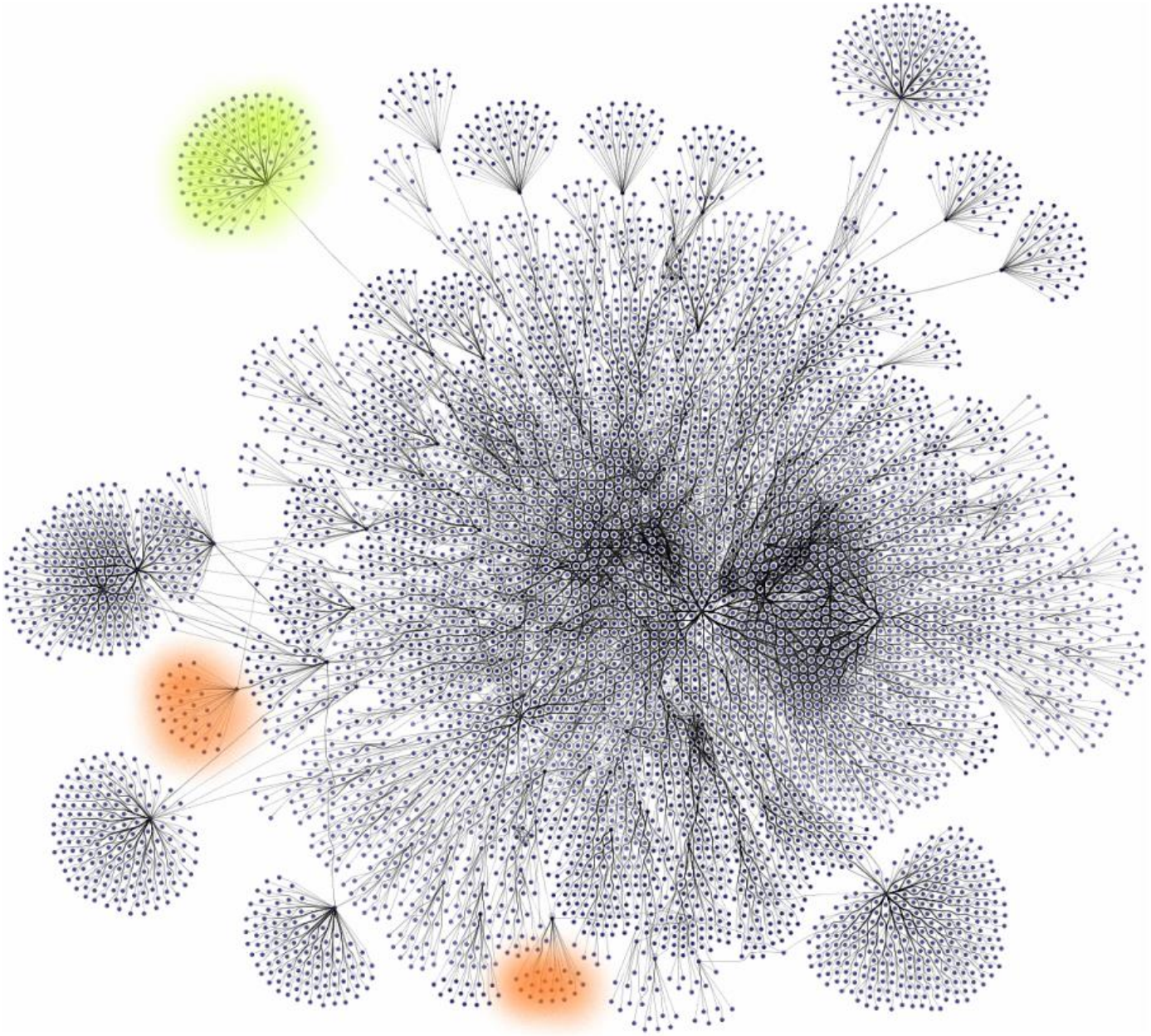
(In reality it's more complex, the actual algorithm is called RETE and it's based on radix trees)



tresorit



tresorit



Why is static analysis important?

- QA is expensive
  - Money: Get the bugs fixed in the earliest stage, cut the administration and release overhead
  - Developer experience: less round-trips -> better focus on one task
- Learning by example
- Insights for project and code health
- It scales across companies
  - Patterns that lead to bugs can be shared
  - Find bugs in your code already found by Microsoft, Facebook, Google

What does the new database enable?

- Granularity & scope
- Developer empowerment
- Maintainability

Granularity – now

- Linters work within files
- TypeScript Compiler and other IDE tools create “interfaces of the imported files” for specific use-cases (e.g. type inference)

Granularity – future

- Complete project
  - Every JS file, together
- Multiple projects
  - Every project – think npm install or npm update
- Over time, over Git branches
  - A project doesn't have 10000 states, but 1 initial state and 9999 changes

Developer empowerment – now

- CTRL + F
- Find class/method/function in IDE
- Class maps for OOP
- Scaffolded Babylon, Acorn or TS compiler script
- Generated + searchable docs based on JSDoc

Developer empowerment – future

- Where is this code used?
- What parts of the code can modify this variable?
- What side effects can this call or assignment have?
- Did I change my libs API? Is it a breaking change?
- How to structure my code?
- Where to cut modules and bundles?

-> ingraph enables such queries

Maintainability – now

- We have unified data structures (similar AST formats)
- De facto standard language: XPath
- Unique visitor patterns
- Hard testability of plugin system
  - Plugins mutate state
  - Problem of “multi-pass” analysis

Maintainability – future

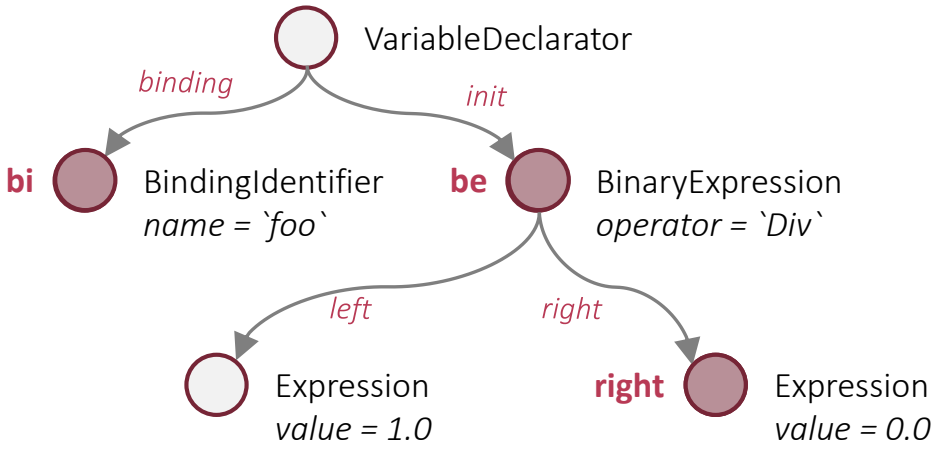
- Adding abstraction without losing information
- Common declarative query language – openCypher

```
MATCH  (bi:BindingIdentifier)
       <-[:binding]-()-->
       (be:BinaryExpression)
       -[:right]->(right:Expression)

WHERE be.operator = 'Div'
  AND right.value = 0.0

RETURN bi
```

var foo = 1 / 0;




```
MATCH  (bi:BindingIdentifier)
       <-[:binding]-()-->
       (be:BinaryExpression)
       -[:right]->(right:Expression)

WHERE be.operator = 'Div'
  AND right.value = 0.0

RETURN bi
```

Graphs are powerful

- Existing optimal algorithms and good heuristics instead of „not that bad code”
- Incremental query caching is possible – eg. RETE or TREAT

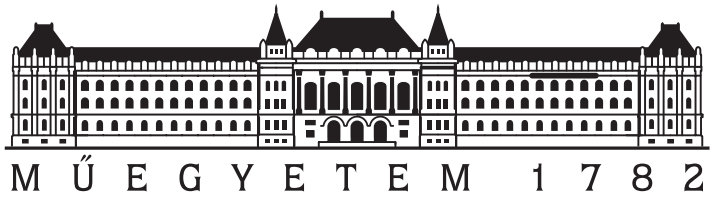
Use cases for incremental pattern matching

- Type propagation and checking (type inference)
- Dead code elimination
- (Asynchronous) code flow checks – can the program reach a specified state, can a value be undefined etc.
- Fuzzing like behavior, e.g. integration test generation
- Code vectorization -> AI

The right tool is

- Declarative
  - what instead of how
- Stateful and incremental
  - cache the existing knowledge
- Instant
  - inside your IDE

Codemodel-rifle



1. Parsing JS using Shift Java
2. Transforming
3. Loading the model into Neo4j or ingraph
4. Executing queries on top of it
   - https://github.com/steindani
   - https://github.com/luczsoma

Thank you!