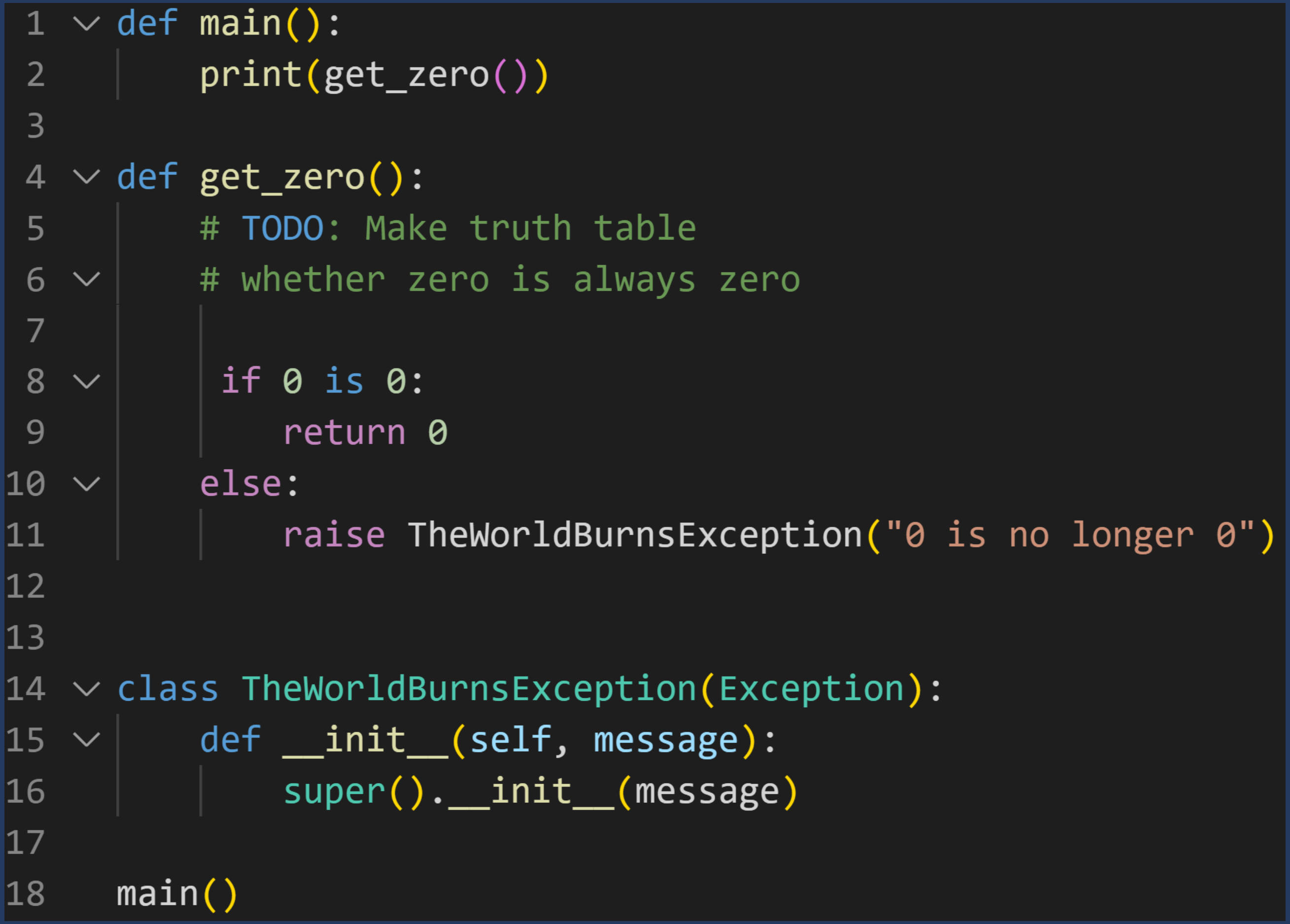
Microsoft

Compilers: Why, What and How?

Programming languages


```python
def main():
    print(get_zero())

def get_zero():
    # TODO: Make truth table
    # whether zero is always zero

     if 0 is 0:
        return 0
    else:
        raise TheWorldBurnsException("0 is no longer 0")


class TheWorldBurnsException(Exception):
    def __init__(self, message):
        super().__init__(message)

main()
```

Domain specific

SQL
Cypher
MATLAB
Regex

Compiled

C#
Java
C/C++
Rust

Interpreted

Python
JavaScript
Lua
Ruby
Powershell

Esoteric

Chicken
LOLCODE
Whitespace

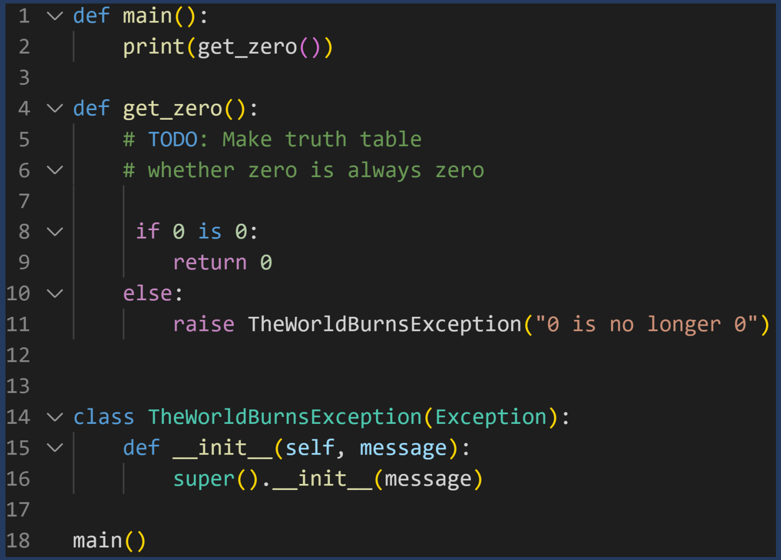
Declarative

Prolog
Lisp

Functional

Haskell
F#

Why?

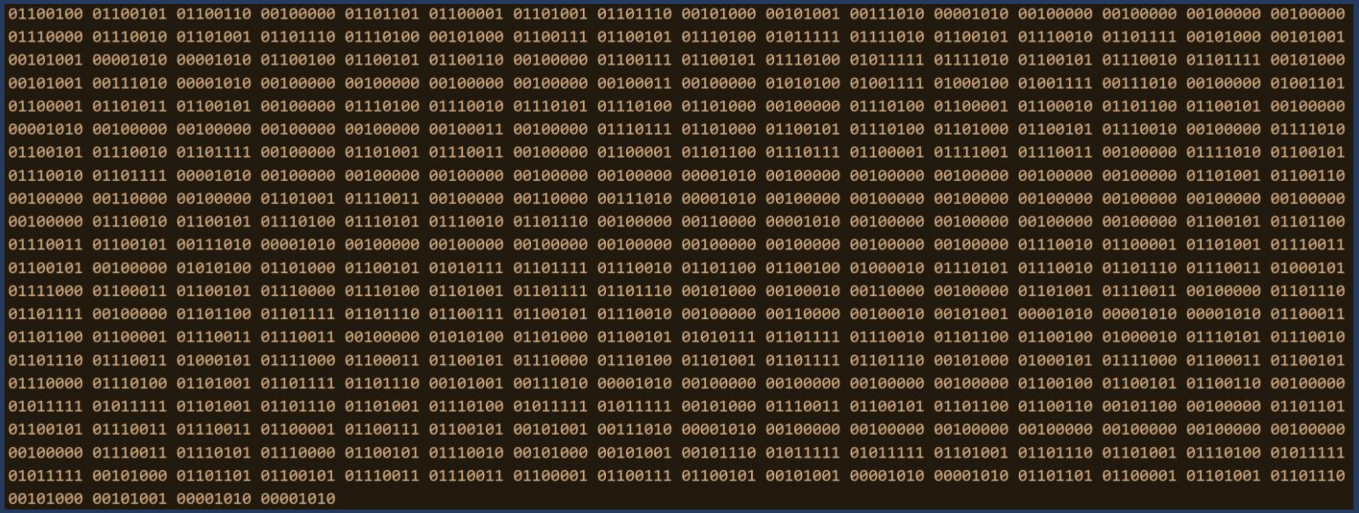


```python
def main():
    print(get_zero())

def get_zero():
    # TODO: Make truth table
    # whether zero is always zero
     
     if 0 is 0:
        return 0
    else:
        raise TheWorldBurnsException("0 is no longer 0")


class TheWorldBurnsException(Exception):
    def __init__(self, message):
        super().__init__(message)

main()
```



```
01100100 01100101 01100110 00100000 01101101 01100001 01101001 01101110 00101000 00101001 00111010 00001010 00100000 00100000 00100000 00100000
01110000 01110010 01101001 01101110 01110100 00101000 01100111 01100101 01110100 01011111 01111010 01100101 01110010 01101111 00101000 00101001
00101001 00001010 00001010 01100100 01100101 01100110 00100000 01100111 01100101 01110100 01011111 01111010 01100101 01110010 01101111 00101000
00101001 00111010 00001010 00100000 00100000 00100000 00100000 00100011 00100000 01010100 01001111 01000100 01001111 00111010 00100000 01001101
01100001 01101011 01100101 00100000 01110100 01110010 01110101 01110100 01101000 00100000 01110100 01100001 01100010 01101100 01100101 00100000
00001010 00100000 00100000 00100000 00100000 00100011 00100000 01110111 01101000 01100101 01110100 01101000 01100101 01110010 00100000 01111010
01100101 01110010 01101111 00100000 01101001 01110011 00100000 01100001 01101100 01110111 01100001 01111001 01110011 00100000 01111010 01100101
01110010 01101111 00001010 00100000 00100000 00100000 00100000 00100000 00001010 00100000 00100000 00100000 00100000 00100000 01101001 01100110
00100000 00110000 00100000 01101001 01110011 00100000 00110000 00111010 00001010 00100000 00100000 00100000 00100000 00100000 00100000 00100000
00100000 01110010 01100101 01110100 01110101 01110010 01101110 00100000 00110000 00001010 00100000 00100000 00100000 00100000 01100101 01101100
01110011 01100101 00111010 00001010 00100000 00100000 00100000 00100000 00100000 00100000 00100000 00100000 01110010 01100001 01101001 01110011
01100101 00100000 01010100 01101000 01100101 01010111 01101111 01110010 01101100 01100100 01000010 01110101 01110010 01101110 01110011 01000101
01111000 01100011 01100101 01110000 01110100 01101001 01101111 01101110 00101000 00100010 00110000 00100000 01101001 01110011 00100000 01101110
01101111 00100000 01101100 01101111 01101110 01100111 01100101 01110010 00100000 00110000 00100010 00101001 00001010 00001010 00001010 01100011
01101100 01100001 01110011 01110011 00100000 01010100 01101000 01100101 01010111 01101111 01110010 01101100 01100100 01000010 01110101 01110010
01101110 01110011 01000101 01111000 01100011 01100101 01110000 01110100 01101001 01101111 01101110 00101000 01000101 01111000 01100011 01100101
01110000 01110100 01101001 01101111 01101110 00101001 00111010 00001010 00100000 00100000 00100000 00100000 01100100 01100101 01100110 00100000
01011111 01011111 01101001 01101110 01101001 01110100 01011111 01011111 00101000 01110011 01100101 01101100 01100110 00101100 00100000 01101101
01100101 01110011 01110011 01100001 01100111 01100101 00101001 00111010 00001010 00100000 00100000 00100000 00100000 00100000 00100000 00100000
00100000 01110011 01110101 01110000 01100101 01110010 00101000 00101001 00101110 01011111 01011111 01101001 01101110 01101001 01110100 01011111
01011111 00101000 01101101 01100101 01110011 01110011 01100001 01100111 01100101 00101001 00001010 00001010 01101101 01100001 01101001 01101110
00101000 00101001 00001010 00001010
```

Compiler overview

Split the program text into tokens.

Combine tokens into expressions and build an Abstract Syntax Tree (AST)

Semantic analysis, Typechecking

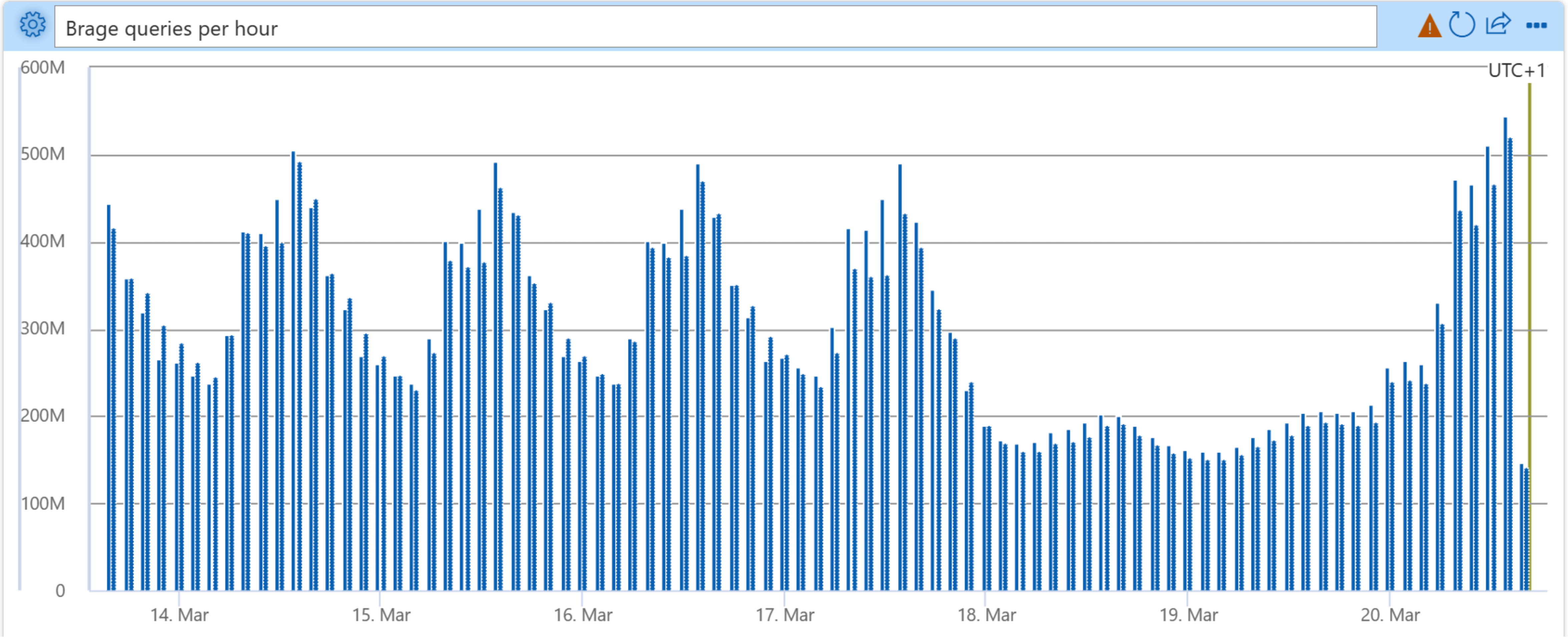
Reduce runtime and hardware requirements

Write instructions for an executable program.

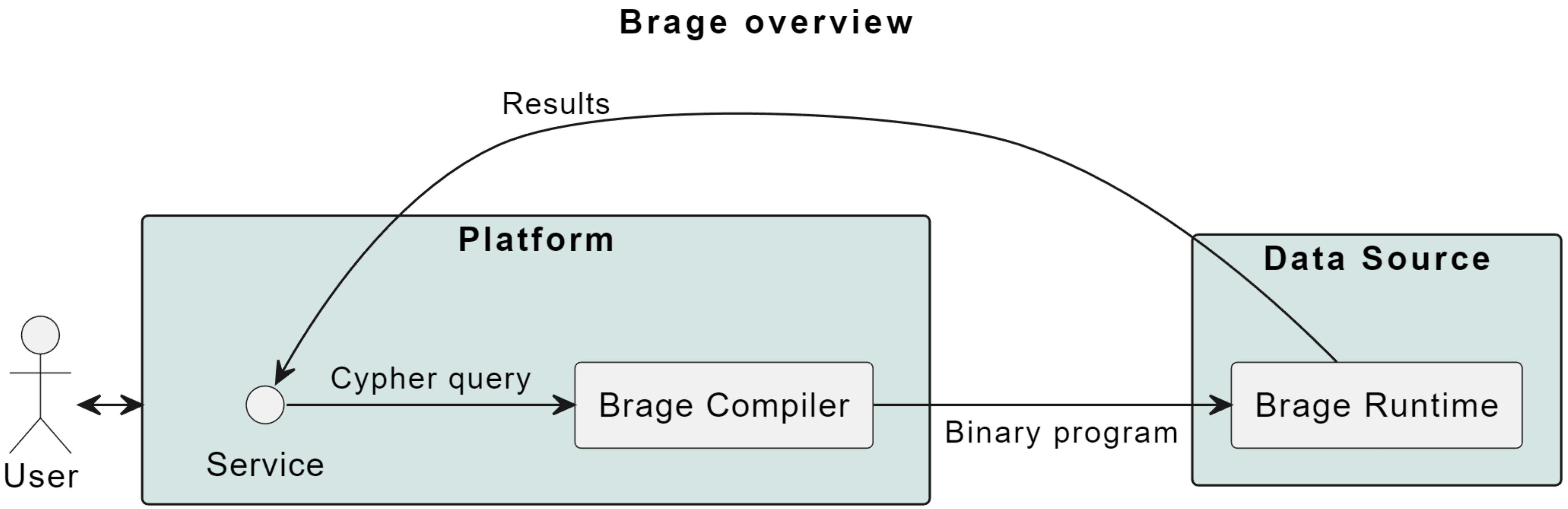
Lexing/Tokenization **Parsing** **Analyze** **Optimize** **Code generation**

Example: Brage

Graph query engine

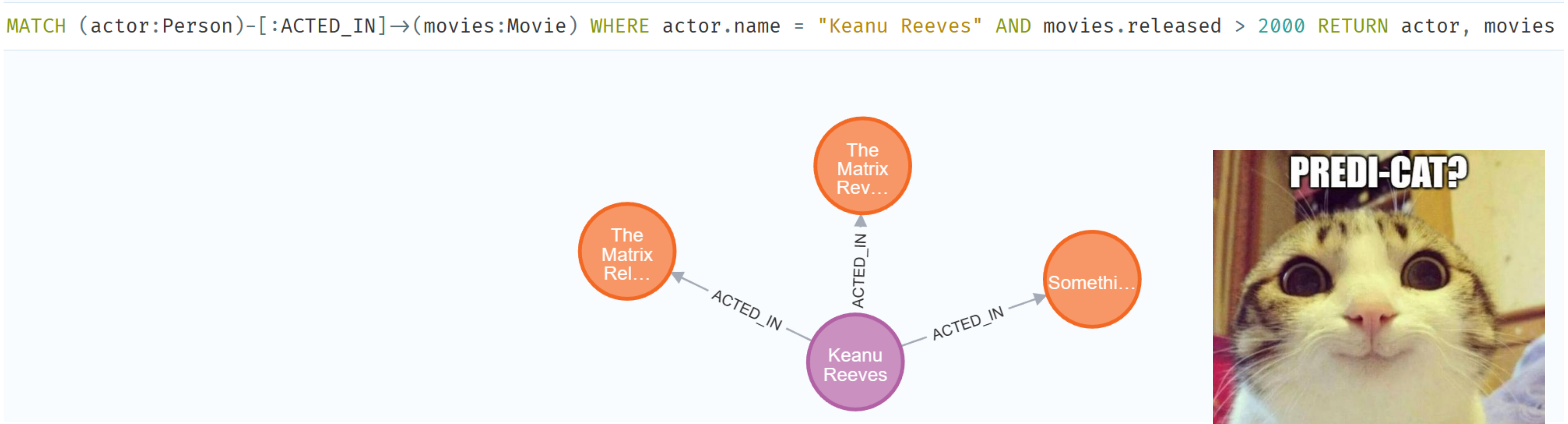


4 billion queries daily

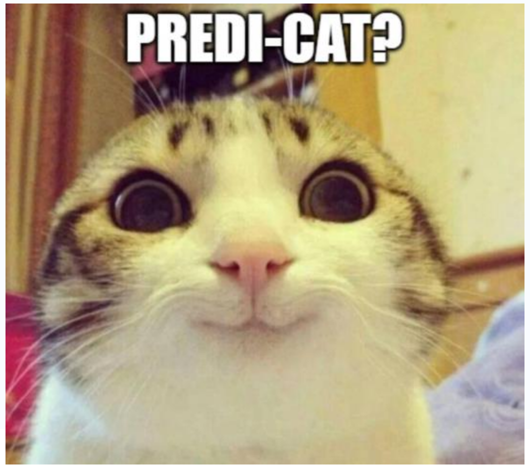

Brage overview
Results
Platform
Data Source
Cypher query
Brage Compiler
Brage Runtime
Binary program
Service
User

Cypher: Graph query language

"Get me all movies that Keanu Reeves acted in, that released after 2000"



```
MATCH (actor:Person)-[:ACTED_IN]→(movies:Movie) WHERE actor.name = "Keanu Reeves" AND movies.released > 2000 RETURN actor, movies
```



Neo4j Sandbox

Example from production

```
MATCH (user)-[myModifications:ModifiedByUser]->(file)
WHERE id(user) = 13 AND myModifications.ActivityDateTime > datetime() -
duration({days: 30})
OPTIONAL MATCH (file)<-[othersModifications: ModifiedByUser]-(otherUsers)
WHERE othersModifications. ActivityDateTime > datetime() -
duration({days: 30})
RETURN file.Name,
       collect( { ModificationTime: othersModifications.ActivityDateTime,
                  UserName:         otherUsers.Name})
```

1

Lexing

Program text => Sequence of tokens

Unknown tokens => Syntax error!

1

Lexing

```
MATCH (user)-[myModifications:ModifiedByUser]->(file)
```

⬇

```
KwMatch, LParen, Identifier, RParen, Dash, LBrack, Identifier,
Colon, Identifier, RBrack, Dash, RAngle, LParen, Identifier, RParen
```

1

Lexing

```
KwMatch, LParen, Identifier, RParen, Dash, LBrack, Identifier,
Colon, Identifier, RBrack, Dash, RAngle, LParen, Identifier,
RParen, KwWhere, Identifier, LParen, Identifier, RParen, Eq,
UnsignedLongInteger, KwAnd, Identifier, Dot, Identifier, RAngle,
Identifier, LParen, RParen, Dash, Identifier, LParen, LCurl,
Identifier, Colon, Integer, RCurl, RParen, KwOptional, KwMatch,
LParen, Identifier, RParen, LAngle, Dash, LBrack, Identifier,
Colon, Identifier, RBrack, Dash, LParen, Identifier, RParen,
KwWhere, Identifier, Dot, Identifier, RAngle, Identifier, LParen,
RParen, Dash, Identifier, LParen, LCurl, Identifier, Colon,
Integer, RCurl, RParen, KwReturn, Identifier, Dot, Identifier,
Comma, Identifier, LParen, LCurl, Identifier, Colon, Identifier,
Dot, Identifier, Comma, Identifier, Colon, Identifier, Dot,
Identifier, RCurl, RParen, Eof
```

Parsing

Token sequence => Abstract Syntax Tree

Unknown token pattern => Syntax error!

Language rules defined in Extended Backus-Naur form (EBNF)

Our EBNF: https://opencypher.org/resources/

Parsing

```
KwMatch, LParen, Identifier, Rparen
```

⬇

```
AllNodes[ user ]
```

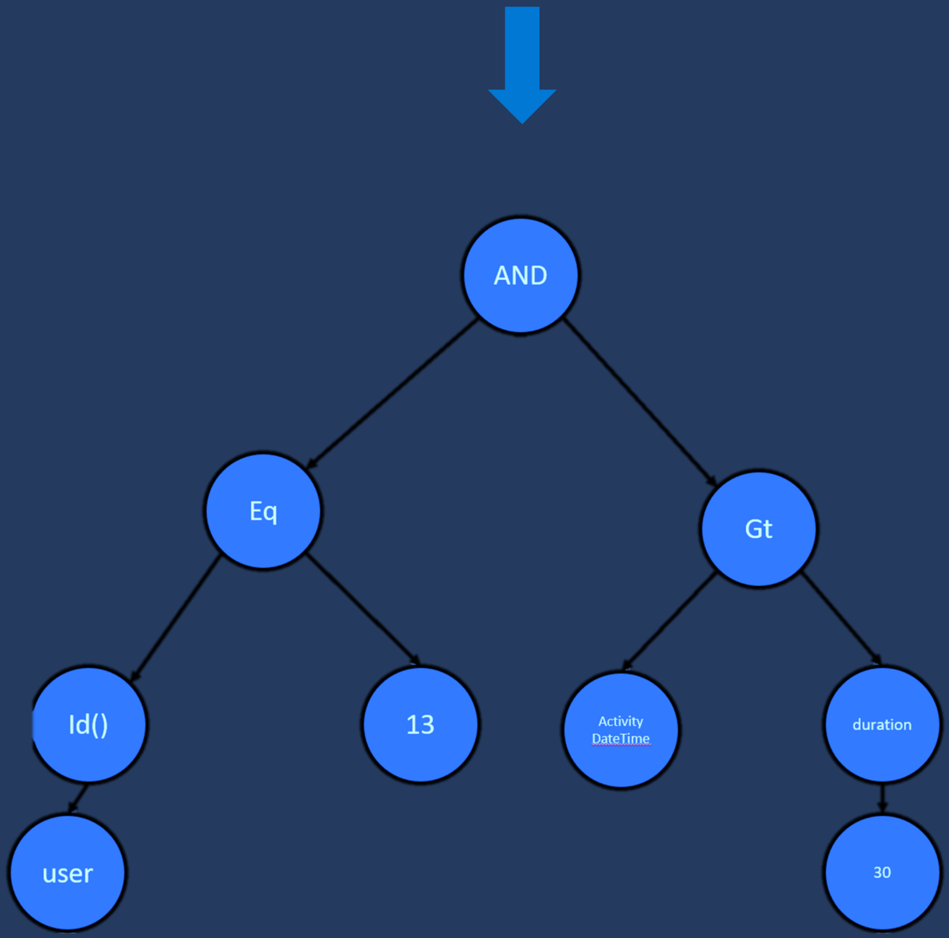
```
Dash, LBrack, Identifier, Colon, Identifier, RBrack, Dash, Rangle

LParen, Identifier, RParen
```

⬇

```
Expand[ () myModifications, user --> file, myModifications :
Properties: (), file : Properties: () ]
```

```
KwWhere, Identifier, LParen, Identifier, RParen, Eq, UnsignedLongInteger,
KwAnd, Identifier, Dot, Identifier, RAngle, Identifier, LParen, RParen, Dash,
Identifier, LParen, LCurl, Identifier, Colon, Integer, RCurl, Rparen
```



OR

```
Where[ ((id(user) Eq 13) AND (myModifications.ActivityDateTime Gt datetime() - duration({days:"30"}))) ]
```

2

Parsing

```
+Return[  ]
|
+Materialize[
file.Name,collect({ModificationTime:"othersModifications.ActivityDa
teTime", UserName:"otherUsers.Name"}) ]
|
+Apply
|\
| +Option[  ]
| |
| +Where[ (othersModifications.ActivityDateTime Gt datetime() -

| |  duration({days:"30"})) ]
| |
| +Expand[ () othersModifications, file <-- otherUsers,

| |othersModifications : Properties:(), otherUsers : Properties:()]
| |
| +AllNodes[ file ]
|
+Where[ ((id(user) Eq 13) AND (myModifications.ActivityDateTime Gt
| datetime() - duration({days:"30"}))) ]
|
+Expand[ () myModifications, user --> file, myModifications :

| Properties: (), file : Properties: () ]
|
+AllNodes[ user ]
```

Analyze

3

3

Analyze: Type checking

- int x = “hello UIT”
- bool y = 5 / false

```
public bool number(int x) {
        return “hello”;
}

number(“hello”);
```

3

Analyze: Scope checking

```
// Here we have nodes a,b,c,x,y,z

…
WITH x, y, z
….

// Here we have nodes x,y,z


WHERE a.name = “Lisa”
```

4

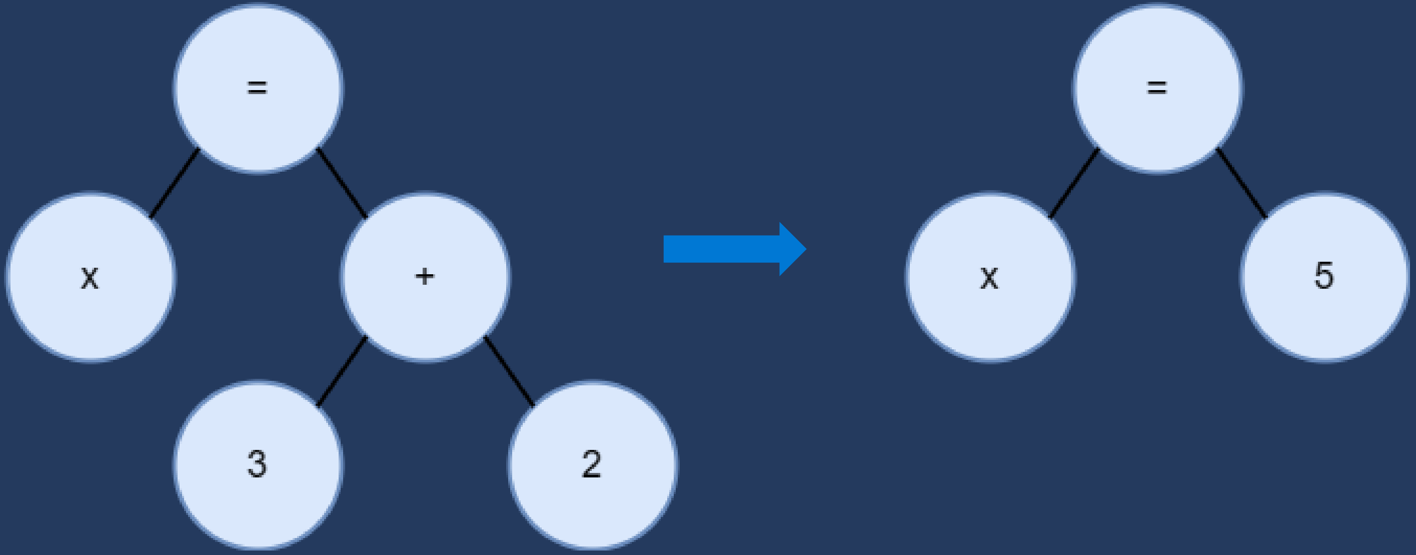
Optimization

- Never change the meaning of the program
- GCC is black magic

- Brage optimizations made for our use case

Optimization: Eager evaluation

```
x = 3 + 2
```



Optimization: Eager evaluation

```
…
+Apply
|\
| +Option[  ]
| |
| +Where[ (othersModifications.ActivityDateTime Gt datetime() –
| | duration({days:"30"})) ]
| |
| +Expand[ () othersModifications, file <-- otherUsers,
| | othersModifications : Properties:(), otherUsers: Properties:()]
| |
| +AllNodes[ file ]
|
+Where[ ((id(user) Eq 13) AND (myModifications.ActivityDateTime Gt
| datetime()duration({days:"30"}))) ]
|
+Expand[ () myModifications, user --> file, myModifications :
| Properties: (), file : Properties: () ]
|
+AllNodes[ user ]
```

4

Optimization: Eager evaluation

```
…
+Apply
|\
| +Option[  ]
| |
| +Where[ (othersModifications.ActivityDateTime Gt datetime() –
| | 30.00:00:00) ]
| |
| +Expand[ () othersModifications, file <-- otherUsers,
| | othersModifications : Properties:(), otherUsers: Properties:()]
| |
| +AllNodes[ file ]
|
+Where[ ((id(user) Eq 13) AND (myModifications.ActivityDateTime Gt
| datetime() - 30.00:00:00)) ]
|
+Expand[ () myModifications, user --> file, myModifications :
| Properties: (), file : Properties: () ]
|
+AllNodes[ user ]
```

4

Optimization: Anchor argument

```
…
+Apply
|\
| +Option[  ]
| |
| +Where[ (othersModifications.ActivityDateTime Gt datetime() –

| | 30.00:00:00) ]
| |
| +Expand[ () othersModifications, file <-- otherUsers,

| | othersModifications : Properties:(), otherUsers: Properties:()]
| |
| +AllNodes[ file ]
|
+Where[ ((id(user) Eq 13) AND (myModifications.ActivityDateTime Gt
| datetime() - 30.00:00:00)) ]
|
+Expand[ () myModifications, user --> file, myModifications :

| Properties: (), file : Properties: () ]
|
+AllNodes[ user ]
```

4

Optimization: Anchor argument

```
…
+Apply
|\
| +Option[  ]
| |
| +Where[ (othersModifications.ActivityDateTime Gt datetime() –
| | 30.00:00:00) ]
| |
| +Expand[ () othersModifications, file <-- otherUsers,
| | othersModifications : Properties:(), otherUsers: Properties:()]
| |
| +Argument[ file ]
|
+Where[ ((id(user) Eq 13) AND (myModifications.ActivityDateTime Gt
| datetime() - 30.00:00:00)) ]
|
+Expand[ () myModifications, user --> file, myModifications :
| Properties: (), file : Properties: () ]
|
+AllNodes[ user ]
```

Optimization: Load properties

```
…
+Apply
|\
| +Option[  ]
| |
| +Where[ (othersModifications.ActivityDateTime Gt datetime() –

| | 30.00:00:00) ]
| |
| +Expand[ () othersModifications, file <-- otherUsers,

| | othersModifications : Properties:(), otherUsers: Properties:()]
| |
| +Argument[ file ]
|
+Where[ ((id(user) Eq 13) AND (myModifications.ActivityDateTime Gt
| datetime()duration({days:"30"}))) ]
|
+Expand[ () myModifications, user --> file, myModifications :

| Properties: (), file : Properties: () ]
|
+AllNodes[ user ]
```

Optimization: Load properties

```
…
+Apply
|\
| +Option[  ]
| |
| +Where[ (othersModifications.ActivityDateTime Gt datetime() –
| 30.00:00:00) ]
| |
| +Expand[ () othersModifications, file <-- otherUsers,
| | othersModifications : Properties: (ActivityDateTime),
| | otherUsers : Properties: (Name) ]
| |
| +Argument[ file ]
|
+Where[ ((id(user) Eq 13) AND (myModifications.ActivityDateTime Gt
| datetime() - 30.00:00:00)) ]
|
+Expand[ () myModifications, user --> file, myModifications :
| Properties: (ActivityDateTime), file : Properties: (Name) ]
|
+AllNodes[ user ]
```

Optimization: Predicate pushdown

```
…
+Apply
|\
| +Option[  ]
| |
| +Where[ (othersModifications.ActivityDateTime Gt datetime() –

| 30.00:00:00) ]
| |
| +Expand[ () othersModifications, file <-- otherUsers,

| | othersModifications : Properties: (ActivityDateTime),

| | otherUsers : Properties: (Name) ]
| |
| +Argument[ file ]
|
+Where[ ((id(user) Eq 13) AND (myModifications.ActivityDateTime Gt
| datetime() - 30.00:00:00)) ]
|
+Expand[ () myModifications, user --> file, myModifications :

| Properties: (ActivityDateTime), file : Properties: (Name) ]
|
+AllNodes[ user ]
```

4

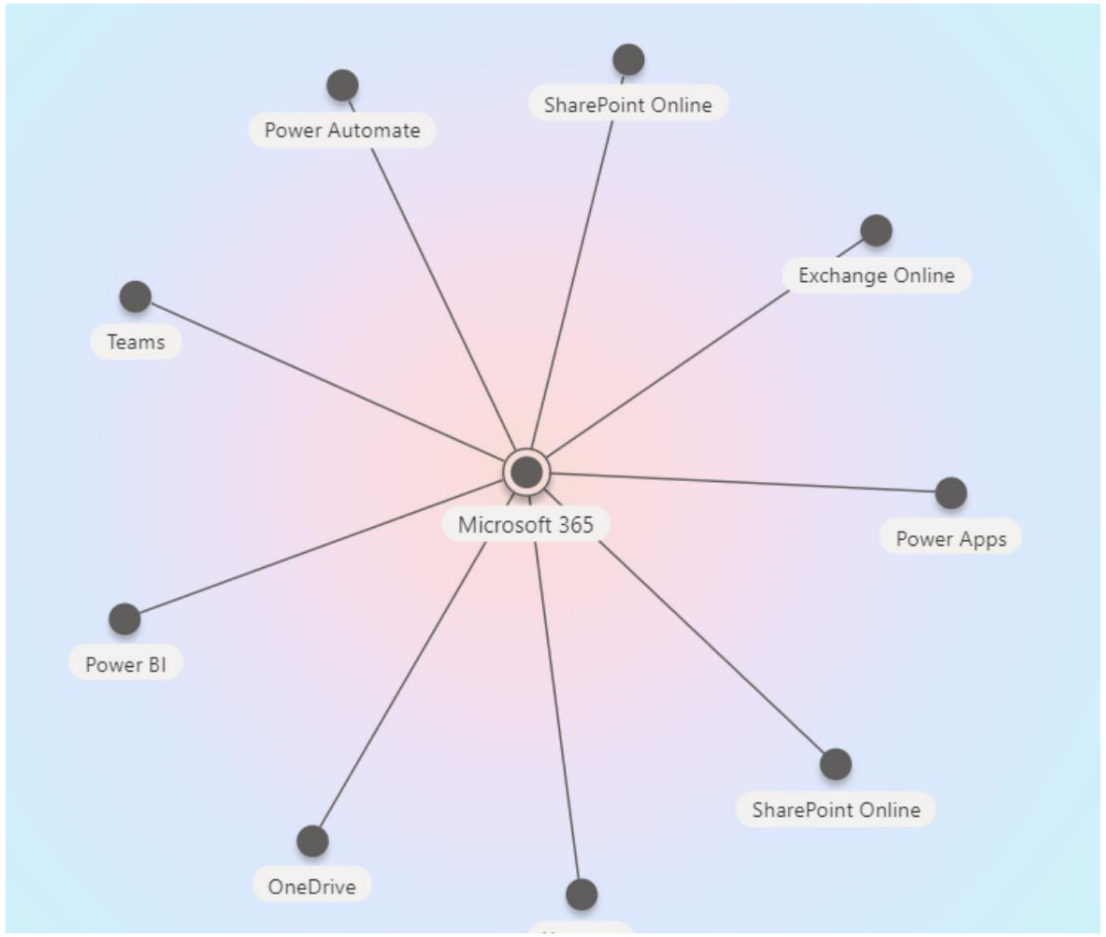
Optimization: Predicate pushdown

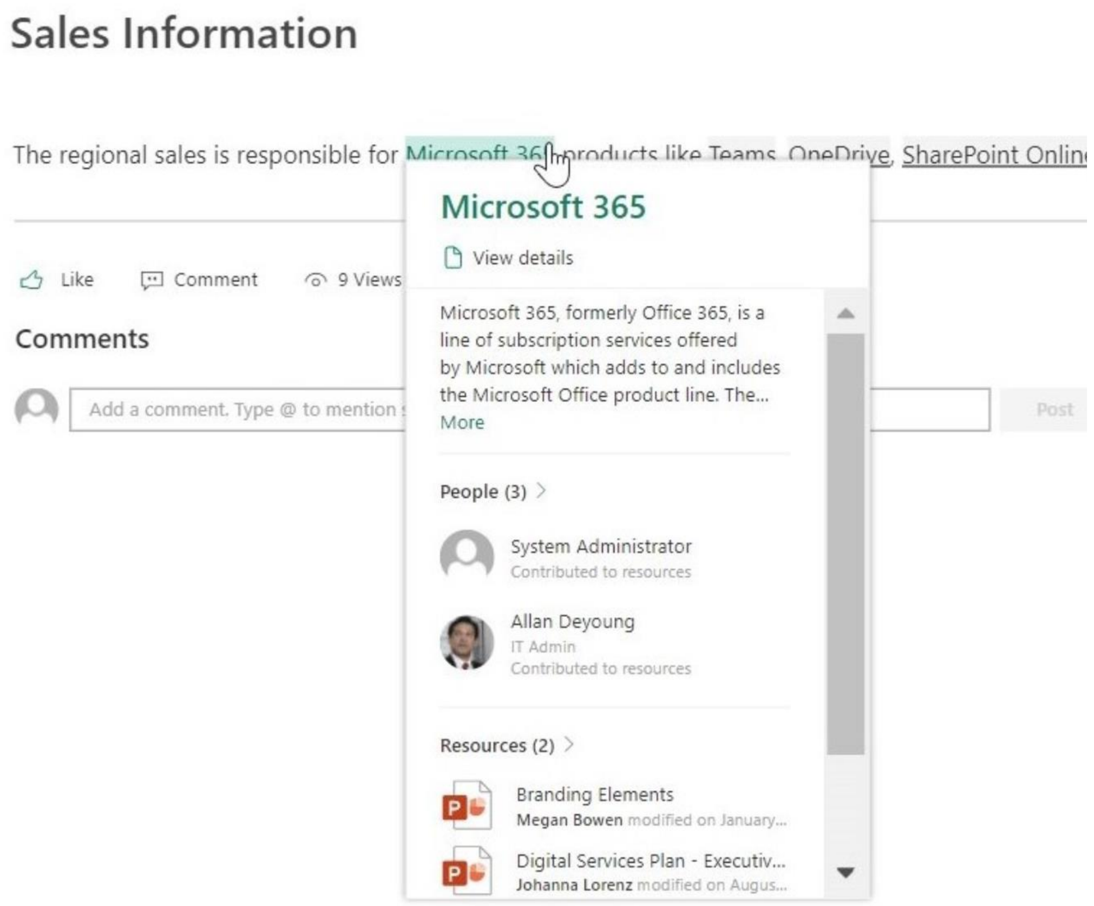
```
…
+Apply
|\
| +Option[  ]
| |
| +Where[ (othersModifications.ActivityDateTime Gt datetime() –
| 30.00:00:00) ]
| |
| +Expand[ () othersModifications, file <-- otherUsers,
| | othersModifications : Properties: (ActivityDateTime),
| | otherUsers : Properties: (Name) ]
| |
| +Argument[ file ]
|
+Where[(myModifications.ActivityDateTime Gt datetime() –
| 30.00:00:00) ]
|
+Expand[ () myModifications, user --> file, myModifications :
| Properties: (ActivityDateTime), file : Properties: (Name) ]
|
+NodeById[ user, ids:13 Properties: () ]
```

5

Code generation

Powered by Brage: Viva Topics




Sales Information

The regional sales is responsible for Microsoft 365 products like Teams, OneDrive, SharePoint Onlin

Like Comment 9 Views

Comments

Add a comment. Type @ to mention

Post

Microsoft 365

View details

Microsoft 365, formerly Office 365, is a line of subscription services offered by Microsoft which adds to and includes the Microsoft Office product line. The...

More

People (3)

System Administrator
Contributed to resources

Allan Deyoung
IT Admin
Contributed to resources

Resources (2)

Branding Elements
Megan Bowen modified on January...

Digital Services Plan - Executiv...
Johanna Lorenz modified on Augus...

Microsoft

Careers @ Microsoft

https://careers.microsoft.com/students

Lisa:
https://www.linkedin.com/in/lisa-jonsson-6a932a151/
Anders:
anders.t.gjerdrum@uit.no