Querying code: a database theoretician’s perspective

Jan Van den Bussche

DSI
DATA SCIENCE INSTITUTE
UHASSELT
WWW.UHASSELT.BE/DSI

What is a database query?

It is a function:

Input: A database D

Output: Selected data retrieved from D, possibly regrouped or transformed

A **query language** is a declarative programming language for expressing such functions

- typically limited expressiveness

So far **code search** fits this well

- data: strings
- query language: regex

Code is more than strings

- Versioning (not this talk)
- Grammatical structure

Suggests the use of **patterns** as a query language

- Several talks here
- Older work on **mining** of code repositories

Pattern-based query languages

Most database query languages follow this pattern (sic):

- have a basis in **patterns**
- further apply **operators** on sets of matchings

E.g., SQL:

- patterns are select—from—where
- operators: union, set difference, grouping, aggregation, subquery

Similarly, XQuery (XML, JSON), SPARQL (RDF), Cypher (graphs)

Code analysts

Like data analysts, but for code

- Look for bad coding style / coding guidelines violations
- Deprecated library usage
- Opportunities for optimization
- “We’ve done that before…where was that again?”
- etc.

Could benefit from a fully fledged XQuery-like query language

- **done in SE:** Datalog, rule-based program transformation, …

Code is more than strings!

- Versioning (not this talk)
- Grammatical structure
- Code can be **evaluated / executed**

Suggests querying code for **behavior:**

- "On this file, which of our string-to-int functions return a negative number?"
- "If we replaced function `foo()` by function `bar()`, which of our programs would give a different result on this input?"

Meta-SQL: Querying queries

- Project we did 20 years ago, several papers, low impact
- Leverage **XML columns** in SQL/XML
    - store SQL queries in table cells
    - SQL/XML allows to apply arbitrary XQuery functions
    - XML aggregation
    - query, **transform** stored SQL code
- Add **eval** function to execute queries
    - result (table) is represented again in XML
    - **closed** data model

Meta-SQL example

- Querying a **query log**
- ‘‘For each query in the log whose answer includes a price column, give the maximum of that column’’

```
select L.Q, max(get_price(x))
from Log L, x in UEVAL(L.Q)
where has_price(x) = ‘true’
group by L.Q
```

- Further research we did:
  - **type system** for such meta queries
  - expressive power (**transitive closure** query becomes expressible)