Do Programming Languages need Query Languages?

Jelle Hellings

Department of Computing and Software
McMaster University
1280 Main St. W., Hamilton, ON L8S 4L7, Canada

Do Programming Languages need Query Languages?

(spoiler alert)

Yes they do!

(and vice versa)

Data processing

Claims

1. Data processing plays a central role in programming.

Data processing

Claims

1. Data processing plays a central role in programming.

Examples: standard support libraries

1. *Collections* to store data
   binary search trees, hash tables, tuples, dynamic arrays, ...
2. *Algorithms* to operate on data
   sorting, filtering, transforming, set operations, ...

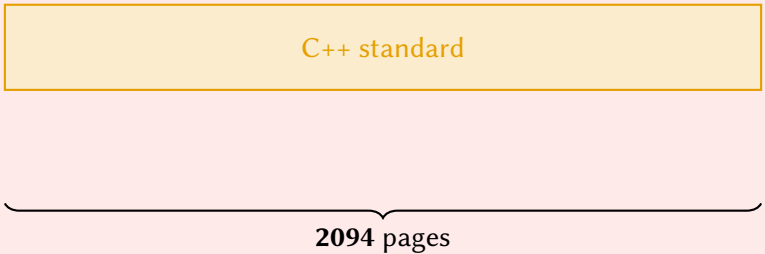
Data processing

Claims

1. Data processing plays a central role in programming.

Working Draft of the C++ standard

(Document N4981, Date 2024-04-16.)



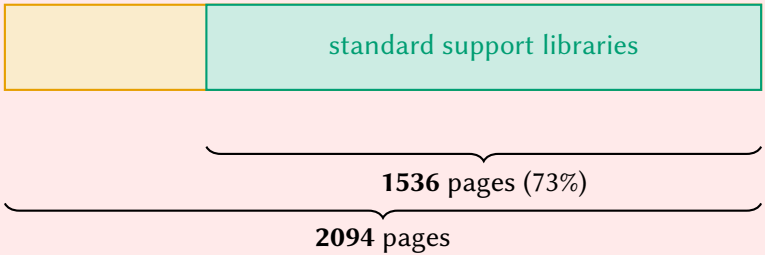
Data processing

Claims

1. Data processing plays a central role in programming.

Working Draft of the C++ standard

(Document N4981, Date 2024-04-16.)



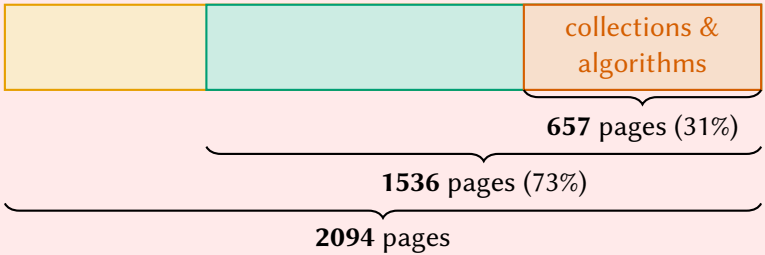
Data processing

Claims

1. Data processing plays a central role in programming.

Working Draft of the C++ standard

(Document N4981, Date 2024-04-16.)



Not counting: numeric, time, formatting, IO, threads, ...

An example: How to *program* relational algebra in C++

$$\pi_{R.\text{parent}}(\sigma_{R.\text{child}=S.\text{name}}(\rho_R(\text{parentOf}) \times \sigma_{S.\text{place}=\text{“Hamilton”}}(\rho_S(\text{person}))))$$

An example: How to *program* relational algebra in C++

$$\pi_{R.\text{parent}}(\sigma_{R.\text{child}=S.\text{name}}(\rho_R(\text{parentOf}) \times \sigma_{S.\text{place}=\text{“Hamilton”}}(\rho_S(\text{person}))))$$

```
using namespace std::views;

auto where_pred = [](auto l) { return l.place == "Hamilton"; };

auto product_pred = [](auto t) { auto [po, p] = t;
                                 return po.child == p.name; };

auto join = cartesian_product(parents, persons | filter(where_pred));
for (auto [po, p] : join | filter(product_pred)) {
    std::cout << po.parent << std::endl;
}
```

Programming and data processing

Claims

1. Data processing plays a central role in programming.
2. Programming languages are *bad* at data processing.

Programming and data processing

Claims

1. Data processing plays a central role in programming.
2. Programming languages are *bad* at data processing.

Efficient data processing

"*Complex*" data processing algorithms even for *simple* data processing tasks.
E.g., join algorithms, join ordering, index usage, selection push down, ...

This complexity is delegated to the programmer.

An example: A *more-efficient* query in C++

```
/* parents is a set, ordered on (parent, child).
 * persons is a set, ordered on (name, place). */

auto where_pred = [](auto l) { return l.place == "Hamilton"; };
auto filtered = persons | filter(where_pred)
                        | std::ranges::to<std::vector>();

for (auto& [pname, cname] : parents) {
  person_t p{cname, "Hamilton"};
  bool has_child = std::binary_search(filtered.begin(),
                                      filtered.end(), p);

  if (has_child) std::cout << pname << std::endl;
}
```

An example: A *more-efficient* query in C++

```
/* parents is a set, ordered on (parent, child).
 * persons is a set, ordered on (name, place). */

auto where_pred = [](auto l) { return l.place == "Hamilton"; };
auto filtered = persons | filter(where_pred)
                        | std::ranges::to<std::vector>();

for (auto& [pname, cname] : parents) {
  person_t p{cname, "Hamilton"};
  bool has_child = std::binary_search(filtered.begin(),
                                      filtered.end(), p);

  if (has_child) std::cout << pname << std::endl;
}
```

What if we want *unique* parents?

An example: A *more-efficient* query in SQL

```
SELECT pname
FROM parents R, persons S
WHERE R.child = S.name AND S.place = "Hamilton";
```

An example: A *more-efficient* query in SQL

```
SELECT DISTINCT pname
FROM parents R, persons S
WHERE R.child = S.name AND S.place = "Hamilton";
```

An example: A *more-efficient* query in SQL

```
SELECT DISTINCT pname
FROM parents R, persons S
WHERE R.child = S.name AND S.place = "Hamilton";
```

Some database systems need a helping hand…

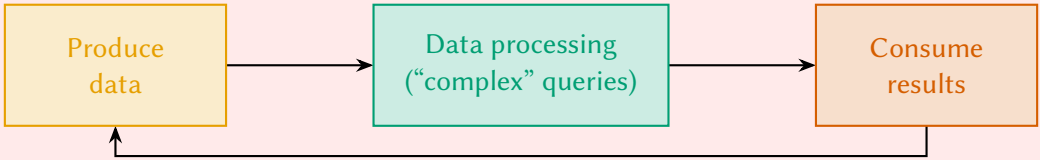
```
SELECT DISTINCT pname
FROM parents
WHERE child IN (SELECT name
                FROM persons
                WHERE place = "Hamilton");
```

Programming and data processing

Claims

1. Data processing plays a central role in programming.
2. Programming languages are *bad* at data processing.
3. We need database technology in our programming languages.

Potential solution?

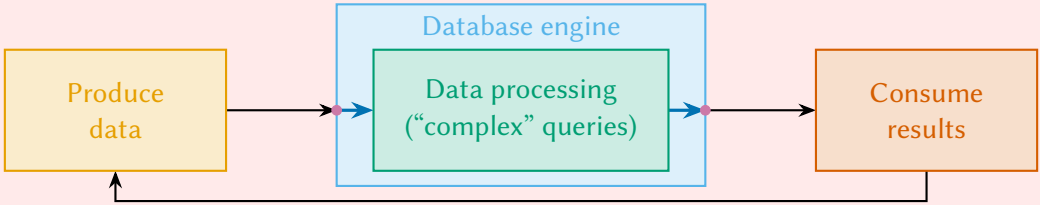


Programming and data processing

Claims

1. Data processing plays a central role in programming.
2. Programming languages are *bad* at data processing.
3. We need database technology in our programming languages.

Potential solution?

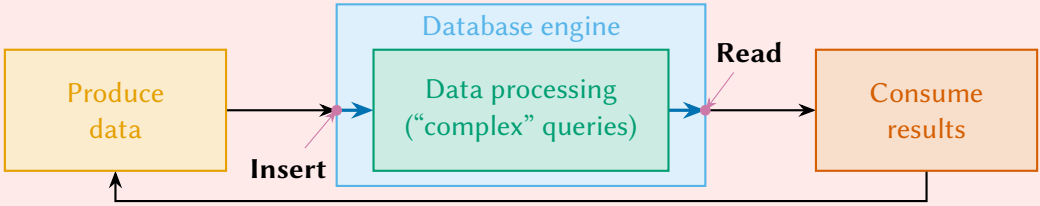


Programming and data processing

Claims

1. Data processing plays a central role in programming.
2. Programming languages are *bad* at data processing.
3. We need database technology in our programming languages.

Potential solution?



Proposed solution

What if…

```
#include "my_fancy_library.hpp"
          ⋮
query<"%(parent) :- parents(parent, c),"
      "             persons(c, 'Hamilton')">(dataset);
```

is valid C++ code in which **query** generates an *optimized* C++ algorithm (at compile time).

Proposed solution

What if…

```
#include "my_fancy_library.hpp"
          ⋮
query<"%(parent) :- parents(parent, c),"
      "             persons(c, 'Hamilton')">(dataset);
```

is valid C++ code in which **query** generates an *optimized* C++ algorithm (at compile time).

How to do so

Create a C++ library that

- Provides a *domain specific query language* for C++.
- At compile time: use database technologies to turn *queries* into C++ algorithms.
- At runtime: these algorithms are *normal C++ functions* that answer queries efficiently.

Proposed solution: Detailed plans

1. Design the *domain specific query language*:
   Target: multiset aware Datalog with stratified negation and aggregation.

2. A *compile-time parser* for the query language:
   Result: the Datalog program in some internal representation.

3. Use database technology to find an *optimal query plan*:
   Goal: turn the internal representation into an efficient query algorithm.

4. At runtime: *execute* the optimal query plan.
   With zero overhead: the query plan representation *is* the algorithm &
   a programmer should not be able to do significantly better.

C++ compile-time support for *domain specific languages*

Almost feature complete (optimistic: publication in 2024?).

What we already have

- A *high-level grammar language* that controls parsing and syntax tree generation.
- A C++ compile-time parser generator that turns grammars into *compile-time parsers*.
- Utilities to transform syntax trees into *C++ functionality* (e.g., algorithms).
- Runtime parser generator support and debugging utilities.

C++ compile-time support for *domain specific languages*

Challenges

Source: https://xkcd.com/303/ by Randall Munroe.

Database technology and finding *optimal query plans*

Idea

Look at existing database systems and use their approach to query optimization.
A compile-time environment is different, however: e.g., no access to data.

Database technology and finding *optimal query plans*

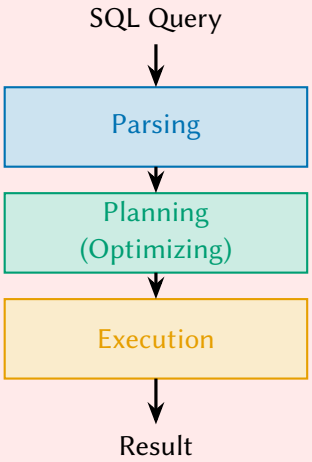
Idea

Look at existing database systems and use their approach to query optimization.
A compile-time environment is different, however: e.g., no access to data.

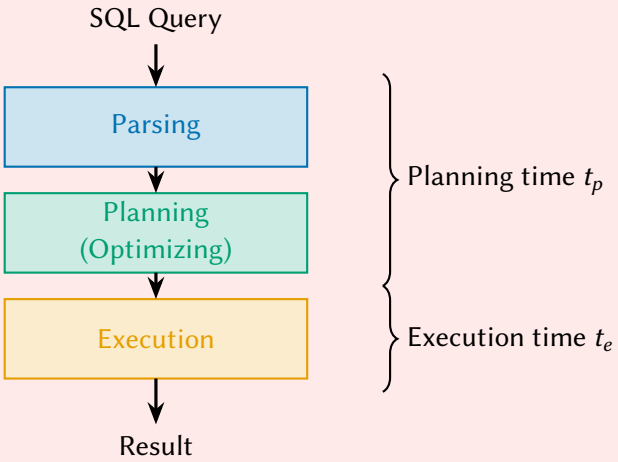
Claims

1. Data processing plays a central role in programming.
2. Programming languages are *bad* at data processing.
3. We need database technology in our programming languages.
4. Typical database technology is *not good* at finding optimal query plans.

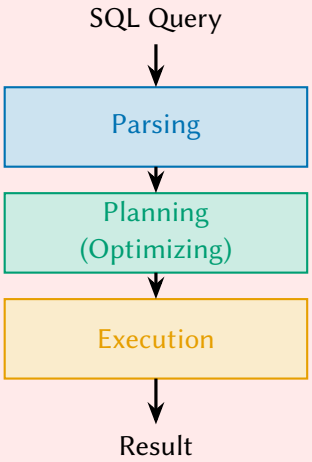
Typical query evaluation



Typical query evaluation



Typical query evaluation



Planning time t_p

Execution time t_e

Database systems:
Attempt to minimize $t_p + t_e$.

Issue: An SQL-based example [FoIKS 2022b]

```
SELECT DISTINCT C.cname, P.type
FROM customer C, bought B, product P
WHERE C.cname = B.cname AND
      B.pname = P.pname AND
      P.type = 'food';
```

Issue: An SQL-based example [FoIKS 2022b]

```
SELECT DISTINCT C.cname, P.type
FROM customer C, bought B, product P
WHERE C.cname = B.cname AND
      B.pname = P.pname AND
      P.type = 'food';
```

Planning time:	14.014 ms
Execution time:	192.833 ms

Issue: An SQL-based example [FoIKS 2022b]

```
SELECT DISTINCT C.cname, P.type
FROM customer C, bought B, product P
WHERE C.cname = B.cname AND
      B.pname = P.pname AND
      P.type = 'food';
```

Planning time: 14.014 ms
Execution time: 192.833 ms

Manual optimizations

```
SELECT cname, 'food' AS type
FROM customer WHERE cname IN (
    SELECT cname FROM bought
    WHERE pname IN (
        SELECT pname FROM product
        WHERE type = 'food'));
```

Planning time: 8.700 ms
Execution time: 75.195 ms

Issue: A graph-inspired example [FoIKS 2022b]

```
SELECT DISTINCT S.nfrom
FROM edges S, edges R, edges T, edges U
WHERE S.nto = R.nfrom AND
      R.nto = T.nfrom AND
      T.nto = U.nfrom;
```

Issue: A graph-inspired example [FoIKS 2022b]

```
SELECT DISTINCT S.nfrom
FROM edges S, edges R, edges T, edges U
WHERE S.nto = R.nfrom AND
      R.nto = T.nfrom AND
      T.nto = U.nfrom;
```

$\pm\mathcal{O}(|\text{edges}|^3)$–$\mathcal{O}(|\text{edges}|^4)$:
Evaluation will not complete.

Issue: A graph-inspired example [FoIKS 2022b]

```
SELECT DISTINCT S.nfrom
FROM edges S, edges R, edges T
WHERE S.nto = R.nfrom AND
      R.nto = T.nfrom AND
      T.nto = T.nfrom;
```

Planning time:	20.026 ms
Execution time:	107 405.232 ms

Issue: A graph-inspired example [FoIKS 2022b]

```
SELECT DISTINCT S.nfrom
FROM edges S, edges R, edges T, edges U
WHERE S.nto = R.nfrom AND
      R.nto = T.nfrom AND
      T.nto = U.nfrom;
```

Planning time: 20.026 ms
Execution time: 107 405.232 ms

Manual optimizations

```
SELECT DISTINCT nfrom FROM edges
WHERE nto IN (
  SELECT nfrom FROM edges
  WHERE nto IN (
    SELECT nfrom FROM edges
    WHERE nto IN (
      SELECT nfrom FROM edges)));
```

Planning time: 8.700 ms
Execution time: 75.195 ms

Should we focus on finding optimal query plans?

Do we need optimal query plans?

Database technology with some tweaking is good-enough for practical situations???

Should we focus on finding optimal query plans?

A typical *query pattern P* in a website

```
SELECT a.name, art.id, art.title, art.content, art.tcreate
FROM author a, article art
WHERE a.id = art.author_id AND art.id = ?;
```

- Pattern *P* is used every time a user visits an article on the website.
- Pattern *P* only changes when the source code of the website changes.

Should we focus on finding optimal query plans?

A typical *query pattern P* in a website

```
SELECT a.name, art.id, art.title, art.content, art.tcreate
FROM author a, article art
WHERE a.id = art.author_id AND art.id = ?;
```

- Pattern *P* is used every time a user visits an article on the website.
- Pattern *P* only changes when the source code of the website changes.

Say our website can force the database system to fully optimize *P once*

- The system takes 10 min to optimize *P*.
- The system makes evaluating *P only* 10 ms faster on average.
- The website is not very busy: only 1 visitors/s.

Should we focus on finding optimal query plans?

A typical *query pattern P* in a website

```
SELECT a.name, art.id, art.title, art.content, art.tcreate
FROM author a, article art
WHERE a.id = art.author_id AND art.id = ?;
```

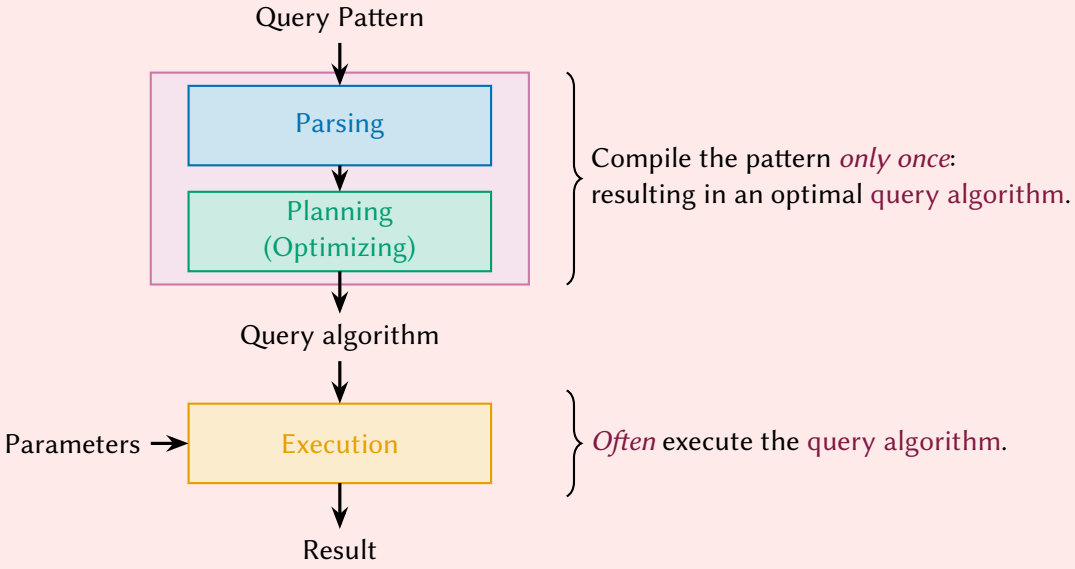
- Pattern *P* is used every time a user visits an article on the website.
- Pattern *P* only changes when the source code of the website changes.

Say our website can force the database system to fully optimize *P once*

- The system takes 10 min to optimize *P*.
- The system makes evaluating *P only* 10 ms faster on average.
- The website is not very busy: only 1 visitors/s.

Overall benefit within 17 h.

Research focus: Fully-optimized query evaluation



Evaluation: Two perspectives

First perspective: How do we compare with existing database products.

Evaluation: Two perspectives

First perspective: How do we compare with existing database products.

Second perspective: How do we compare with (C++) *programmers*?

- Performance?
- Readability?
- Ease-of-use?

Evaluation: Two perspectives

First perspective: How do we compare with existing database products.

Second perspective: How do we compare with (C++) *programmers*?

- Performance?
- Readability?
- Ease-of-use?

Central question

Should we switch to *declarative languages* even in traditional *procedural* source code?

Research volunteers needed

To evaluate our research, we will compare with data processing programs written in C++.

You can provide these C++ programs via an online questionnaire.

- Letter of Information https://jhellings.nl/q/loi.pdf.
- Questionnaire at https://jhellings.nl/q/.

The questionnaire takes 30min–120min and participation is entirely voluntarily.

This study has been reviewed by and received ethics clearance from the *McMaster Research Ethics Board* (#6885).

Conclusion

Programming Languages do need Query Languages!

References at https://jhellings.nl/.

Questionnaire at https://jhellings.nl/q/.