Unifying Query Interpretation and Compilation

Philipp M. Grulich, Aljoscha Lepping, Dwi Nugroho, Varun Pandey, Bonaventura Del Monte, Steffen Zeuch, Volker Markl
Three Query Compilers
Take Away

Our query compilers enable high performance through data and hardware-tailored specialization.
Our query compilers enable **high performance** through **data and hardware-tailored specialization**.

However, query compilers lead to **high system complexity** and require a **high engineering effort**!
“[Query Compilation] is great for performance, but it is difficult for students to maintain and debug the code.”
April 2021, Database Deep Dives with Andy Pavlo
Decreasing Industry-Adoption

“Code generators are harder to build and debug than interpreted-engines.”
Sigmod 2022, Photon: A Fast Query Engine for Lakehouse Systems

“Query Compilation increases engine complexity, makes it harder to onboard new engineers, and retain high development velocity.”
PVLDB 2022, Photon: A Fast Query Engine for Lakehouse Systems

“Use cases where codegen provides clear benefits, outweighing compilation delays, decreased developer productivity, and debuggability are [still] under investigation.”
PVLDB 2022, Velox: Meta’s Unified Execution Engine
Let's take a step back!
Let's take a step back!

Could we unify interpretation and compilation?
Goals
1. Push-based query interpretation
   - Aligns control and data-flow within execution.
   - Fits well with task/morsel-driven parallelization.
Goals

1. Push-based query interpretation

2. Native Operator Implementations
   - Support for standard control flow, virtual functions, abstractions.
   - Native support for debugging and testing.

```cpp
class Selection : public ExecutableOperator{
  void execute (RuntimeContext& ctx, Tuple& tuple){
    // calls child operator only if expression returns true
    if (expression->execute(tuple))
      child->execute(ctx, tuple);
  }
};

class LessThanExpression : public Expression{
  Value execute(Tuple& tuple){
    auto leftValue = leftSubExpression->execute(tuple);
    auto rightValue = rightSubExpression->execute(tuple);
    return leftValue < rightValue;
  }
};
```
Goals

1. Push-based query interpretation

2. Native Operator Implementations

3. Automatic query compilation
   - Generate IR from interpretation-based operators.
   - Selects compilation backend depending on specific workload requirements.
Conclusion

Summary:
✔ Framework with focus on developer experience.
✔ IR to target specialized code-generation backends.