

Exploring Just-in-Time Compilation in Relational Database Engines


Nicolaas van der Merwe

z5467476

University of New South Wales



Contents

1. Introduction – What is query compilation and JIT?
 2. Background – What existing technologies are there?
 3. Benchmarks – Where is the technology up to now?
 4. Proposal – Where to from here and areas to explore
- 

Contents

1. Introduction – What is query compilation and JIT?
2. Background – What existing technologies are there?
3. Benchmarks – Where is the technology up to now?
4. Proposal – Where to from here and areas to explore

Importance of databases

1. Relational databases are used by almost every company – finance, technology, manufacturing, and healthcare are some industries.
2. Tens of thousands of companies use PostgreSQL.
3. It's widely accepted that in most contexts, the database is the bottleneck of the entire system. Some of these systems handle trillions of queries a day.
4. Meaning, even small changes can save billions of dollars of electricity in data centres.

TODO: Sources needed. Oddly hard to find this other than people saying it as generally accepted facts

What is Just-In-Time compilation (JIT)?

1. Interpreted languages (Python, JVM languages, C#) can be very slow
2. They get around this by using JIT, which triggers after a section of code has run a certain number of times
3. When triggered, it compiles a chunk of code into machine code during execution of the program
4. There are situations where JVM code that has been can be faster than optimised C++ code because JIT has metrics about how the code runs. It's very effective and impactful

TODO – find better source for #4 <https://stackoverflow.com/questions/4516778/when-is-java-faster-than-c-or-when-is-jit-faster-then-precompiled>

What tools support JIT?

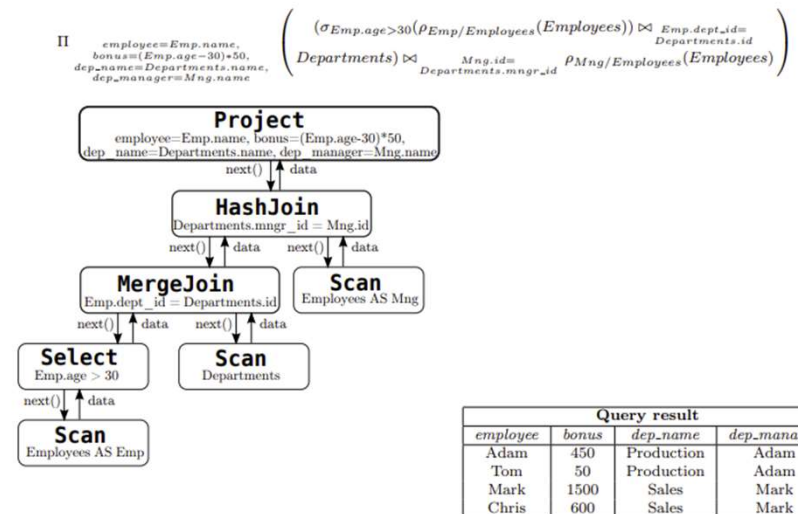
1. Due to the difficulty of implementing JIT itself, most applications of it use some pre-existing tooling
2. It's generally accepted that the most developed JIT is in the JVM due to its age
3. LLVM is another library that supports JIT, with Julia being a major user
4. Some languages do still make their own JIT, like PyPy or WASM (WebAssembly)

What other options are there than JIT?

1. There are some languages that simply don't do JIT, like Python
2. Others do give you the ability to do ahead of time compilation (AOT), like C#
 1. This is a popular choice in UI development because there's a reasonable chance the code paths will never be triggered enough for JIT, but still important enough for requiring optimization

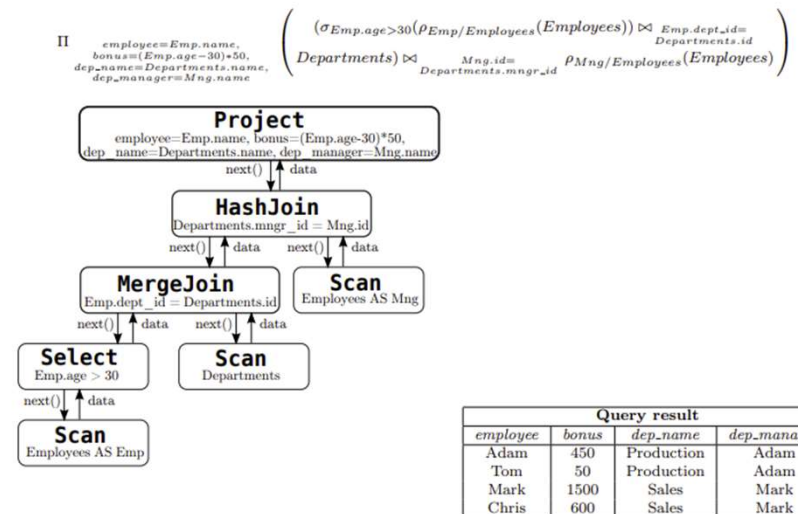
Iterator model

1. Most popular database structure internally
2. Also known as a Volcano model
3. Works by having a main loop that calls next() on an iterator which
4. For instance, Select would call next() on it's child there and filter the result. It will only return the result if it satisfies its filter to the parent.
5. Isn't very effective at using caches because it only handles one tuple at a time

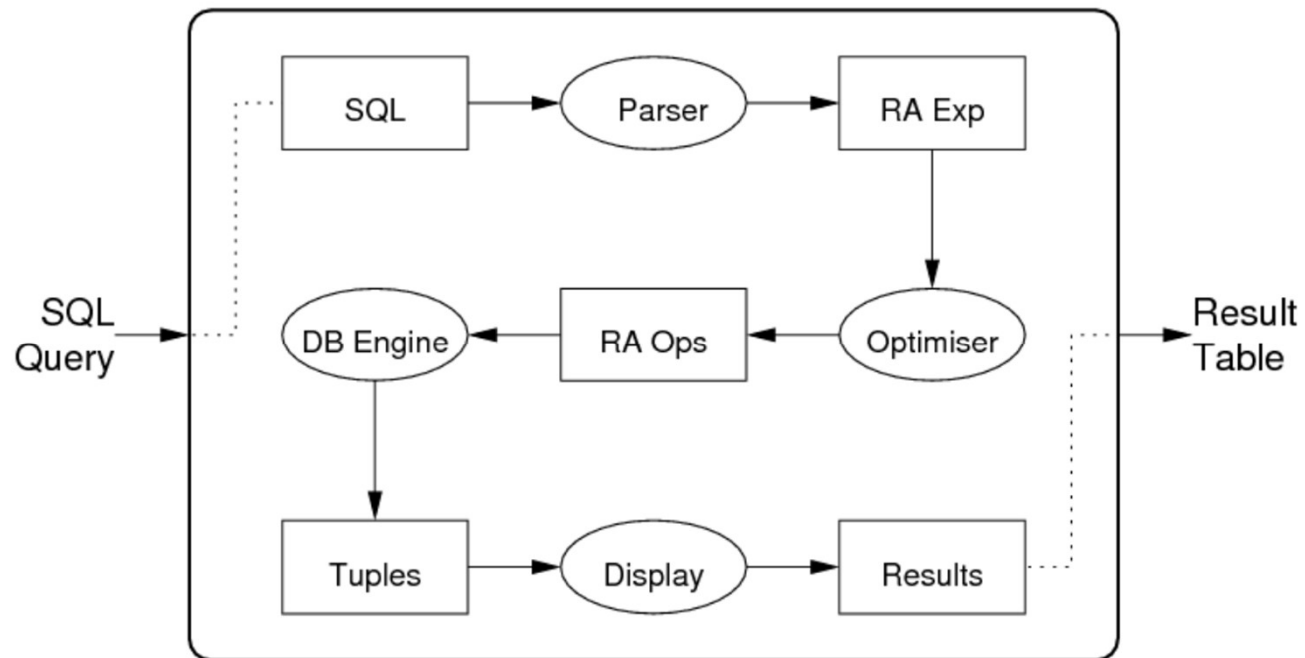


Vector Model

1. Instead of individual tuples on each next() function, they work on batches
2. This reduces overhead of repeated function calls
3. Mostly used in column-wise databases
4. Has less flexibility, and needs more RAM
5. Still worth applying JIT to this

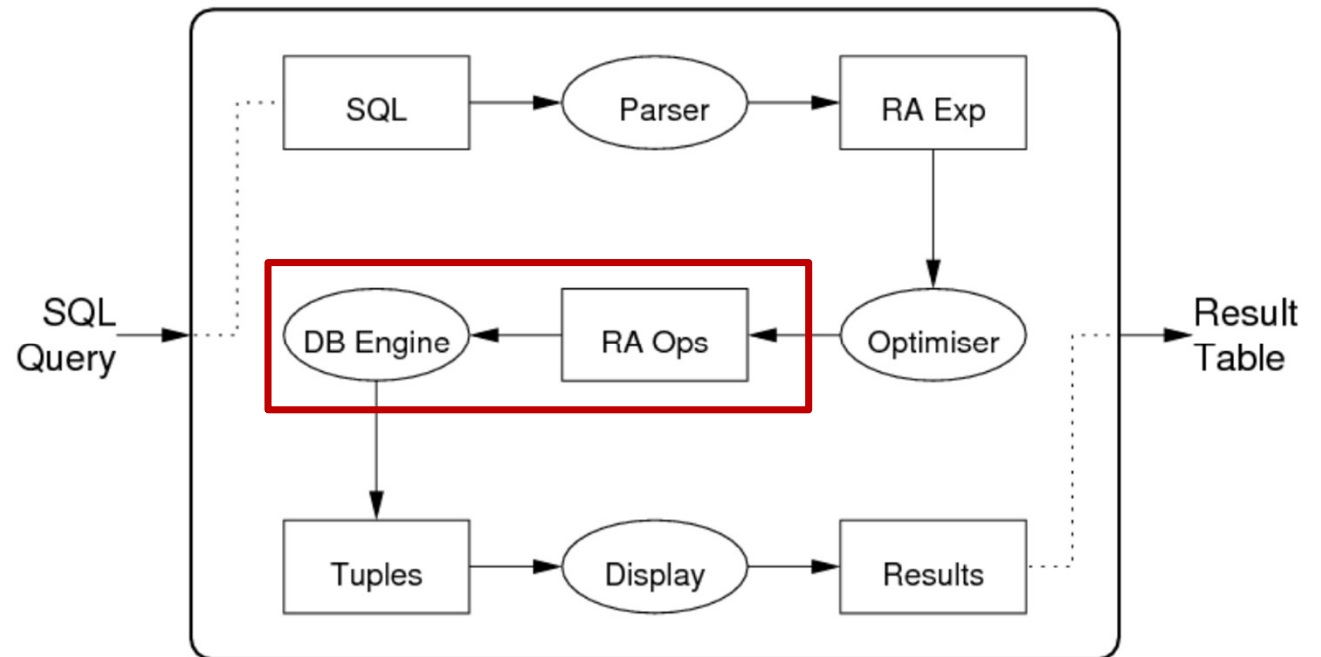


Databasing Query Execution



Where JIT fits

JIT would be used here inside the pipeline, which would be most of the execution time



Contents

1. Introduction – What is query compilation and JIT?
2. Background – What existing technologies are there?
3. Benchmarks – Where is the technology up to now?
4. Proposal – Where to from here and areas to explore

What are the different types of JIT choices in databases

1. Most applications of JIT in databases can be split into having JIT in Query Plan Execution (QPE) and Expression (EXP)
2. QPE-optimised databases are when you use JIT on the entire query plan.
3. At this stage, QPE mostly exists in research databases
4. EXP is when specific operators are compiled, like `age > 30`
5. Only some databases have EXP jit-support, with the flagship one being PostgreSQL
6. PostgreSQL also supports tuple deforming, which is transforming on-disk tuples into in-memory representations

LLVM based JIT compilers in databases

1. HyPer
 1. The pioneer in the space. However, it's hard to test them because they're commercial
2. LingoDB
 1. Uses MLIR and focuses on being a concise implementation of their idea
3. Apache Impala
 1. One of the more established JIT databases, and brags a5x improvement due to JIT. It still lacks common features like nested schemas and indexes
4. PostgreSQL
 1. Very established and strong support, but it only compiles expressions, and tuple transforms so it doesn't fully use JIT

JVM based JIT compilers in databases

1. Derby
2. Neo4J
3. PrestoDB
4. Apache Spark

Other JIT databases

1. Mutable

1. Quite a promising and well-known JIT-supported research database. It compiles to WebAssembly so it's also a unique JIT

2. GreenPlum

There's also NoSQL or specialised databases

1. QuestDB – a time series database
2. RaptorDB – a key-value store for JSON documents

Case: HyPer's use of JIT

1. Postgres introduced JIT in version 12

Case: Postgres's use of JIT

1. Postgres introduced JIT in version 12

Case: Lingo-DB's use of JIT

1. Postgres introduced JIT in version 12

Specialised Databases using JIT

Short list of ones people don't usually look at

Challenges faced when adding JIT

1. Complexity

...

Contents

1. Introduction – What is query compilation and JIT?
2. Background – What existing technologies are there?
3. Benchmarks – Where is the technology up to now?
4. Proposal – Where to from here and areas to explore

Summary

1. ABC seems to claim it's the current fastest

Paper #1's benchmarks

1. Lingo DB's paper

Paper #2's benchmarks

Short list of ones people don't usually look at

Paper #3's benchmarks

Short list of ones people don't usually look at

Benchmarking method

1. Using TPC H

Outcomes

Graph showing results . . .

Revisiting: How much impact would this have?

Some calculations for how much energy large companies use in postgresql

Contents

1. Introduction – What is query compilation and JIT?
2. Background – What existing technologies are there?
3. Benchmarks – Where is the technology up to now?
4. Proposal – Where to from here and areas to explore

Proposed future works by others

1. Putting MLIR into an existing database
2. Expand operators on one of these smaller papers
3. Using novel hardware
4. Further benchmarking to compare the best way to proceed

MLIR

Some more information on MLIR – it's going to be easy to put in, have impact, and be extendable