Towards an Efficient OLAP engine based on Linear Algebra

João Afonso

Supervisors:

Alberto Proença

José N. Oliveira

29-10-2018

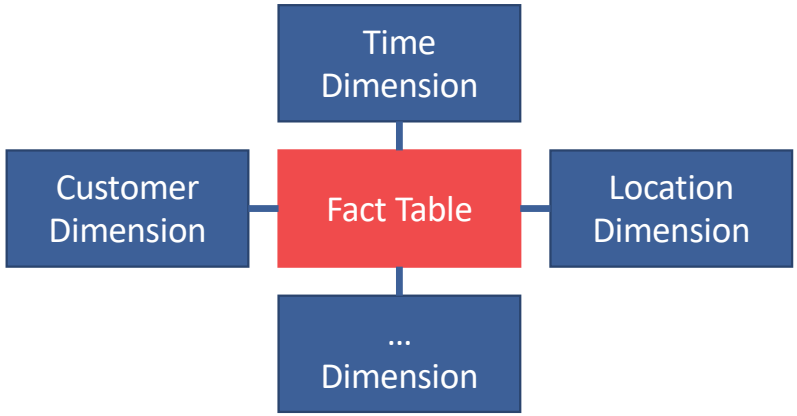
Motivation

Relational systems

- Row oriented
- Low performance in complex business queries

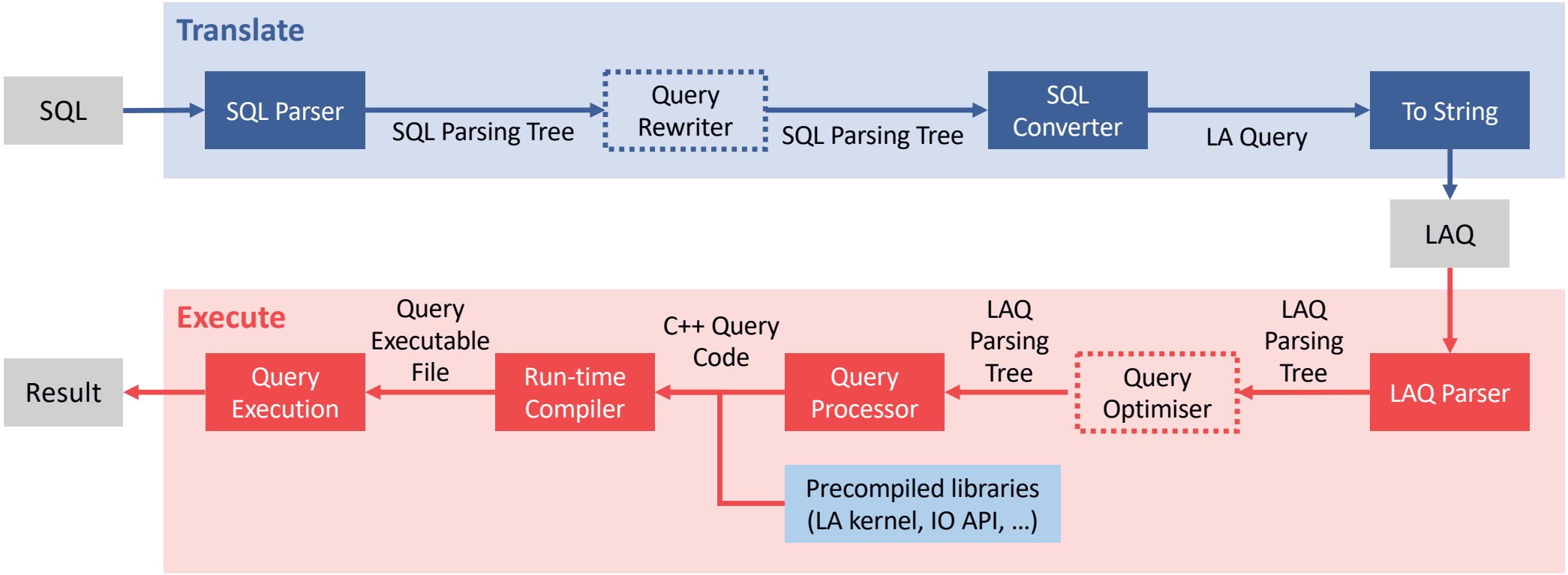
OLAP systems

- Faster by storing pre-computations (cubes)
- **Still based in relational theories!**



How can a columnar theory impact this performance?

TLA-DB Engine



Data Representation

Precompiled libraries
(LA kernel, IO API, …)

Orders.Quantity

1
2
1
3
2

Data Representation

Precompiled libraries
(LA kernel, IO API, …)

Orders.Quantity

1
2
1
3
2

Quantity

	#0	#1	#2	#3	#4
1	1	0	1	0	0
2	0	1	0	0	1
3	0	0	0	1	0

Orders.Quantity

Q ← #o

Data Representation

Precompiled libraries
(LA kernel, IO API, …)

Orders.Quantity

1
2
1
3
2

Quantity

	#0	#1	#2	#3	#4
1	1	0	1	0	0
2	0	1	0	0	1
3	0	0	0	1	0

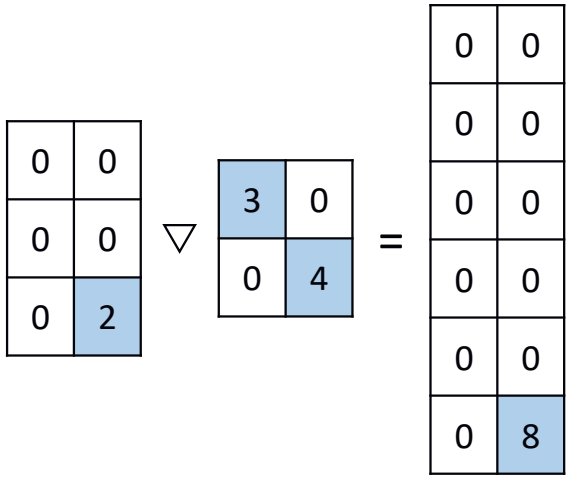
Orders.Quantity
Q ← #o

Quantity (CSC)

Values	1	1	1	1	1	
Rows	0	1	0	2	1	
Col. pointer	0	1	2	3	4	5

Linear Algebra Query language (LAQ)

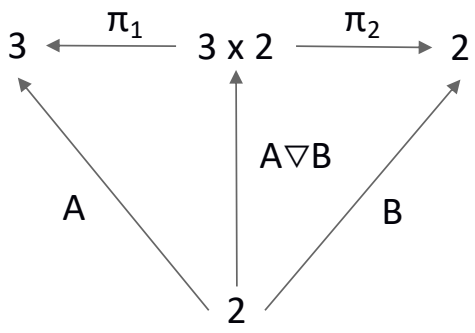
Khatri-Rao



Linear Algebra Query language (LAQ)

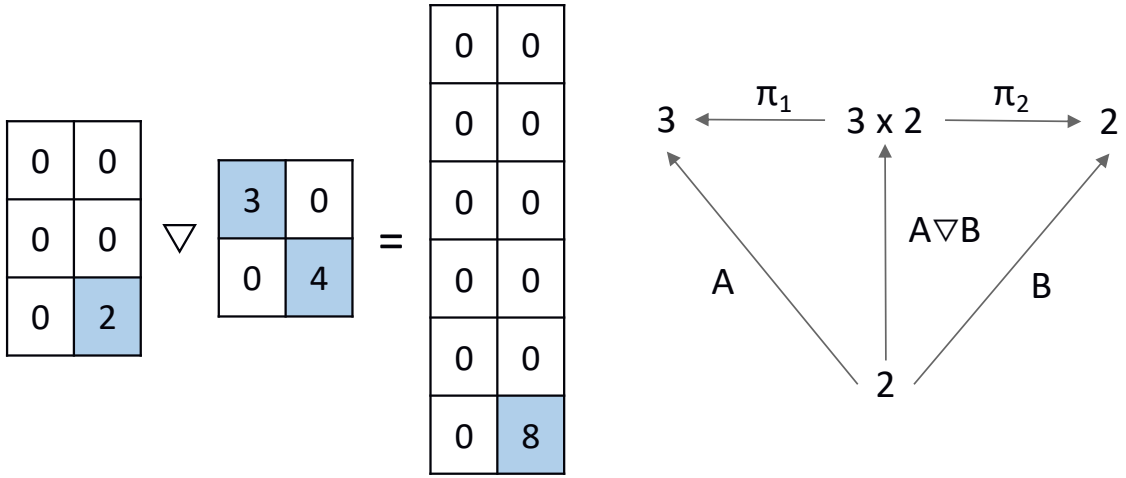
Khatri-Rao

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 2 \end{bmatrix} \triangledown \begin{bmatrix} 3 & 0 \\ 0 & 4 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 8 \end{bmatrix}$$



Linear Algebra Query language (LAQ)

Khatri-Rao



A

V	2		
R	2		
C	0	0	1

B

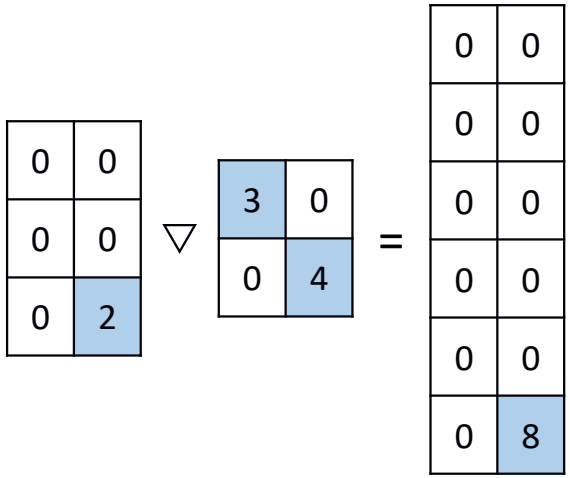
V	3	4	
R	0	1	
C	0	1	2

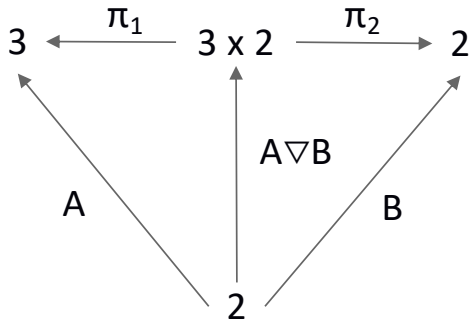
A▽B

V	8		
R	5		
C	0	0	1

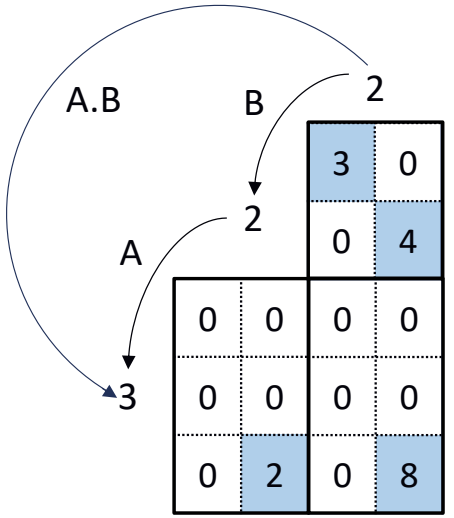
Linear Algebra Query language (LAQ)

Khatri-Rao





Dot Product



A

V	2		
R	2		
C	0	0	1

B

V	3	4	
R	0	1	
C	0	1	2

A▽B

V	8		
R	5		
C	0	0	1

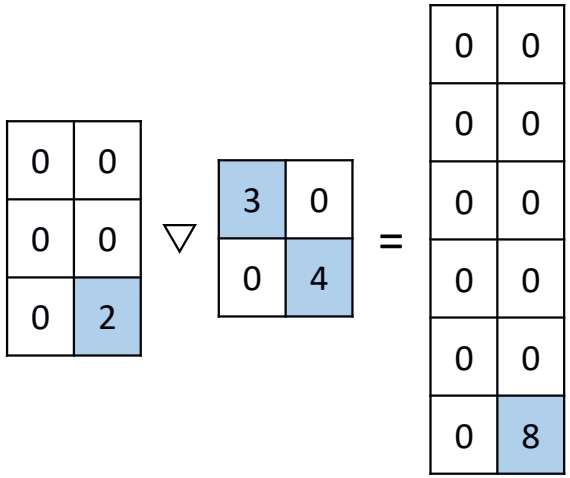
A.B

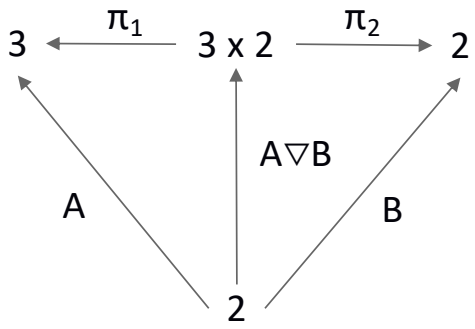
V	8		
R	2		
C	0	0	1

Linear Algebra Query language (LAQ)

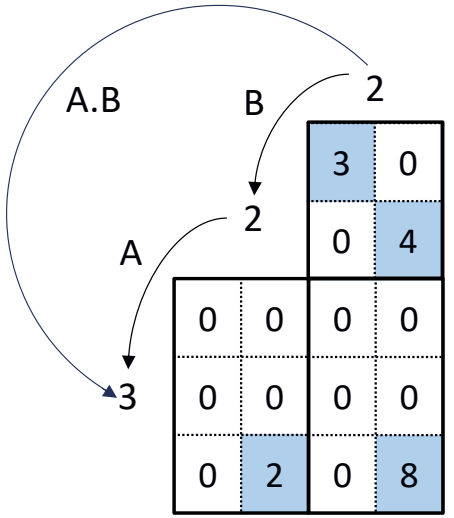
LAQ

Khatri-Rao





Dot Product



Hadamard

Filter

Fold

Lift

A

V	2		
R	2		
C	0	0	1

B

V	3	4	
R	0	1	
C	0	1	2

A▽B

V	8		
R	5		
C	0	0	1

A.B

V	8		
R	2		
C	0	0	1

TPC-H Query 6 – Type Diagram

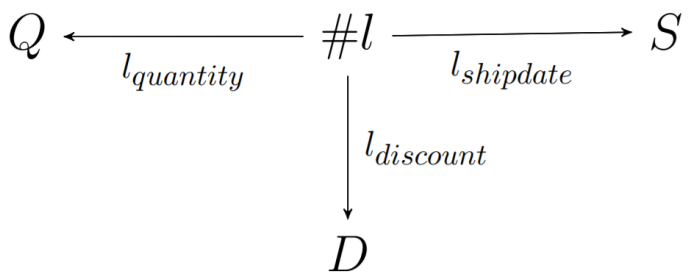
SQL Parser

```
SELECT
  sum(l_extendedprice * l_discount) AS revenue
FROM
  lineitem
WHERE
  l_shipdate >= '1995-03-10'
  AND l_shipdate < '1996-03-10'
  AND l_discount BETWEEN 0.49 AND 0.51
  AND l_quantity < 3;
```

TPC-H Query 6 – Type Diagram

SQL Parser

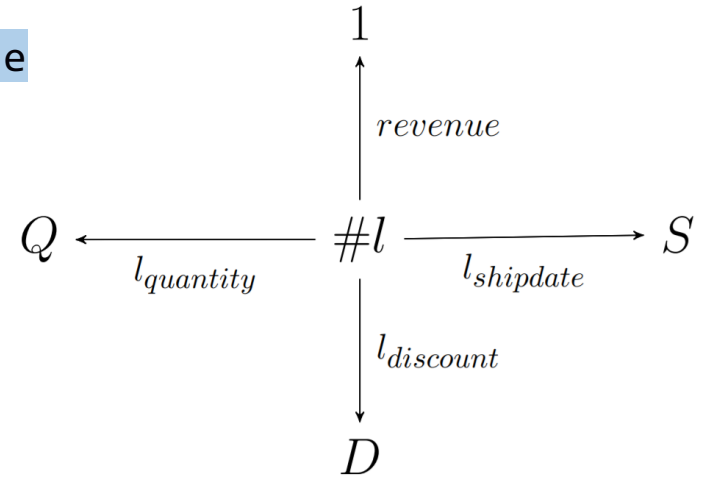
```
SELECT
  sum(l_extendedprice * l_discount) AS revenue
FROM
  lineitem
WHERE
  l_shipdate >= '1995-03-10'
  AND l_shipdate < '1996-03-10'
  AND l_discount BETWEEN 0.49 AND 0.51
  AND l_quantity < 3;
```



TPC-H Query 6 – Type Diagram

SQL Parser

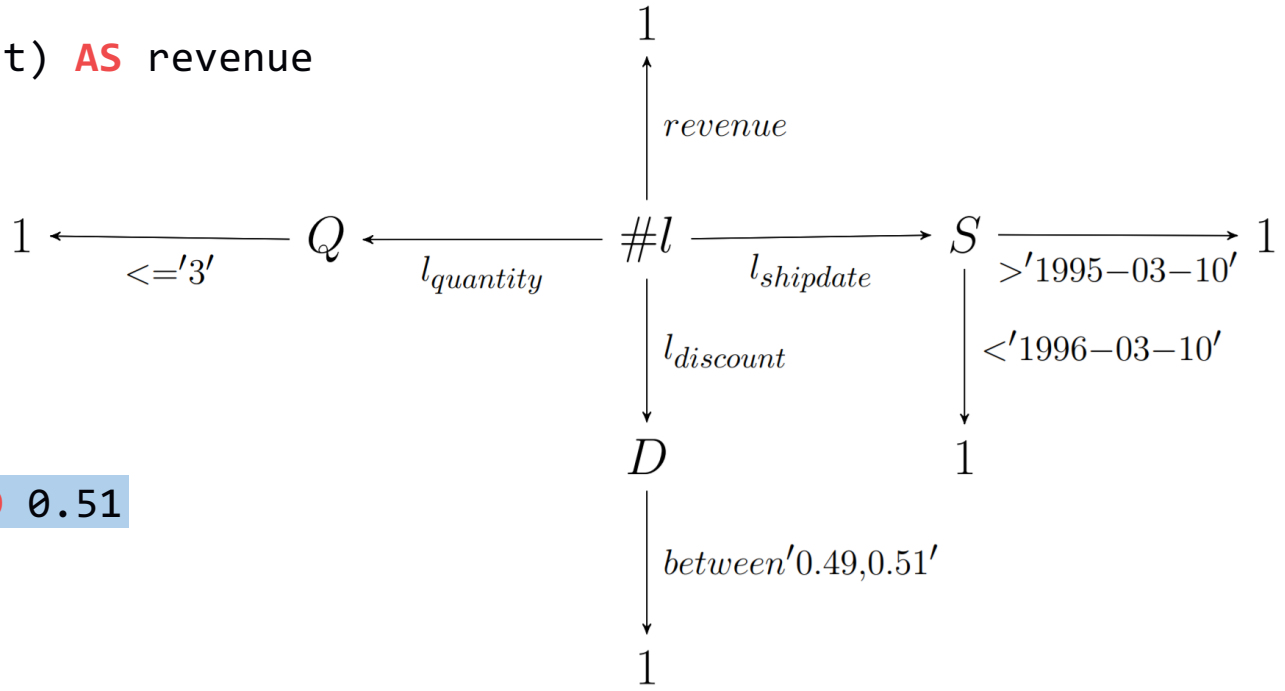
```
SELECT
  sum(l_extendedprice * l_discount) AS revenue
FROM
  lineitem
WHERE
  l_shipdate >= '1995-03-10'
  AND l_shipdate < '1996-03-10'
  AND l_discount BETWEEN 0.49 AND 0.51
  AND l_quantity < 3;
```



TPC-H Query 6 – Type Diagram

SQL Parser

```
SELECT
  sum(l_extendedprice * l_discount) AS revenue
FROM
  lineitem
WHERE
  l_shipdate >= '1995-03-10'
  AND l_shipdate < '1996-03-10'
  AND l_discount BETWEEN 0.49 AND 0.51
  AND l_quantity < 3;
```



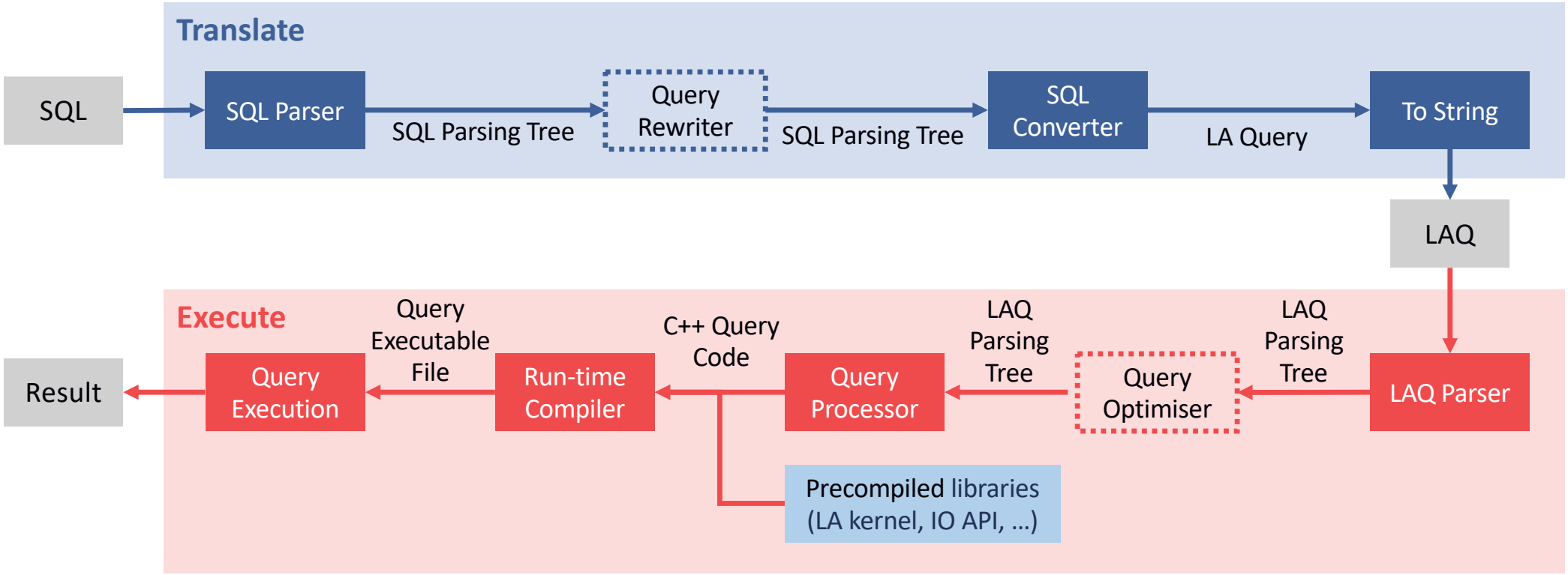
TPC-H Query 6 – LAQ

SQL Converter

```
SELECT
  sum(l_extendedprice * l_discount) AS revenue
FROM
  lineitem
WHERE
  l_shipdate >= ‘1995-03-10’
  AND l_shipdate < ‘1996-03-10’
  AND l_discount BETWEEN 0.49 AND 0.51
  AND l_quantity < 3;
```

```
A = filter( l_shipdate >= “1995-03-10” AND l_shipdate < “1996-03-10” )
B = filter( l_discount >= 0.49 AND l_discount <= 0.51 )
C = hadamard( A, B )
D = filter( l_quantity < 3 )
E = hadamard( C, D )
F = lift( l_extendedprice * l_discount )
G = hadamard( E, F )
H = sum( G )
return ( H )
```

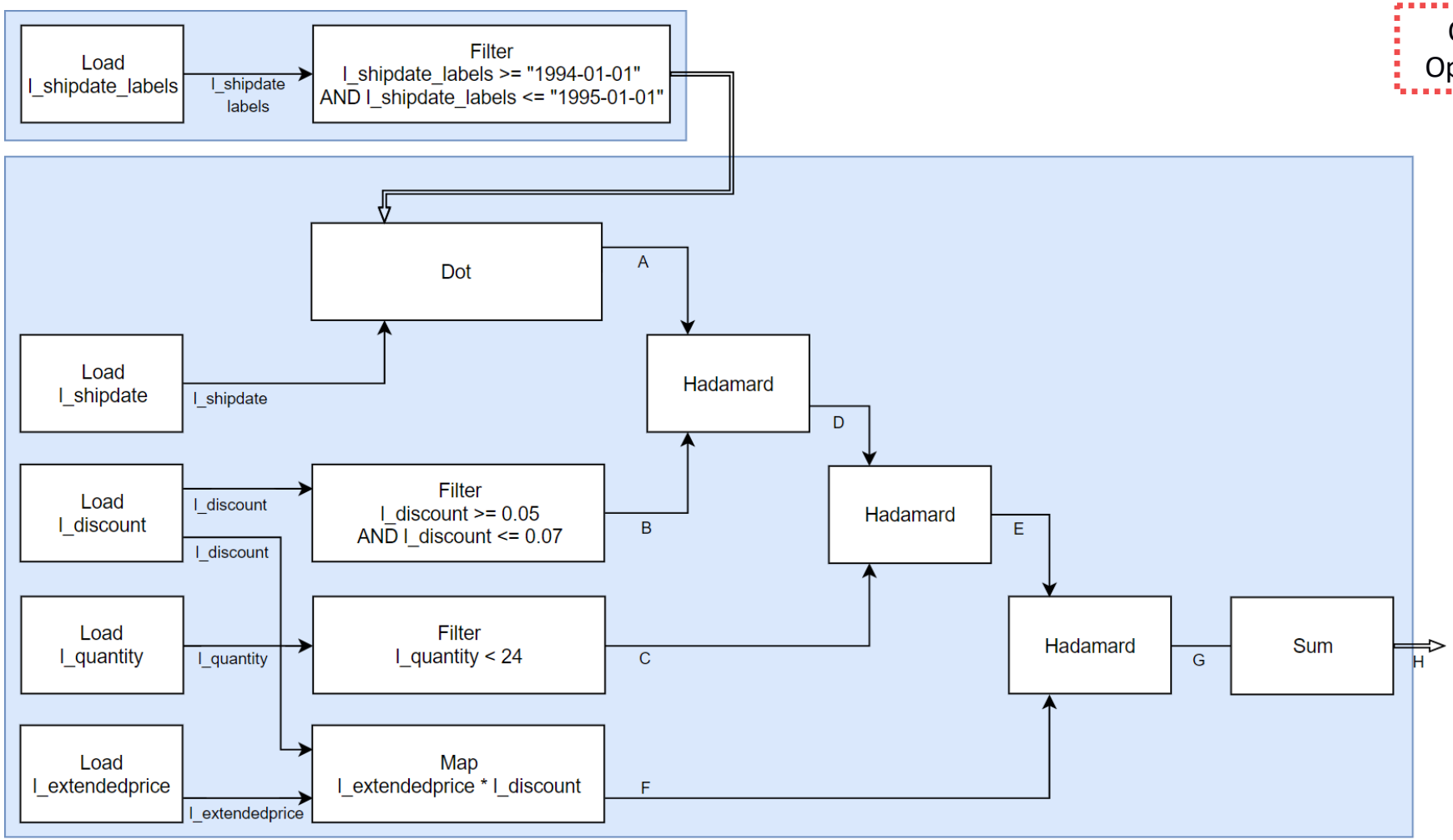
TLA-DB Engine



TPC-H Query 6 – Execution Plan

LAQ Parser

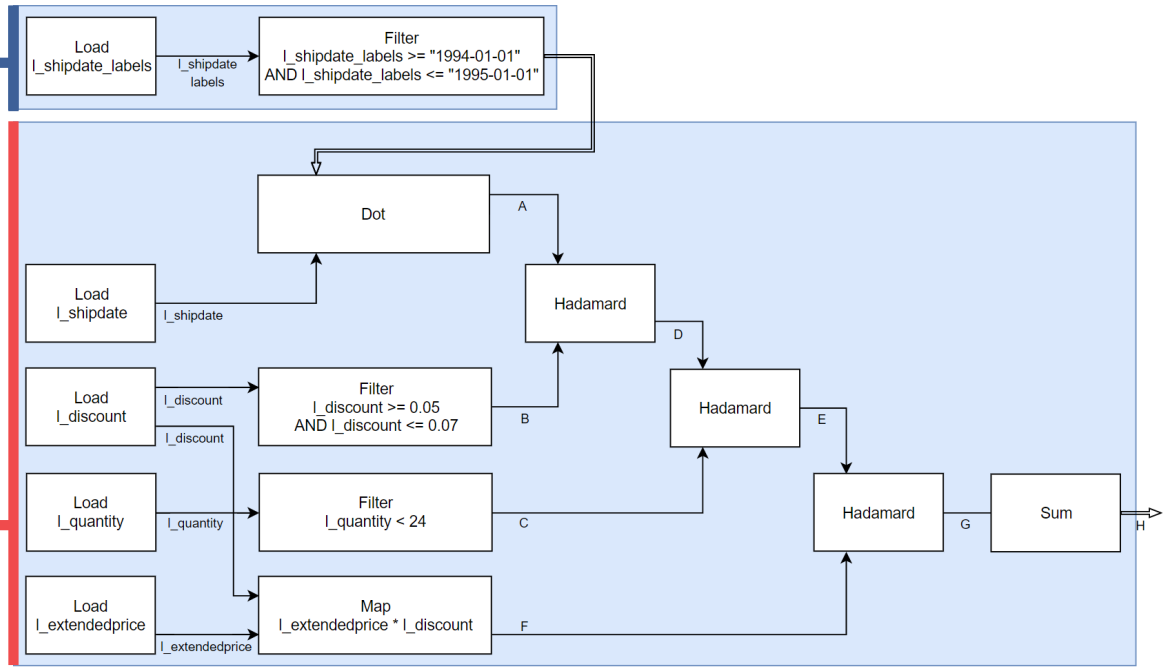
Query Optimiser



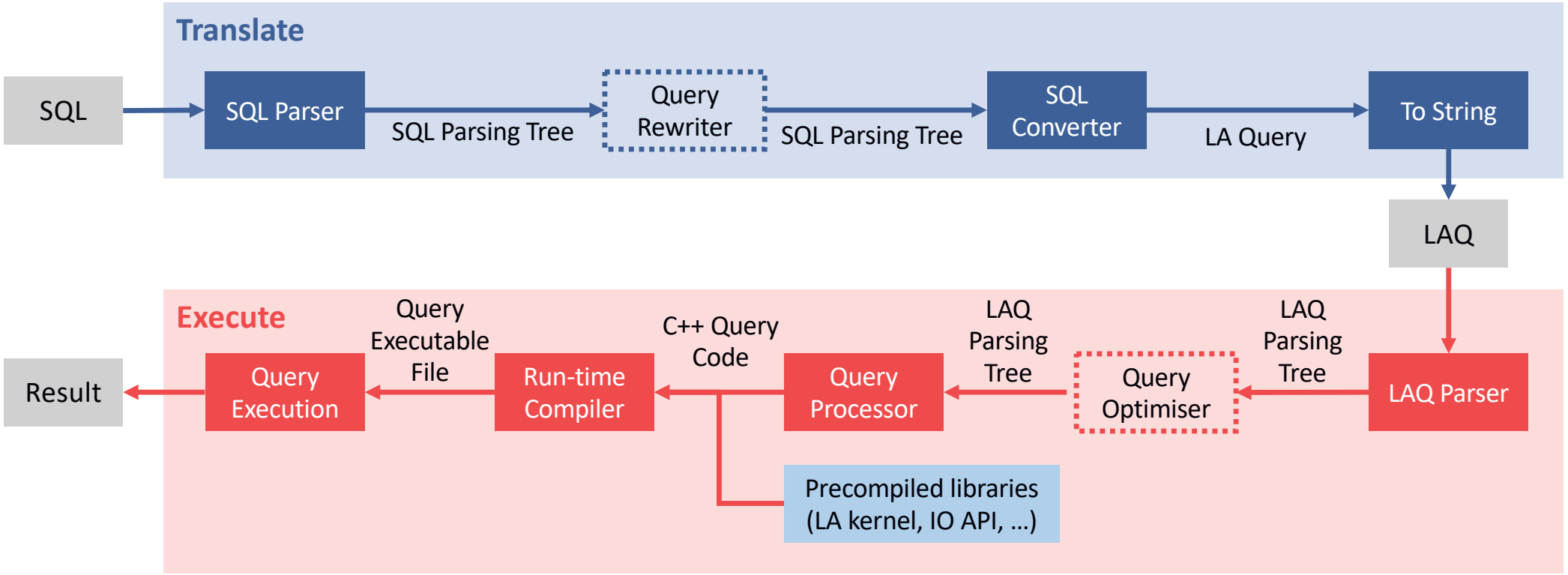
TPC-H Query 6 – Generating C++

Query Processor

```
// 1 – Include precompiled libraries

// 2 – Define necessary expressions

int main() {
  // 3 – Select the database

  // 4 – Load the attributes metadata

  // 5 – Declare temporary matrices

  // 6 – Build the streaming loops
  for (…) {
    …
  }
  for (…) {
    …
  }
  // 7 – Present result
}
```

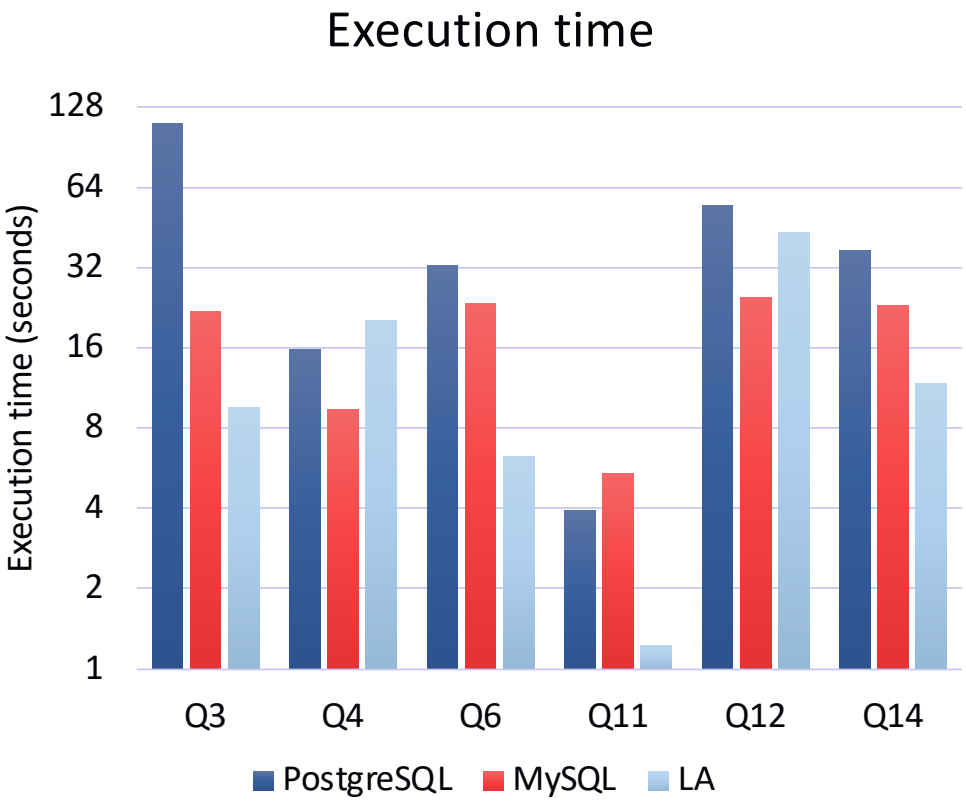


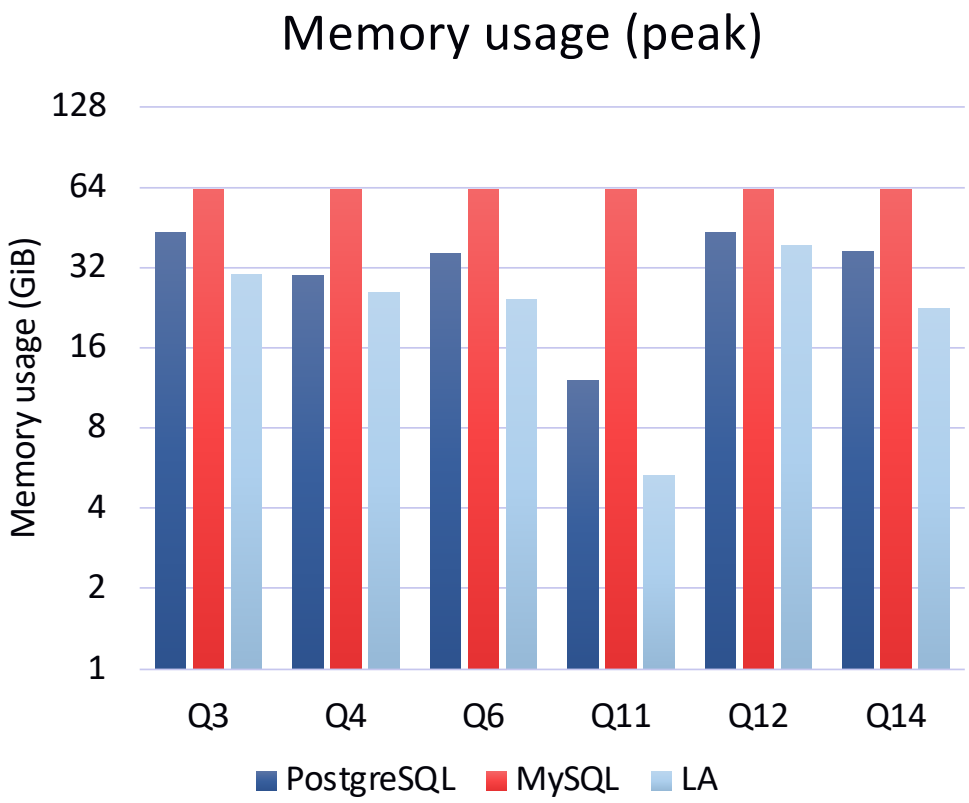
TLA-DB Engine



Benchmarks

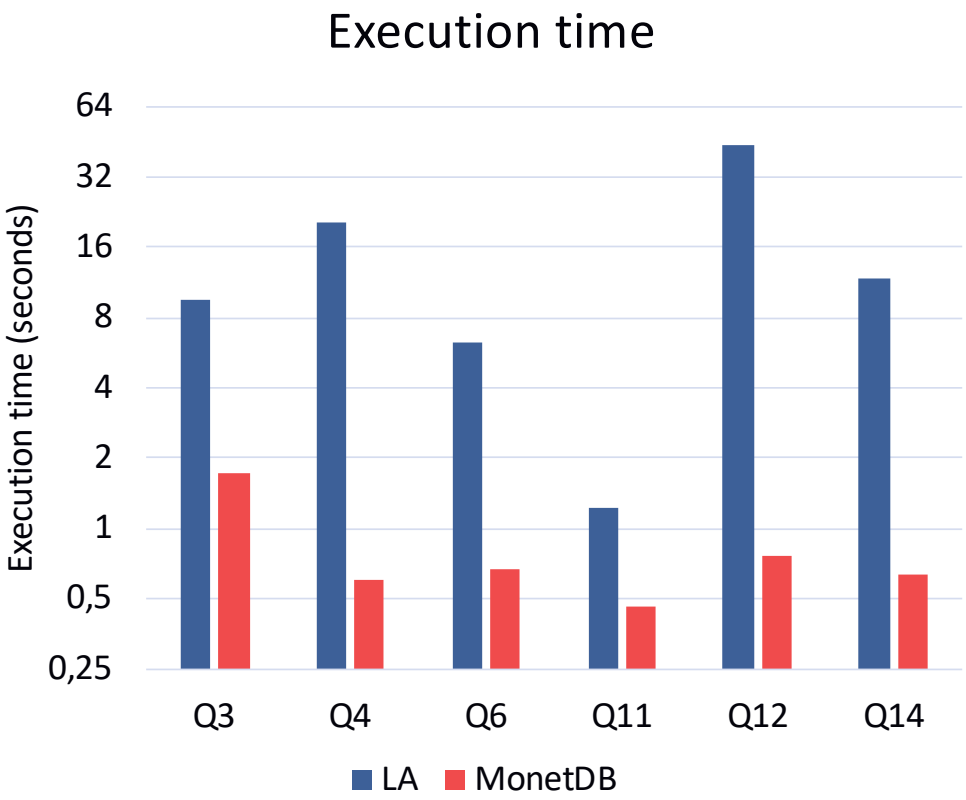
(TPC-H SF 32)

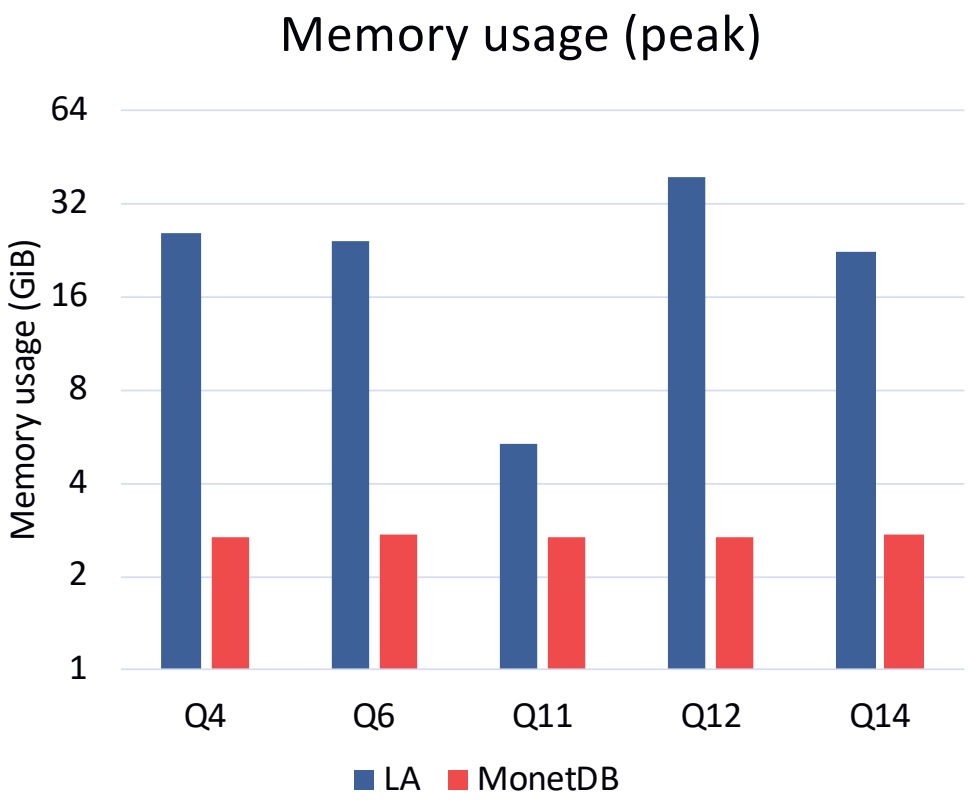




Benchmarks

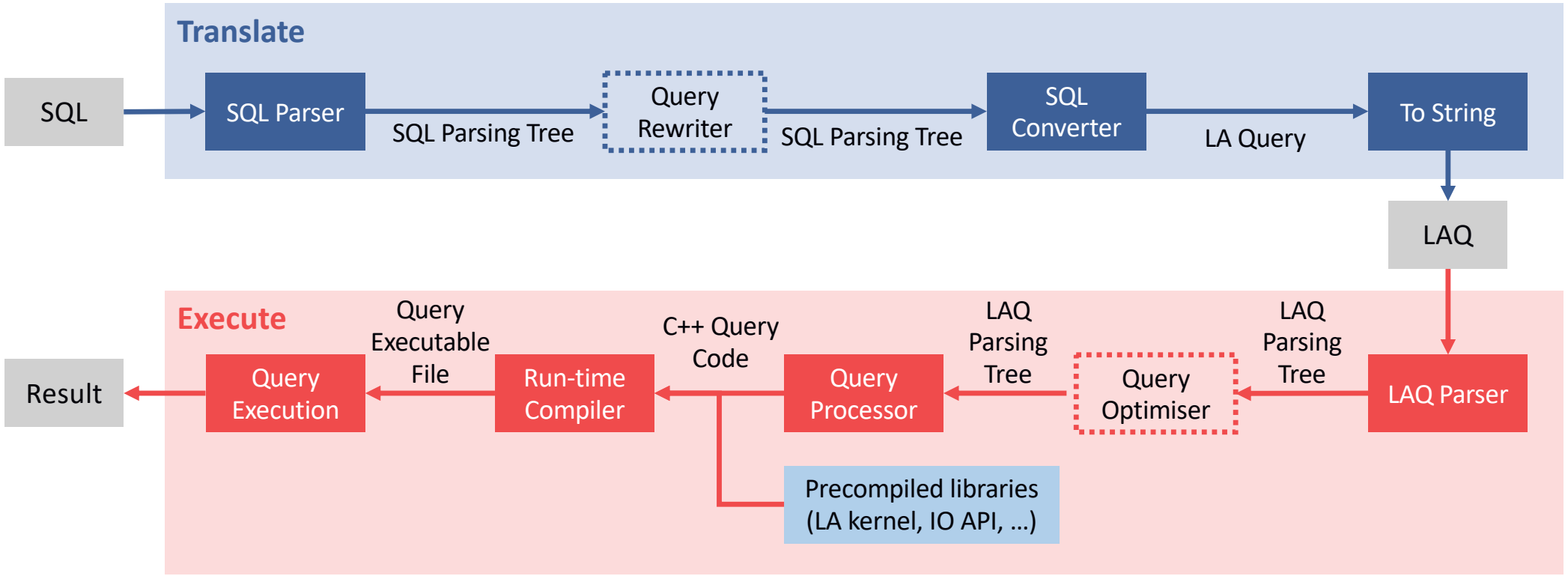
(TPC-H SF 32)





Conclusions

- Difficulties:
  - SQL and LAQ follow distinct paradigms
  - Extension of the project
- The proposed framework:
  - Simple and modular architecture
  - Outperforms PostgreSQL and MySQL in most queries
- Further study required in columnar databases


Translate
SQL
SQL Parser
SQL Parsing Tree
Query Rewriter
SQL Parsing Tree
SQL Converter
LA Query
To String
LAQ
Execute
Result
Query Execution
Query Executable File
Run-time Compiler
C++ Query Code
Query Processor
LAQ Parsing Tree
Query Optimiser
LAQ Parsing Tree
LAQ Parser
Precompiled libraries (LA kernel, IO API, ...)

Questions?