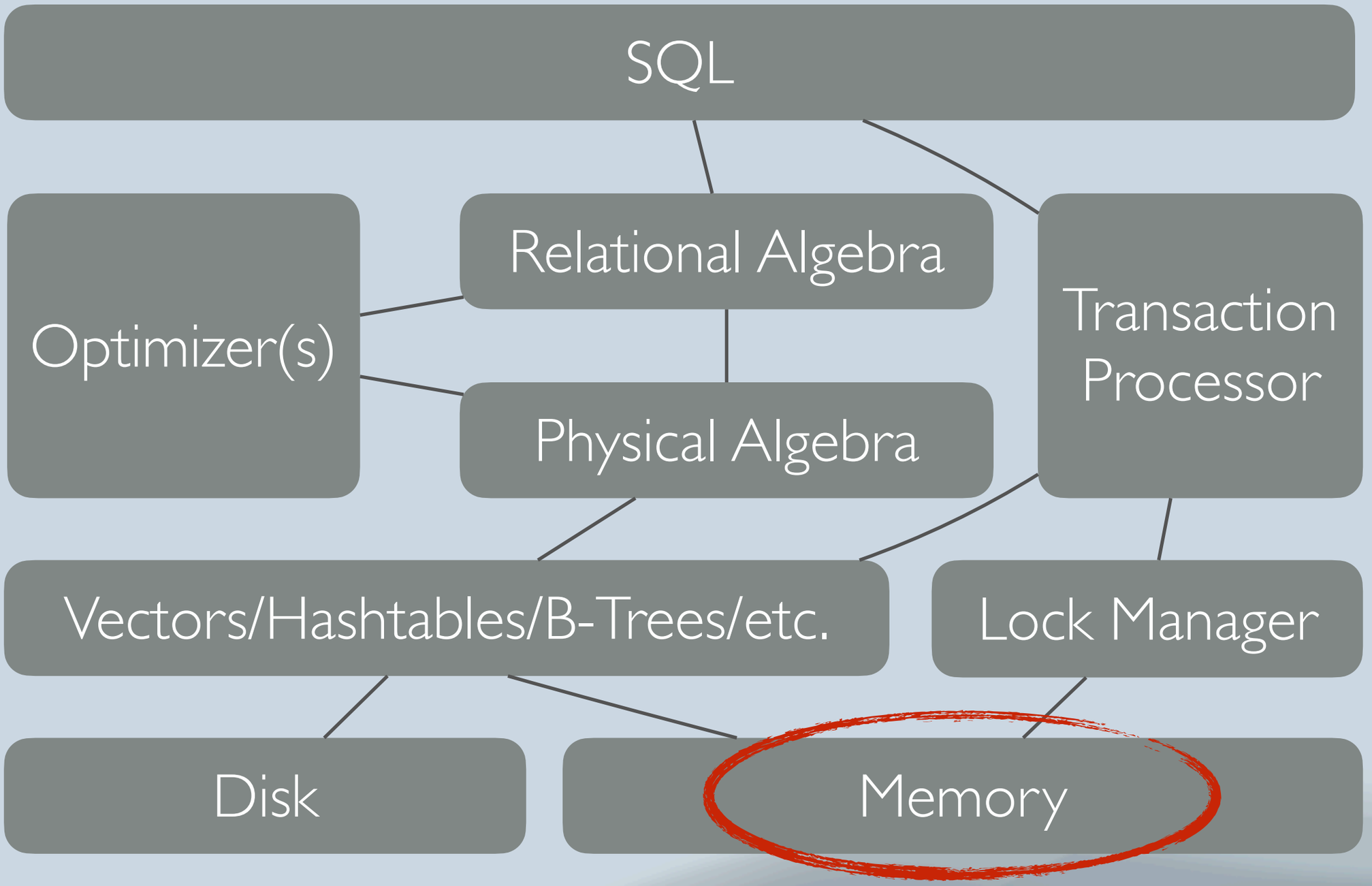
LANGUAGE-ORIENTED DATA PROCESSING SYSTEMS

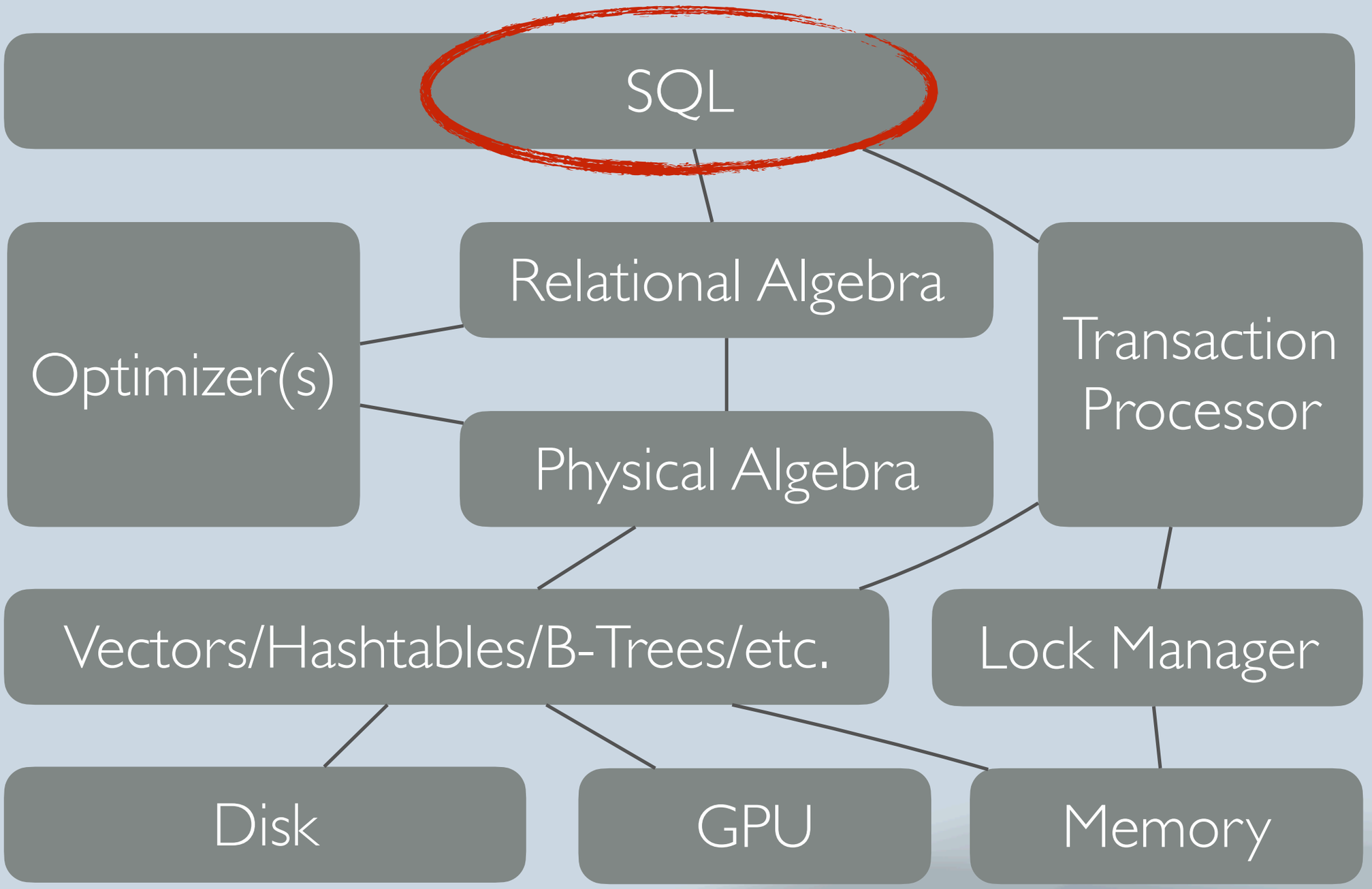
Holger Pirk <hlgr@imperial.ac.uk>

Imperial College London

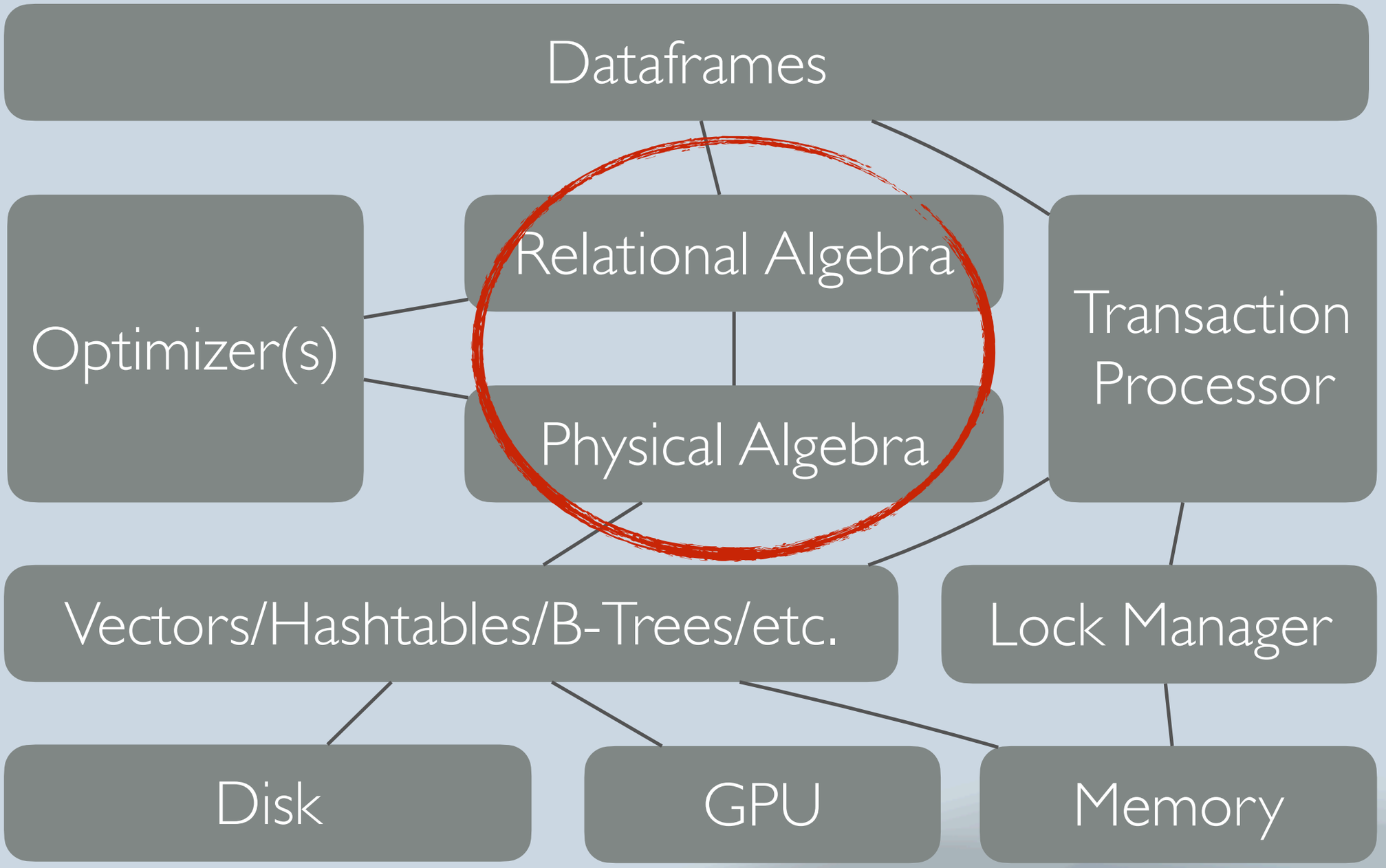
Data Processing Systems



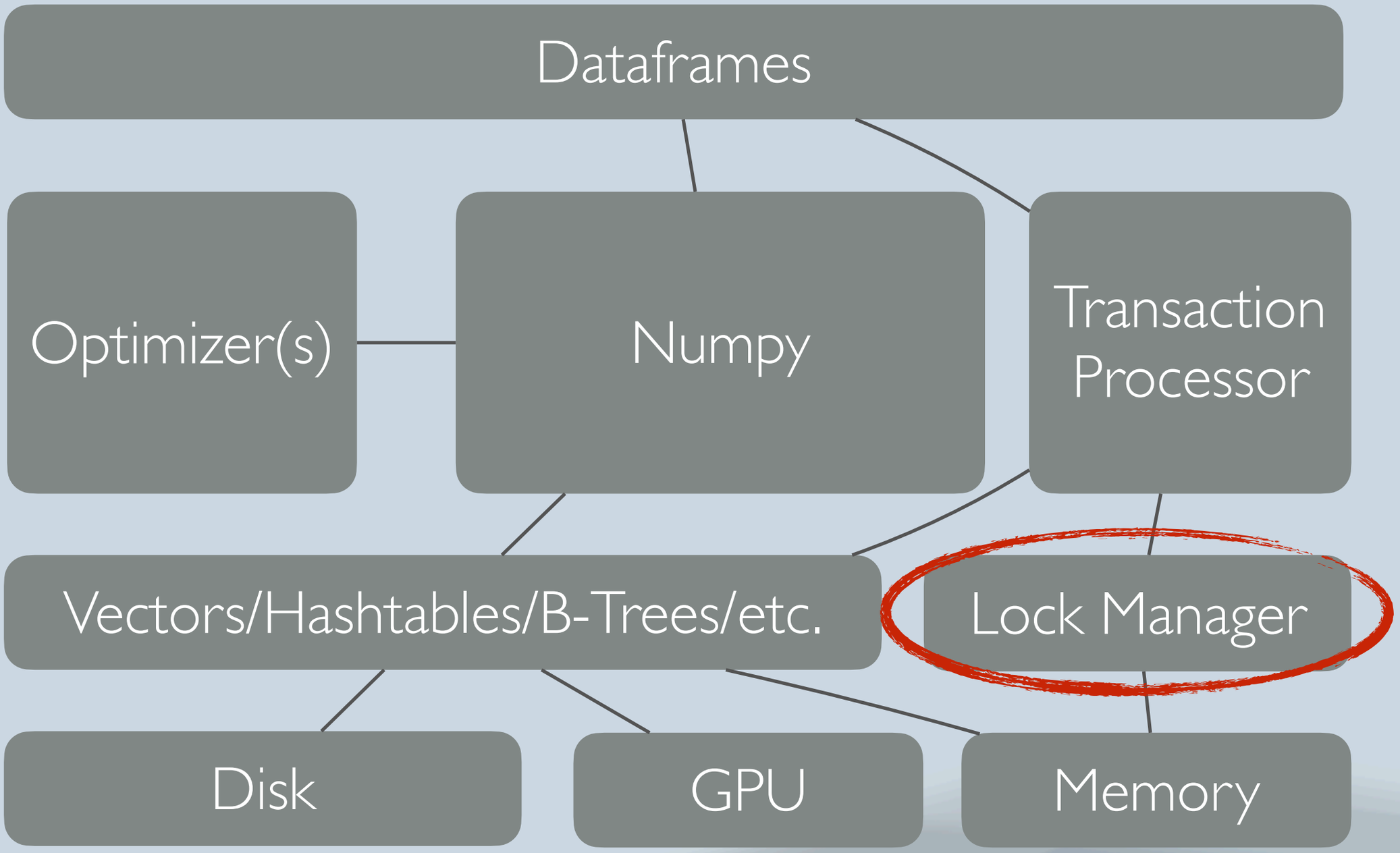
Composable Data Processing Systems



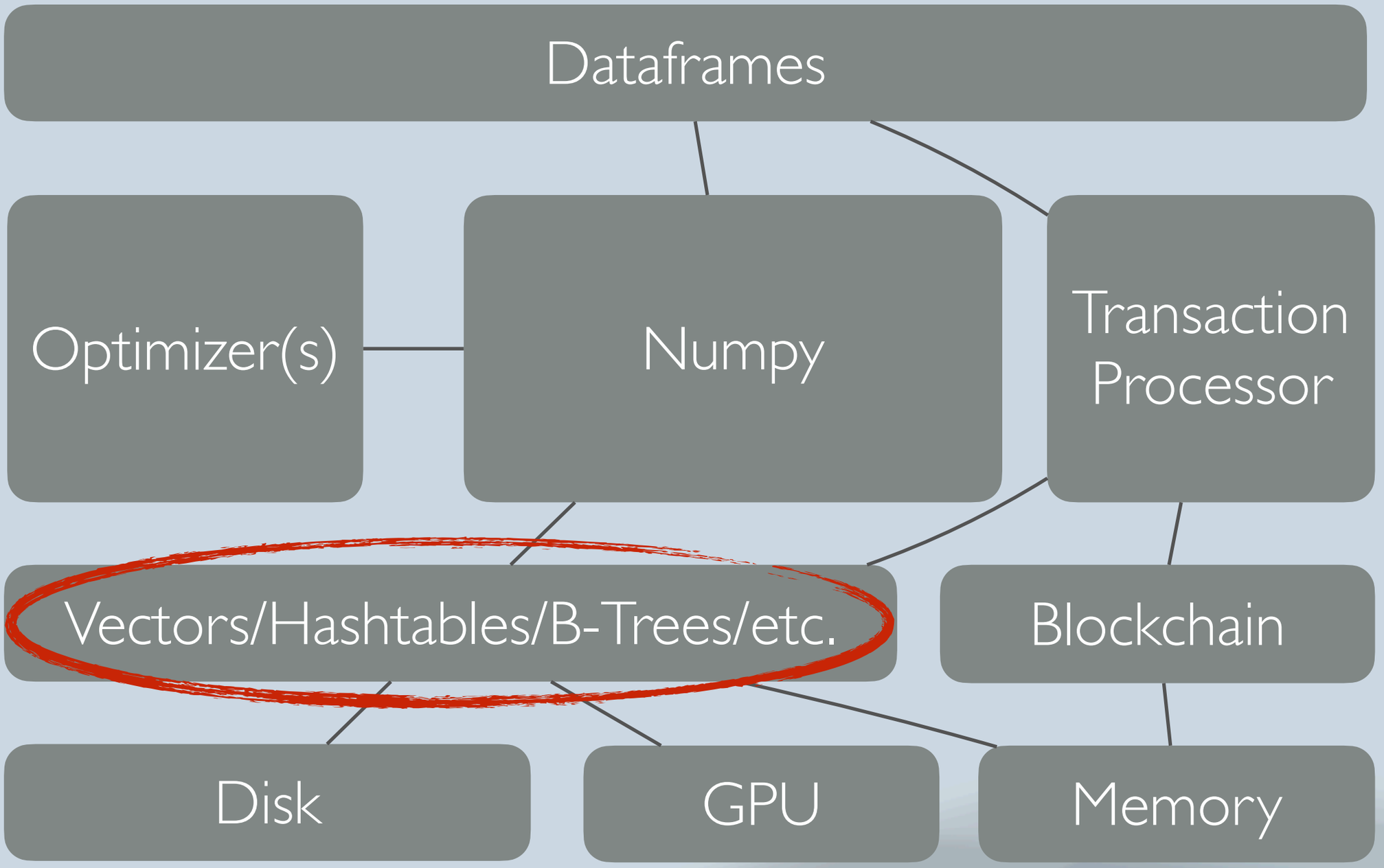
Composable Data Processing Systems



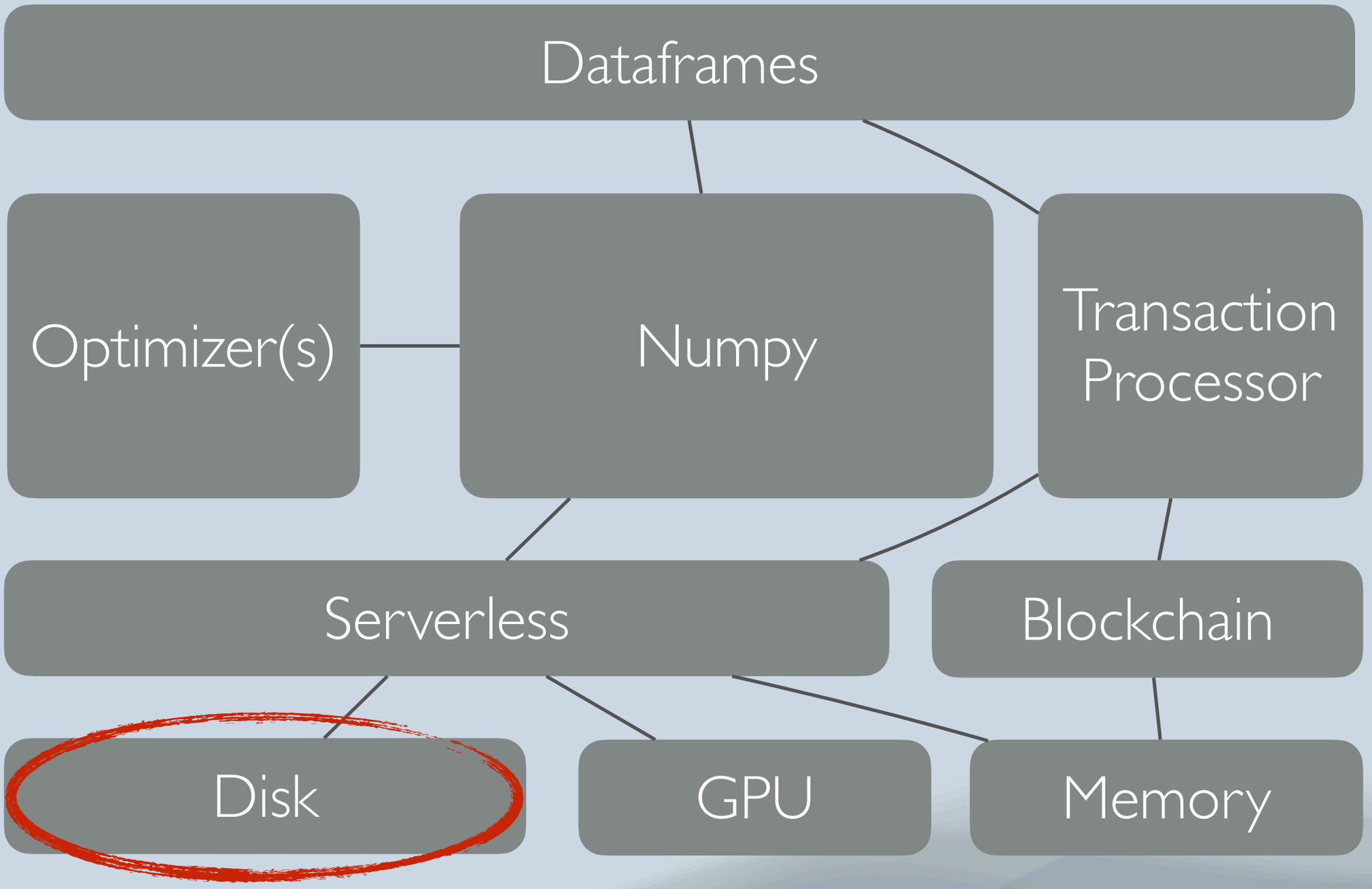
Composable Data Processing Systems



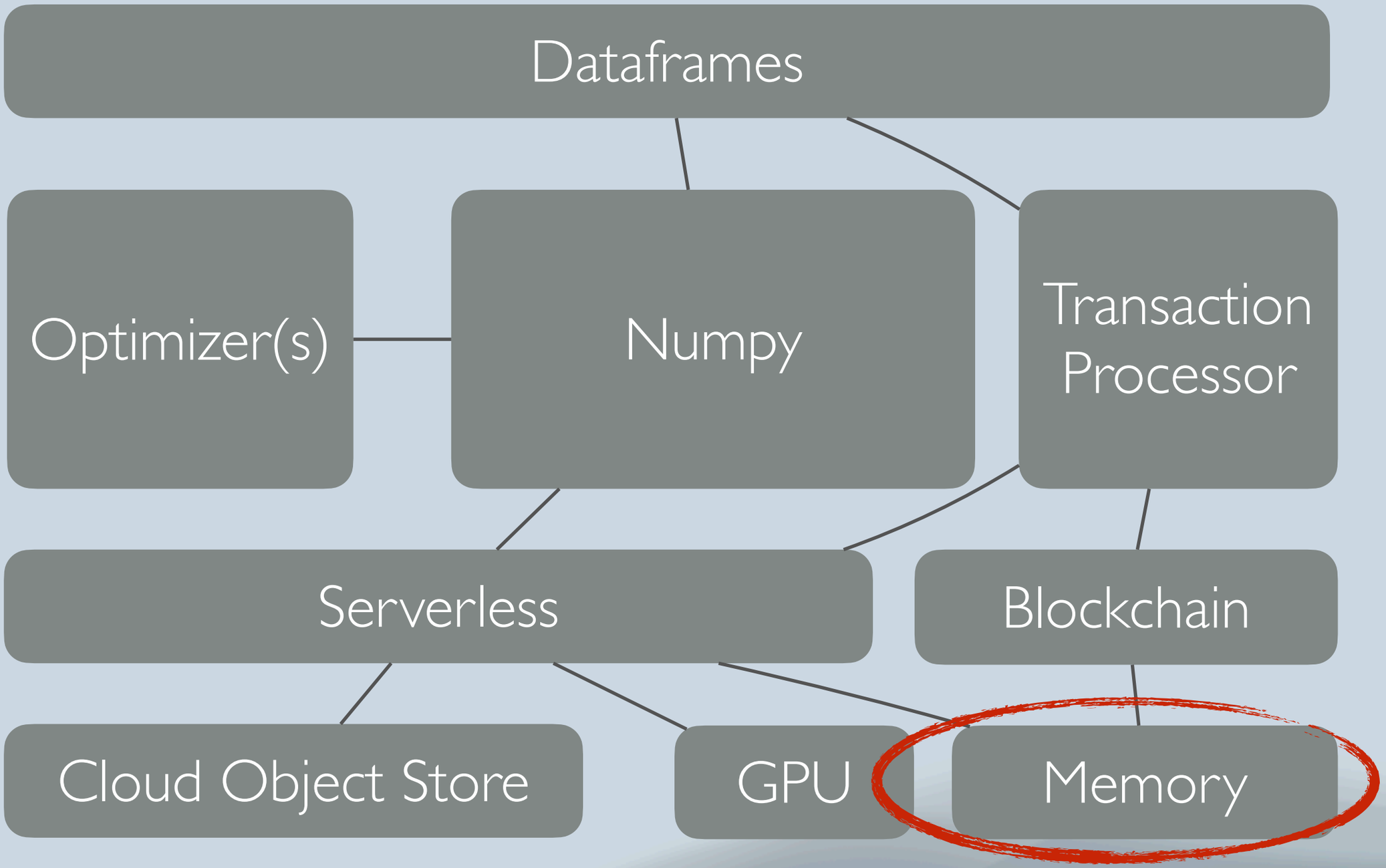
Composable Data Processing Systems



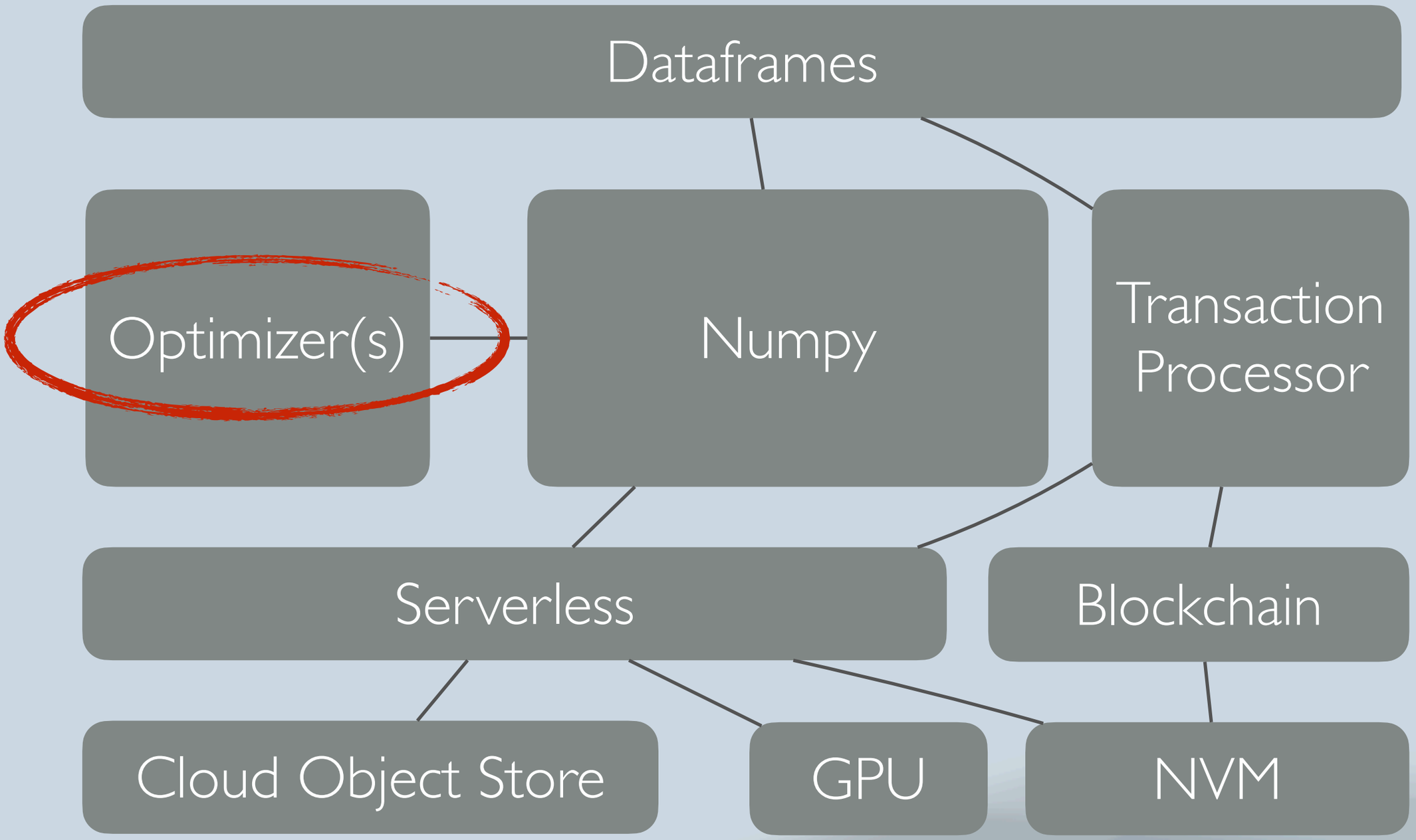
Composable Data Processing Systems



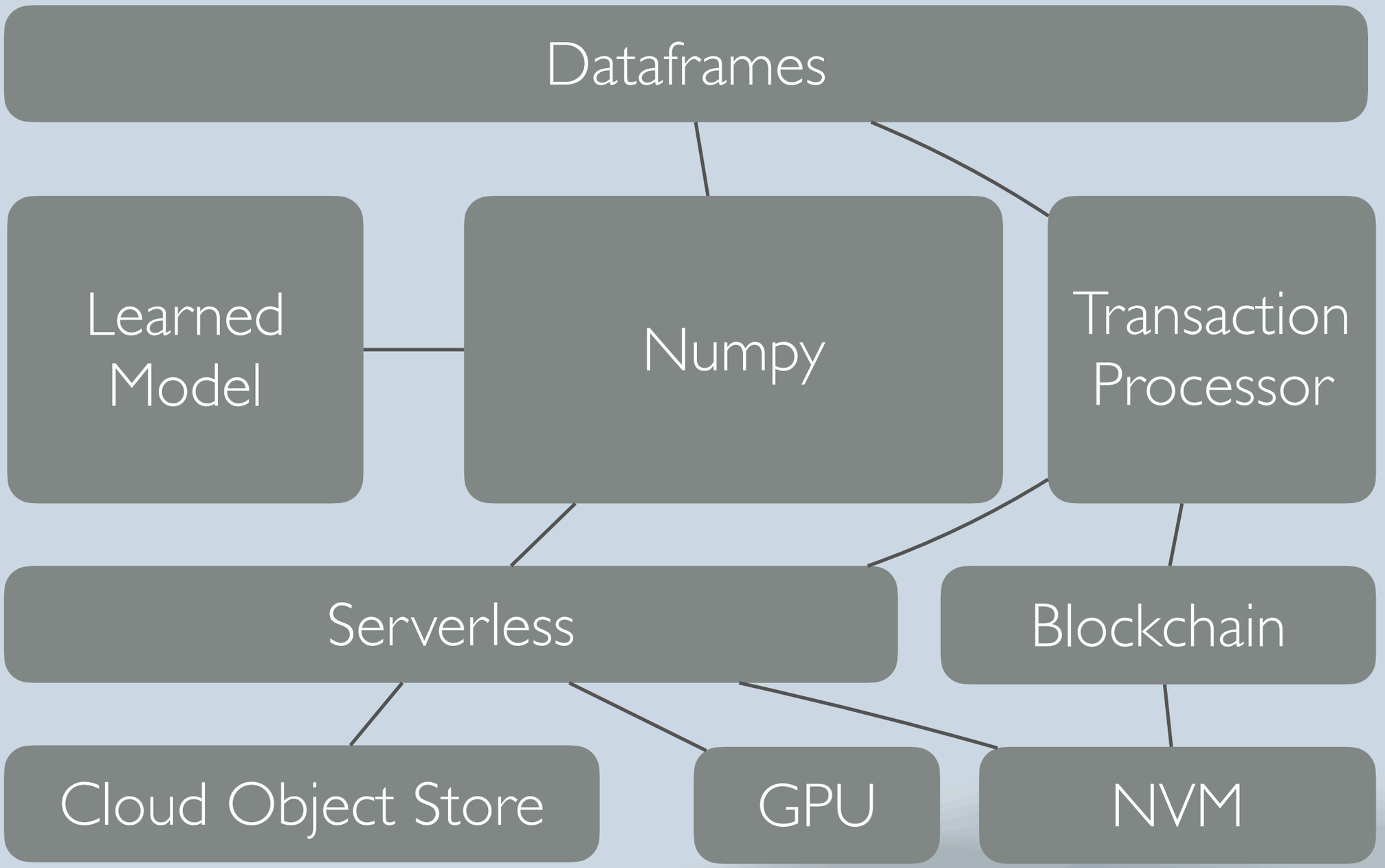
Composable Data Processing Systems



Composable Data Processing Systems



Composable Data Processing Systems



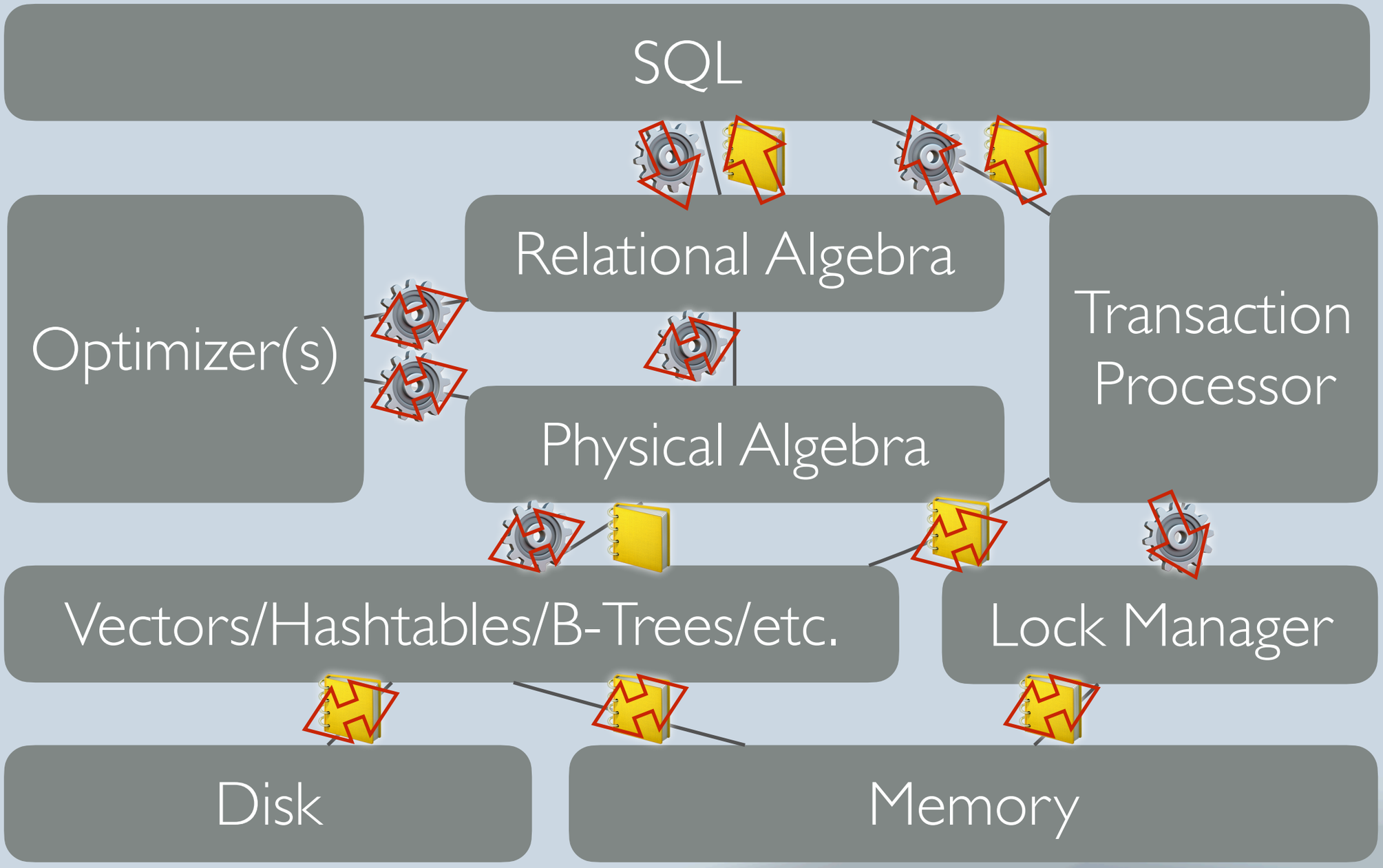
Imperial College London

All Systems Process Two Things

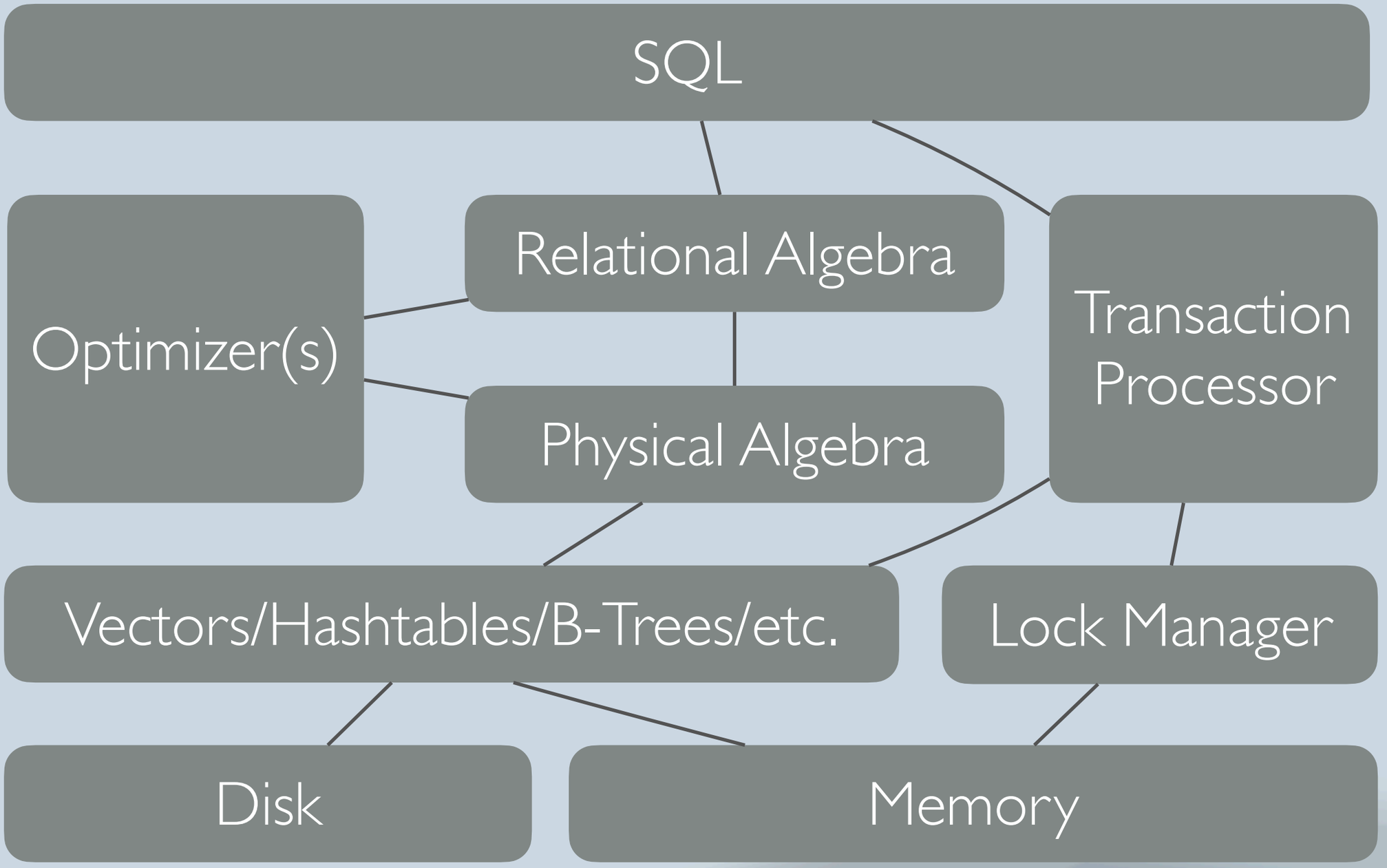
Code/Queries/Plans/UDFs/…

Data

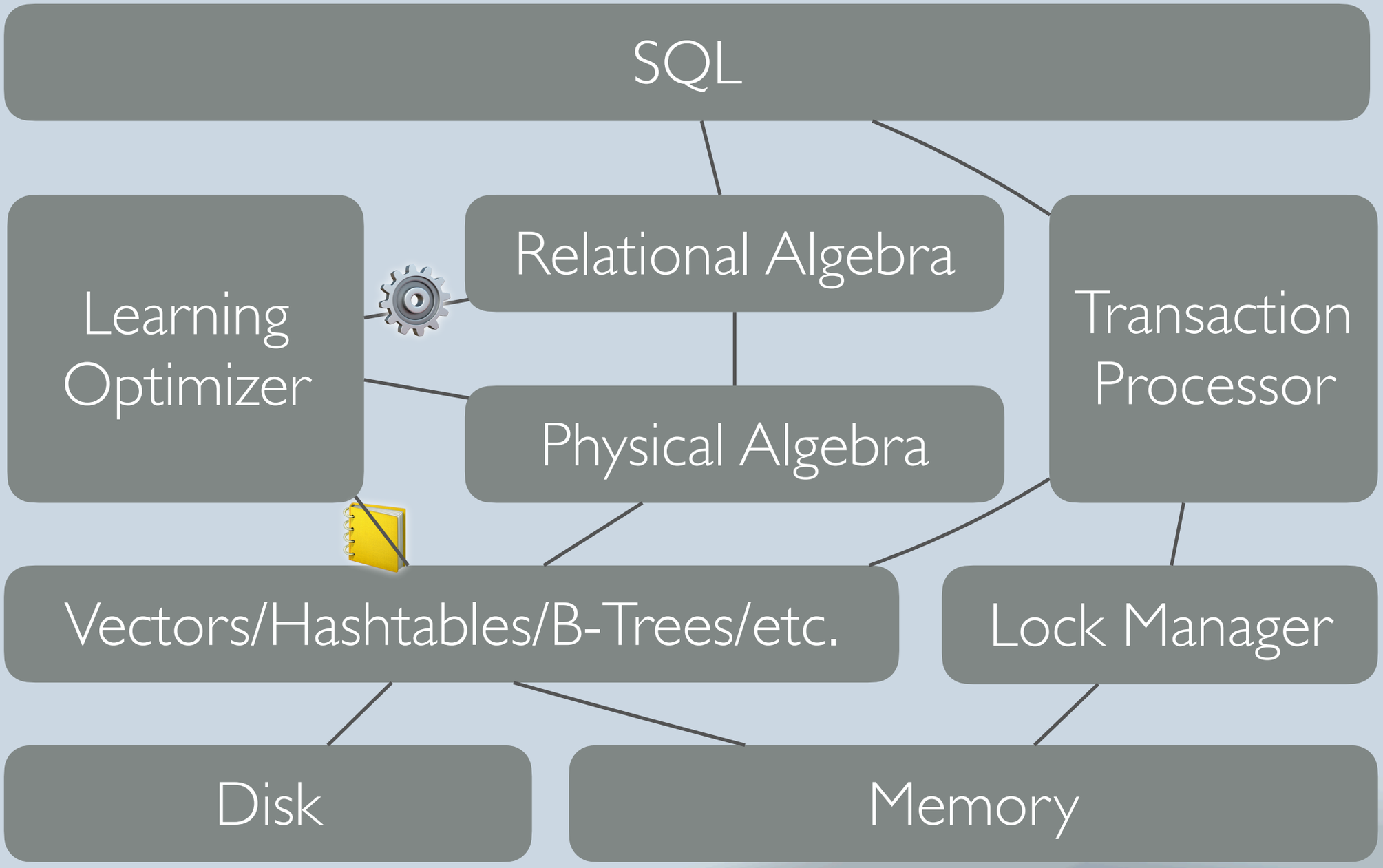
Composable Data Processing Systems



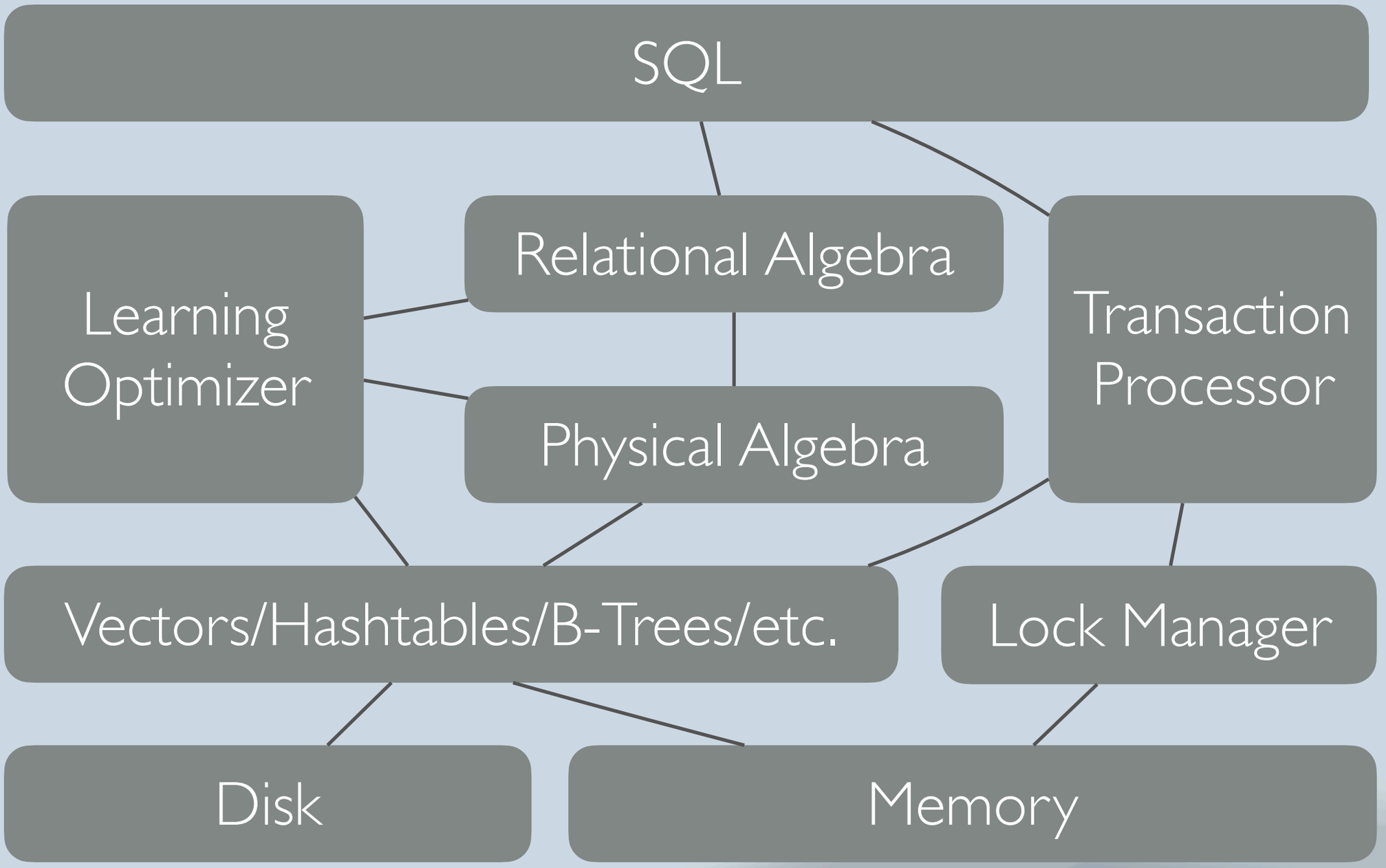
Composable Data Processing Systems



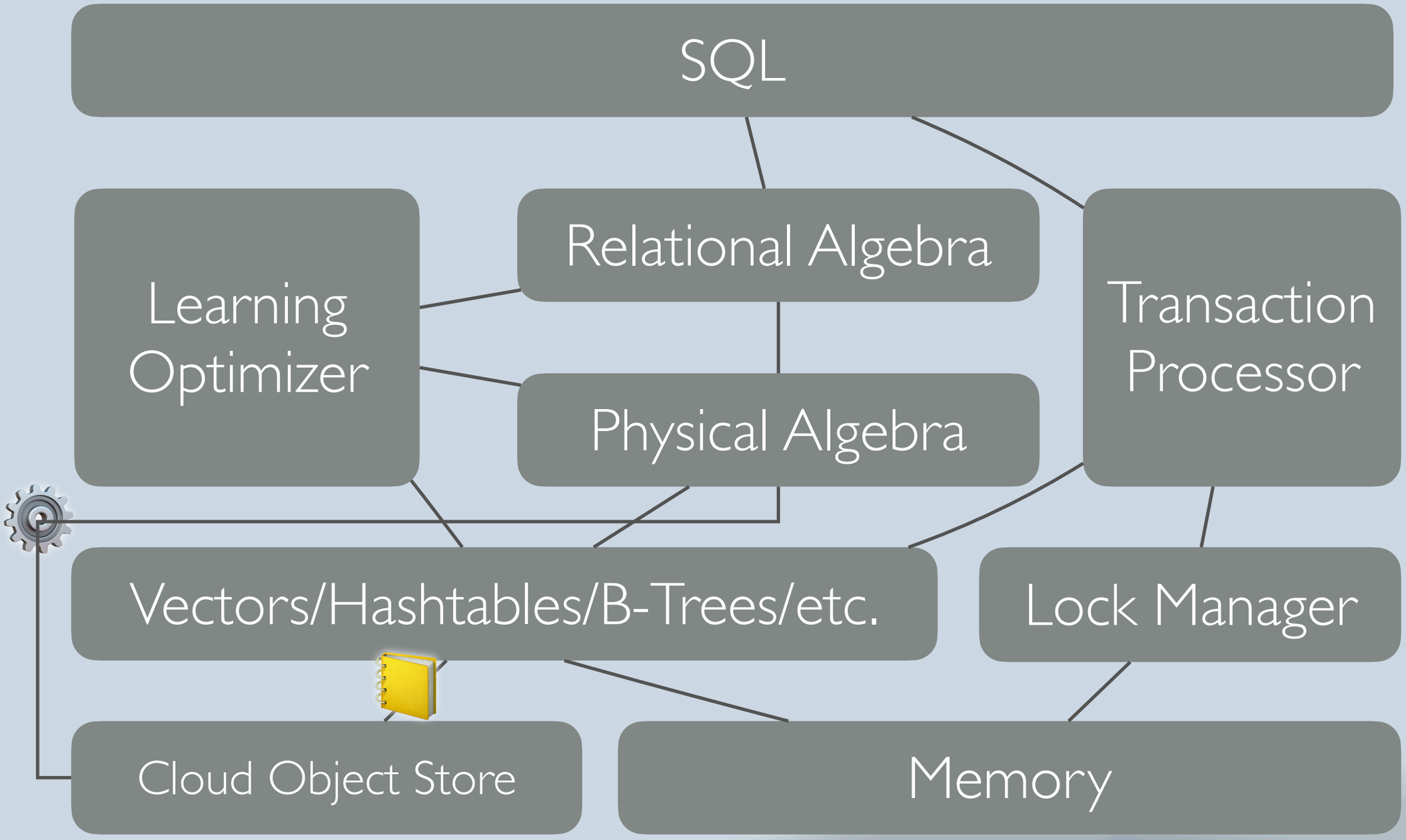
Composable Data Processing Systems



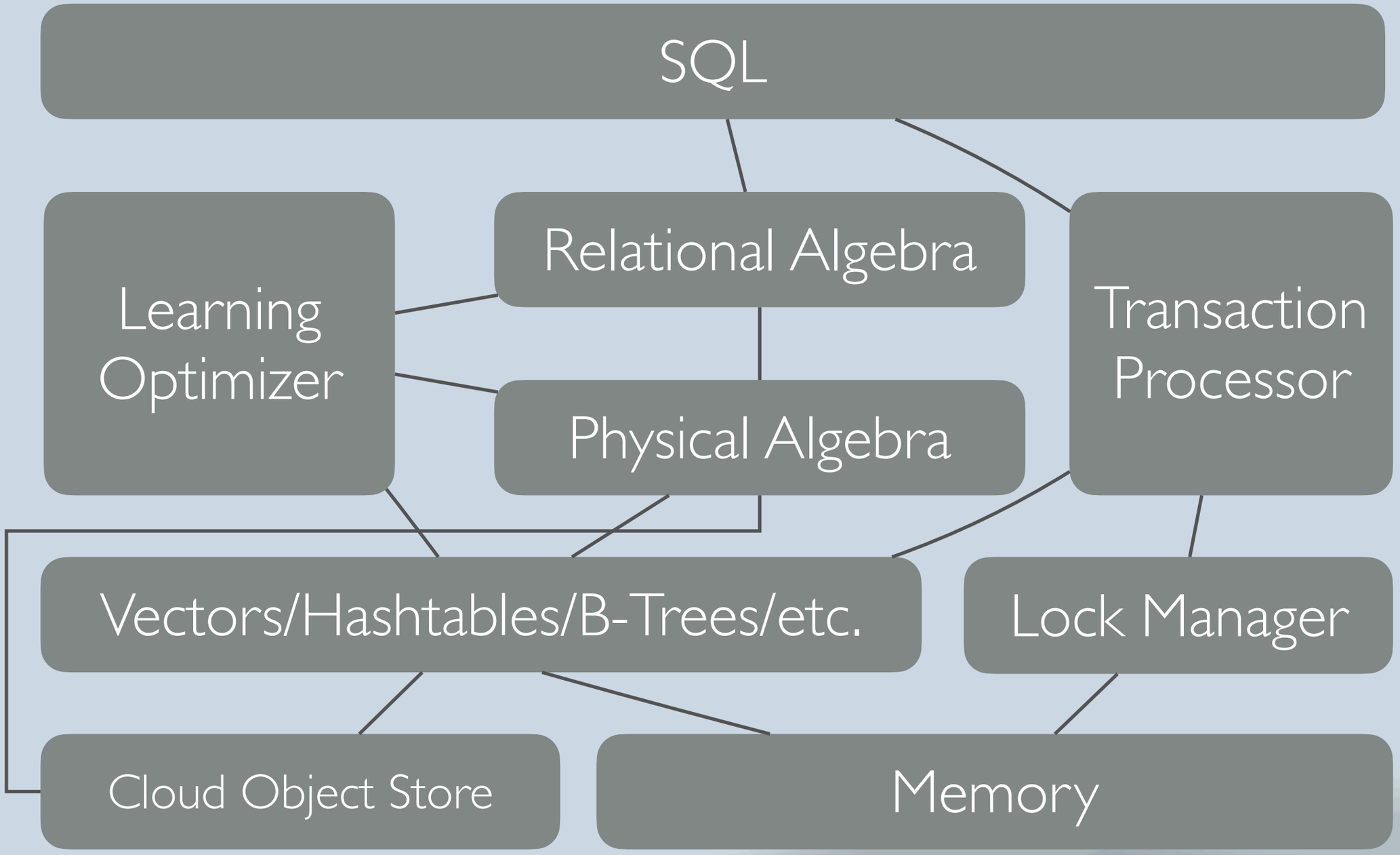
Composable Data Processing Systems



Composable Data Processing Systems

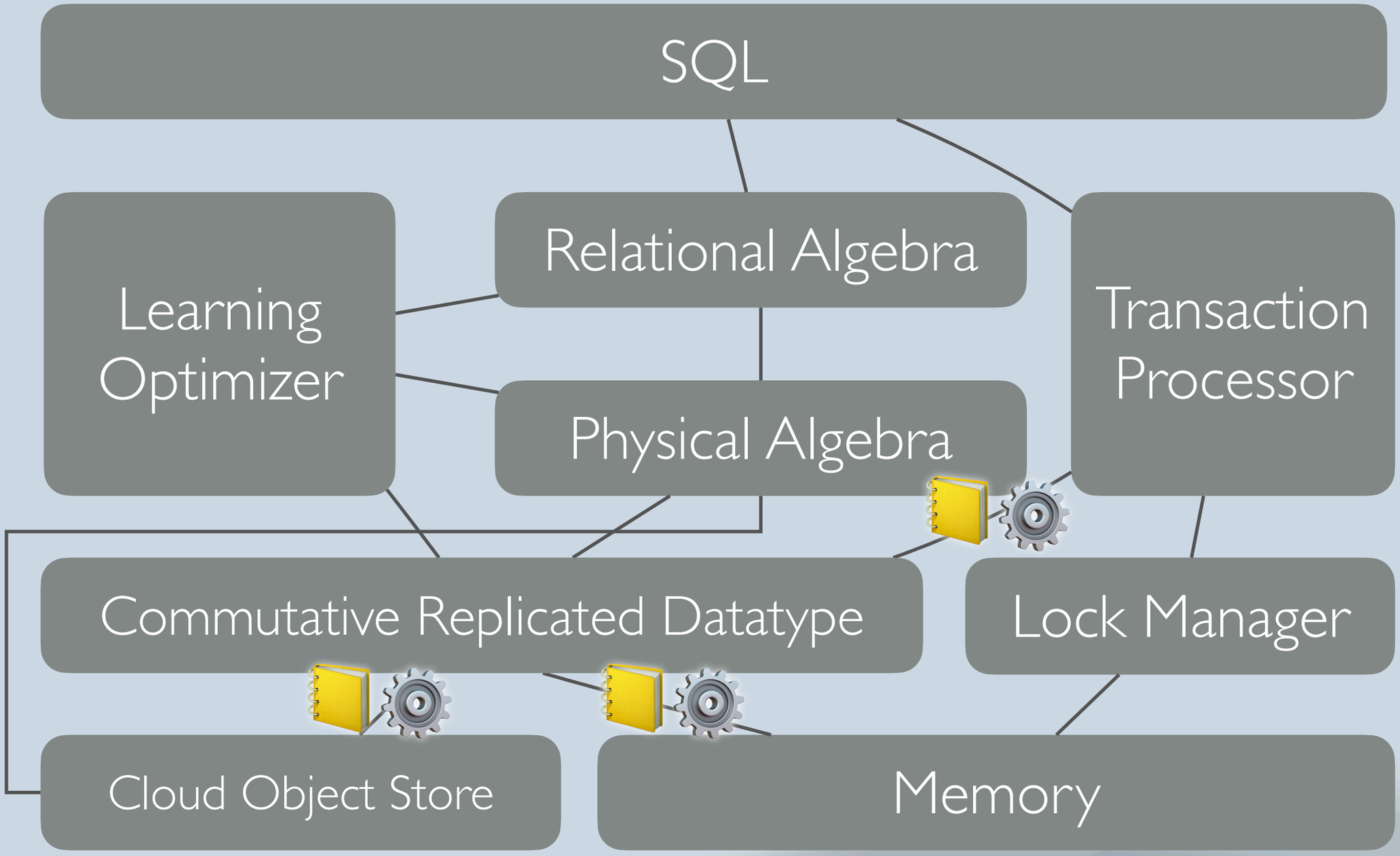


Composable Data Processing Systems

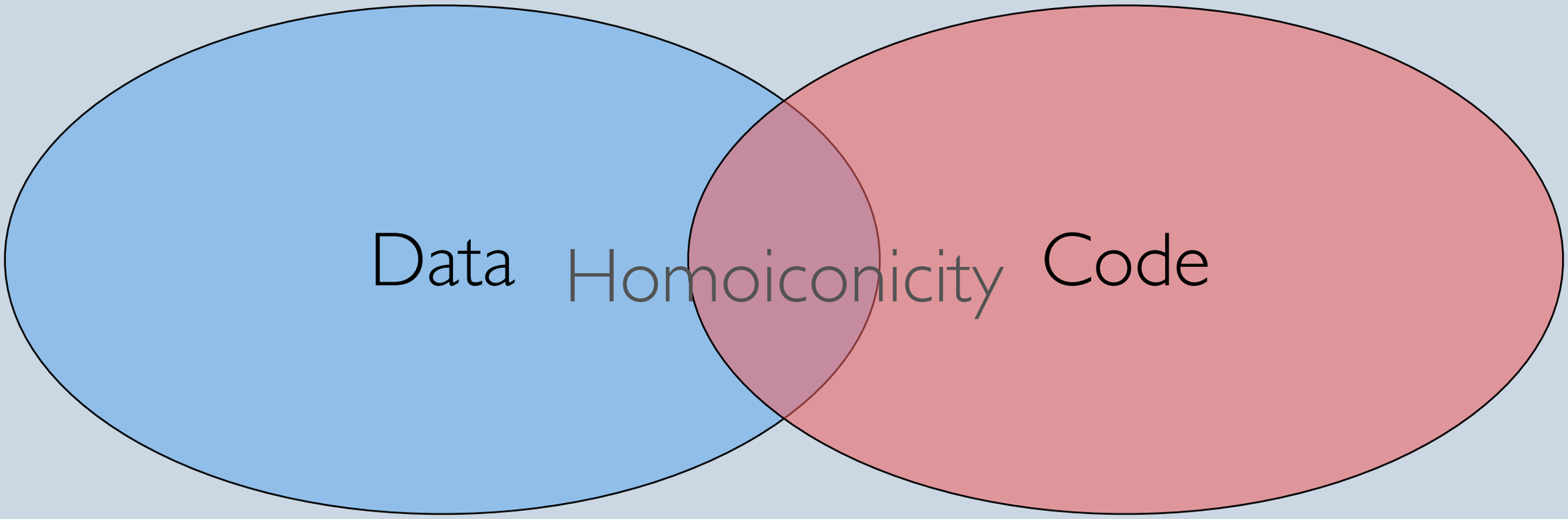


Imperial College London

Composable Data Processing Systems



Option B



Classic Homoiconicity

(list 5 1 9 17 5) **Data**

Code (function square (x) (* x x))

(set input (list 5 1 9 17 5)) **???**

Relational Query (select input (where (> v 5)))

Homoiconic Data Processing

- Flavor 1
  - Data as Code
- Flavor 2
  - Code as Data

Flavor 1: Data as Code

Homoiconic Data Processing

(Create Table Caseload date numberOfCases)

(Insert Into Caseload 17 23)

(Insert Into Caseload 18 23)

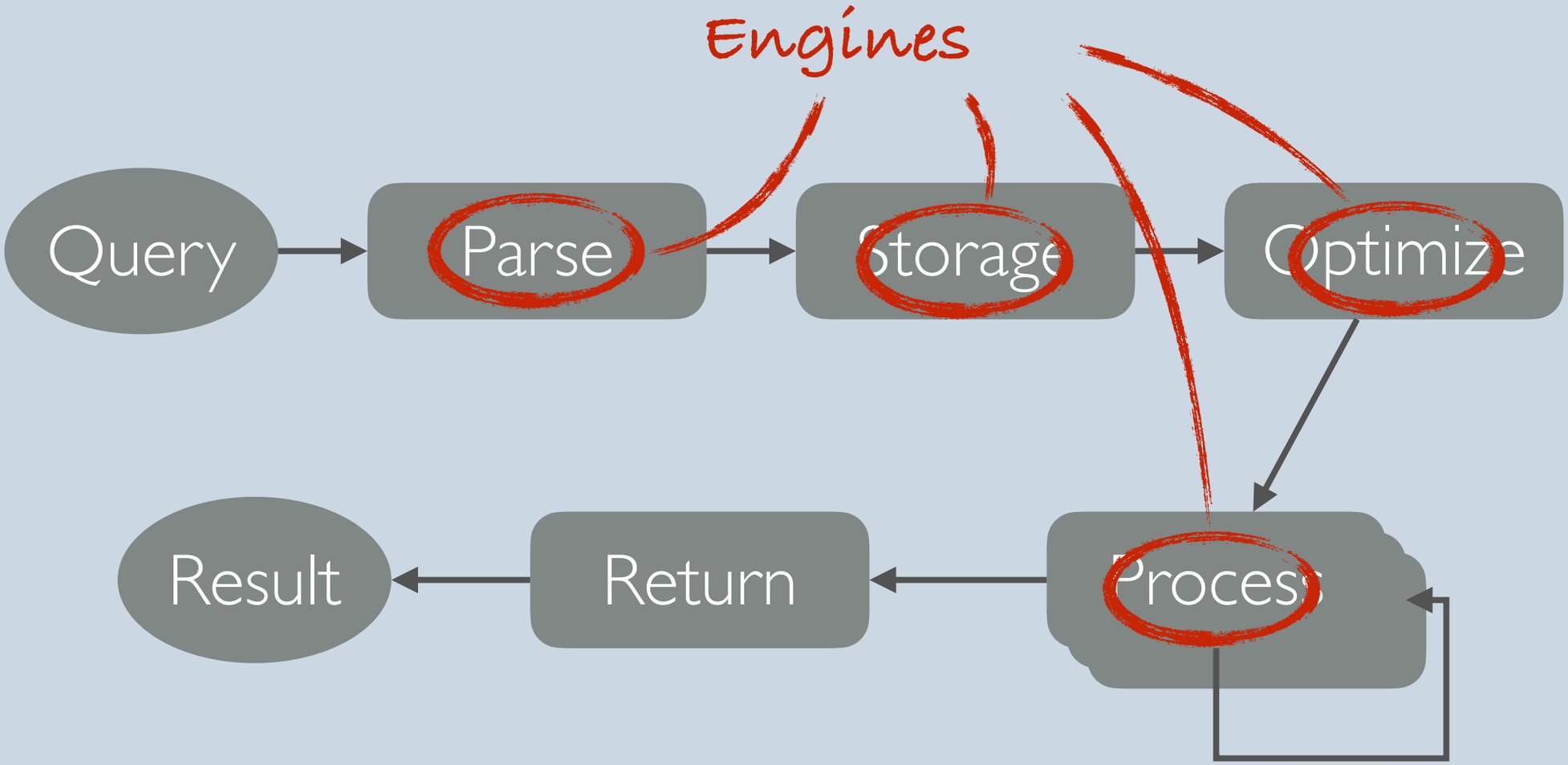
(Insert Into Caseload 19 23)

(Select From Caseload (Where (> numberOfCases 5)))

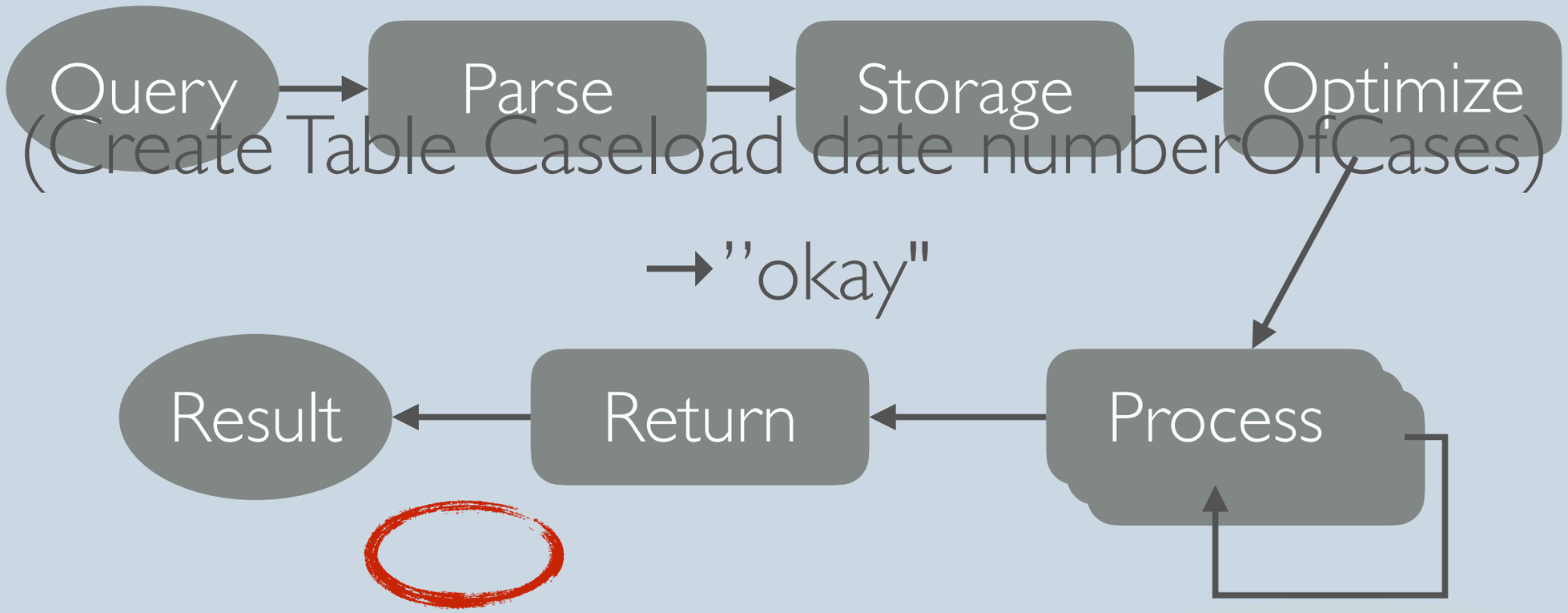
Partial Query Evaluation

Imperial College London

Partial Query Evaluation



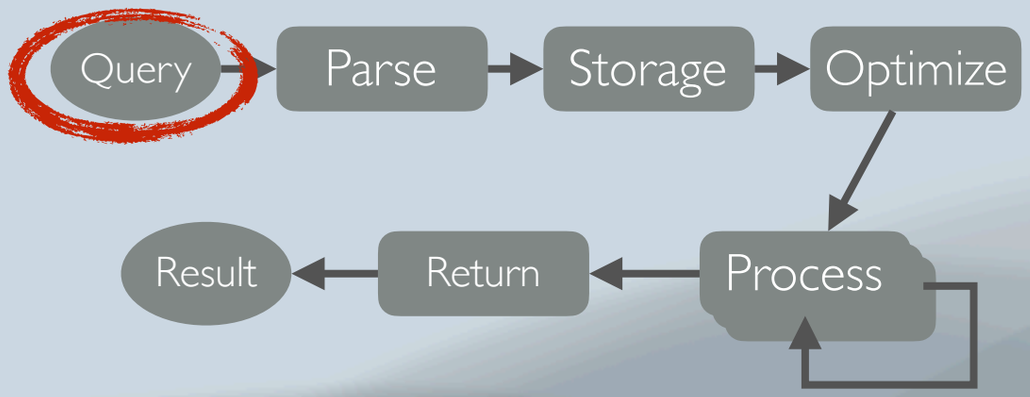
Partial Query Evaluation



Partial Query Evaluation

(Insert Into Caseload 17 23)

→"okay"

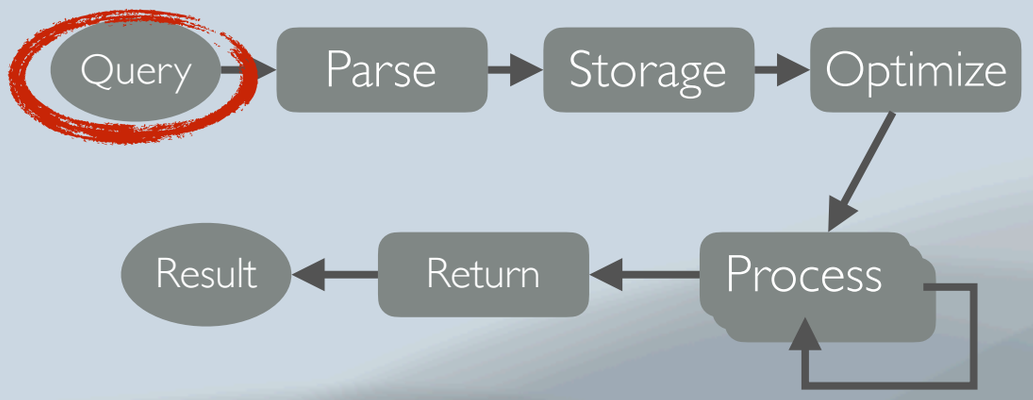


Partial Query Evaluation

(Select From Caseload (Where (> numberOfCases 5)))

→

(Select From
(Table
(data (List 18 19 17))
(numberOfCases (List 5 74 23))))
(Where (> numberOfCases 5)))

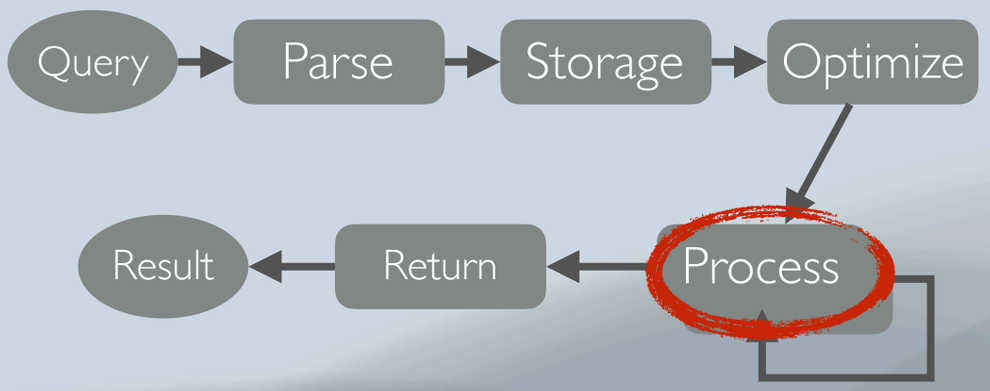


Partial Query Evaluation

```
(Select From
  (Table
      (data (List 18 19 17))
      (numberOfCases (List 5 74 23)))
  (Where (> numberOfCases 5)))
```

→

```
(Table
      (data (List 19 17))
      (numberOfCases (List 74 23)))
```



Flavor 2: Code as Data

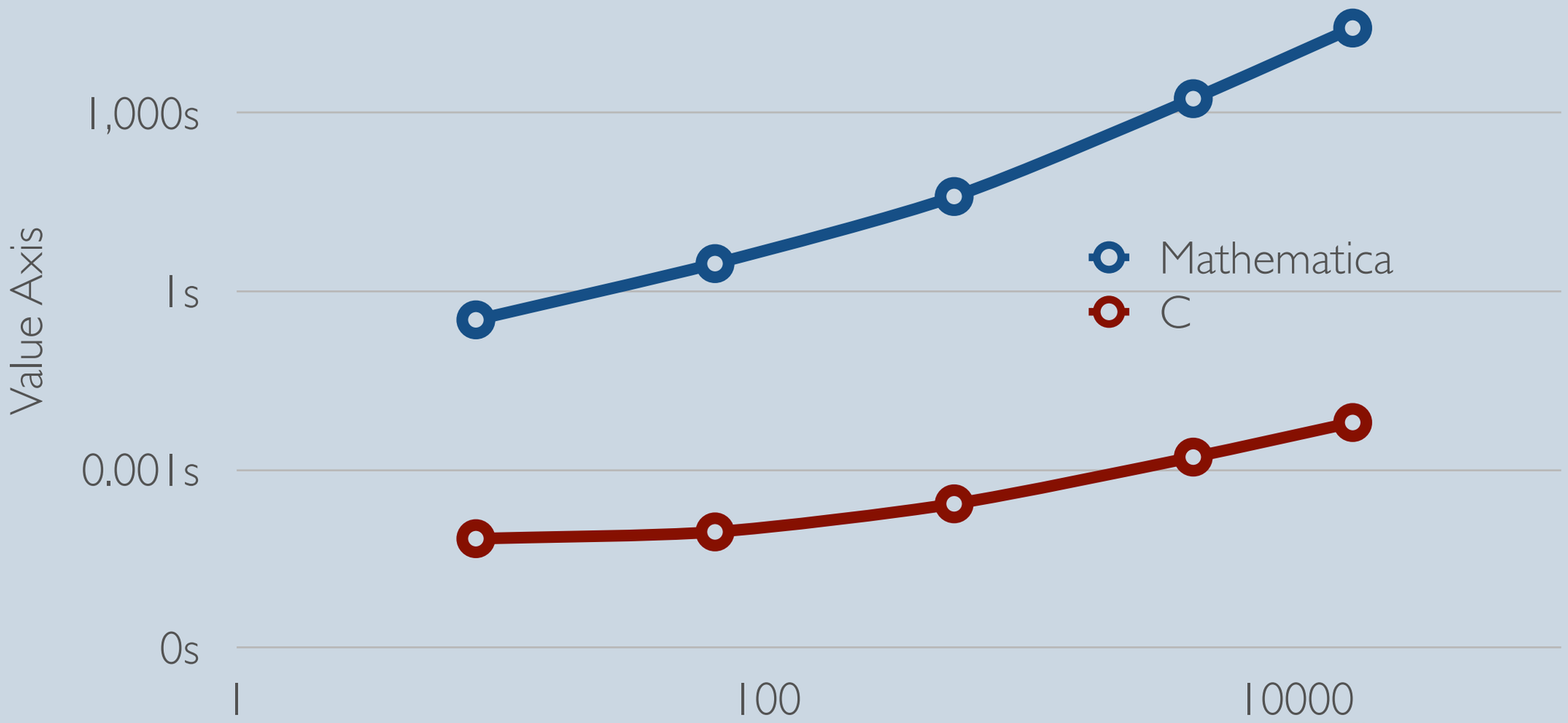
Code in a Database

Id	Name	Description	Price
4	“Cherry"	*'NULL*	0.09
6	“Apple"	“Yummy"	*(Impute)*
(UniqueValue)	“Pear"	"Sweet"	0.69
(UniqueValue)	“Raspberry"	*(fetch “http://…)*	0.04

You're probably thinking

"This all sounds really expensive"

The Problem Homoiconicity: Performance



Select sum(a) from R

Imperial College London

So how do we make it fast?

BOSS - A Bulk-Oriented Symbol Store

Imperial College London

BOSS Engine Interface

```
class Engine {
public:
  Expression evaluate(Expression&&);
};
```

Semi-static Typing

Imperial College London

Physical BOSS Type-System

For Convenient C++-Integration

For Speed & Data Exchange

For flexibility

```
class Symbol : public string {}

template <typename ElementType> struct Span {
  ElementType* element;
  size_t size;
};

template <typename... StaticArguments> class Expression {
  Symbol head;
  tuple<StaticArguments...> staticArguments;
  vector<variant<long, double, string, Expression>> dynamicArguments;
  vector<variant<Span<int>, Span<double>, Span<string>>> spans;
  auto getArgument(int);
};
```

Unified API

Imperial College London

Language Integration Through Semi-Static Typing

Concepts restrict Types

Type Inference during instantiation

Dynamically typed expression

```
operators["Greater"_] =
    [] NumericType FirstArgument>(
        ComplexExpression<FirstArgument>&& input) -> Expression {
  return input.getArgument(0) > (FirstArgument) input.getArgument(1);
};
```

C++ Lambas define operators

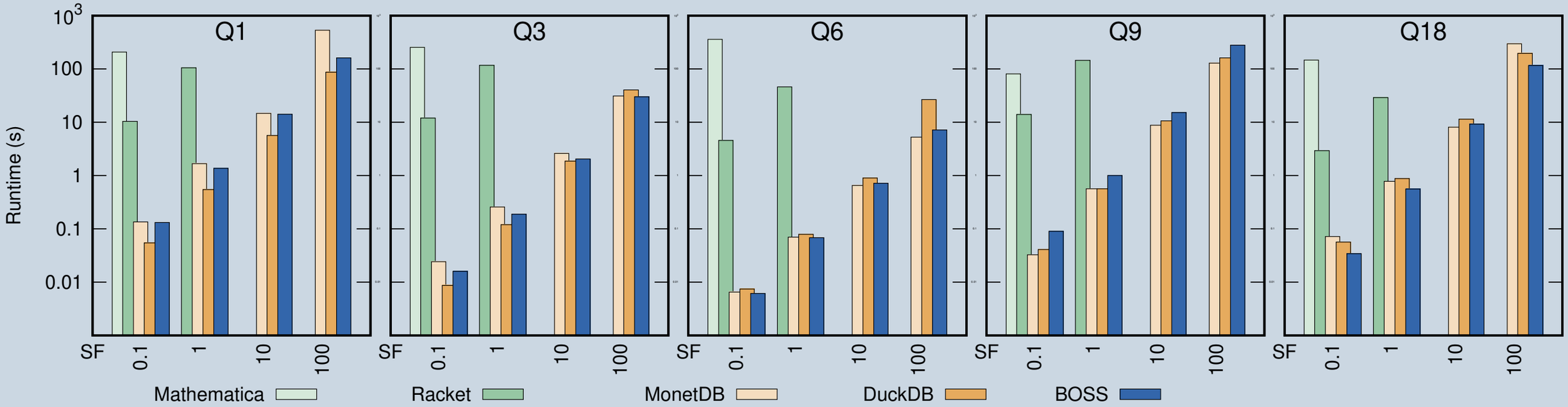
Statically typed

Cast to Statically typed expression

Zero-Copy Data Exchange Through Spans

- Spans are basically C-arrays (with some sugar-coating)
- Virtually all in-memory DBMSs use C-arrays
- Apache Arrow, C++ Vectors, Velox Vectors, ArrayFire Tensors provide zero-copy construction & extraction of C-arrays

TPC-H Performance Is Competitive



Ongoing Work

Storage Engine: Apache Arrow	✔
Processing Engine: Bulk	✔
Processing Engine: Mathematica	✔
Processing Engine: Velox	…
Processing Engine: ArrayFire	…
Adaptive Indexing Engine	
CRDT Engine	…
Data Imputation Engine	✔

These require code in data

Shape-wise Decomposition

Imperial College London

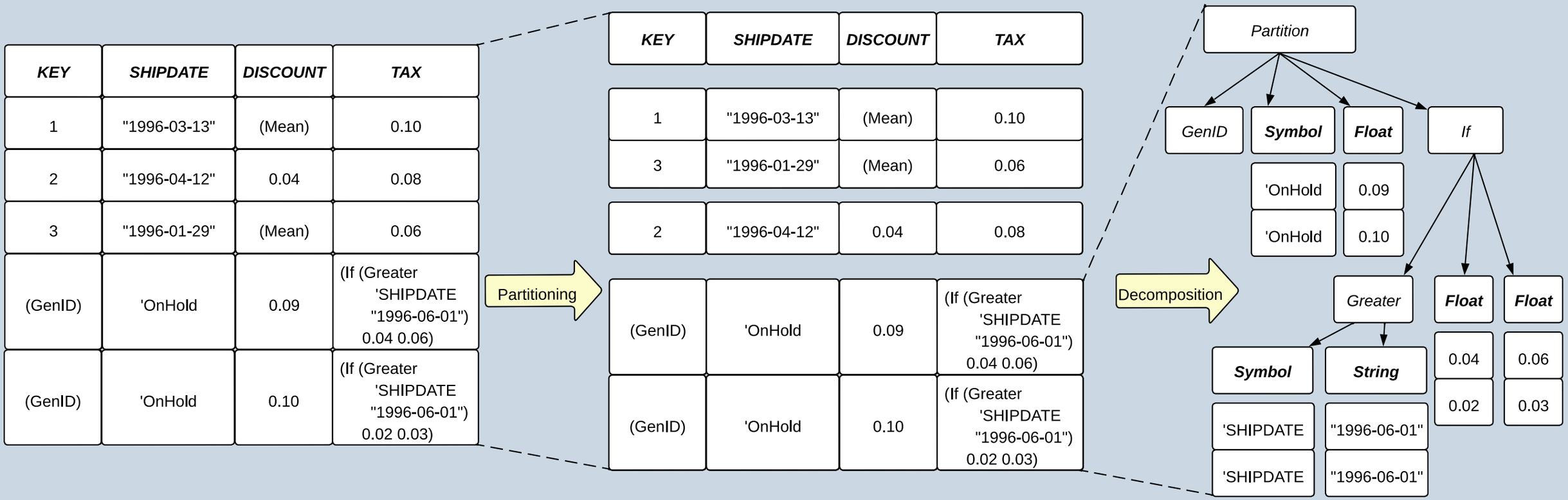
Why Code in Data Is Slow

Sum

KEY	SHIPDATE	DISCOUNT	TAX
1	"1996-03-13"	(Mean)	0.10
2	"1996-04-12"	0.04	0.08
3	"1996-01-29"	(Mean)	0.06
(GenID)	'OnHold	0.09	(If (Greater 'SHIPDATE "1996-06-01") 0.04 0.06)
(GenID)	'OnHold	0.10	(If (Greater 'SHIPDATE "1996-06-01") 0.02 0.03)

Virtual Function Call

Efficiently Representing Homoiconic Databases



Shape-wise Partitioning & Decomposition

Imperial College London

Efficiently Representing Homoiconic Databases

KEY	*SHIPDATE*	*DISCOUNT*	*TAX*
1	"1996-03-13"	(Mean)	0.10
2	"1996-04-12"	0.04	0.08
3	"1996-01-29"	(Mean)	0.06
(GenID)	'OnHold	0.09	(If (Greater 'SHIPDATE "1996-06-01") 0.04 0.06)
(GenID)	'OnHold	0.10	(If (Greater 'SHIPDATE "1996-06-01") 0.02 0.03)

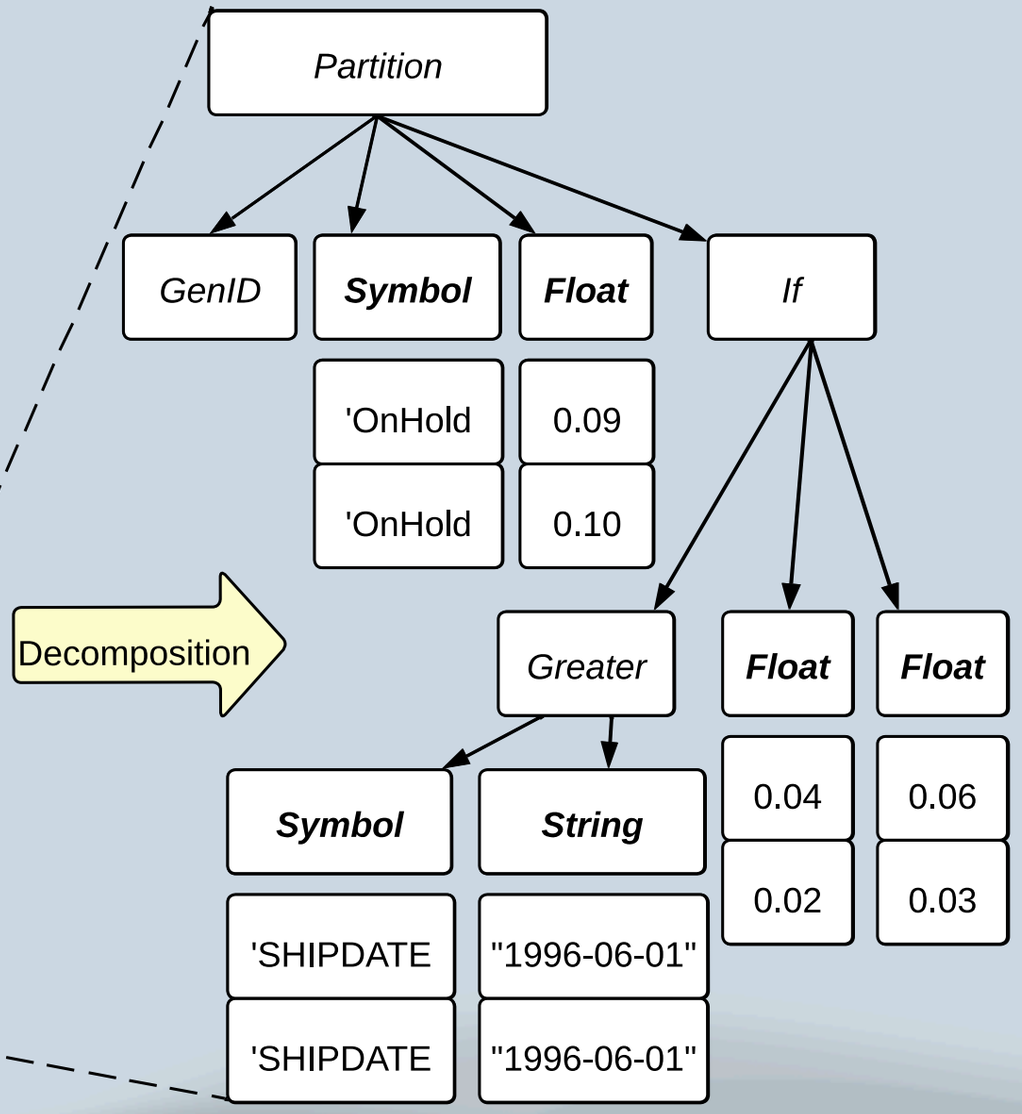
Partitioning

KEY	*SHIPDATE*	*DISCOUNT*	*TAX*
1	"1996-03-13"	(Mean)	0.10
3	"1996-01-29"	(Mean)	0.06
2	"1996-04-12"	0.04	0.08
(GenID)	'OnHold	0.09	(If (Greater 'SHIPDATE "1996-06-01") 0.04 0.06)
(GenID)	'OnHold	0.10	(If (Greater 'SHIPDATE "1996-06-01") 0.02 0.03)

Efficiently Representing Homoiconic Databases

KEY	SHIPDATE	DISCOUNT	TAX
1	"1996-03-13"	(Mean)	0.10
3	"1996-01-29"	(Mean)	0.06
2	"1996-04-12"	0.04	0.08
(GenID)	'OnHold	0.09	(If (Greater 'SHIPDATE "1996-06-01") 0.04 0.06)
(GenID)	'OnHold	0.10	(If (Greater 'SHIPDATE "1996-06-01") 0.02 0.03)

Decomposition



Efficiently Processing Homoiconic Databases

Virtual Function Call

KEY	*SHIPDATE*	*DISCOUNT*	*TAX*
1	"1996-03-13"	(Mean)	0.10
2	"1996-04-12"	0.04	0.08
3	"1996-01-29"	(Mean)	0.06
(GenID)	'OnHold	0.09	(If (Greater 'SHIPDATE "1996-06-01") 0.04 0.06)
(GenID)	'OnHold	0.10	(If (Greater 'SHIPDATE "1996-06-01") 0.02 0.03)

Imperial College London

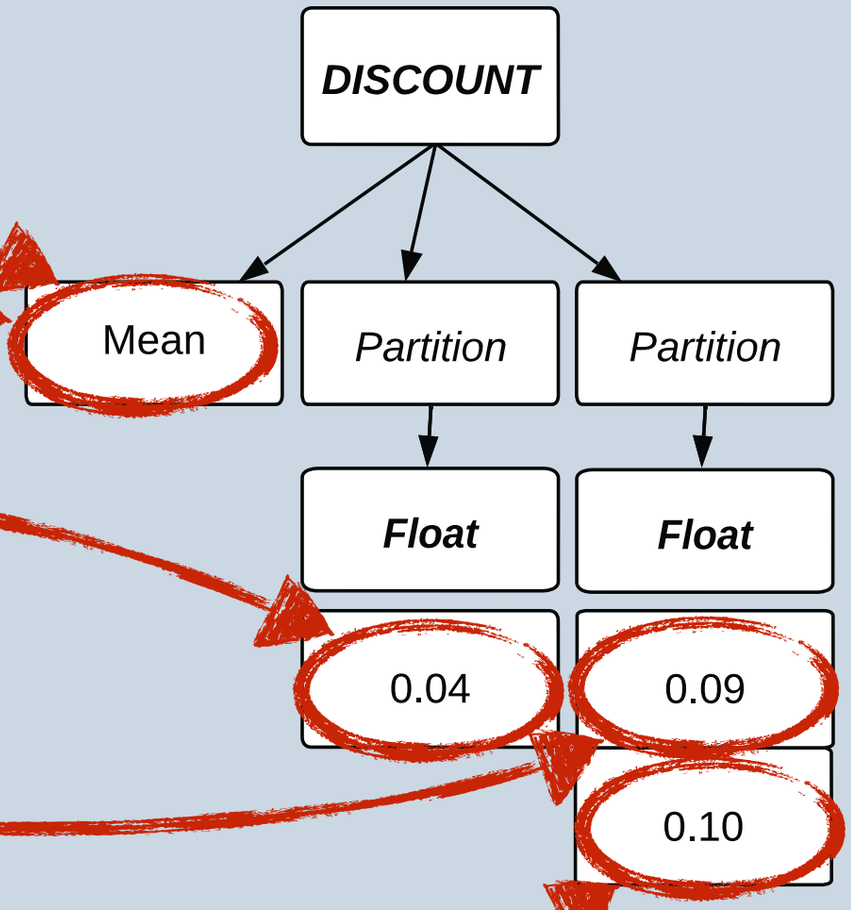
Efficiently Processing Homoiconic Databases

KEY	*SHIPDATE*	*DISCOUNT*	*TAX*
1	"1996-03-13"	(Mean)	0.10
3	"1996-01-29"	(Mean)	0.06
2	"1996-04-12"	0.04	0.08
(GenID)	'OnHold	0.09	(If (Greater 'SHIPDATE "1996-06-01") 0.04 0.06)
(GenID)	'OnHold	0.10	(If (Greater 'SHIPDATE "1996-06-01") 0.02 0.03)

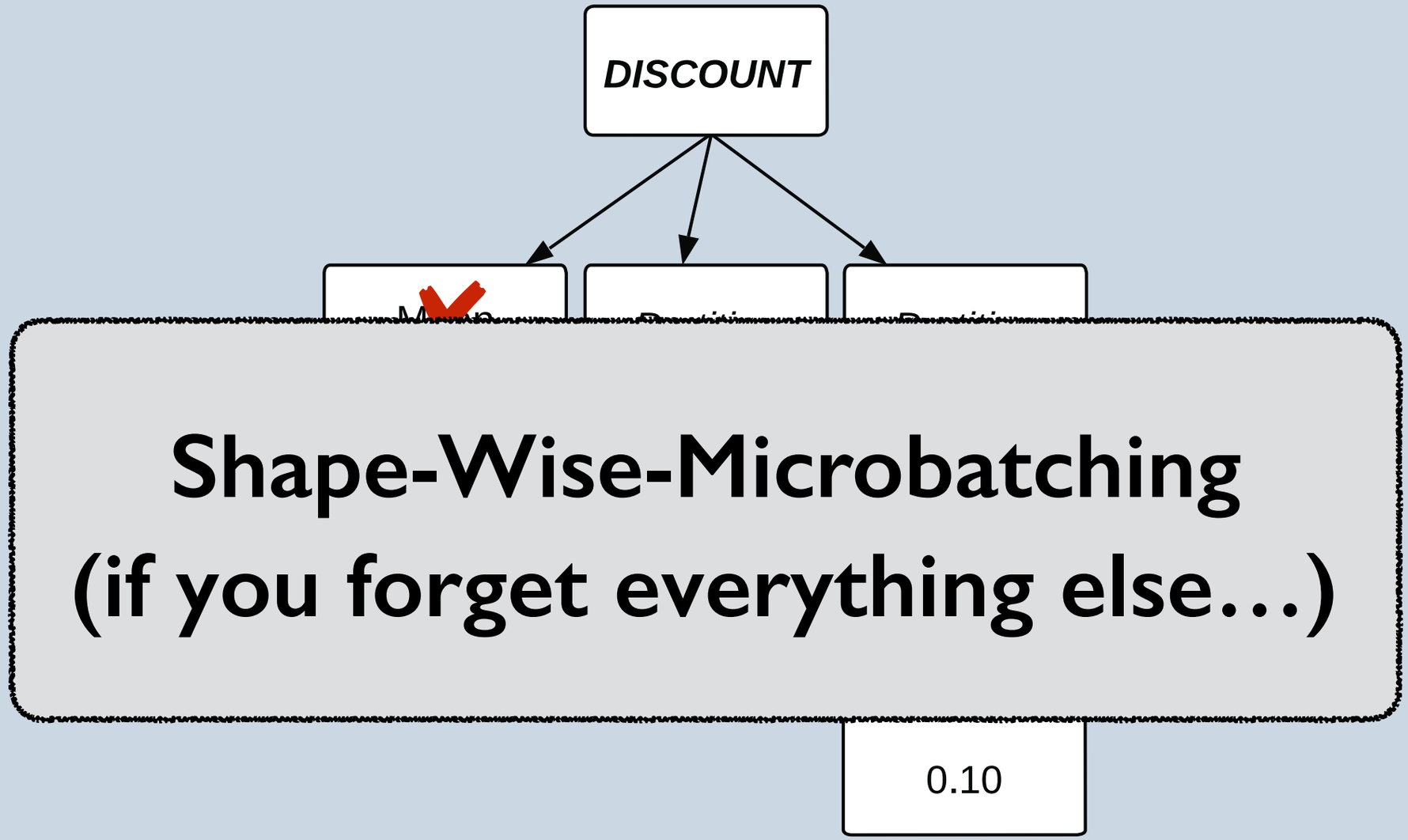
Imperial College London

Efficiently Processing Homoiconic Databases

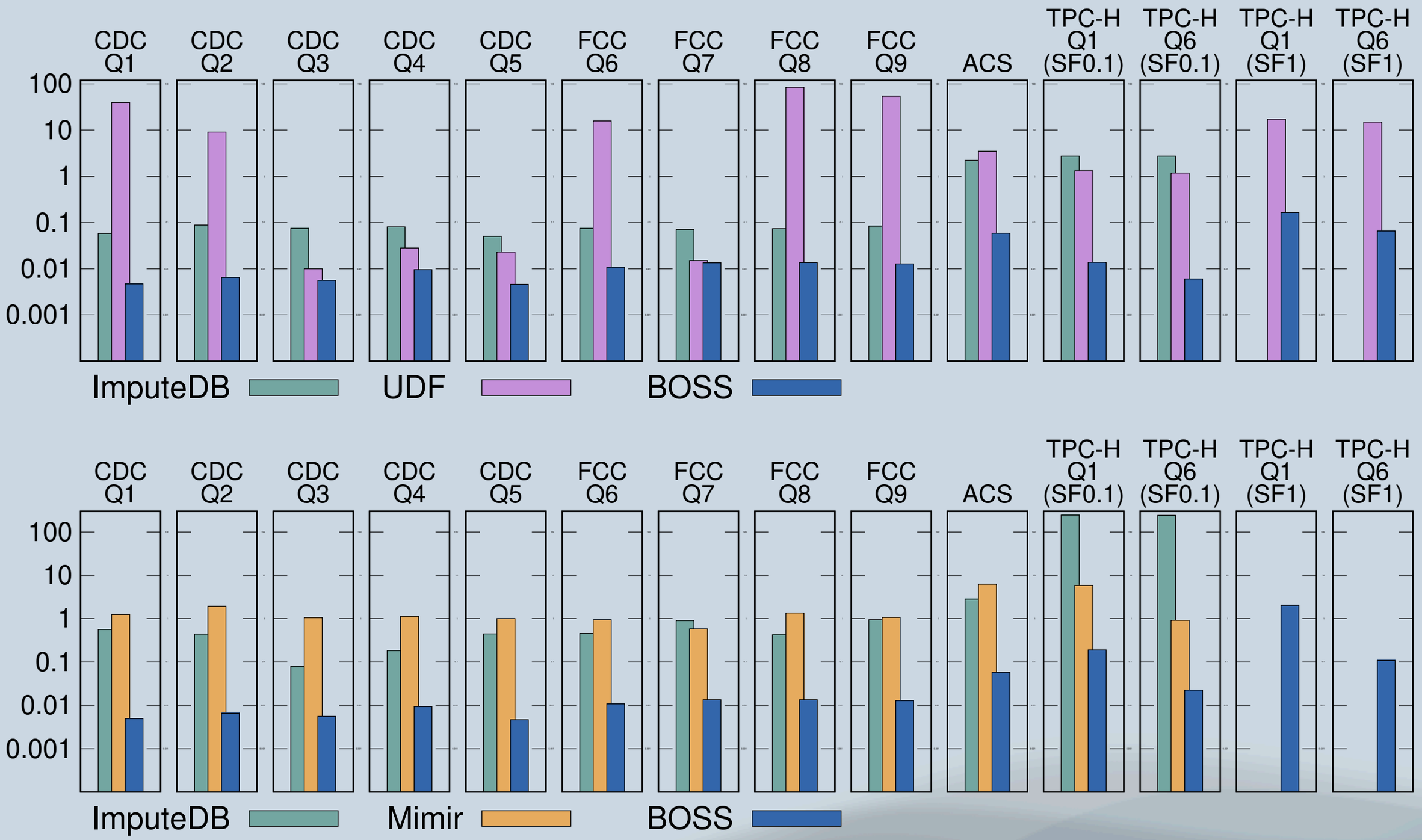
KEY	*SHIPDATE*	*DISCOUNT*	*TAX*
1	"1996-03-13"	(Mean)	0.10
3	"1996-01-29"	(Mean)	0.06
2	"1996-04-12"	0.04	0.08
(GenID)	'OnHold	0.09	(If (Greater 'SHIPDATE "1996-06-01") 0.04 0.06)
(GenID)	'OnHold	0.10	(If (Greater 'SHIPDATE "1996-06-01") 0.02 0.03)



Efficiently Processing Homoiconic Databases



Performance Evaluation



Conclusion

- Separating code and data works well for monolithic systems
- Composable systems need more flexibility
- High-Performance Homoniconicity is
  - Possible
  - Useful

Thank you

Contributors:




Imperial College London