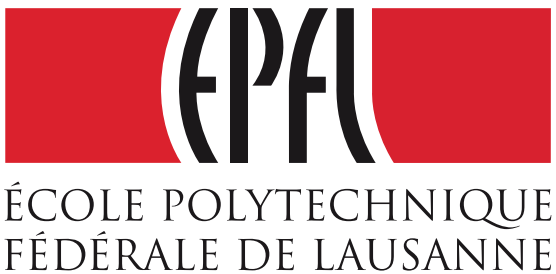
An Embedded Query Language in Scala

Amir Shaikhha

School of Computer and Communication Sciences, EPFL
Typesafe, Lausanne

Master Thesis, August 2013



Outline

Introduction

Problem Statement

Write the code to access database

Introduction

Instead of writing database code in SQL

```
select c.NAME from COFFEES c where c.ID = 10
```

Introduction

Instead of writing database code in SQL

```
select c.NAME from COFFEES c where c.ID = 10
```

Write database code in Scala

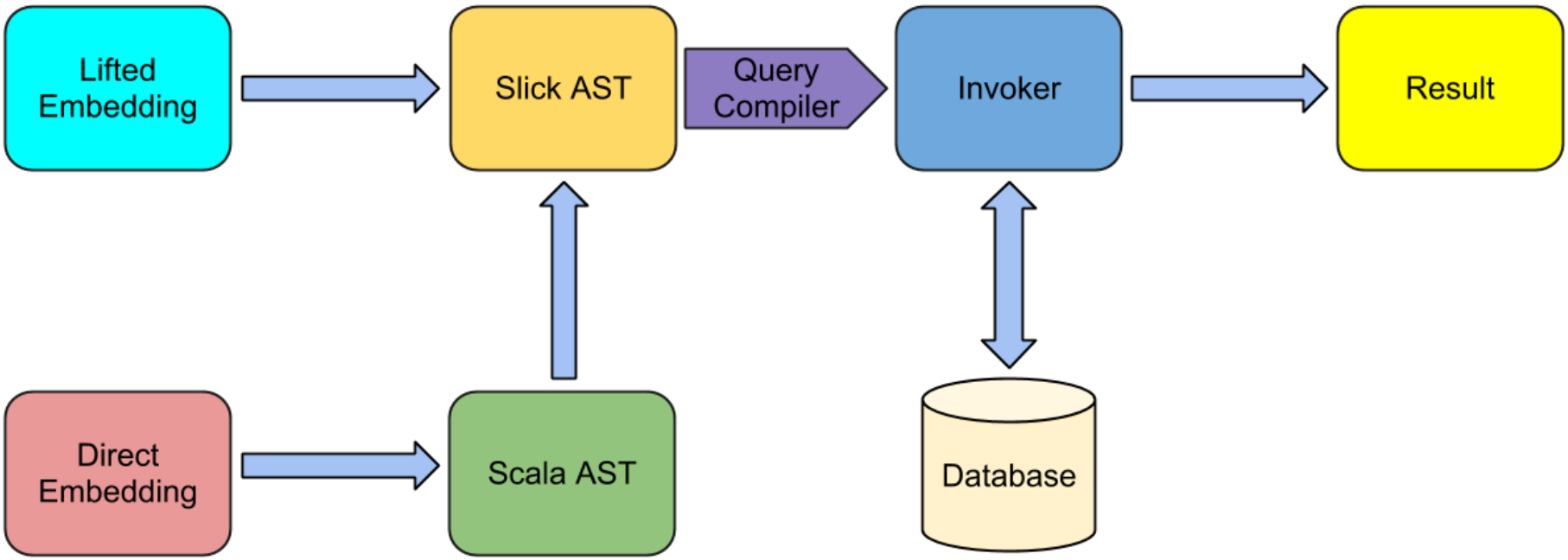
```
for (c <- coffees if c.id == 10) yield c.name
```

Slick

Scala Language-Integrated Connection Kit

Slick

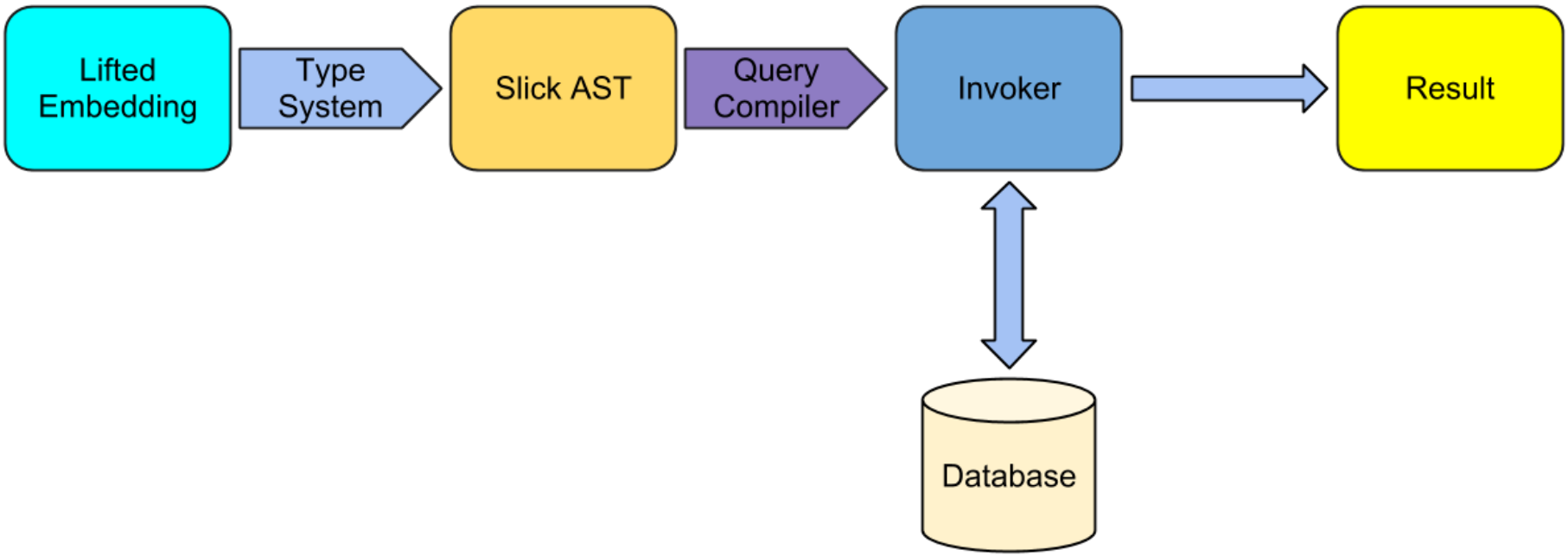
Architecture



Outline

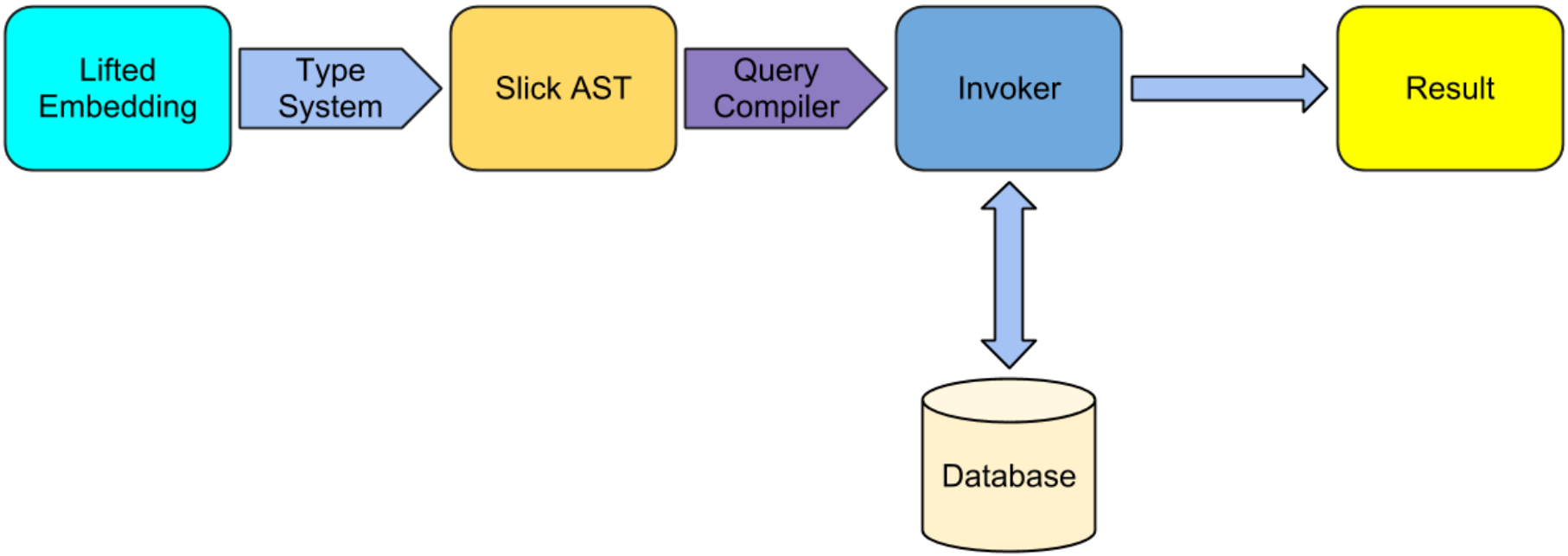
Lifted Embedding

Architecture



Lifted Embedding

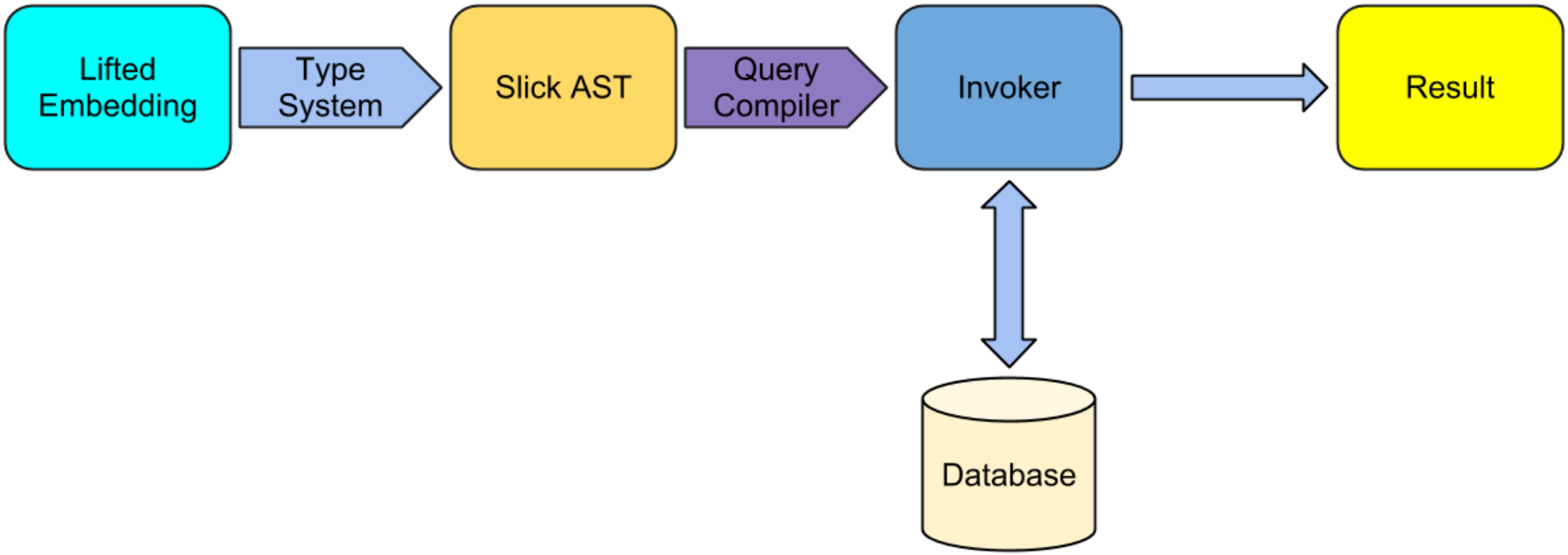
Architecture



- Uses standard Scala

Lifted Embedding

Architecture



- Uses standard Scala
- Not Scala-Virtualized

Lifted Embedding

Example

Lifted Embedding Example

```
Query(Coffees) filter
  (c => c.id === 10) map
  (c => c.name)
```

Lifted Embedding

Example

Lifted Embedding Example

```
Query(Coffees) filter
  (c => c.id === 10) map
  (c => c.name)
```

Scala for-comprehension

```
for (c <- Query(Coffees) if c.id === 10)
  yield c.name
```

Lifted Embedding

Example

Lifted Embedding Example

```
Query(Coffees) filter
  (c => c.id === 10) map
  (c => c.name)
```

SQL Statement

```
select c.NAME from COFFEES c where c.ID = 10
```

Lifted Embedding

Type Information

```
Query(Coffees) filter
  (c => c.id === 10) map
  (c => c.name)
```

Lifted Embedding

Type Information

```
Query(Coffees) filter
  (c => c.id:Rep[Int] === 10:Rep[Int]) map
  (c => c.name:Rep[String])
```

Lifted Embedding

Problem 1

```
Query(Coffees) filter
  (c => c.id === 10) map
  (c => c.name)
```

Lifted Embedding

Problem 1

```
Query(Coffees) filter
  (c => c.id === 10) map
  (c => c.name)
```

How to create Lifted Embedding Table?

Lifted Embedding

```
object Coffees extends Table[(Int, String, Double,
    String, Int)]("COFFEES") {
  def id = column[Int]("ID", O.PrimaryKey)
  def name = column[String]("NAME")
  //...
}
```

Lifted Embedding

```
object Coffees extends Table[(Int, String, Double,
    String, Int)]("COFFEES") {
  def id = column[Int]("ID", O.PrimaryKey)
  def name = column[String]("NAME")
  //...
}
```

Boilerplate!

Type Providers

Generate the types

Type Providers

Generate the types out of:

- Existing Schema

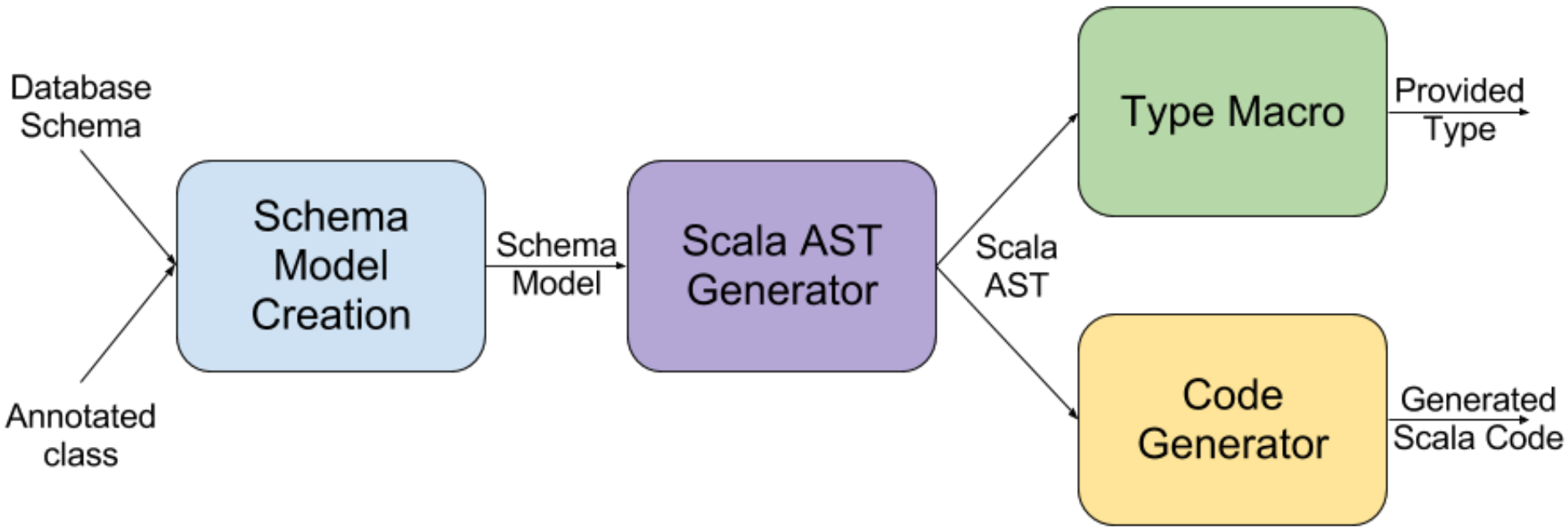
Type Providers

Generate the types out of:

- Existing Schema
- Annotated classes

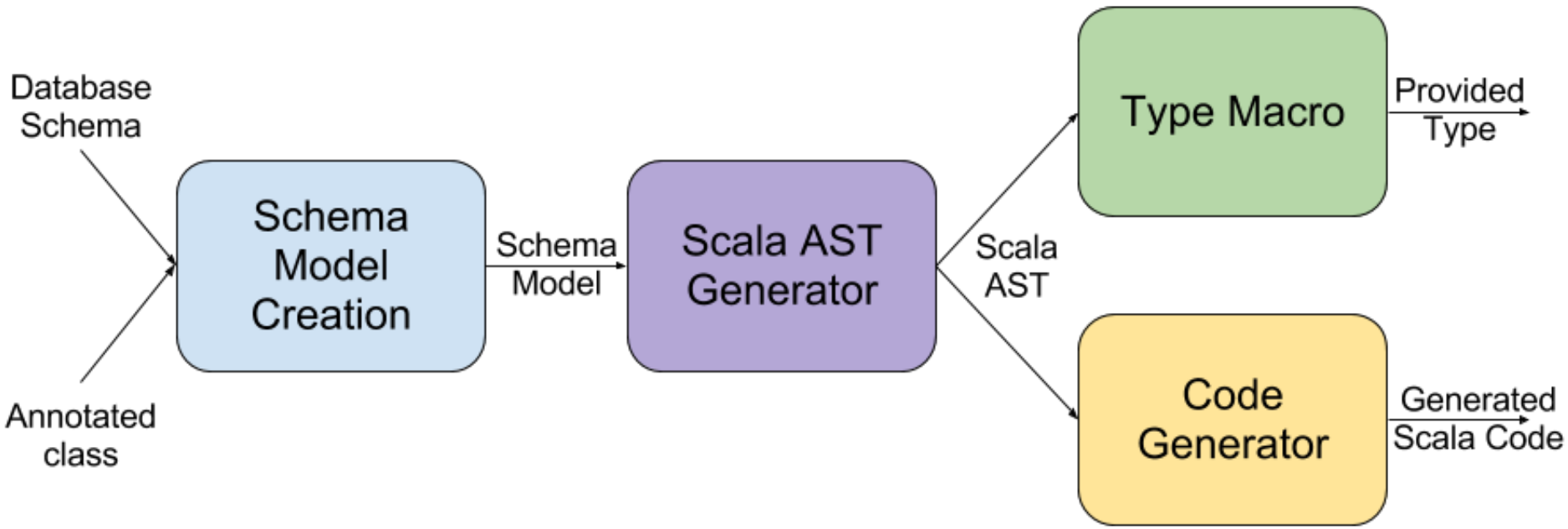
Type Providers

Architecture



Type Providers

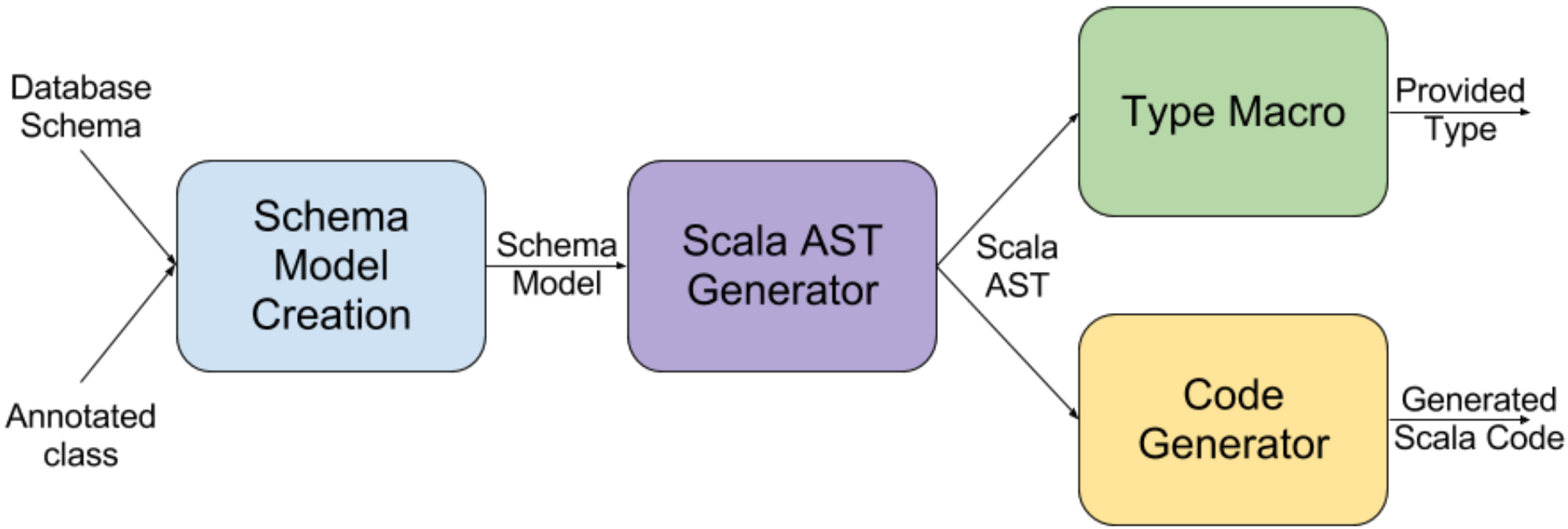
Architecture



- Type Macros are in macro paradise

Type Providers

Architecture



- Type Macros are in macro paradise
- Code Generation uses standard Scala

Lifted Embedding

Problem 2

How to catch the errors?

Lifted Embedding

Type Errors - Good Part

```
Query(Coffees) map
  (c => c.id.toDouble)
```

Lifted Embedding

Type Errors - Good Part

```
Query(Coffees) map
  (c => c.id.toDouble)
```

Compile Error

value toDouble is not a member of scala.slick.lifted.Column[Int]

Lifted Embedding

Type Errors - Bad Part

```
Query(Coffees) map
  (c => c.id substring 2)
```

Lifted Embedding

Type Errors - Bad Part

```
Query(Coffees) map
  (c => c.id substring 2)
```

Compile Error

value substring is not a member of scala.slick.lifted.Column[Int]

Lifted Embedding

Type Errors - Bad Part

```
Query(Coffees) map
  (c => c.id substring 2)
```

Compile Error

value substring is not a member of **scala.slick.lifted.Column[Int]**

Lifted Embedding

Type Errors - Even Worse!

```
Query(Coffees) map (c =>
  if(c.origin == "Iran")
    "Good"
  else
    c.quality
)
```

Lifted Embedding

Type Errors - Even Worse!

```
Query(Coffees) map (c =>
  if(c.origin == "Iran")
    "Good"
  else
    c.quality
)
```

Compile Error

- Don't know how to unpack Any to T and pack to G
- not enough arguments for method map: (implicit shape: scala.slick.lifted.Shape[Any,T,G])scala.slick.lifted.Query[G,T]. Unspecified value parameter

Lifted Embedding

Type Errors - Even Worse!

```
Query(Coffees) map (c =>
  if(c.origin == "Iran")
    "Good"
  else
    c.quality
)
```

Compile Error

- Don't know how to unpack Any to T and pack to G
- not enough arguments for method map: (implicit shape: scala.slick.lifted.Shape[Any,T,G])scala.slick.lifted.Query[G,T]. Unspecified value parameter

Scala-Virtualized has not this problem

Lifted Embedding

Type Errors

Adapted from http://thumbs.dreamstime.com/z/old-bus-desert-7703223.jpg

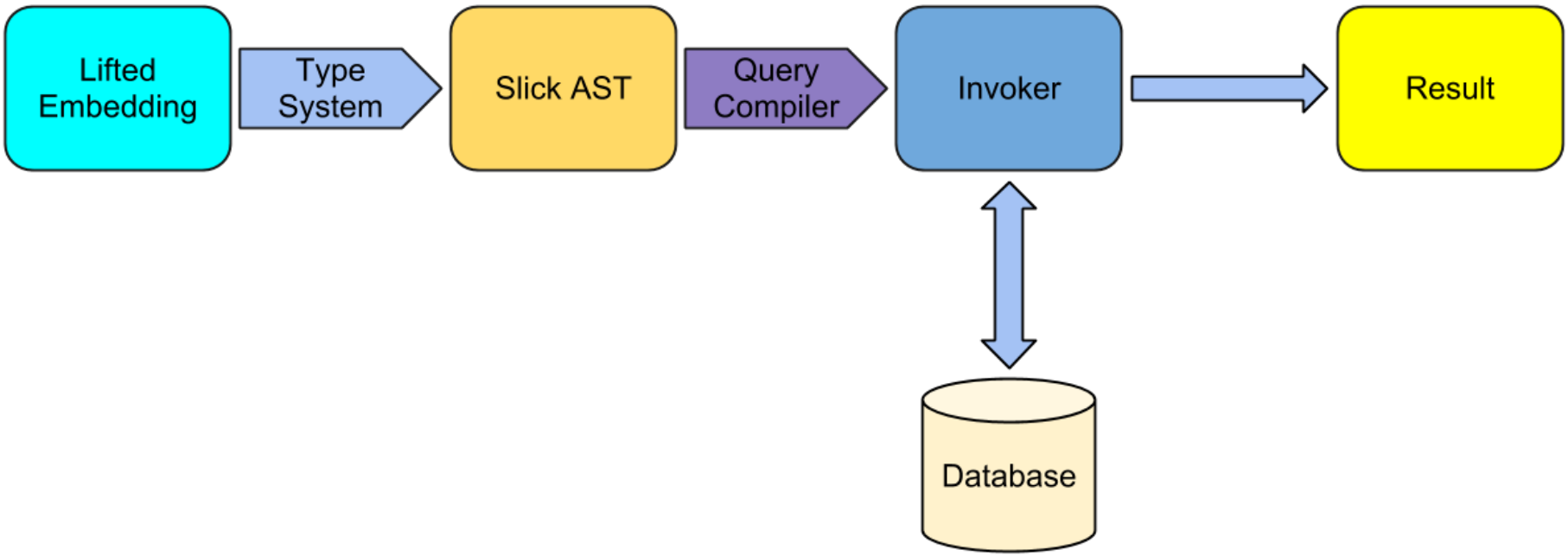
Lifted Embedding

Problem 3

How to have high performance?

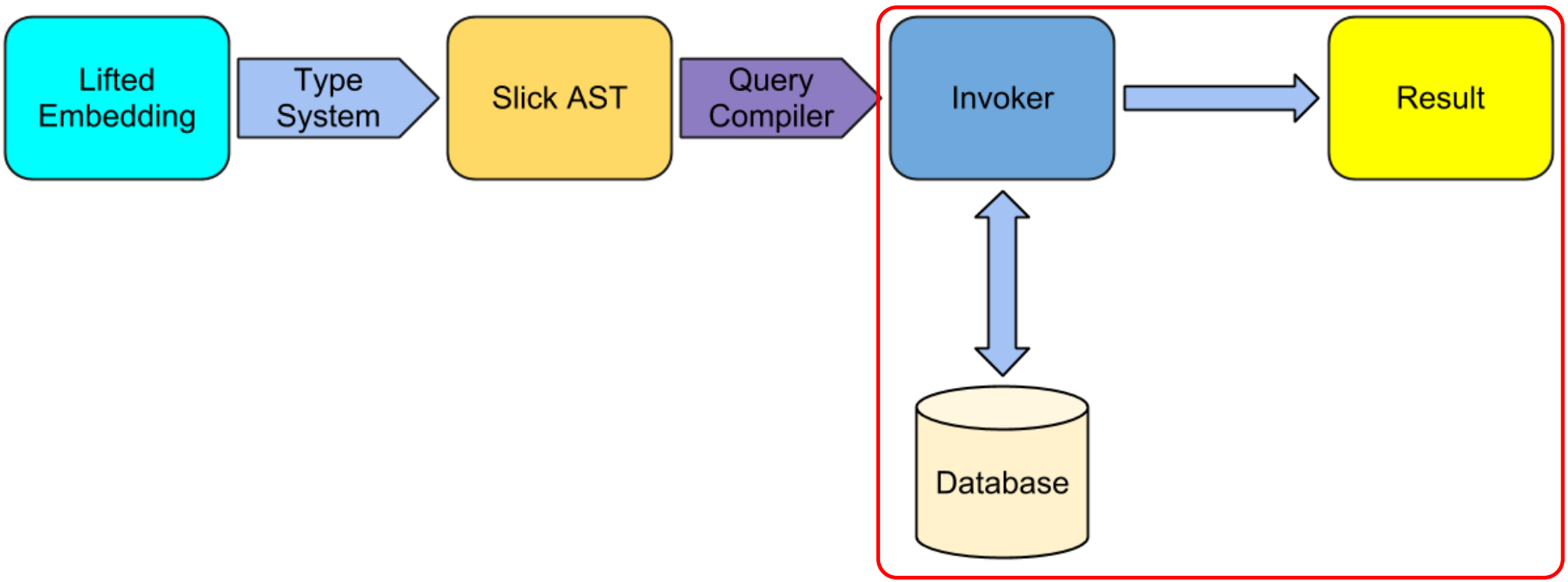
Lifted Embedding

Performance



Lifted Embedding

Performance



Lifted Embedding

Performance

- Caching invokers

Lifted Embedding

Performance

- Caching invokers
- Query templates

Lifted Embedding

Query Template

Lifted Embedding Query Template

```
val getCoffee = for {
  id <- Parameters[Int]
  c <- Query(Coffees) if c.id === id
} yield c.name

getCoffee(10)
```

Lifted Embedding

Query Template

Lifted Embedding Query Template

```
val getCoffee = for {
  id <- Parameters[Int]
  c <- Query(Coffees) if c.id === id
} yield c.name

getCoffee(10)
```

JDBC Prepared Statement

```
"select c.NAME from COFFEES c where c.ID = ?"
```

Lifted Embedding

Summary

Lifted Embedding

Summary

Problem 1

How to create Lifted Embedding Table?

Lifted Embedding

Summary

Problem 1

How to create Lifted Embedding Table?

Type Providers

Lifted Embedding

Summary

Problem 1

How to create Lifted Embedding Table?

Type Providers

Problem 2

How to catch the errors?

Lifted Embedding

Summary

Problem 1

How to create Lifted Embedding Table?

Type Providers

Problem 2

How to catch the errors?

Comprehensive type errors

Lifted Embedding

Summary

Problem 1

How to create Lifted Embedding Table?

Type Providers

Problem 2

How to catch the errors?

Comprehensive type errors

Nonunderstandable type errors

Lifted Embedding

Summary

Problem 1

How to create Lifted Embedding Table?

Type Providers

Problem 2

How to catch the errors?

Comprehensive type errors

Nonunderstandable type errors

Problem 3

How to have high performance?

Lifted Embedding

Summary

Problem 1

How to create Lifted Embedding Table?

Type Providers

Problem 2

How to catch the errors?

Comprehensive type errors

Nonunderstandable type errors

Problem 3

How to have high performance?

Caching Invokers and Query Templates

Lifted Embedding

Summary

Problem 1

How to create Lifted Embedding Table?

Type Providers

Problem 2

How to catch the errors?

Comprehensive type errors

Nonunderstandable type errors

Problem 3

How to have high performance?

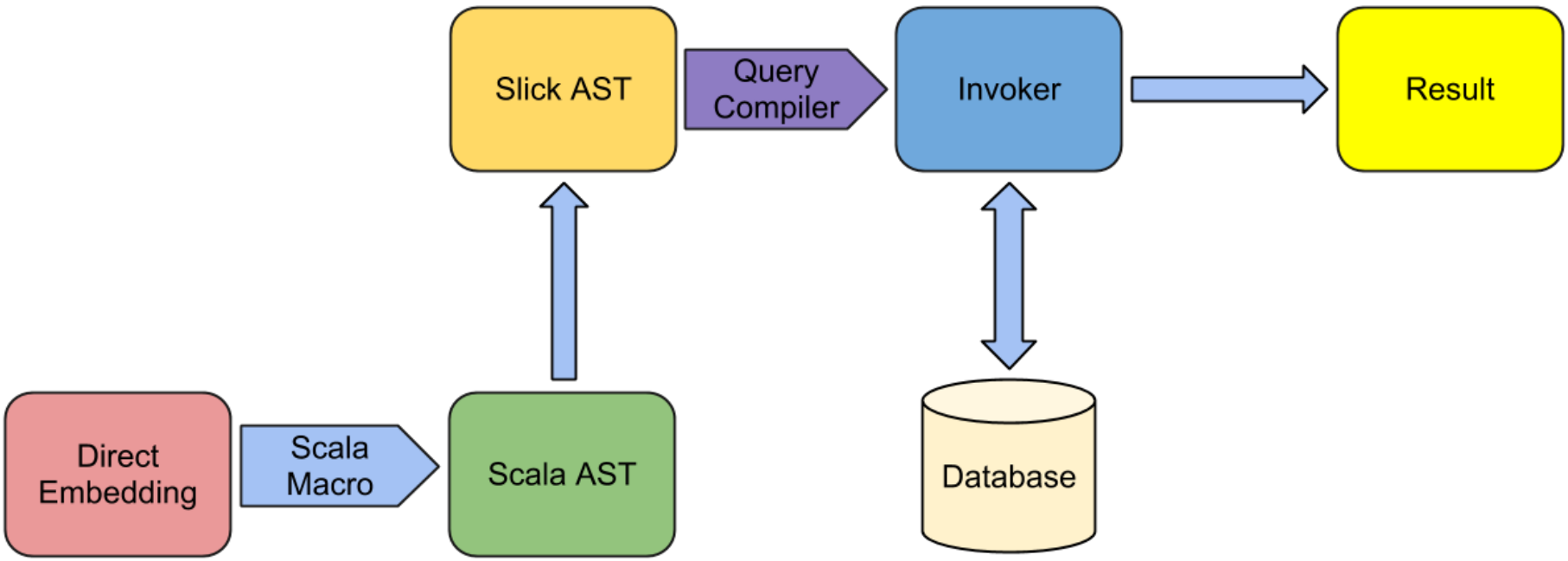
Caching Invokers and Query Templates

User effort needed

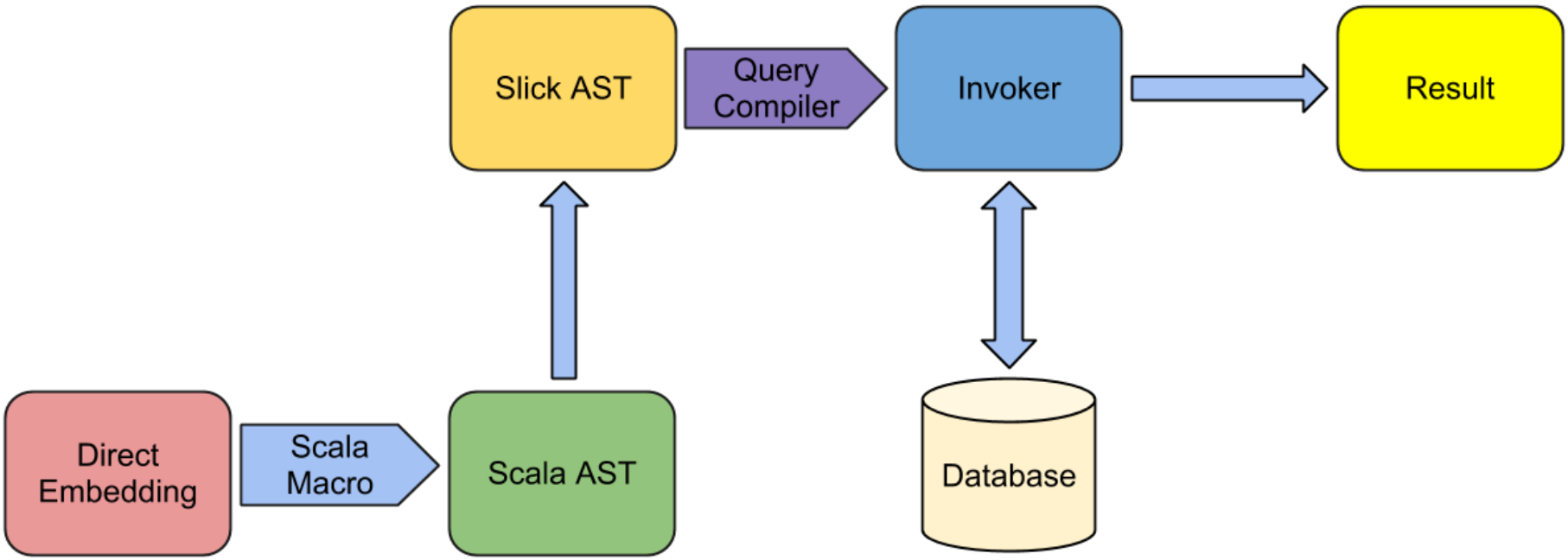
Is it possible to have comprehensible type errors?

Outline

Direct Embedding

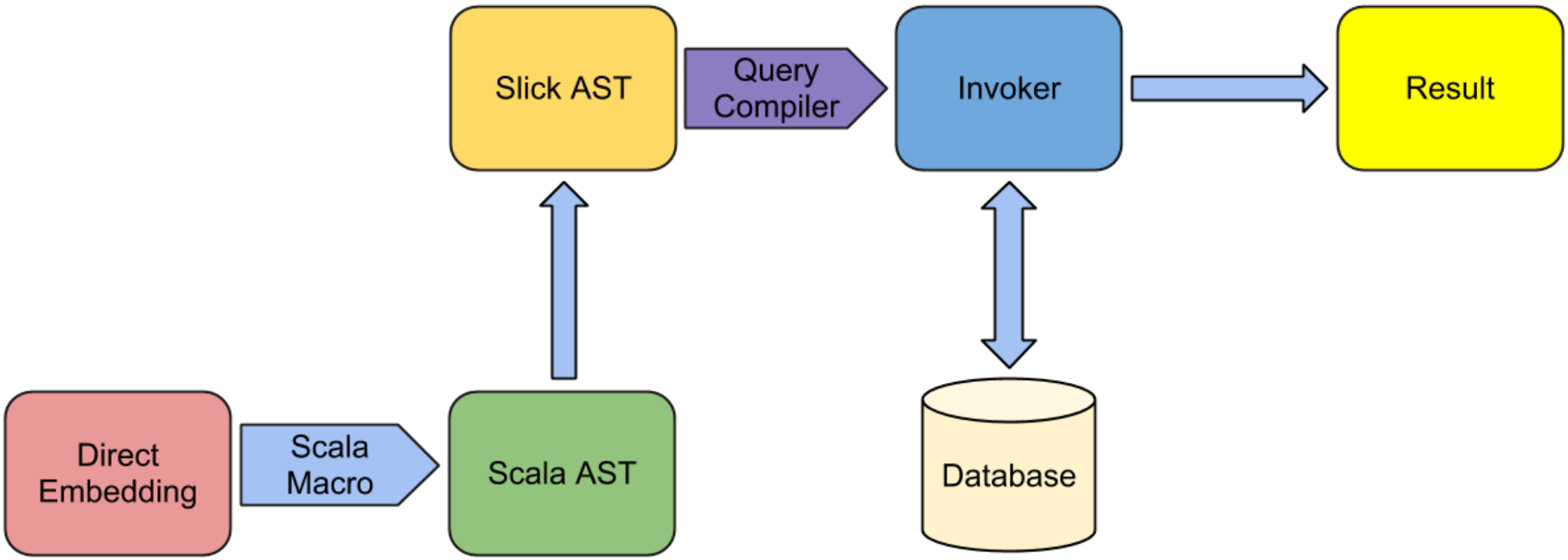


Direct Embedding



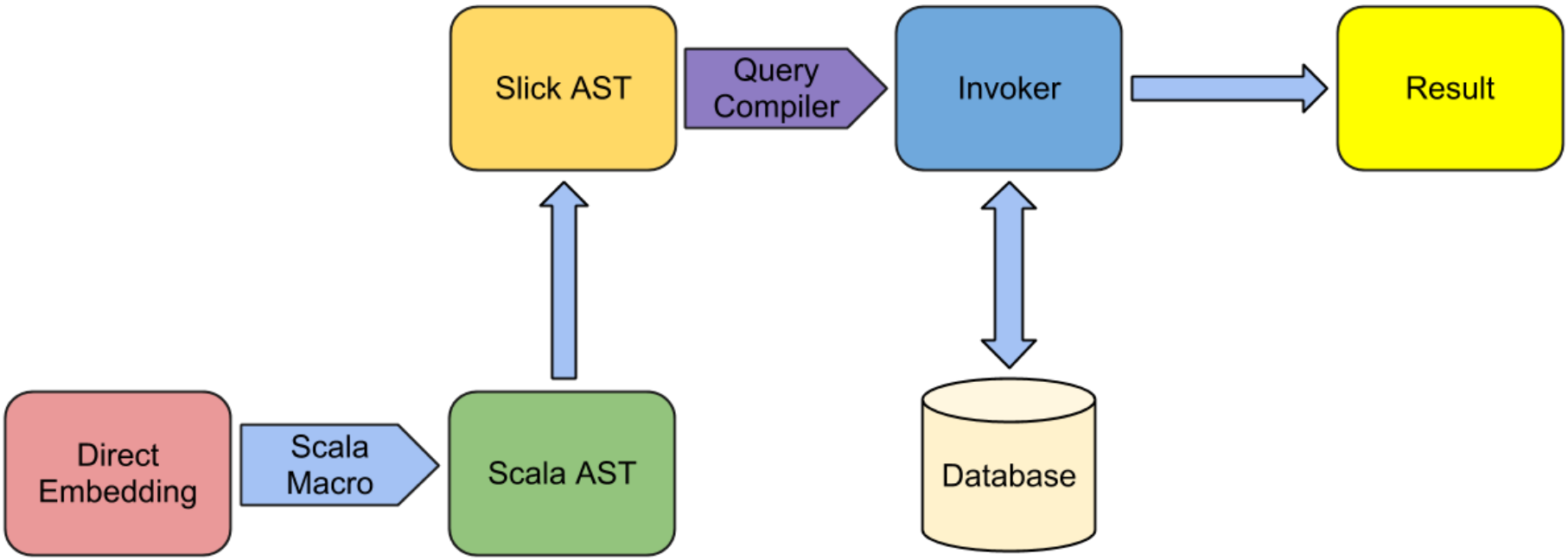
- Query expression to Scala AST (compile-time)

Direct Embedding



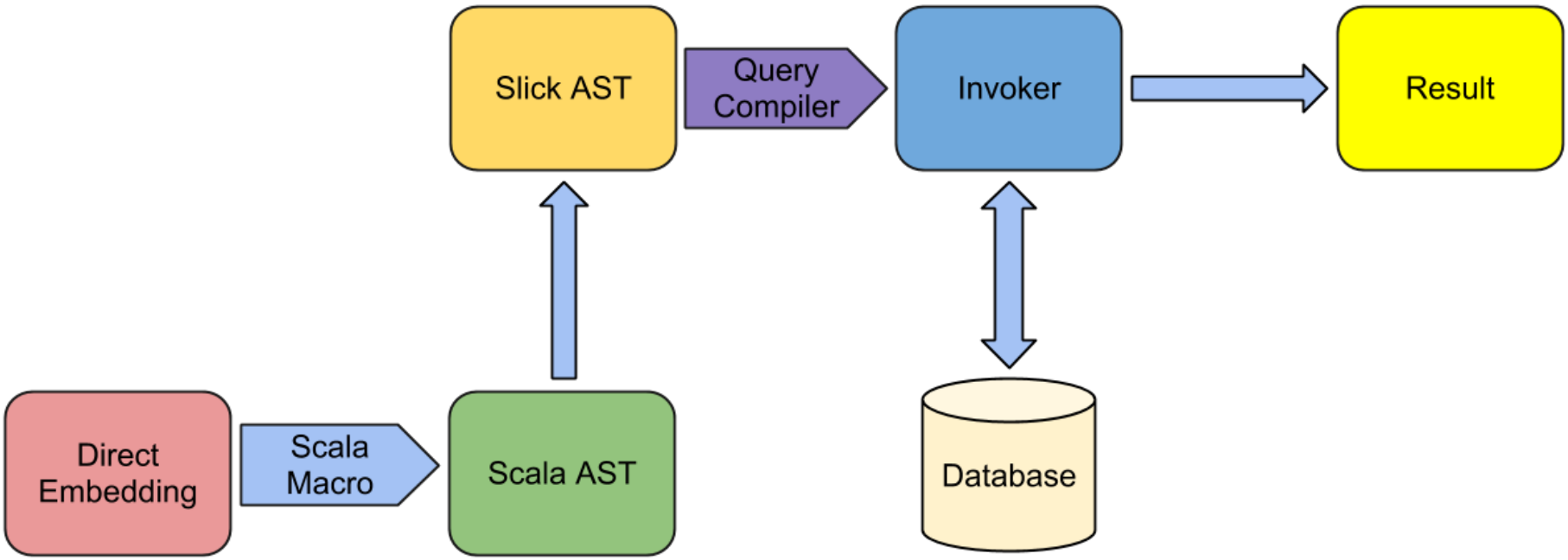
- Query expression to Scala AST (compile-time)
- Scala AST to Slick AST (run time)

Direct Embedding



- Query expression to Scala AST (compile-time)
- Scala AST to Slick AST (run time)
- Similar to LINQ

Direct Embedding



- Query expression to Scala AST (compile-time)
- Scala AST to Slick AST (run time)
- Similar to LINQ
- A prototype

Direct Embedding

Example

Direct Embedding Example

```
Query[Coffee] filter
  (c => c.id == 10) map
  (c => c.name)
```

Direct Embedding

Example

Direct Embedding Example

```
Query[Coffee] filter
  (c => c.id == 10) map
  (c => c.name)
```

SQL Statement

```
select c.NAME from COFFEES c where c.ID = 10
```

Direct Embedding

Type Information

```
Query[Coffee] filter
  (c => c.id == 10) map
  (c => c.name)
```

Direct Embedding

Type Information

```
Query[Coffee] filter
  (c => c.id:Int == 10:Int) map
  (c => c.name:String)
```

Direct Embedding

Type Errors - Good Part

```
Query[Coffee] map
  (c => c.id substring 2)
```

Direct Embedding

Type Errors - Good Part

```
Query[Coffee] map
  (c => c.id substring 2)
```

Compile Error

value substring is not a member of Int

Direct Embedding

Type Errors - Good Part

```
Query[Coffee] map
  (c => c.id substring 2)
```

Compile Error

value substring is not a member of **Int**

Direct Embedding

Type Errors - Bad Part

```
Query[Coffee] map
  (c => c.id.toDouble)
```

Direct Embedding

Type Errors - Bad Part

```
Query[Coffee] map
  (c => c.id.toDouble)
```

Compiles!

Direct Embedding

Type Errors - Bad Part

```
Query[Coffee] map
  (c => c.id.toDouble)
```

Compiles!

Run time error!

Direct Embedding

Type Errors

Adapted from http://r32argent.ca/R32%20information_files/VW%20ads/vw_bus.jpg

Direct Embedding

Summary

Problem 2 (recap)

How to catch the errors?

Direct Embedding

Summary

Problem 2 (recap)

How to catch the errors?

Comprehensible type errors

Direct Embedding

Summary

Problem 2 (recap)

How to catch the errors?

Comprehensible type errors

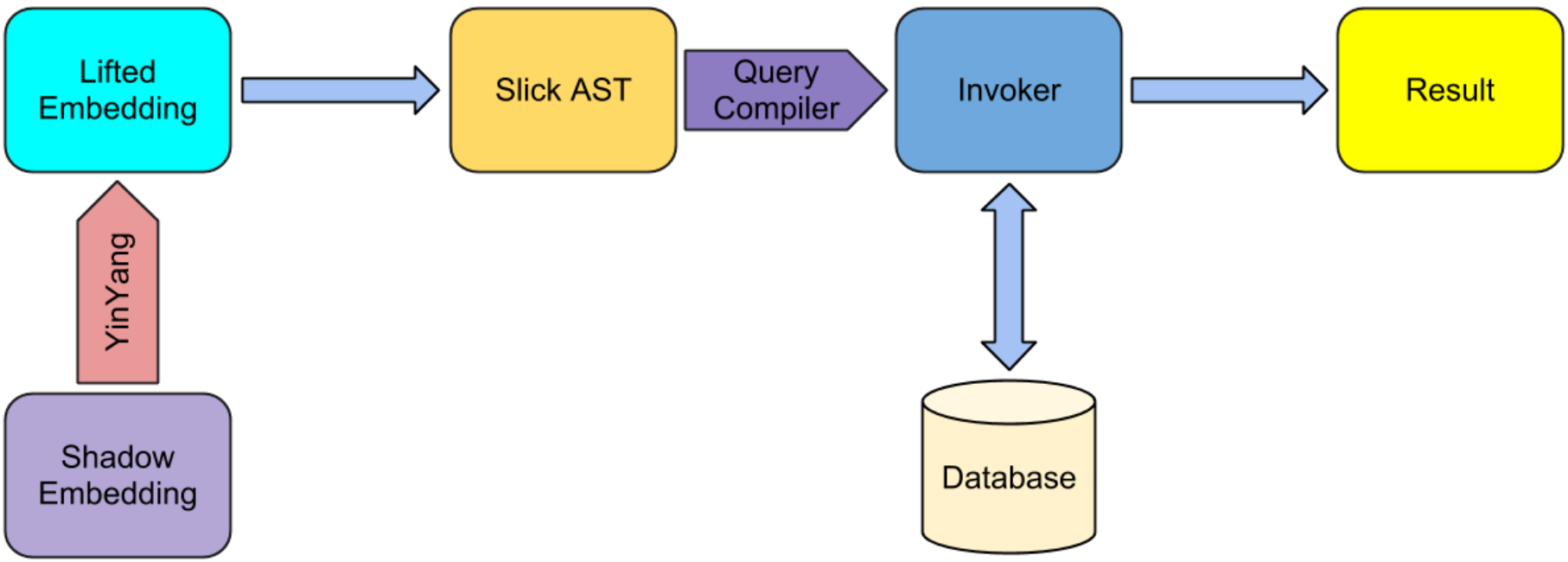
Incomprehensive type errors

Is it possible to have comprehensive and comprehensible type errors at the same time?

Outline

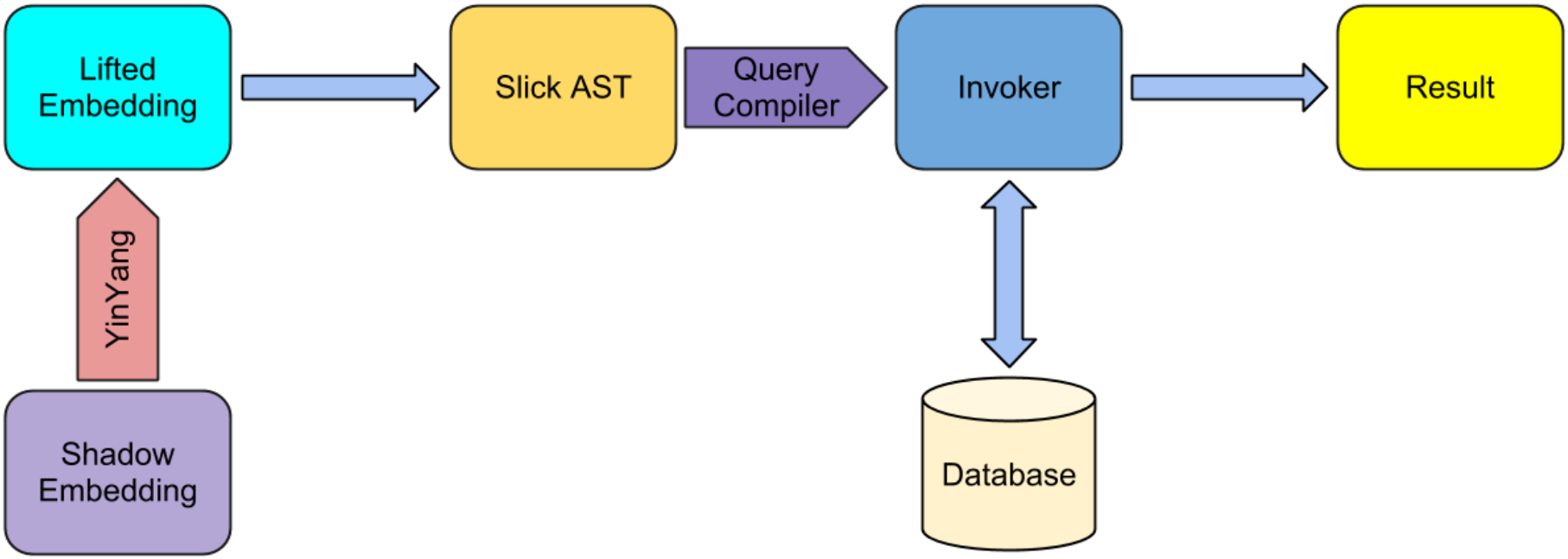
Shadow Embedding

Architecture



Shadow Embedding

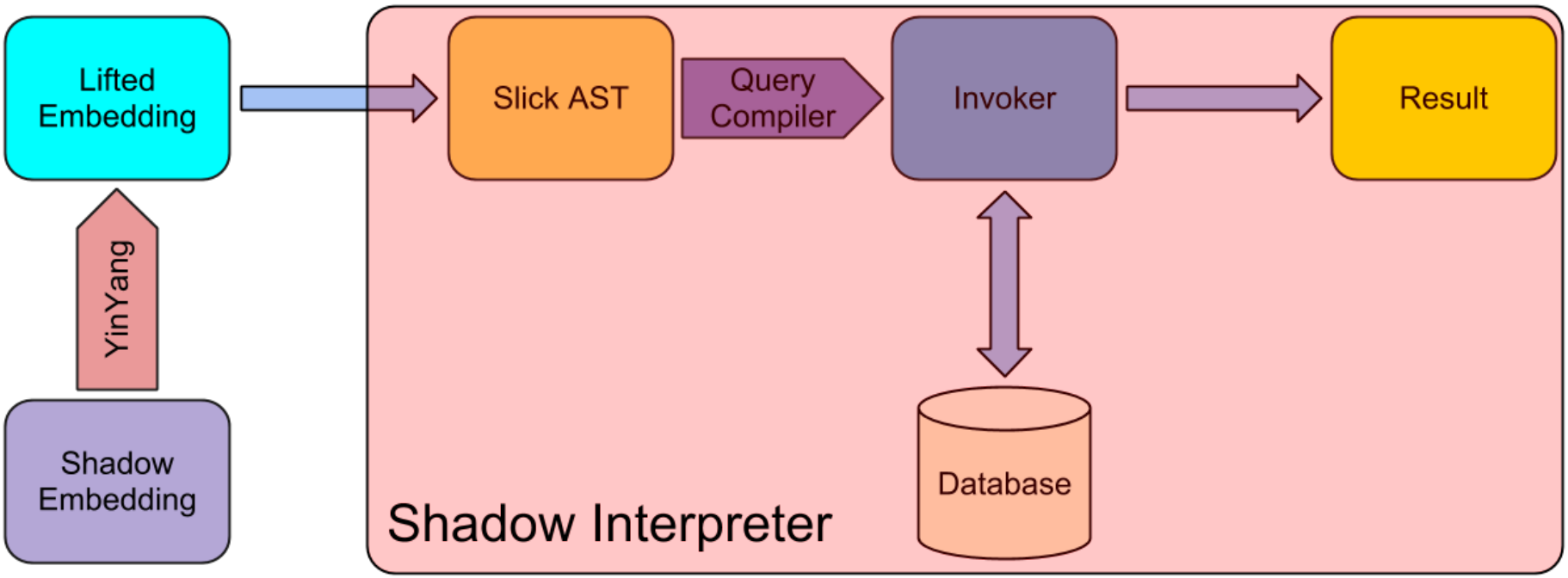
Architecture



$$shadow = shallow + deep$$

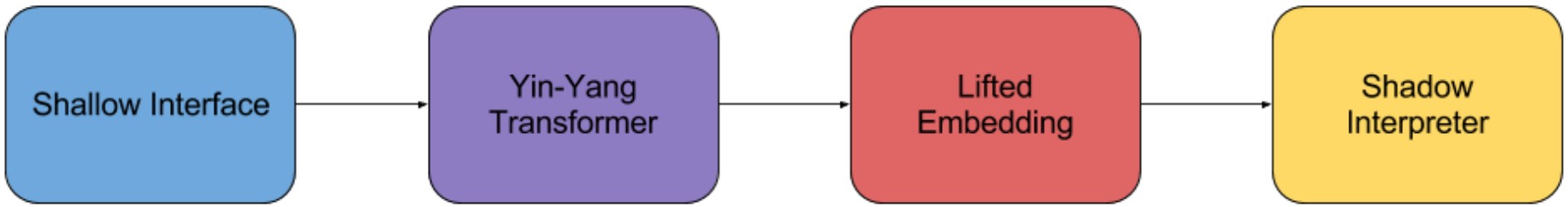
Shadow Embedding

Architecture



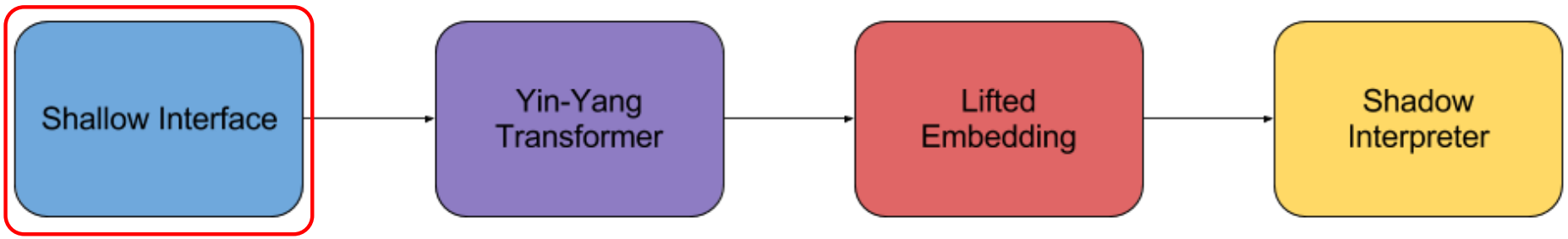
Shadow Embedding

Architecture



Shadow Embedding

Shallow Interface



Shadow Embedding

Shallow Interface

Query interface:

Shadow Embedding

Shallow Interface

Query interface:

```
class Query[T] {
  def map[S](f: T => S): Query[S]
  def filter(f: T => Boolean): Query[T]
  def flatMap[S](f: T => Query[S]): Query[S]
  def groupBy[S](f: T => S): Query[(S, Query[T])]
  def union(q2: Query[T]): Query[T]
  def join[S](q2: Query[S]): JoinQuery[T, S]
  // ...
}
```

Shadow Embedding

Example

Shallow Embedding Example

```
stage {
  Query[Coffee] filter
    (c => c.id == 10) map
    (c => c.name)
}
```

Shadow Embedding

Type Information

```
Query[Coffee] filter
  (c => c.id == 10) map
  (c => c.name)
```

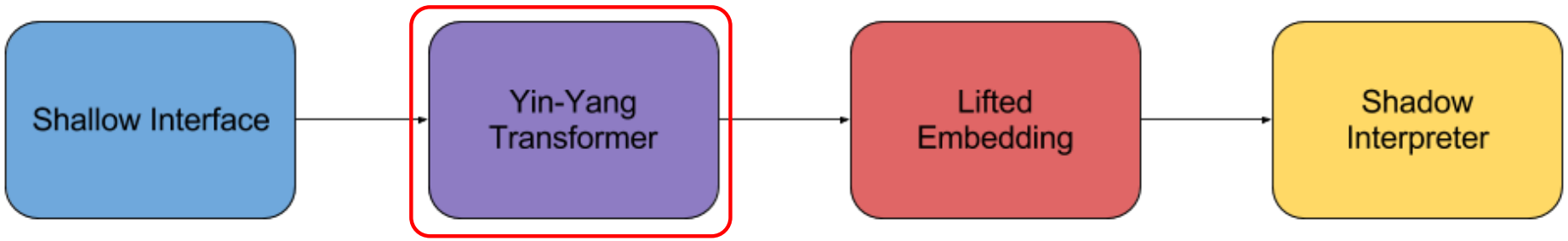
Shadow Embedding

Type Information

```
Query[Coffee] filter
  (c => c.id:Int == 10:Int) map
  (c => c.name:String)
```

Shadow Embedding

Yin-Yang Transformation

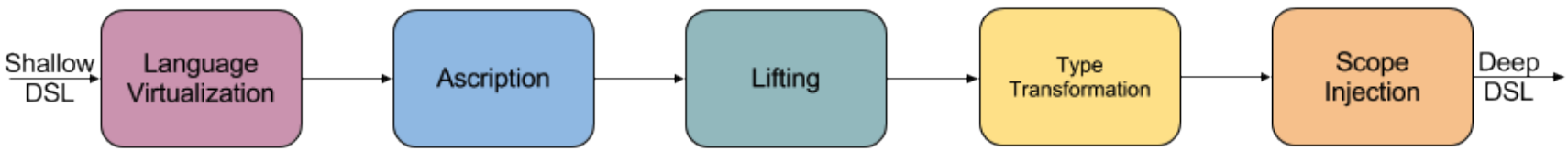


Shadow Embedding

Yin-Yang Transformation

Shallow Query

```
stage {
  Query(1) filter (x => x == 10)
}
```

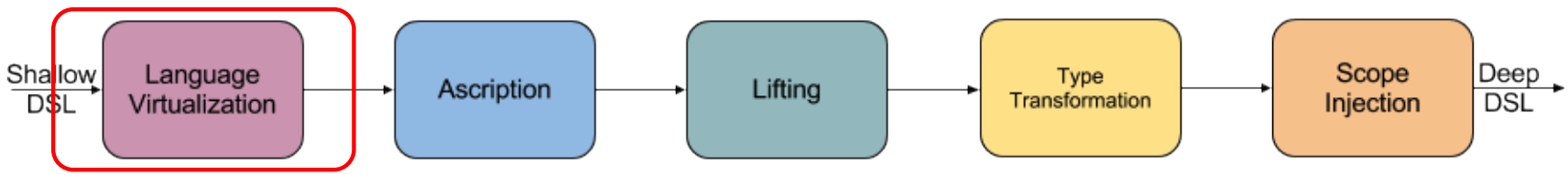


Shadow Embedding

Yin-Yang Transformation

After Language Virtualization

```
Query(1) filter (x => x __== 10)
```

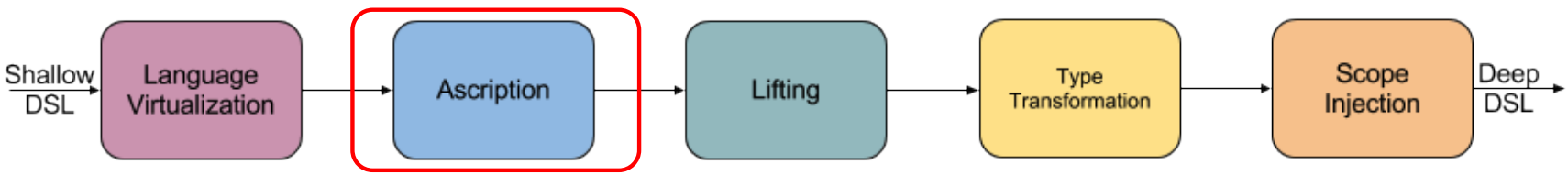


Shadow Embedding

Yin-Yang Transformation

After Ascription

```
Query(1:Int) filter
  ((x:Int) => (x:Int) __== (10:Int))
```

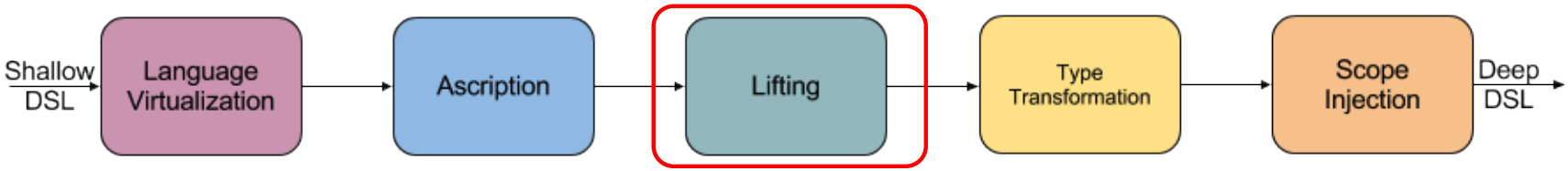


Shadow Embedding

Yin-Yang Transformation

After Lifting

```
Query(lift(1):Int) filter
  ((x:Int) => (x:Int) __== (lift(10):Int))
```

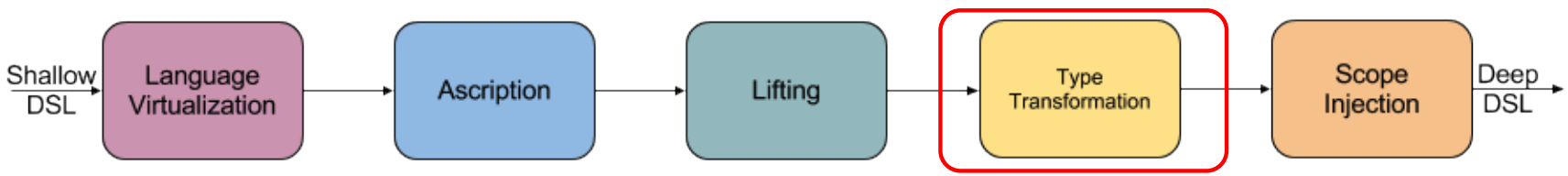


Shadow Embedding

Yin-Yang Transformation

After Type Transformation

```
Query(lift(1):this.Int) filter
  ((x:this.Int) =>
    (x:this.Int) __== (lift(10):this.Int))
```

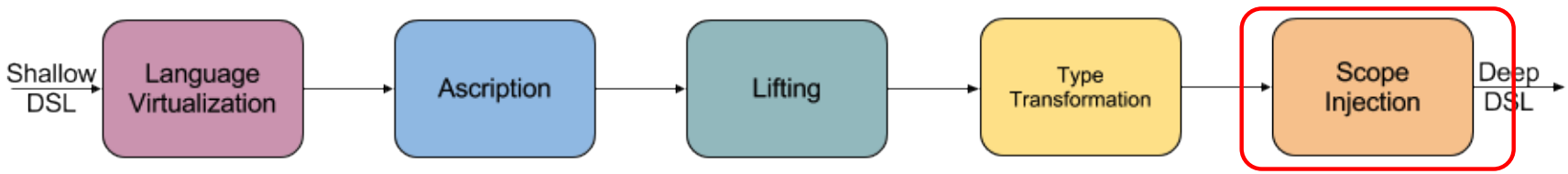


Shadow Embedding

Yin-Yang Transformation

After Scope Injection

```
new ShadowDSLComponent {
  this.Query(lift(1):this.Int) filter
    ((x:this.Int) =>
      (x:this.Int) __== (lift(10):this.Int))
}
```

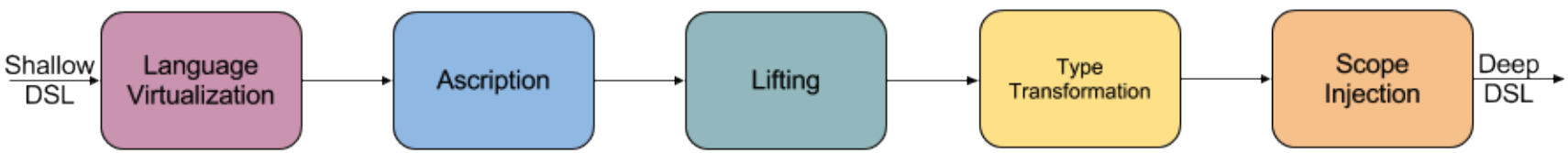


Shadow Embedding

Yin-Yang Transformation

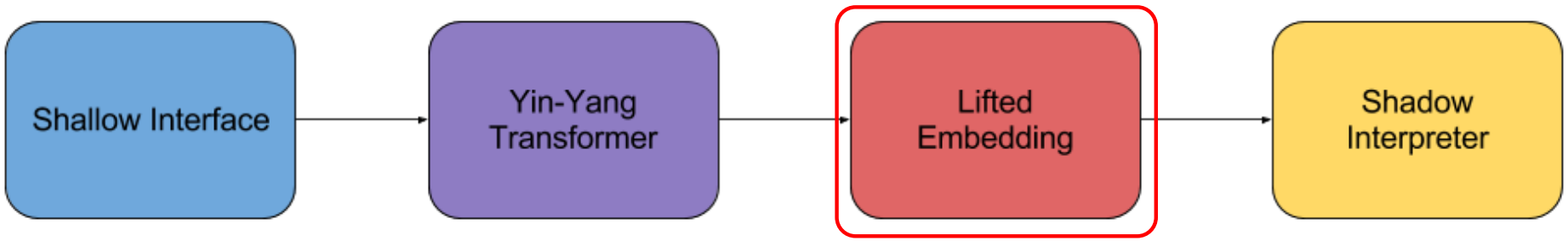
Lifted Embedding Query

```
new ShadowDSLComponent {
  this.Query(lift(1):this.Int) filter
    ((x:this.Int) =>
      (x:this.Int) __== (lift(10):this.Int))
}
```



Shadow Embedding

Lifted Embedding



Shadow Embedding

Lifted Embedding

- No need to convert from Scala AST to Slick AST

Shadow Embedding

Lifted Embedding

- No need to convert from Scala AST to Slick AST
- Interoperable with Lifted Embedding

Shadow Embedding

A Problem similar to Problem 1

Shadow Embedding

A Problem similar to Problem 1

Problem 1 (recap)

How to create Lifted Embedding Table?

Shadow Embedding

A Problem similar to Problem 1

Problem 1 (recap)

How to create Lifted Embedding Table?

```
stage {
  Query[Coffee] map (c => c.id)
}
```

Shadow Embedding

A Problem similar to Problem 1

Problem 1 (recap)

How to create Lifted Embedding Table?

```
stage {
  Query[Coffee] map (c => c.id)
}
```

How to create Shadow Embedding Table?

Shadow Embedding

A Problem similar to Problem 1

Problem 1 (recap)

How to create Lifted Embedding Table?

```
stage {
  Query[Coffee] map (c => c.id)
}
```

How to create Shadow Embedding Table?

Reuse Type Providers of Lifted Embedding!

Shadow Embedding

Problem 2

Shadow Embedding

Problem 2

Problem 2 (recap)

How to catch the errors?

Shadow Embedding

Type Errors - Good Part

```
stage {
  Query[Coffee] map
    (c => c.id substring 2)
}
```

Shadow Embedding

Type Errors - Good Part

```
stage {
  Query[Coffee] map
    (c => c.id substring 2)
}
```

Compile Error

value substring is not a member of Int

Shadow Embedding

Type Errors - Good Part

```
stage {
  Query[Coffee] map
    (c => c.id substring 2)
}
```

Compile Error

value substring is not a member of **Int**

Shadow Embedding

Type Errors - Good Part Again!

```
stage {
  Query[Coffee] map
    (c => c.id.toDouble)
}
```

Shadow Embedding

Type Errors - Good Part Again!

```
stage {
  Query[Coffee] map
    (c => c.id.toDouble)
}
```

Compile Error

in Slick method toDouble is not a member of Int

Shadow Embedding

Type Errors - Surprise!

```
stage {
  Query[Coffee] map (c =>
    if(c.origin == "Iran")
      "Good"
    else
      c.quality
  )
}
```

Shadow Embedding

Type Errors - Surprise!

```
stage {
  Query[Coffee] map (c =>
    if(c.origin == "Iran")
      "Good"
    else
      c.quality
  )
}
```

Compiles and works!

Shadow Embedding

Type Errors

Adapted from http://www.littlerocktours.com/images/vehicles/buses/109-lg.jpg

Shadow Embedding

Problem 3

Shadow Embedding

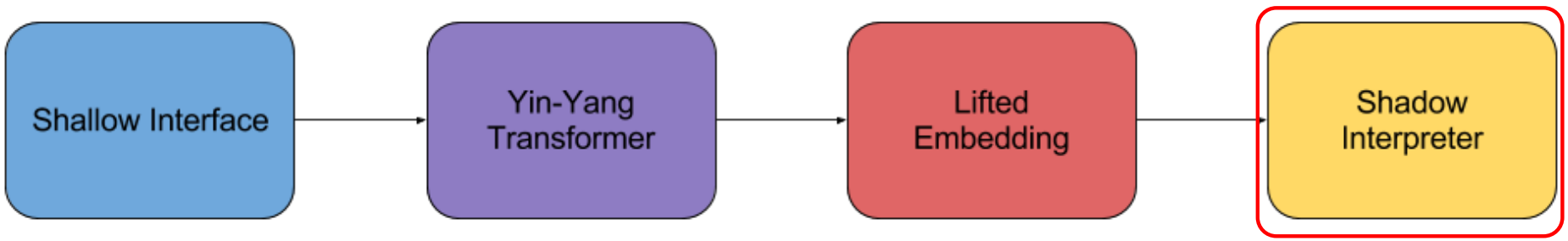
Problem 3

Problem 3 (recap)

How to have high performance?

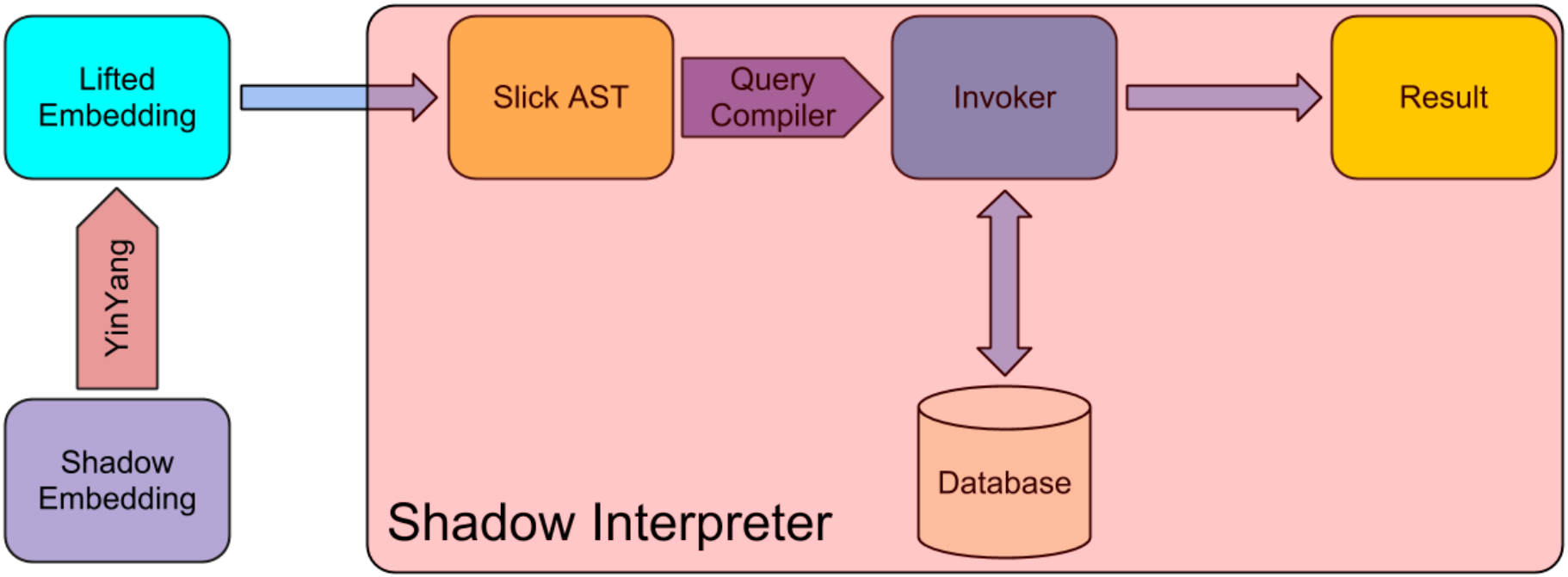
Shadow Embedding

Shadow Interpreter



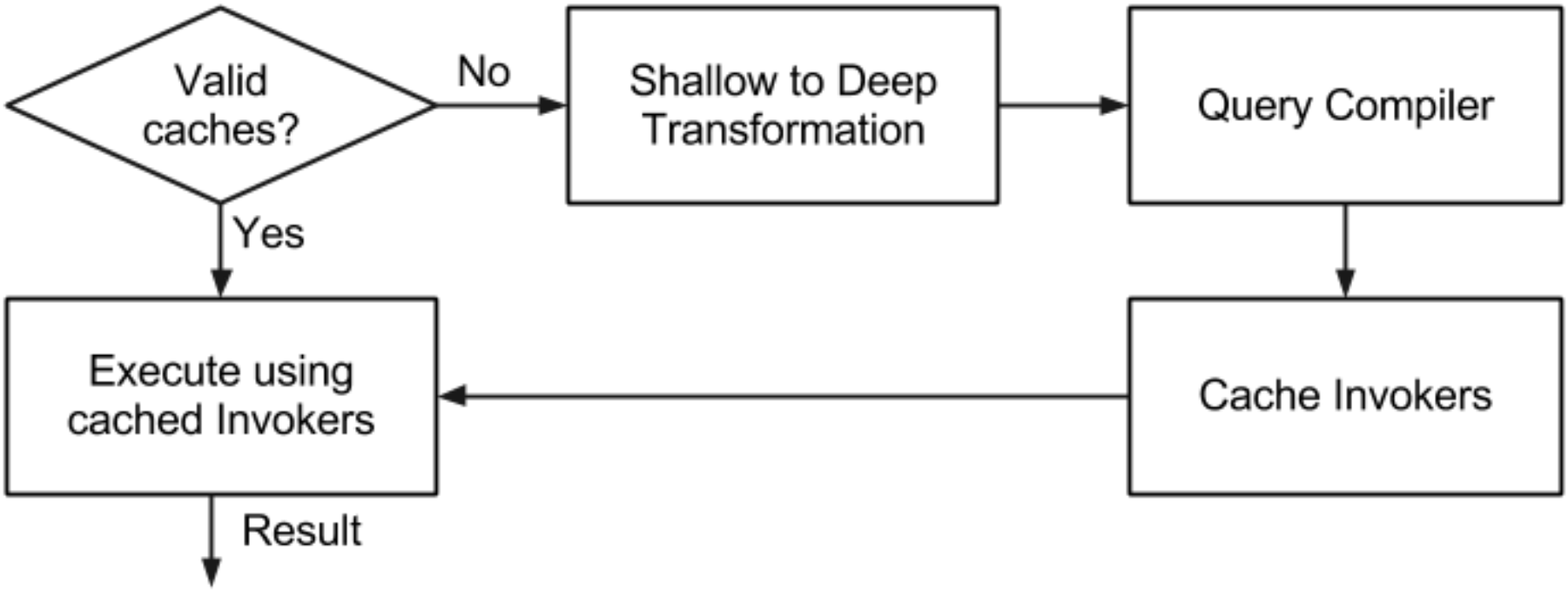
Shadow Embedding

Shadow Interpreter



Shadow Embedding

Shadow Interpreter



Shadow Embedding

Query Template

Shadow Embedding

Query Template

Shadow Embedding Query Template

```
def getCoffee(id: Int) = stage {
  for {
    c <- Query[Coffee] if c.id == id
  } yield c.name
}

getCoffee(10)
```

Shadow Embedding

Query Template

Shadow Embedding Query Template

```
def getCoffee(id: Int) = stage {
  for {
    c <- Query[Coffee] if c.id == id
  } yield c.name
}

getCoffee(10)
```

JDBC Prepared Statement

```
"select c.NAME from COFFEES c where c.ID = ?"
```

Shadow Embedding

Query Template - Shadow vs. Lifted

```
def getCoffee(id: Int) = stage {
  for {
    c <- Query[Coffee] if c.id == id
  } yield c.name
}

getCoffee(10)
```

vs.

```
val getCoffee = for {
  id <- Parameters[Int]
  c <- Query(Coffees) if c.id === id
} yield c.name

getCoffee(10)
```

Shadow Embedding

Composability

```
val query: Query[Coffee] = stage {
  Query[Coffee] filter (_.origin == "Iran")
}
```

Shadow Embedding

Composability

```
val query: Query[Coffee] = stage {
  Query[Coffee] filter (_.origin == "Iran")
}
```

```
stage {
  query map (_.name)
}
```

Shadow Embedding

Summary

Shadow Embedding

Summary

Problem 1 (recap)

How to create Lifted Embedding Table?

Shadow Embedding

Summary

Problem 1 (recap)

How to create Lifted Embedding Table?

Type Providers

Shadow Embedding

Summary

Problem 1 (recap)

How to create Lifted Embedding Table?

Type Providers

Problem 2 (recap)

How to catch the errors?

Shadow Embedding

Summary

Problem 1 (recap)

How to create Lifted Embedding Table?

Type Providers

Problem 2 (recap)

How to catch the errors?

Shallow Interface makes it comprehensible

Shadow Embedding

Summary

Problem 1 (recap)

How to create Lifted Embedding Table?

Type Providers

Problem 2 (recap)

How to catch the errors?

Shallow Interface makes it comprehensible

Yin-Yang makes it comprehensive

Shadow Embedding

Summary

Problem 1 (recap)

How to create Lifted Embedding Table?

Type Providers

Problem 2 (recap)

How to catch the errors?

Shallow Interface makes it comprehensible

Yin-Yang makes it comprehensive

Problem 3 (recap)

How to have high performance?

Shadow Embedding

Summary

Problem 1 (recap)

How to create Lifted Embedding Table?

Type Providers

Problem 2 (recap)

How to catch the errors?

Shallow Interface makes it comprehensible

Yin-Yang makes it comprehensive

Problem 3 (recap)

How to have high performance?

Shadow Interpreter reduces the user effort

Outline

Correctness

- Several basic tests

Correctness

- Several basic tests
- All Direct Embedding test suites

Correctness

- Several basic tests
- All Direct Embedding test suites
- Important Lifted Embedding test suites

Performance

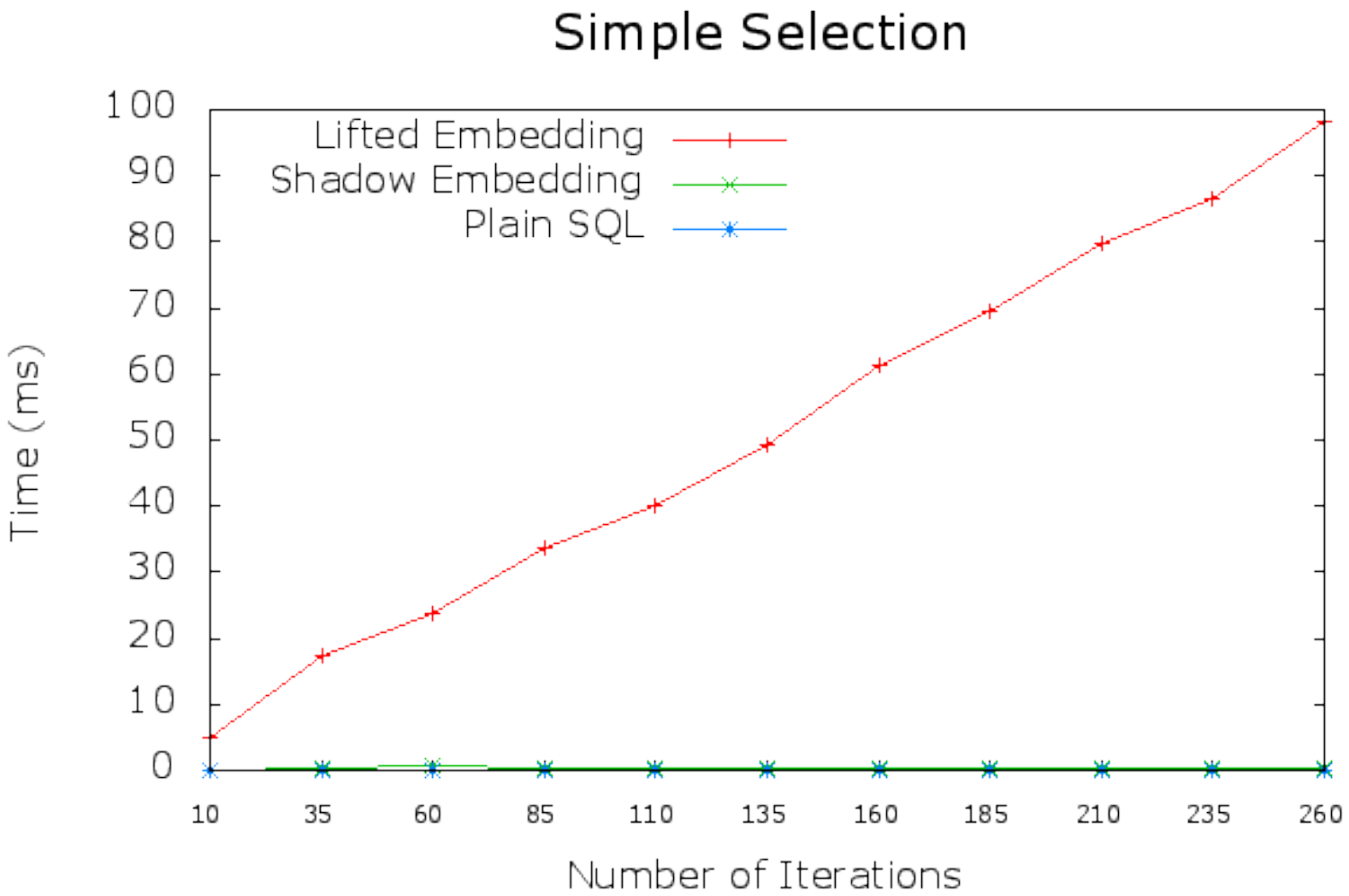
Microbenchmarking

Shadow Embedding Simple Selection

```
for (i <- range) {
  stage {
    for (c <- Query[Coffee] if c.id == 1) yield c
  }
}
```

Performance

Microbenchmarking



Performance

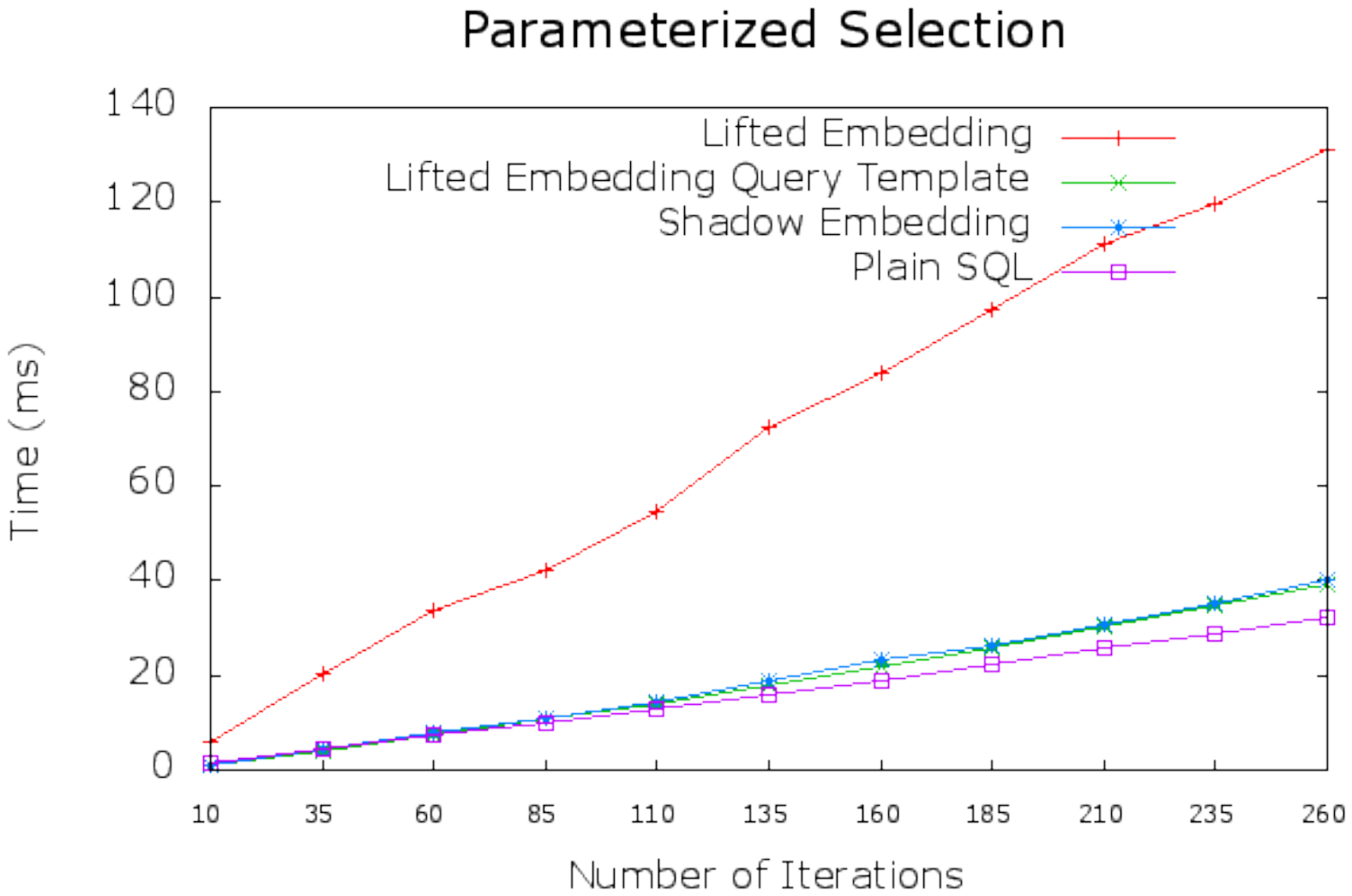
Microbenchmarking

Shadow Embedding Parameterized Selection

```
for (i <- range) {
  stage {
    for (c <- Query[Coffee] if c.id < i) yield c
  }
}
```

Performance

Microbenchmarking



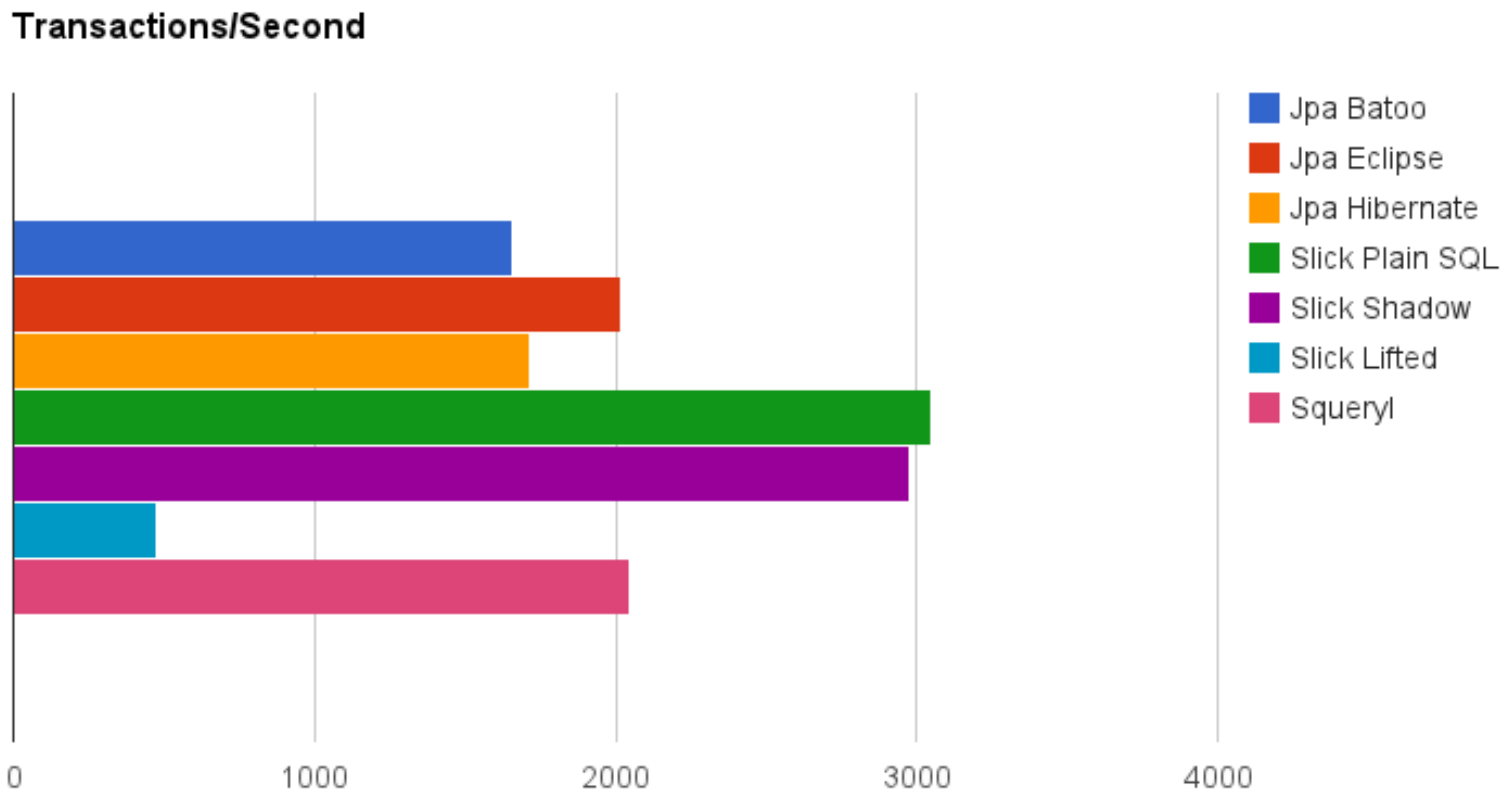
Performance

Databench

- 50,000 accounts
- 500,000 transactions
- 20% updating
- 80% reading

Performance

Databench



Conclusion

- User-friendly

Conclusion

- User-friendly
  - Shallow Interface

Conclusion

- User-friendly
  - Shallow Interface
  - Type Providers

Conclusion

- User-friendly
  - Shallow Interface
  - Type Providers
  - Composability

Conclusion

- User-friendly
  - Shallow Interface
  - Type Providers
  - Composability
  - Shadow Interpreter

Conclusion

- User-friendly
  - Shallow Interface
  - Type Providers
  - Composability
  - Shadow Interpreter
- Comprehensive and comprehensible type errors

Conclusion

- User-friendly
  - Shallow Interface
  - Type Providers
  - Composability
  - Shadow Interpreter
- Comprehensive and comprehensible type errors
  - Shallow Interface

Conclusion

- User-friendly
  - Shallow Interface
  - Type Providers
  - Composability
  - Shadow Interpreter
- Comprehensive and comprehensible type errors
  - Shallow Interface
  - Yin-Yang

Conclusion

- User-friendly
  - Shallow Interface
  - Type Providers
  - Composability
  - Shadow Interpreter
- Comprehensive and comprehensible type errors
  - Shallow Interface
  - Yin-Yang
- Highly performant

Conclusion

- User-friendly
  - Shallow Interface
  - Type Providers
  - Composability
  - Shadow Interpreter
- Comprehensive and comprehensible type errors
  - Shallow Interface
  - Yin-Yang
- Highly performant
  - Shadow Interpreter

Conclusion

- User-friendly
  - Shallow Interface
  - Type Providers
  - Composability
  - Shadow Interpreter
- Comprehensive and comprehensible type errors
  - Shallow Interface
  - Yin-Yang
- Highly performant
  - Shadow Interpreter
- Interoperable with Lifted Embedding

Conclusion

- User-friendly
  - Shallow Interface
  - Type Providers
  - Composability
  - Shadow Interpreter
- Comprehensive and comprehensible type errors
  - Shallow Interface
  - Yin-Yang
- Highly performant
  - Shadow Interpreter
- Interoperable with Lifted Embedding
  - Reusing Lifted Embedding

Conclusion

- User-friendly
  - Shallow Interface
  - Type Providers
  - Composability
  - Shadow Interpreter
- Comprehensive and comprehensible type errors
  - Shallow Interface
  - Yin-Yang
- Highly performant
  - Shadow Interpreter
- Interoperable with Lifted Embedding
  - Reusing Lifted Embedding
- Maintainable

Conclusion

- User-friendly
  - Shallow Interface
  - Type Providers
  - Composability
  - Shadow Interpreter
- Comprehensive and comprehensible type errors
  - Shallow Interface
  - Yin-Yang
- Highly performant
  - Shadow Interpreter
- Interoperable with Lifted Embedding
  - Reusing Lifted Embedding
- Maintainable
  - Reusing Lifted Embedding

Future Work

- Macro annotations

Future Work

- Macro annotations
- Shadow Programming

Future Work

- Macro annotations
- Shadow Programming
  - Yin-Yang

Future Work

- Macro annotations
- Shadow Programming
  - Yin-Yang
  - Type providers

Thank You

Thank You!