An Embedded Query Language in Scala

Amir Shaikhha

School of Computer and Communication Sciences, EPFL
Typesafe, Lausanne

Master Thesis, August 2013
Outline

1. Introduction
2. Lifted Embedding
3. Direct Embedding
4. Shadow Embedding
5. Evaluation
Introduction

Problem Statement
Write the code to access database
Instead of writing database code in SQL

```sql
select c.NAME from COFFEEES c where c.ID = 10
```
Instead of writing database code in SQL

```sql
select c.NAME from COFFEEES c where c.ID = 10
```

Write database code in Scala

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

Scala Language-Integrated Connection Kit
Slick
Architecture

- Lifted Embedding
- Slick AST
- Query Compiler
- Invoker
- Result
- Direct Embedding
- Scala AST
- Database
Outline

1. Introduction
2. Lifted Embedding
3. Direct Embedding
4. Shadow Embedding
5. Evaluation
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
Query(Coffees) filter
  (c => c.id === 10) map
  (c => c.name)
Query(Coffees) filter
(c => c.id:Rep[Int] === 10:Rep[Int]) map
(c => c.name:Rep[String])
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?
object Coffees extends Table[(Int, String, Double, String, Int)]("COFFEES") {
  def id = column[Int]("ID", 0 PRIMARY KEY)
  def name = column[String]("NAME")
  //...
}
object Coffees extends Table[(Int, String, Double, String, Int)]("COFFEES") {
    def id = column[Int]("ID", O.PrimaryKey)
    def name = column[String]("NAME")
    //...
}
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
Lifted Embedding

Type Providers

Architecture

- Database Schema
- Annotated class
- Schema Model Creation
- Schema Model
- Scala AST Generator
- Scala AST
- Type Macro
- Provided Type
- Code Generator
- Generated Scala Code
Type Providers

Architecture

- Type Macros are in macro paradise
Type Providers

Architecture

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

```scala
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
Lifted Embedding

Type Errors - Even Worse!

```scala
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

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
Caching invokers
Lifted Embedding

Performance

- Caching invokers
- Query templates
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

```scala
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
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?
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

1. Introduction
2. Lifted Embedding
3. Direct Embedding
4. Shadow Embedding
5. Evaluation
Direct Embedding

- Direct Embedding
- Scala AST

- Query Compiler
- Invoker
- Result

- Database
- Scala Macro
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

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

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

SQL Statement

```sql
select c.NAME from COFFEES c where c.ID = 10
```
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)
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
Problem 2 (recap)

How to catch the errors?
Problem 2 (recap)

How to catch the errors?
Comprehensible type errors
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

1. Introduction
2. Lifted Embedding
3. Direct Embedding
4. Shadow Embedding
5. Evaluation
Shadow Embedding

Architecture
Shadow Embedding

Architecture

shallow = shallow + deep
Shadow Embedding

Architecture
Shadow Embedding

Shallow Interface
Query interface:
Query interface:

```scala
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

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

- Shallow Interface
- Yin-Yang Transformer
- Lifted Embedding
- Shadow Interpreter
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

\[
\text{Query}(1:\text{Int}) \text{ filter} \\
((x:\text{Int}) \Rightarrow (x:\text{Int}) \_\_== (10:\text{Int}))
\]
Shadow Embedding

Yin-Yang Transformation

After Lifting

Query\(\text{lift}(1) : \text{Int})\) filter
\((x : \text{Int}) \Rightarrow (x : \text{Int}) \_\_== \text{lift}(10) : \text{Int})\)
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

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

[Diagram showing the process from Shallow DSL to Deep DSL with stages: Language Virtualization, Ascription, Lifting, Type Transformation, Scope Injection]
Shadow Embedding
Yin-Yang Transformation

Lifted Embedding Query

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

Lifted Embedding

- Shallow Interface
- Yin-Yang Transformer
- Lifted Embedding
- Shadow Interpreter
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?

```scala
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?

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

How to create Shadow Embedding Table?

Reuse Type Providers of Lifted Embedding!
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

```scala
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

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

Compile Error
value substring is not a member of `Int`
stage {
  Query[Coffee] map
    (c => c.id.toDouble)
}
Shadow Embedding
Type Errors - Good Part Again!

```scala
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!

```java
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

Shadow Embedding

Problem 3
Problem 3 (recap)

How to have high performance?
Shadow Embedding

Shadow Interpreter
Shadow Embedding

Shadow Interpreter
Shadow Embedding
Shadow Interpreter

- Valid caches?
  - Yes: Execute using cached Invokers
  - No: Shallow to Deep Transformation
    - Query Compiler
    - Cache Invokers

Result
Shadow Embedding

Query Template

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

cache {getCoffee (10)}
```

JDBC Prepared Statement

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

```scala
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

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

getCoffee(10)
```

vs.

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

getCoffee(10)
```
val query: Query[Coffee] = stage {
    Query[Coffee] filter (_.origin == "Iran")
}
Shadow Embedding

Composability

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

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

Summary
Problem 1 (recap)

How to create Lifted Embedding Table?
**Problem 1 (recap)**

How to create Lifted Embedding Table?

**Type Providers**
## Shadow Embedding

### Summary

<table>
<thead>
<tr>
<th>Problem 1 (recap)</th>
<th>How to create Lifted Embedding Table?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Type Providers</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem 2 (recap)</th>
<th>How to catch the errors?</th>
</tr>
</thead>
</table>

Shallow Interface makes it comprehensible

Yin-Yang makes it comprehensive

Problem 3 (recap)

How to have high performance?

Shadow Interpreter reduces the user effort
## 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

1. Introduction
2. Lifted Embedding
3. Direct Embedding
4. Shadow Embedding
5. Evaluation
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

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

Microbenchmarking

Simple Selection

- Lifted Embedding
- Shadow Embedding
- Plain SQL

Time (ms)

Number of Iterations
Shadow Embedding Parameterized Selection

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

Performance

Microbenchmarking

Parameterized Selection

Time (ms)

Number of Iterations

Lifted Embedding
Lifted Embedding Query Template
Shadow Embedding
Plain SQL
Evaluation

Performance
Databench

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

Transactions/Second

- Jpa Batoo
- Jpa Eclipse
- Jpa Hibernate
- Slick Plain SQL
- Slick Shadow
- Slick Lifted
- Squeryl
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!