Scaling Scala to the Database

Jan Christopher Vogt, EPFL
Stefan Zeiger, Typesafe
Chapter 1: Overview / Key Concepts
WE WRITE SQL SO YOU DON'T HAVE TO
Write database code in Scala

• Instead of SQL, JPQL, Criteria API, etc.

```scala
for { p <- persons } yield p.name
```

```sql
select p.NAME from PERSON p
```
(for {
    p <- persons.filter(_.age < 20) ++
    persons.filter(_.age >= 50)
    if p.name.startsWith("A")
} yield p).groupBy(_.age).map { case (age, ps) =>
    (age, ps.length)
}

select x2.x3, count(1) from ( 
    select * from ( 
        select x4."NAME" as x5, x4."AGE" as x3 
        from "PERSON" x4 where x4."AGE" < 20 
        union all select x6."NAME" as x5, x6."AGE" as x3 
        from "PERSON" x6 where x6."AGE" >= 50 
        ) x7 where x7.x5 like 'A%' escape '^'
    ) x2 group by x2.x3
• Database query and access library for Scala
• Successor of ScalaQuery
• Developed at Typesafe and EPFL
• Open Source
Supported Databases

- PostgreSQL
- MySQL
- H2
- Hsqldb
- Derby / JavaDB
- SQLite
- Access

Closed-Source *Slick Extensions* (with commercial support by Typesafe):

- Oracle
- DB/2
- SQL Server
Components

• Lifted Embedding
• Direct Embedding
• Plain SQL
• Session Management
• Schema Model
Compared to ORMs
Impedance Mismatch: Retrieval

**Colombian**
French_Roast
Espresso
Colombian_Decaf
French_Roast_Decaf

**Espresso**
Price: 9.99
Supplier: The High Ground

```sql
select COF_NAME
from COFFEES
```

```sql
select c.*, s.SUP_NAME
from COFFEES c, SUPPLIERS s
where c.COF_NAME = ?
and c.SUP_ID = s.SUP_ID
```
```scala
def getAllCoffees(): Seq[Coffee] = ...

def printLinks(s: Seq[Coffee]) {
  for(c <- s) println(c.name + " " + c.price)
}

def printDetails(c: Coffee) {
  println(c.name)
  println("Price: " + c.price)
  println("Supplier: " + c.supplier.name)
}
```

- **Coffee**
  - Colombian
  - French_Roast
  - Espresso
  - Colombian_Decaf
  - French_Roast_Decaf

- **Espresso**
  - Price: 9.99
  - Supplier: The High Ground
O/R Mapper

• Mapping low-level programming (OOP) to high-level concepts (relational algebra)

• Not transparent (but pretends to be)
Better Match: Functional Programming

- **Relation**
  - `case class` Coffee(name: String, supplierId: Int, price: Double)

- **Attribute**
  - `val` coffees = Set(
    Coffee("Colombian", 101, 7.99),
    Coffee("Espresso", 150, 9.99),
    Coffee("French_Roast", 49, 8.99),
    
  )

- **Tuple**

- **Relation Value**

- **Relation Variable** - mutable state in the DB
Functional-Relational Mapping

- Embraces the relational model
- No impedance mismatch
- Composable Queries
- Explicit control over statement execution
- Stateless
3

Live Coding Demo
4

Under The Hood
APIs

Lifted Embedding

Slick AST

Scala AST

Direct Embedding

Scala Compiler

DB

Result

Slick Macros

Query Compiler

Executor
val q = for {
  c <- coffees if c.price < 9.0
  s <- c.supplier
} yield (c.name, s.name)

val result = q.run
val q = for {
  c <- coffees if c.price < 9.0
  s <- c.supplier
} yield (c.name, s.name)

val result = q.run
Query Compiler

• Immutable ASTs
  – Types can be mutated until they are observed
• Immutable compiler state
  – containing AST + phase output state
• Phases transform compiler state
  – using mutable state locally
• Drivers provide their own compilers
Compiler Phases: SQL

Clean Up
- inline
- assignUniqueSymbols
- expandTables
- inferTypes
- createResultSetMapping
- forceOuterBinds

Flatten Columns
- expandRefs
- replaceFieldSymbols
- rewritePaths
- relabelUnions
- pruneFields
- assignTypes

SQL Shape
- resolveZipJoins
- convertToComprehensions
- fuseComprehensions
- fixRowNumberOrdering
- hoistClientOps

Generate Code
- codeGen
  (driver-specific)
Compiler Phases: MemoryDriver

Clean Up
- inline
- assignUniqueSymbols
- expandTables
- inferTypes
- createResultSetMapping
- forceOuterBinds

Flatten Columns
- expandRefs
- replaceFieldSymbols
- rewritePaths
- relabelUnions
- pruneFields
- assignTypes

Prepare for Interpreter
- codeGen
Compiler Phases: Scheduling

**Clean Up**
- inline
- assignUniqueSymbols

**Flatten Columns**
- expandRefs
- replaceFieldSymbols
- rewritePaths
- relabelUnions
- pruneFields
- assignTypes

**Distribute**
- distribute (to other drivers' compilers)

**e.g. H2**
- Query Compiler

**MySQL**
- Query Compiler

**...**
- Query Compiler

**Clean Up II**
- expandTables
- inferTypes
- createResultSetMapping
- forceOuterBinds

**Prepare for Interpreter**
- codeGen
5 Outlook
Outlook

- NoSQL support
- Other data sources
- Async / Reactive API