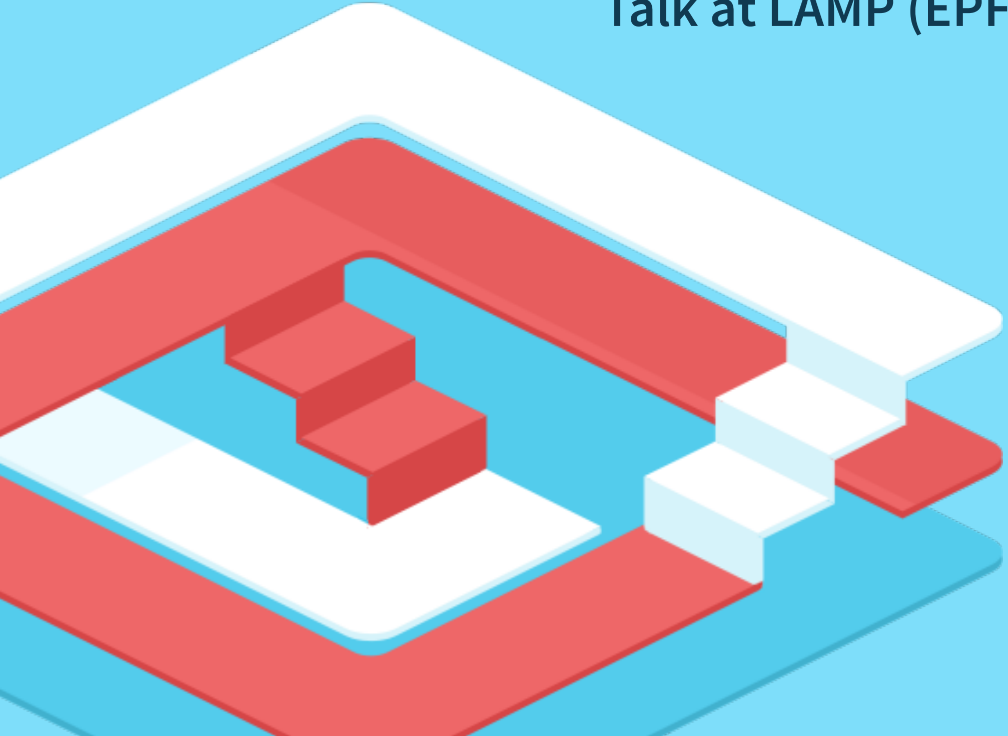
Slick Query Compiler

Stefan Zeiger, Typesafe

Talk at LAMP (EPFL) 2014-03-04



Overview

Slick

Scala Language Integrated Connection Kit

- Database query and access library for Scala
- Successor of ScalaQuery
- Developed at Typesafe and EPFL
- Open Source

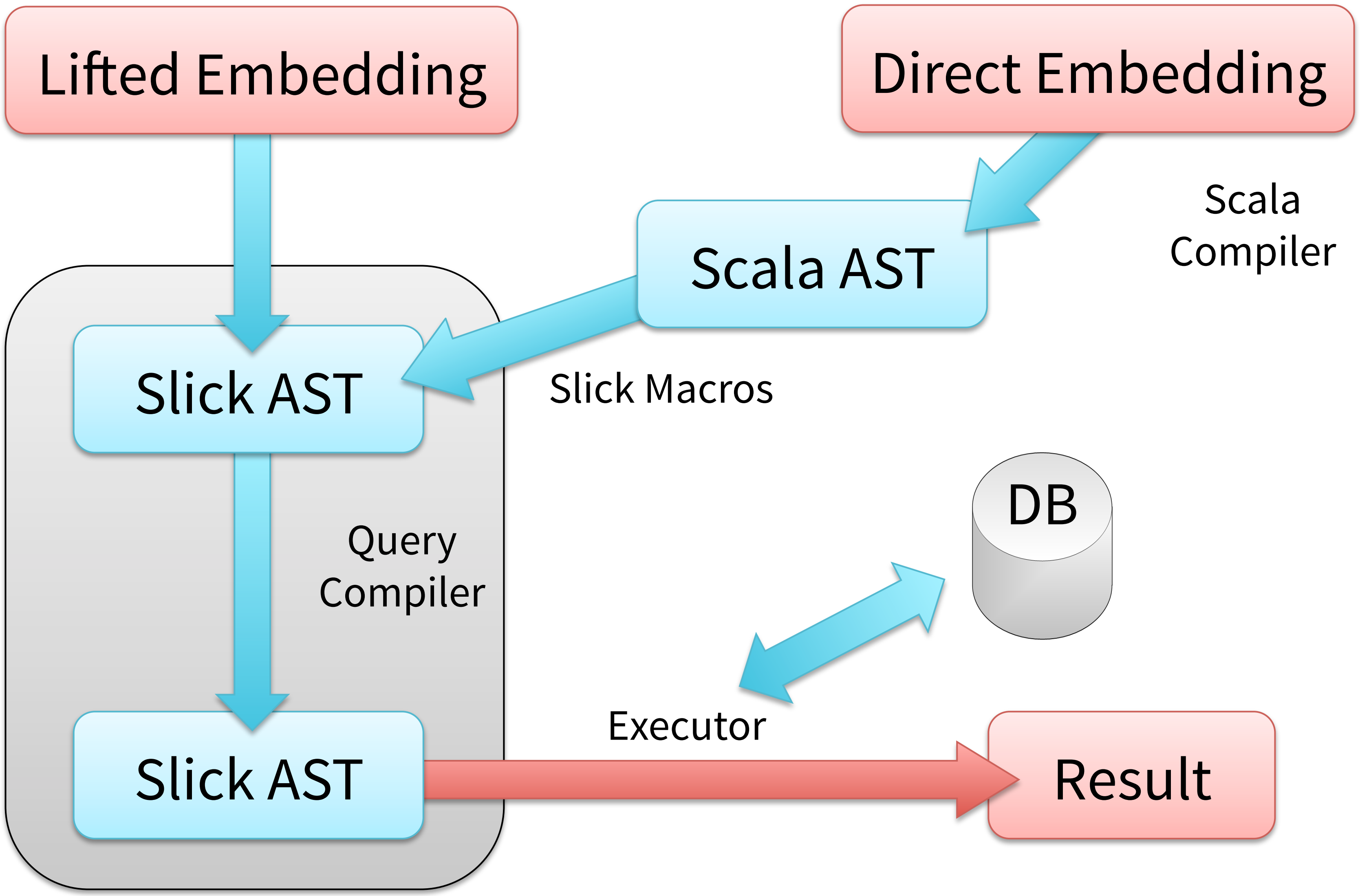
Write database code in Scala

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

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

```
select p.NAME from PERSON p
```

Slick APIs


Lifted Embedding
Direct Embedding
Scala Compiler
Scala AST
Slick Macros
Slick AST
Query Compiler
Slick AST
DB
Executor
Result

Overview

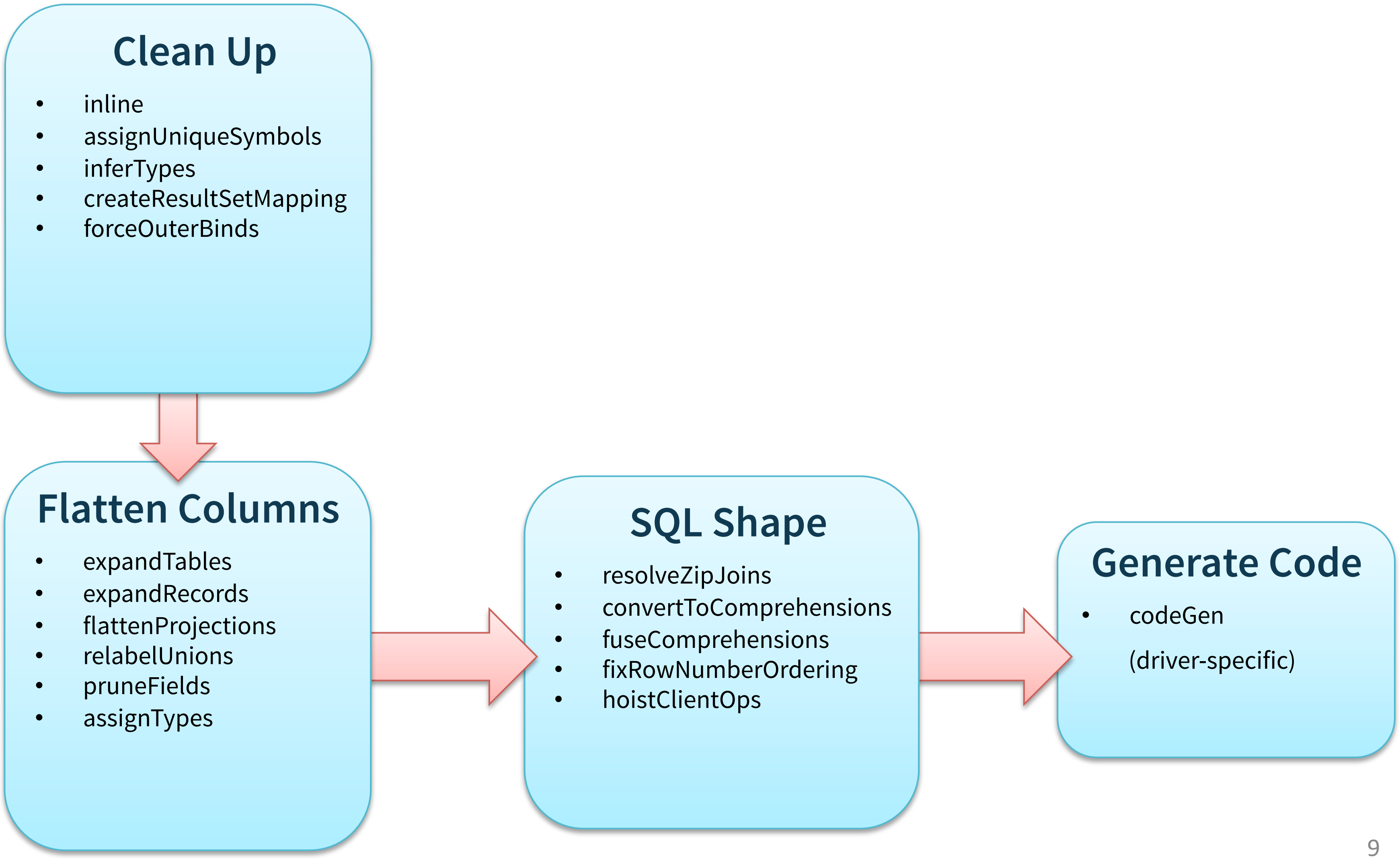
- Compiler Architecture
- Types
- Trees
- Symbols
- Scopes
- Compiler Phases

Compiler Architecture

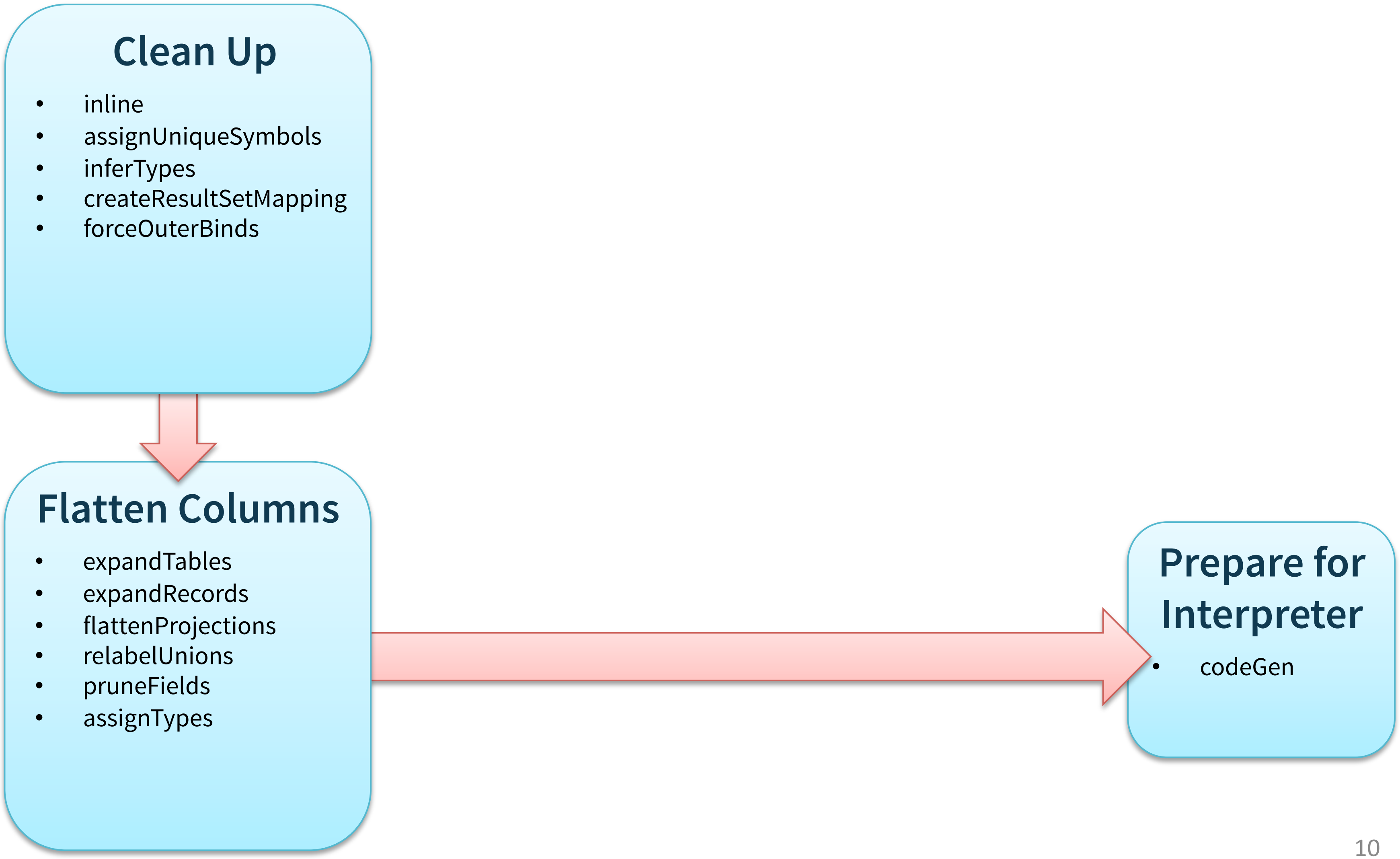
Compiler Architecture

- 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



Compiler Phases: MemoryDriver



Clean Up

- inline
- assignUniqueSymbols
- inferTypes
- createResultSetMapping
- forceOuterBinds

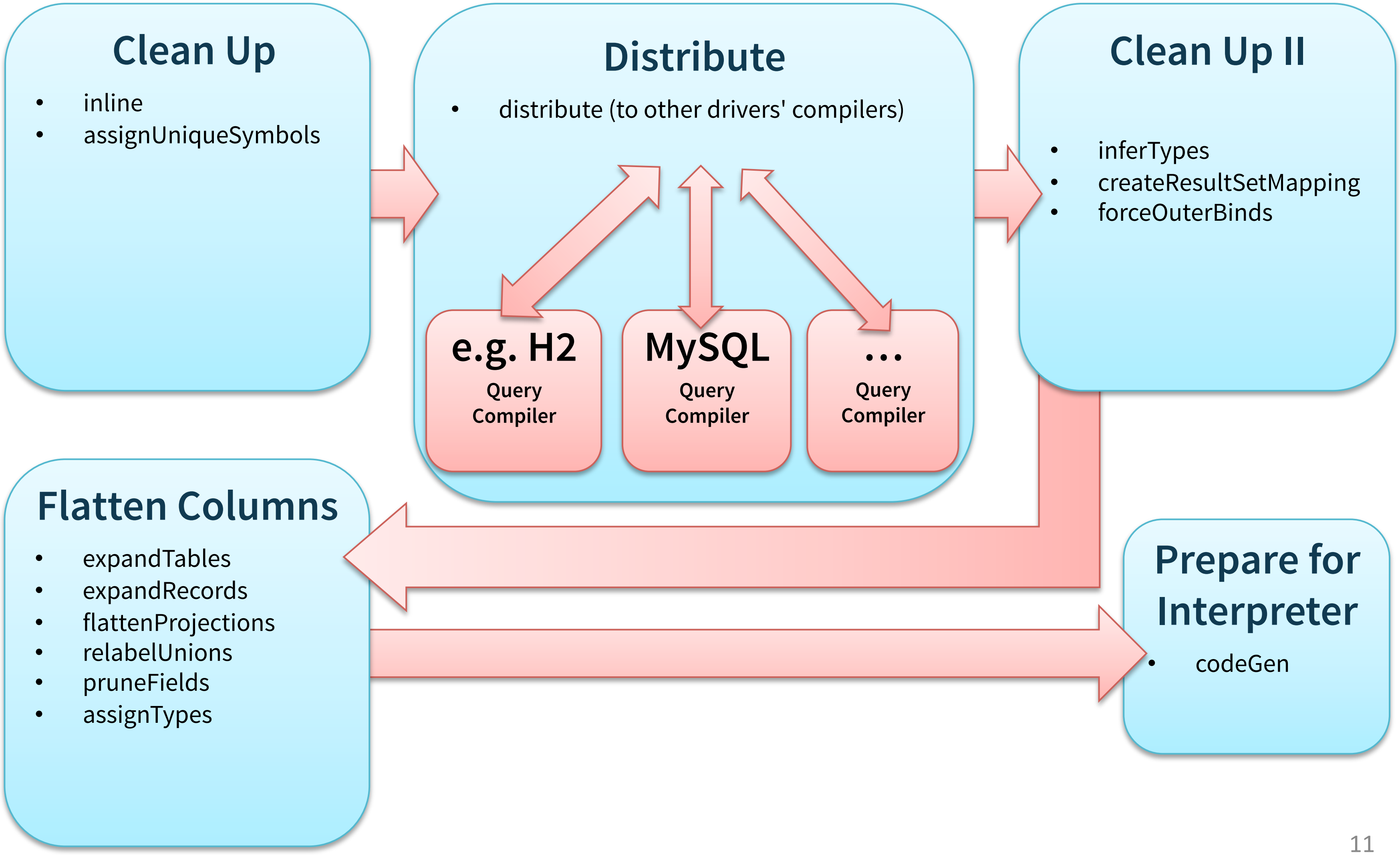
Flatten Columns

- expandTables
- expandRecords
- flattenProjections
- relabelUnions
- pruneFields
- assignTypes

Prepare for Interpreter

- codeGen

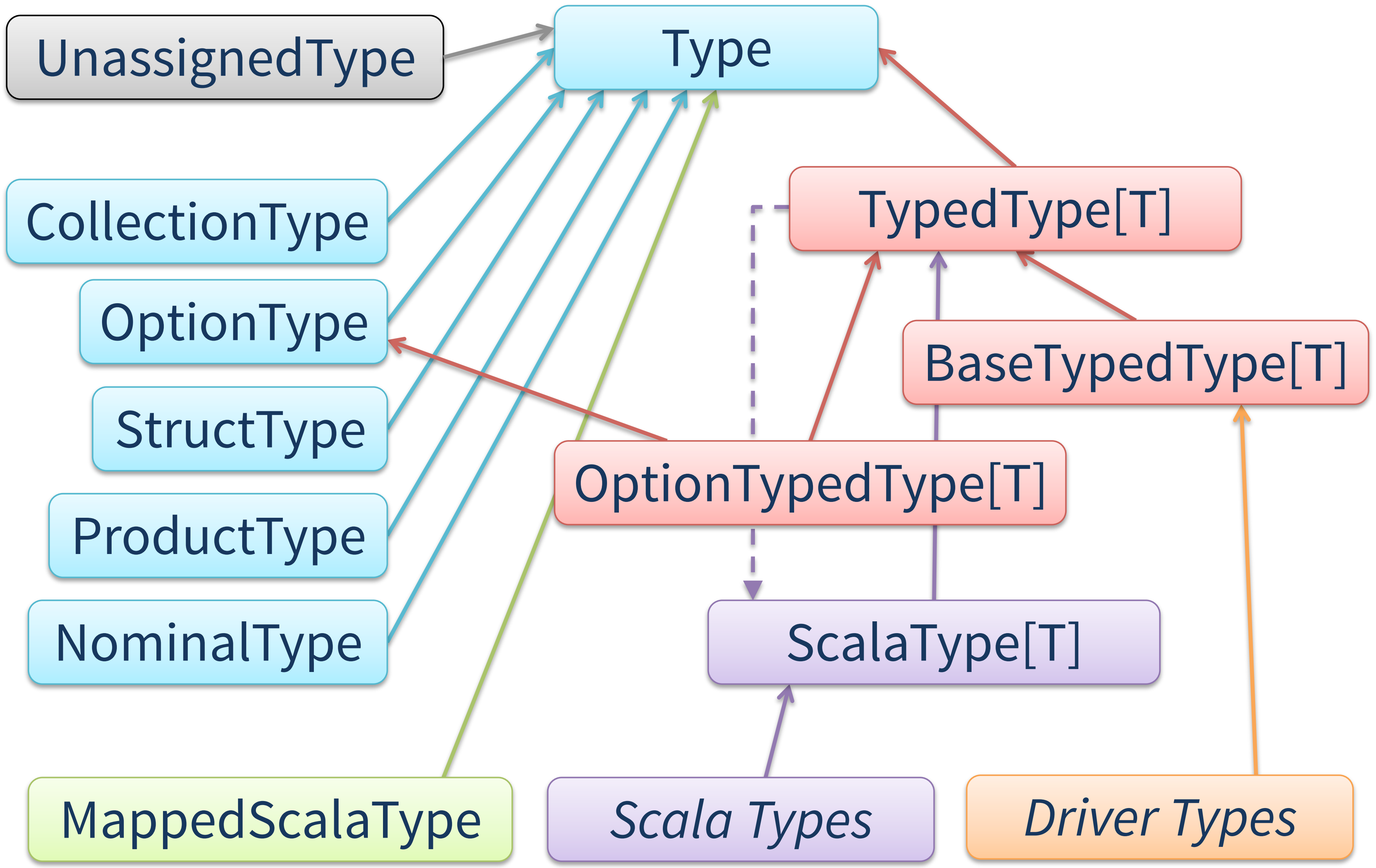
Compiler Phases: Scheduling



Types

Types

```
/** Super-trait for all types */
trait Type {
  /** All children of this Type. */
  def children: Seq[Type]
  /** Apply a transformation to all type children and
      reconstruct this type with the new children, or
      return the original object if no child is changed. */
  def mapChildren(f: Type => Type): Type
  def select(sym: Symbol): Type =
    throw new SlickException("No type for symbol "+sym+
      " found in "+this)
  /** The structural view of this type */
  def structural: Type = this
}
```


UnassignedType
Type
CollectionType
OptionType
StructType
ProductType
NominalType
TypedType[T]
BaseTypedType[T]
OptionTypedType[T]
ScalaType[T]
MappedScalaType
Scala Types
Driver Types

Trees

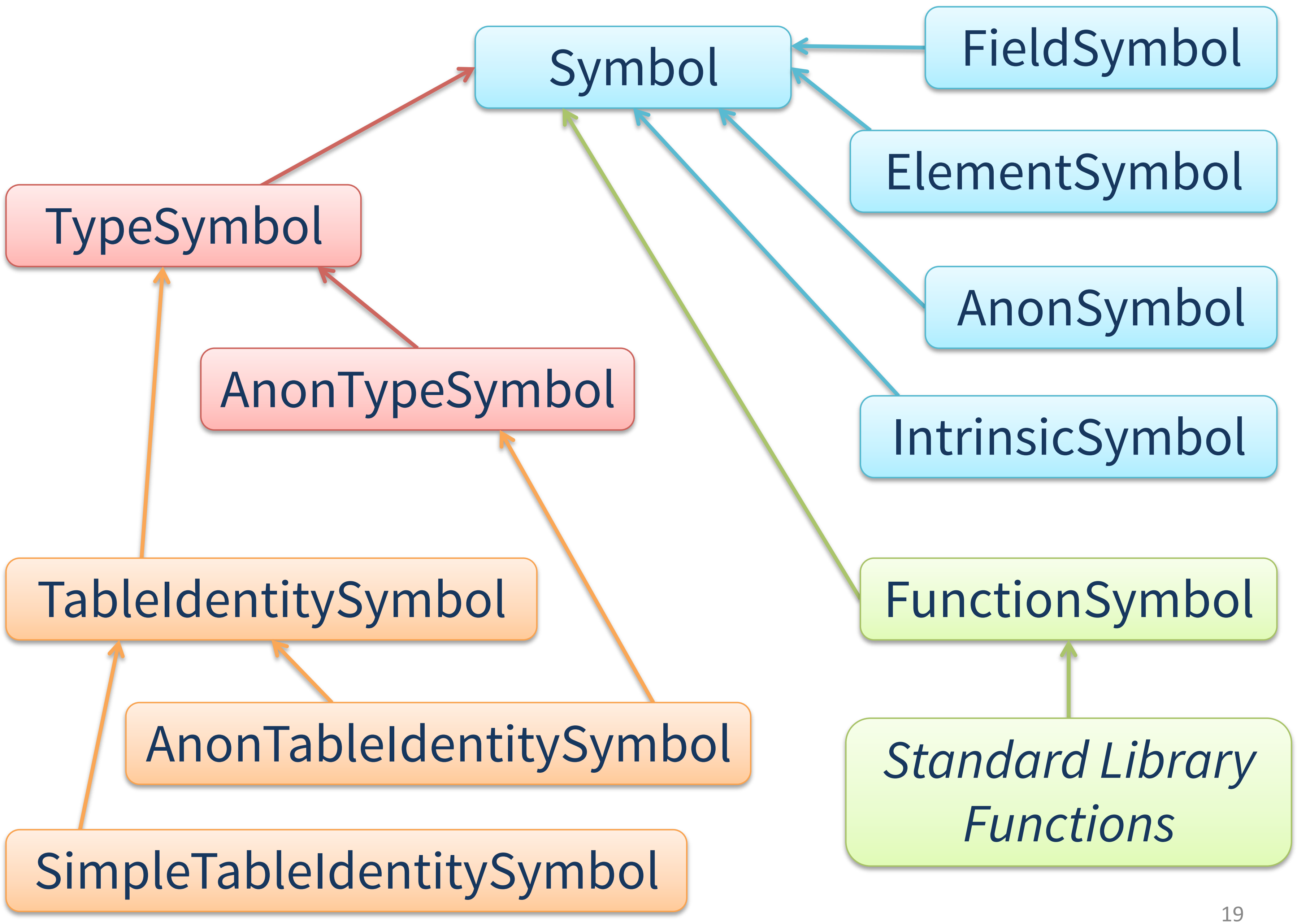
Trees

```
trait Node {
  type Self >: this.type <: Node
  def nodeChildren: Seq[Node]
  protected[this] def nodeRebuild(ch: IndexedSeq[Node]): Self

  final def nodeMapChildren(f: Node => Node,
    keepType: Boolean = false): Self = ...

  def nodeType: Type = ...
  def nodePeekType: Type = ...
  def nodeHasType: Boolean = ...
  final def nodeTypedOrCopy(tpe: Type): Self = ...

  final def nodeWithComputedType(
    scope: SymbolScope = SymbolScope.empty,
    typeChildren: Boolean = false,
    retype: Boolean = false): Self = ...

  ...
}
```

Symbols

Symbols

```
/** A symbol which can be used in the AST. */
trait Symbol {
  def name: String
  override def toString = SymbolNamer(this)
}
```


Symbol
FieldSymbol
ElementSymbol
TypeSymbol
AnonSymbol
AnonTypeSymbol
IntrinsicSymbol
TableIdentitySymbol
FunctionSymbol
AnonTableIdentitySymbol
Standard Library Functions
SimpleTableIdentitySymbol

Scopes

Scopes

```
trait SymbolScope {
  def + (entry: (Symbol, Type)): SymbolScope
  def get(sym: Symbol): Option[Type]
  def withDefault(f: (Symbol => Type)): SymbolScope
}
```

- Mainly for typing
- Most transformations don't need scopes

Compiler Phases

Example

TableQuery

```
val qa = for {
      c <- coffees.take(3)
   } yield (c.supID, (c.name, 42))
```

Source AST

```
*** (s.slick.compiler.QueryCompiler) Source:
  Bind
    from s2: Take 3
      from: TableExpansion
        table s3: Table COFFEES : Coll[NominalType(@(H2Driver._.COFFEES))
(UnassignedStructuralType(@(H2Driver._.COFFEES)))]
        columns: ProductNode
          1: Path s3.COF_NAME : String/VARCHAR
          2: Path s3.SUP_ID : Int/INTEGER
          3: Path s3.PRICE : Int/INTEGER
          4: Path s3.SALES : Int/INTEGER
          5: Apply Function * : Int/INTEGER
            0: Path s3.TOTAL : Int/INTEGER
            1: LiteralNode 10 (volatileHint=false) : Int/INTEGER
    select: Pure t4
      value: ProductNode
        1: Path s2.SUP_ID : Int/INTEGER
        2: ProductNode
          1: Path s2.COF_NAME : String/VARCHAR
          2: LiteralNode 42 (volatileHint=false) : Int/INTEGER
```

Unique Symbols

```
*** (s.slick.compiler.QueryCompiler) After phase inline: (no change)
*** (s.slick.compiler.QueryCompiler) After phase assignUniqueSymbols:
  Bind
    from s2: Take 3
      from: TableExpansion
        table s3: Table COFFEES : Coll[NominalType(@(t5))(UnassignedStructural
Type(@(t5)))]
        columns: ProductNode
          1: Path s3.COF_NAME : String/VARCHAR
          2: Path s3.SUP_ID : Int/INTEGER
          3: Path s3.PRICE : Int/INTEGER
          4: Path s3.SALES : Int/INTEGER
          5: Apply Function * : Int/INTEGER
            0: Path s3.TOTAL : Int/INTEGER
            1: LiteralNode 10 (volatileHint=false) : Int/INTEGER
    select: Pure t4
      value: ProductNode
        1: Path s2.SUP_ID : Int/INTEGER
        2: ProductNode
          1: Path s2.COF_NAME : String/VARCHAR
          2: LiteralNode 42 (volatileHint=false) : Int/INTEGER
```

After Type Inference

```
*** (s.slick.compiler.QueryCompiler) After phase inferTypes:
  Bind : Coll[NominalType(t4)((Int/INTEGER, (String/VARCHAR, Int/INTEGER)))]
    from s2: Take 3 : Coll[NominalType(@(t5))({TOTAL: Int/INTEGER, COF_NAME: String/VARCHAR, PRICE:
Int/INTEGER, SUP_ID: Int/INTEGER, SALES: Int/INTEGER})]
      from: TableExpansion : Coll[NominalType(@(t5))({TOTAL: Int/INTEGER, COF_NAME: String/VARCHAR,
PRICE: Int/INTEGER, SUP_ID: Int/INTEGER, SALES: Int/INTEGER})]
        table s3: Table COFFEES : Coll[NominalType(@(t5))({TOTAL: Int/INTEGER, COF_NAME: String/VARC
HAR, PRICE: Int/INTEGER, SUP_ID: Int/INTEGER, SALES: Int/INTEGER})]
        columns: ProductNode : (String/VARCHAR, Int/INTEGER, Int/INTEGER, Int/INTEGER, Int/INTEGER)
          1: Path s3.COF_NAME : String/VARCHAR
          2: Path s3.SUP_ID : Int/INTEGER
          3: Path s3.PRICE : Int/INTEGER
          4: Path s3.SALES : Int/INTEGER
          5: Apply Function * : Int/INTEGER
            0: Path s3.TOTAL : Int/INTEGER
            1: LiteralNode 10 (volatileHint=false) : Int/INTEGER
    select: Pure t4 : Coll[NominalType(t4)((Int/INTEGER, (String/VARCHAR, Int/INTEGER)))]
      value: ProductNode : (Int/INTEGER, (String/VARCHAR, Int/INTEGER))
        1: Path s2.SUP_ID : Int/INTEGER
        2: ProductNode : (String/VARCHAR, Int/INTEGER)
          1: Path s2.COF_NAME : String/VARCHAR
          2: LiteralNode 42 (volatileHint=false) : Int/INTEGER
```

inferTypes implementation

```scala
class InferTypes extends Phase {
  val name = "inferTypes"

  def apply(state: CompilerState) = state.map { tree =>
    val tree2 =
      tree.nodeWithComputedType(new DefaultSymbolScope(Map.empty), true, false)
    val structs = tree2.collect[(TypeSymbol, (Symbol, Type))] {
      case s @ Select(_ :@ (n: NominalType), sym) =>
        n.sourceNominalType.sym -> (sym -> s.nodeType)
    }.groupBy(_._1).mapValues(v => StructType(v.map(_._2).toMap.toIndexedSeq))
    logger.debug("Found Selects: "+structs.keySet.mkString(", "))
    def tr(n: Node): Node =
      n.nodeMapChildren(tr, keepType = true).nodeTypedOrCopy(n.nodeType.replace {
        case UnassignedStructuralType(tsym) if structs.contains(tsym) =>
          structs(tsym)
      })
    tr(tree2)
  }
}
```

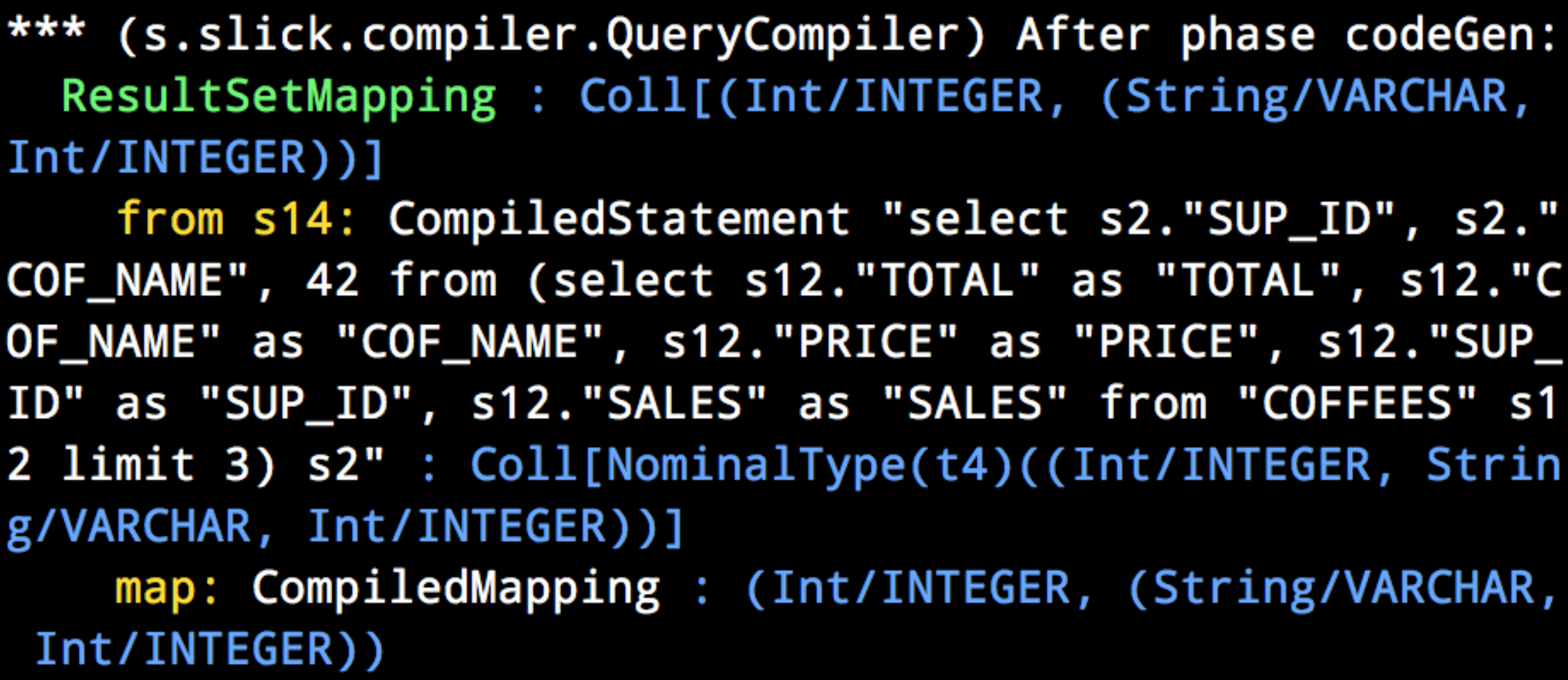
After Flattening

```
*** (s.slick.compiler.QueryCompiler) After phase assignTypes:
  ResultSetMapping : Coll[(Int/INTEGER, (String/VARCHAR, Int/INTEGER))]
    from s6: Bind : Coll[NominalType(t4)((Int/INTEGER, String/VARCHAR, Int/INTEGER
))]
      from s2: Take 3 : Coll[NominalType(@(t5))({TOTAL: Int/INTEGER, COF_NAME: Str
ing/VARCHAR, PRICE: Int/INTEGER, SUP_ID: Int/INTEGER, SALES: Int/INTEGER})]
        from: Table COFFEES : Coll[NominalType(@(t5))({TOTAL: Int/INTEGER, COF_NAM
E: String/VARCHAR, PRICE: Int/INTEGER, SUP_ID: Int/INTEGER, SALES: Int/INTEGER})]
      select: Pure t4 : Coll[NominalType(t4)((Int/INTEGER, String/VARCHAR, Int/INT
EGER))]
        value: ProductNode : (Int/INTEGER, String/VARCHAR, Int/INTEGER)
          1: Path s2.SUP_ID : Int/INTEGER
          2: Path s2.COF_NAME : String/VARCHAR
          3: LiteralNode 42 (volatileHint=false) : Int/INTEGER
    map: ProductNode : (Int/INTEGER, (String/VARCHAR, Int/INTEGER))
      1: Path s6._1 : Int/INTEGER
      2: ProductNode : (String/VARCHAR, Int/INTEGER)
        1: Path s6._2 : String/VARCHAR
        2: Path s6._3 : Int/INTEGER
```

SQL Form

```
*** (s.slick.compiler.QueryCompiler) After phase hoistClientOps:
 ResultSetMapping : Coll[(Int/INTEGER, (String/VARCHAR, Int/INTEGER))]
  from s14: Comprehension(fetch = None, offset = None) : Coll[NominalType(t4)((Int/INTEGER, Strin
g/VARCHAR, Int/INTEGER))]
   from s2: Comprehension(fetch = Some(3), offset = None) : Coll[NominalType(t13)({TOTAL: Int/IN
TEGER, COF_NAME: String/VARCHAR, PRICE: Int/INTEGER, SUP_ID: Int/INTEGER, SALES: Int/INTEGER})]
    from s12: Table COFFEES : Coll[NominalType(@(t5))({TOTAL: Int/INTEGER, COF_NAME: String/VAR
CHAR, PRICE: Int/INTEGER, SUP_ID: Int/INTEGER, SALES: Int/INTEGER})]
    select: Pure t13 : Coll[NominalType(t13)({TOTAL: Int/INTEGER, COF_NAME: String/VARCHAR, PRI
CE: Int/INTEGER, SUP_ID: Int/INTEGER, SALES: Int/INTEGER})]
     value: StructNode : {TOTAL: Int/INTEGER, COF_NAME: String/VARCHAR, PRICE: Int/INTEGER, SU
P_ID: Int/INTEGER, SALES: Int/INTEGER}
      TOTAL: Path s12.TOTAL : Int/INTEGER
      COF_NAME: Path s12.COF_NAME : String/VARCHAR
      PRICE: Path s12.PRICE : Int/INTEGER
      SUP_ID: Path s12.SUP_ID : Int/INTEGER
      SALES: Path s12.SALES : Int/INTEGER
   select: Pure t4 : Coll[NominalType(t4)((Int/INTEGER, String/VARCHAR, Int/INTEGER))]
    value: ProductNode : (Int/INTEGER, String/VARCHAR, Int/INTEGER)
     1: Path s2.SUP_ID : Int/INTEGER
     2: Path s2.COF_NAME : String/VARCHAR
     3: LiteralNode 42 (volatileHint=false) : Int/INTEGER
  map: ProductNode : (Int/INTEGER, (String/VARCHAR, Int/INTEGER))
   1: Path s14._1 : Int/INTEGER
   2: ProductNode : (String/VARCHAR, Int/INTEGER)
    1: Path s14._2 : String/VARCHAR
    2: Path s14._3 : Int/INTEGER
```

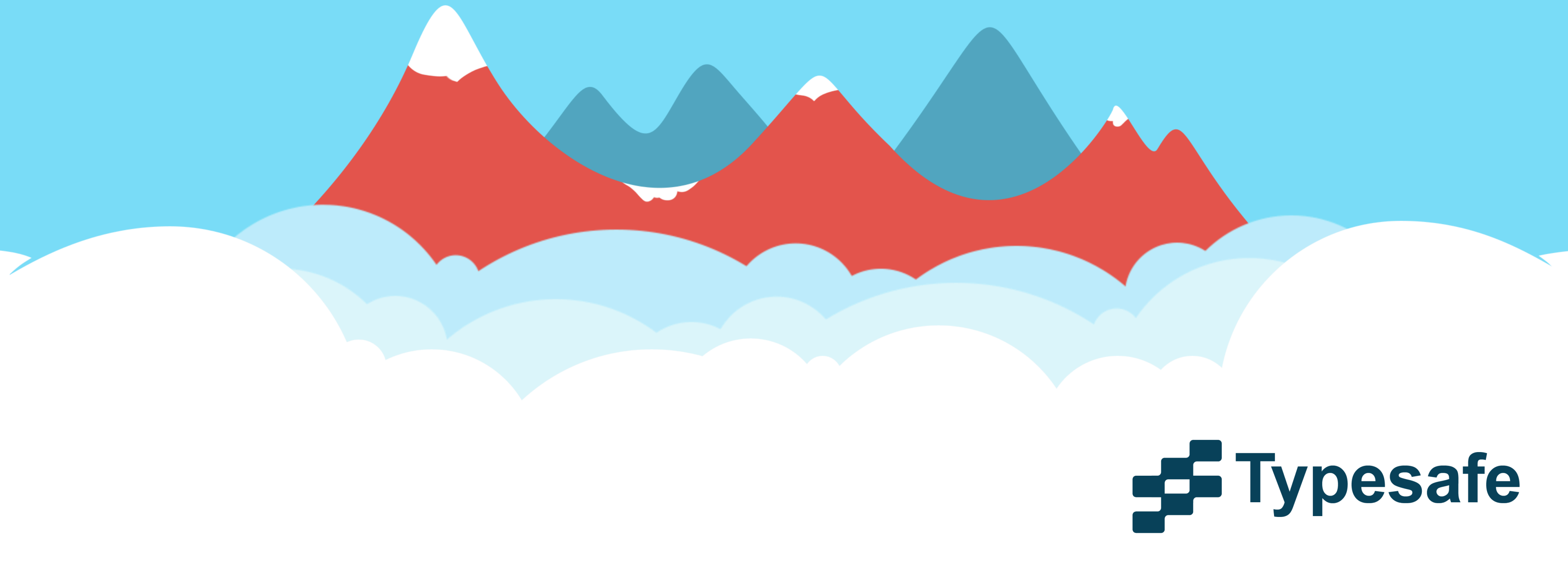
After Code Generator



```
*** (s.slick.compiler.QueryCompiler) After phase codeGen:
  ResultSetMapping : Coll[(Int/INTEGER, (String/VARCHAR,
Int/INTEGER))]
    from s14: CompiledStatement "select s2."SUP_ID", s2."
COF_NAME", 42 from (select s12."TOTAL" as "TOTAL", s12."C
OF_NAME" as "COF_NAME", s12."PRICE" as "PRICE", s12."SUP_
ID" as "SUP_ID", s12."SALES" as "SALES" from "COFFEES" s1
2 limit 3) s2" : Coll[NominalType(t4)((Int/INTEGER, Strin
g/VARCHAR, Int/INTEGER))]
    map: CompiledMapping : (Int/INTEGER, (String/VARCHAR,
 Int/INTEGER))
```

slick.typesafe.com

@StefanZeiger



Typesafe