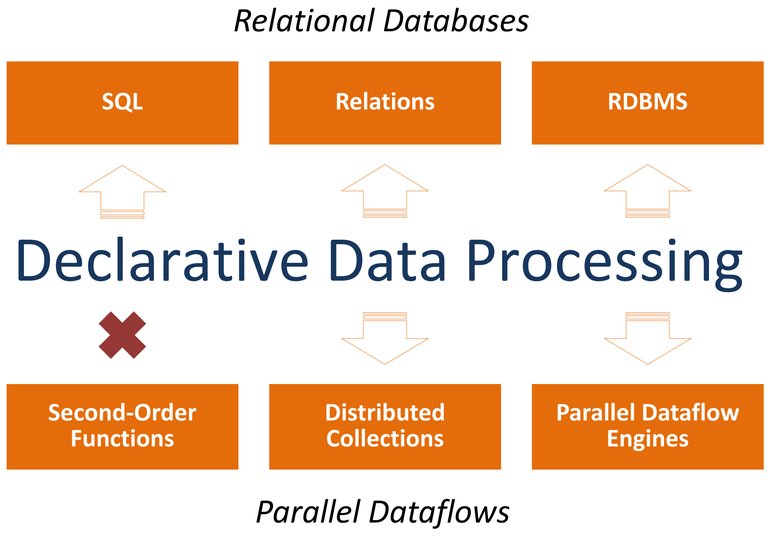
Emma in Action: Declarative Dataflows for Scalable Data Analysis

A. Alexandrov, A. Salzmann, G. Krastev, A. Katsifodimos, V. Markl
firstname.lastname@tu-berlin.de

ACM SIGMOD 2015 Research Highlights Award

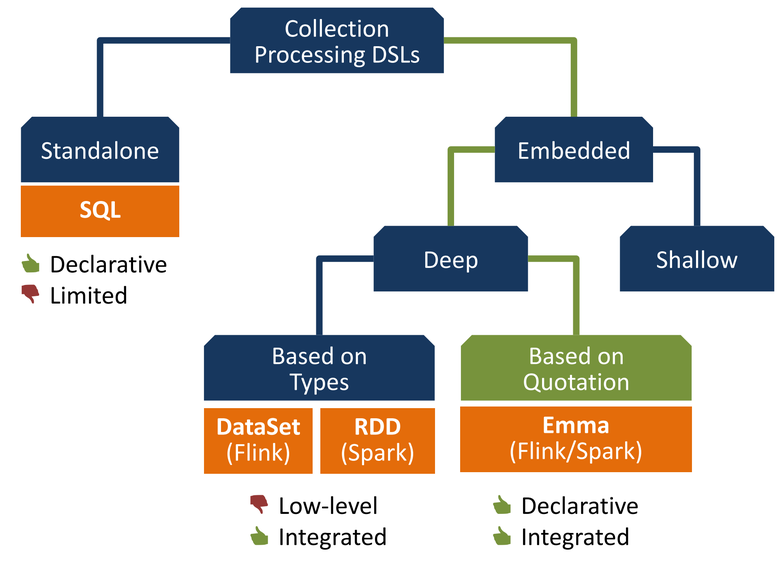
Context



SQL is **declarative**, but is designed for querying data. Advanced dataflows characterized by heavy use of library methods, control flow, and nesting stretch its limits.

Embedded dataflow DSLs overcome these problems, but are **too low-level**. Runtime aspects like caching, partitioning, and aggregation need to be hard-coded by the programmer.

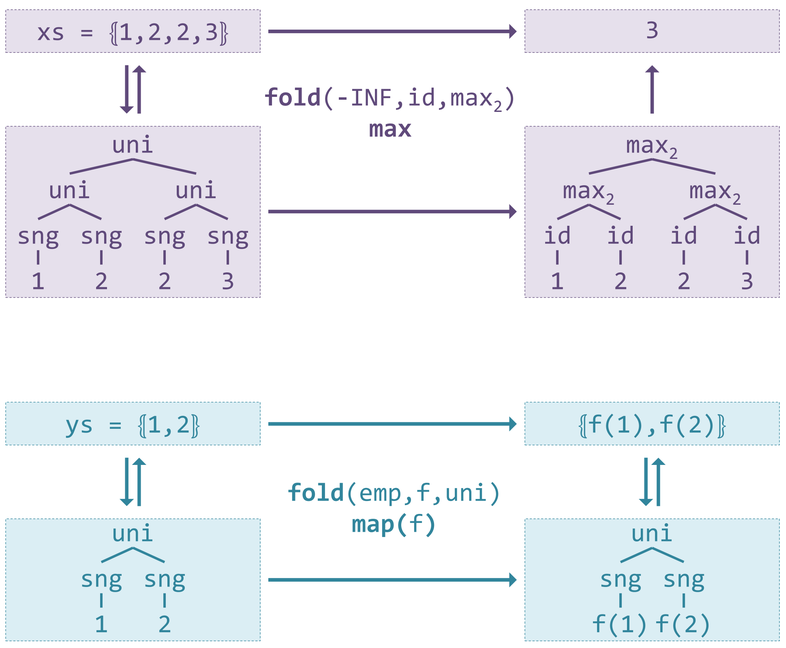
The benefits of the two **can we combined** if we change the embedding strategy.



Declarative Dataflows Beyond SQL

Distributed Bags, Union Representation and Folds

```
def fold[A,B](e: B, s: A => B, u: (B,B) => B)
             (xs: Bag[A]): B = xs match {
  case emp        => e
  case sng(x)     => s(x)
  case uni(ys,zs) => u(fold(e,s,u)(ys), fold(e,s,u)(zs))
}
```



Comprehension Syntax

Comprehensions generalize SQL and are available as first-class syntax in modern general purpose programming languages.

Maths

$$\{ (x, y) \mid x \in xs, y \in ys, x = y \}$$

SQL

```
SELECT x, y FROM x AS xs, y AS ys WHERE x = y
```

Python

```
[ (x, y) for x in xs, y in ys if x == y ]
```

Scala

```
for (x <- xs; y <- ys; if x == y) yield (x, y)
```

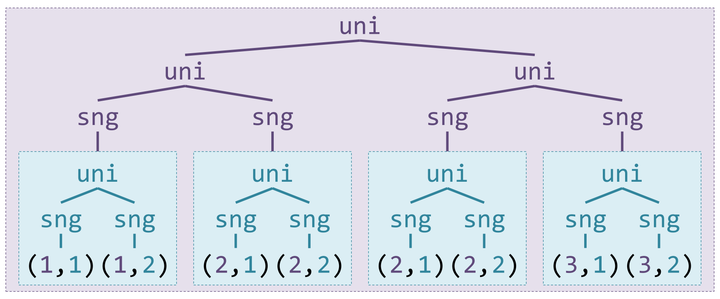
Comprehension Semantics

Comprehension syntax can be enabled in Scala if we extend the bag type to a monad using three second-order functions: map, flatMap, and withFilter.

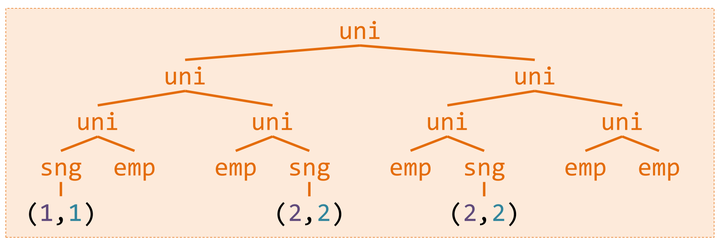
Desugared Comprehension

```
xs.flatMap(x =>
  ys.withFilter(y => x == y).map(y => (x, y)))
```

Nested Map Result



Flattened and Filtered Result



Emma in Action

Basic Principles

Reuse linguistic concepts available in Scala such as *while* loops, *for*-comprehensions, and product types.

Develop and test locally. When the code looks good, wrap it inside an *emma.parallelize { … }* macro.

Emma will (1) identify maximal bag terms, (2) rewrite them holistically, and (3) transparently offload them on a parallel dataflow engine at runtime.

Example: Transitive Closure

```
val algorithm = emma.parallelize {
  var edges = read(input, …).distinct()

  var sizeO = 0L          // old size
  var sizeN = edges.size // new size

  while (sizeN - sizeO > 0) {
    val closure = for {
      e1 <- edges
      e2 <- edges
      if e1.dst == e2.src
    } yield Edge(e1.src, e2.dst)
    edges = (edges plus closure).distinct()
    sizeO = sizeN
    sizeN = edges.size
  }

  write(output, …)(edges)
}

algorithm.run(rt.engine("spark")) // or "flink"
```

Demonstration

Algorithm	*Domain*
TPC-H Queries	Relational
K-Means	Clustering
Naïve Bayes	Classification
Belief Propagation	Statistical inference
Triangle Count	Graph Analysis

Learn More

Implicit Parallelism through Deep Language Embedding. SIGMOD Record 45(1): 51-58 (2016)

Implicit Parallelism through Deep Language Embedding. SIGMOD Conference 2015: 47-61

http://www.emma-language.org

SPONSORED BY THE
Federal Ministry of Education and Research
DFG
ORACLE
eit EIT ICT Labs