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 be combined if we change the embedding strategy.

Declarative Dataflows Beyond SQL

Comprehension Syntax

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

```
xs = {1,2,2,3}
fold(-INF,id,max2)(xs) = (1,1) (1,2) (2,1) (2,2) (3,1) (3,2)
```

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

```
scala
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"
```