**Motivation**

Graph analytics involves viewing the same data as both graphs and tables.

Currently need separate systems to support each view:

- Separate systems increase complexity, lead to unnecessary data movement, and hinder data structure reuse.

**Key Idea**

1. Encode graphs as distributed tables.
2. Express graph computation in relational ops.
3. Recast graph systems optimizations as:
   - A. Distributed join optimization.
   - B. Incremental materialized maintenance.

**System Design**

Horizontally partitioned vertex and edge tables with indexing and join site information.

Graph API Extends the Spark RDDs:

```scala
class Graph (vertices: Table[(Id, V)], edges: Table[(Id, Id, E)])

// Views ---------------------------------------
def vertices: Table[(Id, V)]
def edges: Table[(Id, Id, E)]
def triplets: Table[((Id, V), (Id, V), E)]

// Computation ---------------------------------------
def mrTriplets (mapF: Edge[V, E] => List[(Id, T)], reduceF: (T, T) => T): Graph[T, E]
def mapV (m: (Id, V) => T): Graph[T, E]
def joinV (tbl: Table[(Id, T)]: Graph[(V, T), E]
def subgraph (pV: (Id, V) => Boolean, pE: Edge[V, E] => Boolean): Graph[V, E]
```

Integrate Graph and Table data processing systems. Achieve performance parity with specialized systems.