GraphFrames介绍：构建在DataFrame之上的图处理库

由Databricks、UC Berkeley以及MIT联合为Apache Spark开发了一款图像处理类库，名为：GraphFrames，该类库是构建在DataFrame之上，它既能利用DataFrame良好的扩展性和强大的性能，同时也为Scala、Java和Python提供了统一的图处理API。

什么是GraphFrames

与Apache Spark的GraphX类似，GraphFrames支持多种图处理功能，但得益于DataFrame因此GraphFrames与GraphX库相它有着下面几方面的优势：

1、统一的 API: 为Python、Java和Scala三种语言提供了统一的接口，这是Python和Java首次能够使用GraphX的全部算法。
2、强大的查询功能：GraphFrames使得用户可以构建与Spark SQL以及DataFrame类似的查询语句。
3、图的存储和读取：GraphFrames与DataFrame的数据源完全兼容，支持以Parquet、JSON以及CSV等格式完成图的存储或读取。

在GraphFrames中图的顶点(Vertex)和边(edge)都是以DataFrame形式存储的，所以一个图的所有信息都能够完整保存。

社交网络实例

比如说我们现在有一个社交关系图，每一名用户也就是顶点由他们之间的关系所连接，比如下面的这个例子：

针对这种社交关系图我们可能会有“谁最有影响力”或“应不应该介绍甲乙之间认识”等问题，这类问题可以用图查询算法来解决。这里每一名用户有姓名和年龄两种属性，用户之间的关系也有不同的类型。

id	name	age
a	Alice	34
b	Bob	36
c	Charlie	30
d	David	29
e	Esther	32
f	Fanny	36

src	dst	relationship
a	e	friend
f	b	follow
c	e	friend
a	b	friend
b	c	follow
c	b	follow
f	c	follow
e	f	follow
e	d	friend
d	a	friend

简单查询

使用GraphFrames来进行查询非常容易。由于顶点和边都是以DataFrame存储，很多简单一些的查询语句直接就是DataFrame/SQL语句。

问：图中年龄超过35的用户总数是多少？

g.vertices.filter(“age>35”)

问：有两名以上关注者的用户总数是多少？

g.inDegrees.filter(“inDegree>=2”)

复杂查询

GraphFrames兼容GraphX中所有的算法因此对于复杂查询也能够很好地支持。比如我们想找出图中最重要的用户就可以用pageRank函数：

```
results = g.pageRank(resetProbability=0.15, maxIter=10)
display(results.vertices)
```

GraphFrames还加入了广度优先搜索BFS和模式发现Motif finding两种新算法。

再举一个例子，比如我们想给用户推荐关注 的人就可以寻找图中满足下面这个条件的ABC三个用户:A关注B，B关注C但A并未关注C。代码如下：

```
# Motif: A->B->C but not A->C
results = g.find("(A)-[]->(B); (B)-[]->(C); !(A)-[]->(C)")
# Filter out loops (with DataFrame operation)
results = results.filter("A.id != C.id")
# Select recommendations for A to follow C
results = results.select("A", "C")
display(results)
```

其他GraphFrames支持的算法还有:排序、最短路径、连通分量、强连通分量、三角计数和标签传播LPA。

GraphFrames与GraphX的集成

GraphFrames可以实现与GraphX的完美集成。两者之间相互转换时不会丢失任何数据。

```
val gx: Graph[Row, Row] = g.toGraphX()
val g2: GraphFrame = GraphFrame.fromGraphX(gx)
```

欲了解更多GraphFrames和GraphX之间的转换请参阅GraphFrames API文档。

小结

DataFrames针对图所做出的优化还远未完成，我们在下一版本中还会加入更多的功能。