API

- user define

```
class application {
  virtual generate_one_update();
  virtual apply_one_update();
  virtual update_stream();
}

class pagerank: public application {
  generate_one_update();
  apply_one_update();
  update_stream();
}
```

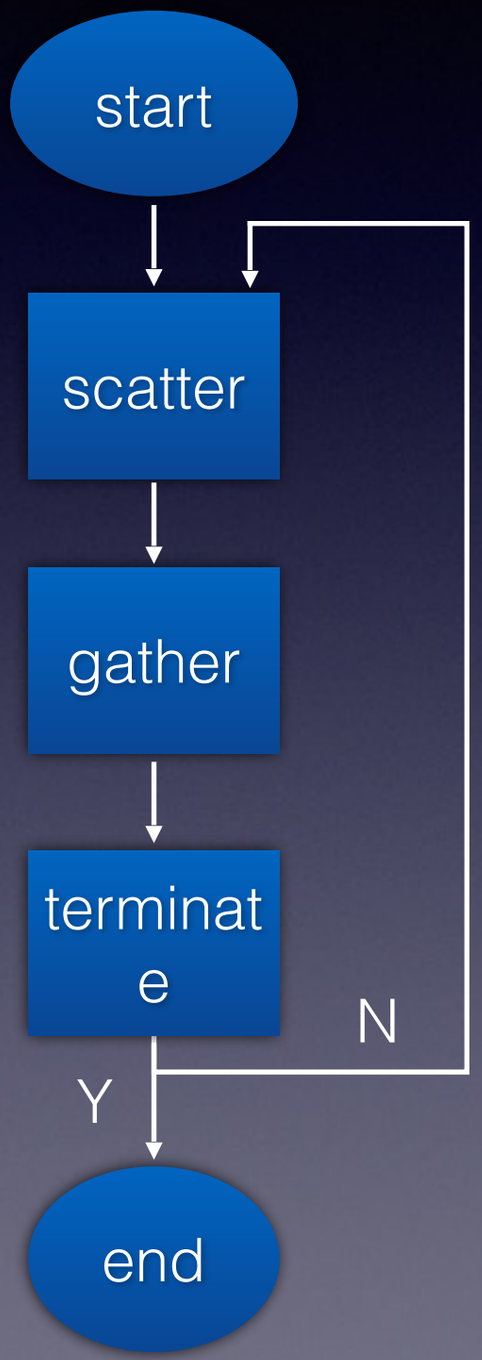
- share same behavior

```
class engine {

  update_stream scatter(edge_stream);
  gather(update_stream);
  shuffle(update_stream, direction =
BY_TARGET);

  // relation algebra support
  update_stream join(update_stream,
edge_stream);
  update_stream
project(update_stream, col_num, …);
  …
}
```

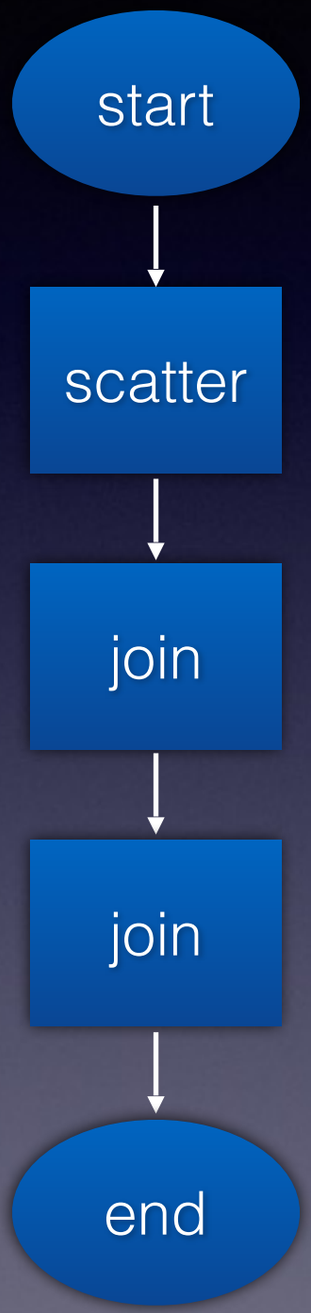
PageRank




```
public update_stream(){…};
public generate_one_update(){…};
public apply_one_update(){…};

while(!terminate()) {
   for(int i = 0; i < num_partitions; i++) {
      update_stream =
engine.scatter(edge_stream(i)) {
            // for each edge
            generate_one_update();
          };
      engine.shuffle(update_stream);
      engine.gather(update_stream(i)) {
          // for each edge
         apply_one_update();
      };
   }
}
```

Triangle Counting



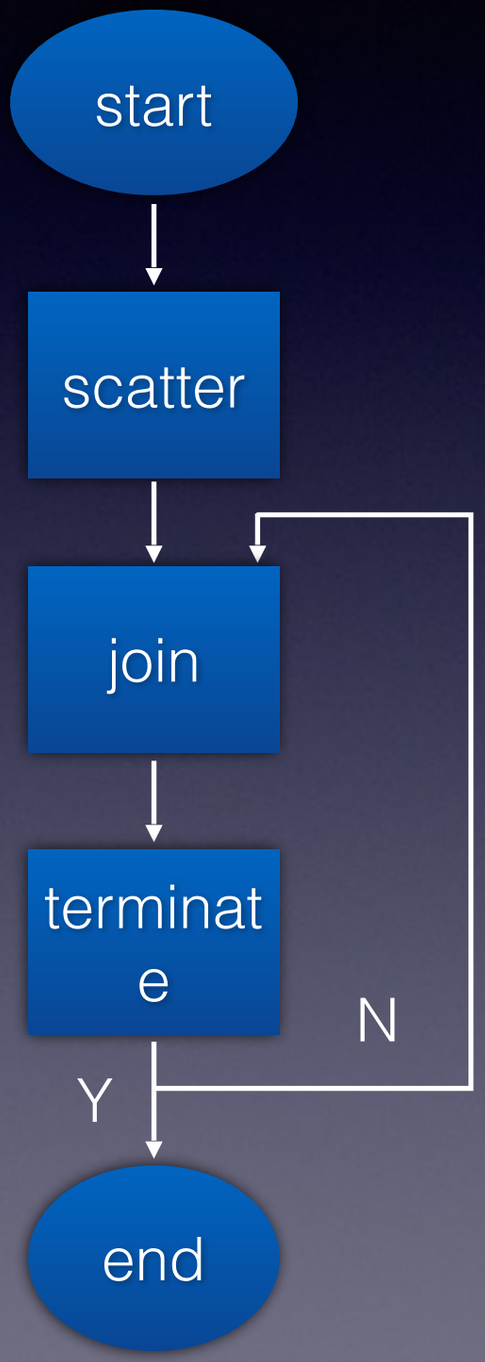

```
// tr(x,y,z) :- edge(x,y), edge(y,z), edge(z,x)

for(int i = 0; i < num_partitions; i++) {
   update_stream =
    engine.scatter(edge_stream(i));
   engine.shuffle(update_stream);
}

for(int i = 0; i < num_partitions; i++) {
   update_stream =
 engine.join(update_stream(i),edge_stream(i));
   update_stream =
engine.project(update_stream, 0, 1, 2);
   engine.shuffle(update_stream);
}

for(int i = 0; i < num_partitions; i++) {
   update_stream =
 engine.join(update_stream(i),edge_stream(i));
   update_stream =
engine.project(update_stream, 0, 1);
   engine.shuffle(update_stream);
}
```

Transitive Closure




```
// tc(x,y) :- tc(x,z), edge(z,y)
for(int i = 0; i < num_partitions; i++) {
   update_stream =
    engine.scatter(edge_stream(i));
   engine.shuffle(update_stream);
}

while(!terminate()) {

   for(int i = 0; i < num_partitions; i++) {
      update_stream =
    engine.join(update_stream(i),edge_stream(i));
      update_stream =
      engine.project(update_stream, 0,
1).distinct();
      engine.shuffle(update_stream);
   }
}
```

System Design

- preprocessing
- disk I/O
- memory management
- multi threading