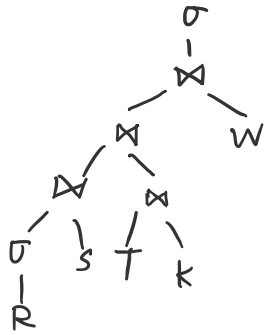
Iterator Interface

Tuesday, May 7, 2019 6:16 PM

- Combine: The Iterator Interface

```
interface Operator {
    void open(..);   // initialize operator state
                     // sets parameters
    // process an input tuple
    // produce output tuple(s), return null when done
    Tuple next();

    //cleans up
    void close()
}
```

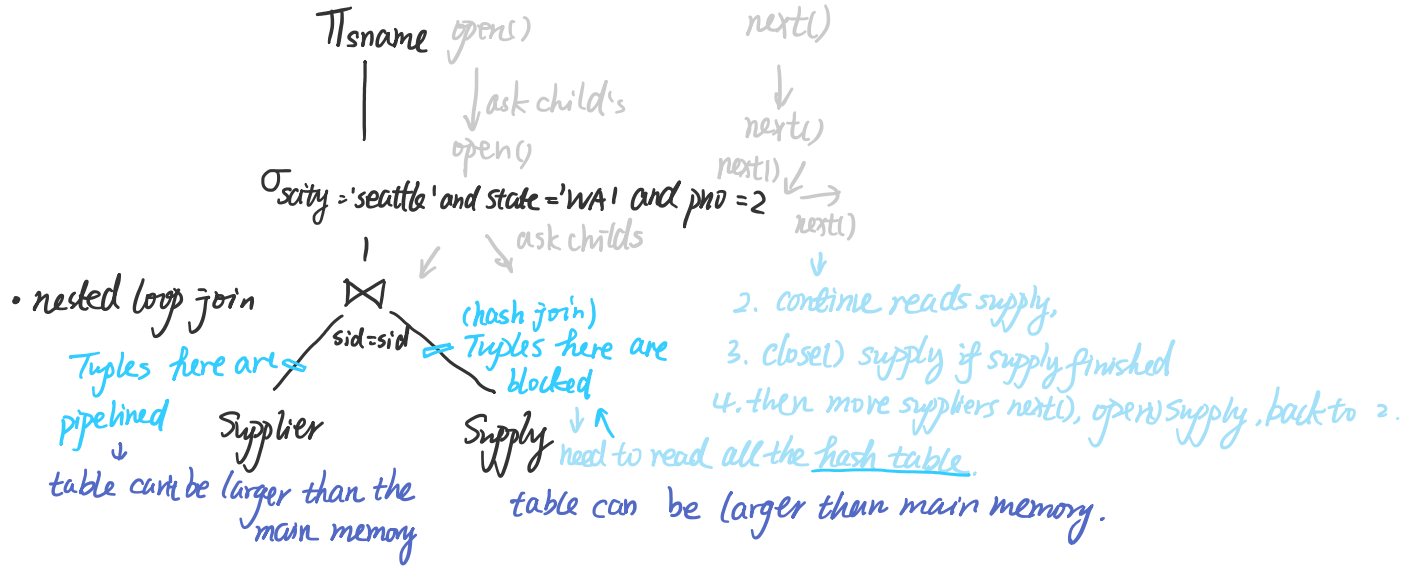



```
class Select implements Operator {
    ...
    void open(Predicate p, operator c) {
        this.p = p; this.c = c; c.open;
    }
    Tuple next() {
        boolean found = false;
        Tuple r = null;
```

```
        while(!found) {
            r=c.next()
            if(r==null) break;
            found = p(r);
        }
        return r;
    }

    void close(){
        c.close();
    }
}
```

- Pipelining



- nested loop join
- If it's hash join:

  After open(), create hash table on supply, probe the hash table

  If list at supply is null, return the first element at that list

- Merge join: both Supply & Supplier are blocked.

- Pipline
  - A tuple moves all the way through up the query plan.
  - Advantage : speed
  - Disadvantage : need all hash at the same time in memory.
- Blocking
  - The entire result of the subplan is computed (and stored to the disk) before the first tuple is sent up the plan
  - Advantage: save memory
  - Disadvantage: slower.