```clojure
(defn simplify [g & ks] (map #(if (st/vartest? %) ? :v) ks))
(defmulti get-from-index simplify)
```

```clojure
(defmethod get-from-index [ ?  ?  ?]
  [{idx :eav} e a v]
  (for [e (keys idx), a (keys (idx e)), v ((idx e) a)]
    [e a v]))
```

```clojure
(defmethod get-from-index [:v  ? :v]
  [{idx :vea} e a v]
  (map vector (get-in idx [v e])))
```

```clojure
(defmethod get-from-index [ ? :v  ?]
  [{idx :ave} e a v]
  (let [edx (idx a)] (for [v (keys edx), e (edx v)]
    [e v])))
```

```clojure
(defmethod get-from-index [:v :v :v]
  [{idx :eav} e a v]
  (if (get-in idx [e a v]) [[]] []))
```

```
def indexAdd(idx, a, b, c):
  adx = idx.get(a)
  if None == adx:
    adx = {}
    idx[a] = adx
  bdx = adx.get(b)
  if None == bdx:
    adx[b] = {c}
  else:
    bdx.add(c)
  return idx
```

```
class GraphIndexed:
  def __init__(self):
    self.eav = {}
    self.ave = {}
    self.vea = {}

  def add(self, e, a, v):
    self.eav = indexAdd(self.eav, e, a, v)
    self.ave = indexAdd(self.ave, a, v, e)
    self.vea = indexAdd(self.vea, v, e, a)
    return self

  def get(self, e, a, v):
    return lookups[simplify(e, a, v)](self, e, a, v)
```