```
(defn index-add
  [idx a b c]
  (update-in idx
             [a b]
             (fn [v] (if (seq v) (conj v c) #{c}))))
```

```
(defrecord GraphIndexed [eav ave vea]
  Graph
  (graph-add [this entity attribute value]
    (let [new-eav (index-add eav entity attribute value)]
      (if (identical? eav new-eav)
        this
        (assoc this :eav new-eav
               :ave (index-add ave attribute value entity)
               :vea (index-add vea value entity attribute)))))

  (resolve-triple [this entity attribute value]
    (get-from-index this entity attribute value)))
```

```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]) [[]] []))
```