Item 271
git_comments:

1. /////////// old subgraph strategy methods
2. **comment:** the edge must pass the edge predicate, and both of its incident vertices must also pass the vertex predicate inV() and/or outV() will be empty if they do not. it is sometimes the case that an edge is unwrapped in which case it may not be filtered. in such cases, the vertices on such edges should be tested.
   label: test
3. **comment:** TODO: why do we have to unwrap? Note that we are not doing f.apply() like the other methods. Is this bad?
   label: code-design
4. **comment:** TODO: we should make sure index hits go first.
   label: code-design
5. ////////////////////////

git_commits:

1. **summary:** Add builder for SubgraphStrategy #417
   message: Add builder for SubgraphStrategy #417 PartitionStrategy no longer extends SubgraphStrategy. This allowed the build() method to stay consistent and set up for major changes to PartitionStrategy in #419

github_issues:

github_issues_comments:

github_pulls:

1. **title:** TINKERPOP-1330: by()-modulation for where()
   body: https://issues.apache.org/jira/browse/TINKERPOP-1330 Added `by()`-modulation support to `where()` predicate-based steps. Added 3 solid `WhereTest` cases to verify proper functioning. Ensured proper `hashCode()` construction in `WhereStepTest`. Also, optimized `TraversalRing` to return `null` if there are no traversals in the ring and thus, `TraversalUtil.applyNullable()` can be leveraged instead which is more efficient than using `IdentityTraversal`. Finally, there was a severe bug in Gremlin-Python that made a complex `WhereTest` fail because `P.and` and `P.or` nesting was reversed! I have fixed Gremlin-Python `P` in this PR. --- Here is an example of the new `where().by()`-model. ``` // give me "a" and "b" is "a" knows "b" and "a" is older than "b". gremlin> g.V().as("a").out("knows").as("b"). ......1> where("a",gt("b")).by("age"). ......2> select("a","b").by("name") ==>[a:marko,b:vadas] gremlin> g.V().as("a").outE("created").as("b"). ......1> inV().as("c"). ......2> in("created").as("d"). ......3> where("a", lt("b").or(gt("c")).and(neq("d"))). ......4> by("age"). ......5> by("weight"). ......6> by(__.in("created").values("age").min()). ......7> select("a", "c", "d").by("name") ==> [a:josh,c:lop,d:marko] ==>[a:josh,c:lop,d:peter] ==>[a:peter,c:lop,d:marko] ==>[a:peter,c:lop,d:josh] ``` In the second query: a -> "age" b -> "weight" c -> in("created")... d -> "age" // TraversalRings are round-robin structures Pretty insane-o. If people start using `where()-by()` heavily, I believe there is an easy translation to `match()` and as such a `TraversalStrategy` would enable us to get the benefits of `match()`-steps runtime query optimizer. VOTE +1
2. **title:** TINKERPOP-1330: by()-modulation for where()
   body: https://issues.apache.org/jira/browse/TINKERPOP-1330 Added `by()`-modulation support to `where()` predicate-based steps. Added 3 solid `WhereTest` cases to verify proper functioning. Ensured proper `hashCode()` construction in `WhereStepTest`. Also, optimized `TraversalRing` to return `null` if there are no traversals in the ring and thus, `TraversalUtil.applyNullable()` can be leveraged instead which is more efficient than using `IdentityTraversal`. Finally, there was a severe bug in Gremlin-Python that made a complex `WhereTest` fail because `P.and` and `P.or` nesting was reversed! I have fixed Gremlin-Python `P` in this PR. --- Here is an example of the new `where().by()`-model. ``` // give me "a" and "b" is "a" knows "b" and "a" is older than "b". gremlin> g.V().as("a").out("knows").as("b"). ......1> where("a",gt("b")).by("age"). ......2> select("a","b").by("name") ==>[a:marko,b:vadas] gremlin> g.V().as("a").outE("created").as("b"). ......1> inV().as("c"). ......2> in("created").as("d"). ......3> where("a", lt("b").or(gt("c")).and(neq("d"))). ......4> by("age"). ......5> by("weight"). ......6> by(__.in("created").values("age").min()). ......7> select("a", "c", "d").by("name") ==>

[a:josh,c:lop,d:marko] ==>[a:josh,c:lop,d:peter] ==>[a:peter,c:lop,d:marko] ==>[a:peter,c:lop,d:josh] ``` In the second query: a -> "age" b -> "weight" c -> in("created")... d -> "age" // TraversalRings are round-robin structures Pretty insane-o. If people start using `where()-by()` heavily, I believe there is an easy translation to `match()` and as such a `TraversalStrategy` would enable us to get the benefits of `match()`-steps runtime query optimizer. VOTE +1

3. **title:** TINKERPOP-1330: by()-modulation for where()
   body: https://issues.apache.org/jira/browse/TINKERPOP-1330 Added `by()`-modulation support to `where()` predicate-based steps. Added 3 solid `WhereTest` cases to verify proper functioning. Ensured proper `hashCode()` construction in `WhereStepTest`. Also, optimized `TraversalRing` to return `null` if there are no traversals in the ring and thus, `TraversalUtil.applyNullable()` can be leveraged instead which is more efficient than using `IdentityTraversal`. Finally, there was a severe bug in Gremlin-Python that made a complex `WhereTest` fail because `P.and` and `P.or` nesting was reversed! I have fixed Gremlin-Python `P` in this PR. --- Here is an example of the new `where().by()`-model. ``` // give me "a" and "b" is "a" knows "b" and "a" is older than "b". gremlin> g.V().as("a").out("knows").as("b"). ......1> where("a",gt("b")).by("age"). ......2> select("a","b").by("name") ==>[a:marko,b:vadas] gremlin> g.V().as("a").outE("created").as("b"). ......1> inV().as("c"). ......2> in("created").as("d"). ......3> where("a", lt("b").or(gt("c")).and(neq("d"))). ......4> by("age"). ......5> by("weight"). ......6> by(__.in("created").values("age").min()). ......7> select("a", "c", "d").by("name") ==> [a:josh,c:lop,d:marko] ==>[a:josh,c:lop,d:peter] ==>[a:peter,c:lop,d:marko] ==>[a:peter,c:lop,d:josh] ``` In the second query: a -> "age" b -> "weight" c -> in("created")... d -> "age" // TraversalRings are round-robin structures Pretty insane-o. If people start using `where()-by()` heavily, I believe there is an easy translation to `match()` and as such a `TraversalStrategy` would enable us to get the benefits of `match()`-steps runtime query optimizer. VOTE +1
4. **title:** TINKERPOP-1330: by()-modulation for where()
   body: https://issues.apache.org/jira/browse/TINKERPOP-1330 Added `by()`-modulation support to `where()` predicate-based steps. Added 3 solid `WhereTest` cases to verify proper functioning. Ensured proper `hashCode()` construction in `WhereStepTest`. Also, optimized `TraversalRing` to return `null` if there are no traversals in the ring and thus, `TraversalUtil.applyNullable()` can be leveraged instead which is more efficient than using `IdentityTraversal`. Finally, there was a severe bug in Gremlin-Python that made a complex `WhereTest` fail because `P.and` and `P.or` nesting was reversed! I have fixed Gremlin-Python `P` in this PR. --- Here is an example of the new `where().by()`-model. ``` // give me "a" and "b" is "a" knows "b" and "a" is older than "b". gremlin> g.V().as("a").out("knows").as("b"). ......1> where("a",gt("b")).by("age"). ......2> select("a","b").by("name") ==>[a:marko,b:vadas] gremlin> g.V().as("a").outE("created").as("b"). ......1> inV().as("c"). ......2> in("created").as("d"). ......3> where("a", lt("b").or(gt("c")).and(neq("d"))). ......4> by("age"). ......5> by("weight"). ......6> by(__.in("created").values("age").min()). ......7> select("a", "c", "d").by("name") ==> [a:josh,c:lop,d:marko] ==>[a:josh,c:lop,d:peter] ==>[a:peter,c:lop,d:marko] ==>[a:peter,c:lop,d:josh] ``` In the second query: a -> "age" b -> "weight" c -> in("created")... d -> "age" // TraversalRings are round-robin structures Pretty insane-o. If people start using `where()-by()` heavily, I believe there is an easy translation to `match()` and as such a `TraversalStrategy` would enable us to get the benefits of `match()`-steps runtime query optimizer. VOTE +1
   label: code-design
5. **title:** TINKERPOP-1330: by()-modulation for where()
   body: https://issues.apache.org/jira/browse/TINKERPOP-1330 Added `by()`-modulation support to `where()` predicate-based steps. Added 3 solid `WhereTest` cases to verify proper functioning. Ensured proper `hashCode()` construction in `WhereStepTest`. Also, optimized `TraversalRing` to return `null` if there are no traversals in the ring and thus, `TraversalUtil.applyNullable()` can be leveraged instead which is more efficient than using `IdentityTraversal`. Finally, there was a severe bug in Gremlin-Python that made a complex `WhereTest` fail because `P.and` and `P.or` nesting was reversed! I have fixed Gremlin-Python `P` in this PR. --- Here is an example of the new `where().by()`-model. ``` // give me "a" and "b" is "a" knows "b" and "a" is older than "b". gremlin> g.V().as("a").out("knows").as("b"). ......1> where("a",gt("b")).by("age"). ......2> select("a","b").by("name") ==>[a:marko,b:vadas] gremlin> g.V().as("a").outE("created").as("b"). ......1> inV().as("c"). ......2> in("created").as("d"). ......3> where("a", lt("b").or(gt("c")).and(neq("d"))). ......4> by("age"). ......5> by("weight"). ......6> by(__.in("created").values("age").min()). ......7> select("a", "c", "d").by("name") ==> [a:josh,c:lop,d:marko] ==>[a:josh,c:lop,d:peter] ==>[a:peter,c:lop,d:marko] ==>[a:peter,c:lop,d:josh] ``` In the second query: a -> "age" b -> "weight" c -> in("created")... d -> "age" // TraversalRings are round-robin structures Pretty insane-o. If people start using `where()-by()` heavily, I believe there is an easy

translation to `match()` and as such a `TraversalStrategy` would enable us to get the benefits of `match()`-steps runtime query optimizer. VOTE +1

6. **title:** TINKERPOP-1330: by()-modulation for where()
   body: https://issues.apache.org/jira/browse/TINKERPOP-1330 Added `by()`-modulation support to `where()` predicate-based steps. Added 3 solid `WhereTest` cases to verify proper functioning. Ensured proper `hashCode()` construction in `WhereStepTest`. Also, optimized `TraversalRing` to return `null` if there are no traversals in the ring and thus, `TraversalUtil.applyNullable()` can be leveraged instead which is more efficient than using `IdentityTraversal`. Finally, there was a severe bug in Gremlin-Python that made a complex `WhereTest` fail because `P.and` and `P.or` nesting was reversed! I have fixed Gremlin-Python `P` in this PR. --- Here is an example of the new `where().by()`-model. ``` // give me "a" and "b" is "a" knows "b" and "a" is older than "b". gremlin> g.V().as("a").out("knows").as("b"). ......1> where("a",gt("b")).by("age"). ......2> select("a","b").by("name") ==>[a:marko,b:vadas] gremlin> g.V().as("a").outE("created").as("b"). ......1> inV().as("c"). ......2> in("created").as("d"). ......3> where("a", lt("b").or(gt("c")).and(neq("d"))). ......4> by("age"). ......5> by("weight"). ......6> by(__.in("created").values("age").min()). ......7> select("a", "c", "d").by("name") ==> [a:josh,c:lop,d:marko] ==>[a:josh,c:lop,d:peter] ==>[a:peter,c:lop,d:marko] ==>[a:peter,c:lop,d:josh] ``` In the second query: a -> "age" b -> "weight" c -> in("created")... d -> "age" // TraversalRings are round-robin structures Pretty insane-o. If people start using `where()-by()` heavily, I believe there is an easy translation to `match()` and as such a `TraversalStrategy` would enable us to get the benefits of `match()`-steps runtime query optimizer. VOTE +1
7. **title:** TINKERPOP-1330: by()-modulation for where()
   body: https://issues.apache.org/jira/browse/TINKERPOP-1330 Added `by()`-modulation support to `where()` predicate-based steps. Added 3 solid `WhereTest` cases to verify proper functioning. Ensured proper `hashCode()` construction in `WhereStepTest`. Also, optimized `TraversalRing` to return `null` if there are no traversals in the ring and thus, `TraversalUtil.applyNullable()` can be leveraged instead which is more efficient than using `IdentityTraversal`. Finally, there was a severe bug in Gremlin-Python that made a complex `WhereTest` fail because `P.and` and `P.or` nesting was reversed! I have fixed Gremlin-Python `P` in this PR. --- Here is an example of the new `where().by()`-model. ``` // give me "a" and "b" is "a" knows "b" and "a" is older than "b". gremlin> g.V().as("a").out("knows").as("b"). ......1> where("a",gt("b")).by("age"). ......2> select("a","b").by("name") ==>[a:marko,b:vadas] gremlin> g.V().as("a").outE("created").as("b"). ......1> inV().as("c"). ......2> in("created").as("d"). ......3> where("a", lt("b").or(gt("c")).and(neq("d"))). ......4> by("age"). ......5> by("weight"). ......6> by(__.in("created").values("age").min()). ......7> select("a", "c", "d").by("name") ==> [a:josh,c:lop,d:marko] ==>[a:josh,c:lop,d:peter] ==>[a:peter,c:lop,d:marko] ==>[a:peter,c:lop,d:josh] ``` In the second query: a -> "age" b -> "weight" c -> in("created")... d -> "age" // TraversalRings are round-robin structures Pretty insane-o. If people start using `where()-by()` heavily, I believe there is an easy translation to `match()` and as such a `TraversalStrategy` would enable us to get the benefits of `match()`-steps runtime query optimizer. VOTE +1
8. **title:** TINKERPOP-1330: by()-modulation for where()
   body: https://issues.apache.org/jira/browse/TINKERPOP-1330 Added `by()`-modulation support to `where()` predicate-based steps. Added 3 solid `WhereTest` cases to verify proper functioning. Ensured proper `hashCode()` construction in `WhereStepTest`. Also, optimized `TraversalRing` to return `null` if there are no traversals in the ring and thus, `TraversalUtil.applyNullable()` can be leveraged instead which is more efficient than using `IdentityTraversal`. Finally, there was a severe bug in Gremlin-Python that made a complex `WhereTest` fail because `P.and` and `P.or` nesting was reversed! I have fixed Gremlin-Python `P` in this PR. --- Here is an example of the new `where().by()`-model. ``` // give me "a" and "b" is "a" knows "b" and "a" is older than "b". gremlin> g.V().as("a").out("knows").as("b"). ......1> where("a",gt("b")).by("age"). ......2> select("a","b").by("name") ==>[a:marko,b:vadas] gremlin> g.V().as("a").outE("created").as("b"). ......1> inV().as("c"). ......2> in("created").as("d"). ......3> where("a", lt("b").or(gt("c")).and(neq("d"))). ......4> by("age"). ......5> by("weight"). ......6> by(__.in("created").values("age").min()). ......7> select("a", "c", "d").by("name") ==> [a:josh,c:lop,d:marko] ==>[a:josh,c:lop,d:peter] ==>[a:peter,c:lop,d:marko] ==>[a:peter,c:lop,d:josh] ``` In the second query: a -> "age" b -> "weight" c -> in("created")... d -> "age" // TraversalRings are round-robin structures Pretty insane-o. If people start using `where()-by()` heavily, I believe there is an easy translation to `match()` and as such a `TraversalStrategy` would enable us to get the benefits of `match()`-steps runtime query optimizer. VOTE +1
9. **title:** TINKERPOP-1330: by()-modulation for where()
   body: https://issues.apache.org/jira/browse/TINKERPOP-1330 Added `by()`-modulation support to `where()` predicate-based steps. Added 3 solid `WhereTest` cases to verify proper functioning. Ensured

proper `hashCode()` construction in `WhereStepTest`. Also, optimized `TraversalRing` to return `null` if there are no traversals in the ring and thus, `TraversalUtil.applyNullable()` can be leveraged instead which is more efficient than using `IdentityTraversal`. Finally, there was a severe bug in Gremlin-Python that made a complex `WhereTest` fail because `P.and` and `P.or` nesting was reversed! I have fixed Gremlin-Python `P` in this PR. --- Here is an example of the new `where().by()`-model. ``` // give me "a" and "b" is "a" knows "b" and "a" is older than "b". gremlin> g.V().as("a").out("knows").as("b"). ......1> where("a",gt("b")).by("age"). ......2> select("a","b").by("name") ==>[a:marko,b:vadas] gremlin> g.V().as("a").outE("created").as("b"). ......1> inV().as("c"). ......2> in("created").as("d"). ......3> where("a", lt("b").or(gt("c")).and(neq("d"))). ......4> by("age"). ......5> by("weight"). ......6> by(__.in("created").values("age").min()). ......7> select("a", "c", "d").by("name") ==> [a:josh,c:lop,d:marko] ==>[a:josh,c:lop,d:peter] ==>[a:peter,c:lop,d:marko] ==>[a:peter,c:lop,d:josh] ``` In the second query: a -> "age" b -> "weight" c -> in("created")... d -> "age" // TraversalRings are round-robin structures Pretty insane-o. If people start using `where()-by()` heavily, I believe there is an easy translation to `match()` and as such a `TraversalStrategy` would enable us to get the benefits of `match()`-steps runtime query optimizer. VOTE +1

github_pulls_comments:

1. Regarding the `match()` translation, it's indeed very easy: ``` gremlin> g.V().as("a").outE("created").as("b"). ......1> inV().as("c").in("created").as("d"). ......2> match(......3> __.as("a").values("age").as("a_by"), ...4> __.as("b").values("weight").as("b_by"), ...5> __.as("c").in("created").values("age").min().as("c_by"), ...6> __.as("d").values("age").as("d_by"), ...7>). ......8> where("a_by", lt("b_by").or(gt("c_by")).and(neq("d_by"))). ......9> select("a", "c", "d").by("name") ==>[a:josh,c:lop,d:marko] ==>[a:josh,c:lop,d:peter] ==>[a:peter,c:lop,d:marko] ==>[a:peter,c:lop,d:josh] ``` I will make a code review and vote tomorrow.
2. I think we did a pretty bad job on upgrade docs in 3.2.2 so i'm kinda on the lookout for things that we might add to them - this might be a nice thing to call attention to in "user" section imo.
3. Could you please update the Traversal Induced Values recipe? https://github.com/apache/tinkerpop/blob/master/docs/src/recipes/traversal-induced-values.asciidoc
4. **body:** @spmallette I updated `traversal-induced-values.asciidoc`. Wow -- `where().by()` is super powerful. That nested `where(select())`-clause you had in there was naaaasty. Given how much easier some things are to express, I'm realizing that people will be wanting to use `where().by()` more often than not and given that `where().by()` is a `PathProcessor` now, OLAP will not allow any `by()` beyond the element id! Eek. No fear, we have `PathProcessorStrategy` that knows how to "flatten" `select().by()` accordingly. With that, I think another ticket should provide two things: 1. `PathProcessorStrategy` should be updated to support `where().by()` flattening. 2. Create a new strategy called `WhereByMatchStrategy` which turns `where().by()` patterns into `match()`-steps. (@dkuppitz's idea) Now, if (1) above happens then (2) above won't do anything. Why would (1) happen, but not (2). The complexity of the `where()`-step's `by()`-clause. If the `by()` clause is within the bounds of the local star graph, then (1) will trigger. Else, it won't and (2) would trigger. Next, we will have to see how these two strategies interact with `MatchPredicateStrategy` as this folds `where()`-clauses into `match()` to get the runtime optimizer benefits... Its all very complex :D
   label: code-design
5. Doc update looks good. I'd still like a short section in the upgrade docs but other than that - All tests pass with `docker/build.sh -t -n -i` VOTE +1
6. Code looks great, although the Python-related fixes are a bit misplaced in this PR. Anyhow, VOTE: +1
7. The Python-related fixes are necessary or else some of the `WhereTests` fail. Python has a bad ordering to the scoping variables with nested `and()`/`or()`. Thus, for the test suite to pass I had to fix a bug in Python.

github_pulls_reviews:

jira_issues:

1. **summary:** by()-modulation for where()
   description: As discussed in https://issues.apache.org/jira/browse/TINKERPOP-1329, it would be nice to have {{by()}}-modulators for {{where()}}. For example: {code} g.V().as("a").out().where(gt("a")).by("age") // both a and current ages are selected

g.V().as("a").out().as("b").where("b",gt("a")).by("age") // both a and b ages are selected
g.V().as("a").out().as("b").where("b",gt("a")).by("age").by("weight") // where b.age > a.weight {code}
2. **summary:** by()-modulation for where()
description: As discussed in https://issues.apache.org/jira/browse/TINKERPOP-1329, it would be nice to have {{by()}}-modulators for {{where()}}. For example: {code}
g.V().as("a").out().where(gt("a")).by("age") // both a and current ages are selected
g.V().as("a").out().as("b").where("b",gt("a")).by("age") // both a and b ages are selected
g.V().as("a").out().as("b").where("b",gt("a")).by("age").by("weight") // where b.age > a.weight {code}

jira_issues_comments:

1. GitHub user okram opened a pull request: https://github.com/apache/tinkerpop/pull/417 TINKERPOP-1330: by()-modulation for where() https://issues.apache.org/jira/browse/TINKERPOP-1330 Added `by()`-modulation support to `where()` predicate-based steps. Added 3 solid `WhereTest` cases to verify proper functioning. Ensured proper `hashCode()` construction in `WhereStepTest`. Also, optimized `TraversalRing` to return `null` if there are no traversals in the ring and thus, `TraversalUtil.applyNullable()` can be leveraged instead which is more efficient than using `IdentityTraversal`. Finally, there was a severe bug in Gremlin-Python that made a complex `WhereTest` fail because `P.and` and `P.or` nesting was reversed! I have fixed Gremlin-Python `P` in this PR. --- Here is an example of the new `where().by()`-model. ``` // give me "a" and "b" is "a" knows "b" and "a" is older than "b". gremlin> g.V().as("a").out("knows").as("b"). ......1> where("a",gt("b")).by("age"). ......2> select("a","b").by("name") ==>[a:marko,b:vadas] gremlin> g.V().as("a").outE("created").as("b"). ......1> inV().as("c"). ......2> in("created").as("d"). ......3> where("a", lt("b").or(gt("c")).and(neq("d"))). ......4> by("age"). ......5> by("weight"). ......6> by(__.in("created").values("age").min()). ......7> select("a", "c", "d").by("name") ==>[a:josh,c:lop,d:marko] ==>[a:josh,c:lop,d:peter] ==>[a:peter,c:lop,d:marko] ==>[a:peter,c:lop,d:josh] ``` In the second query: a -> "age" b -> "weight" c -> in("created")... d -> "age" // TraversalRings are round-robin structures Pretty insane-o. If people start using `where()-by()` heavily, I believe there is an easy translation to `match()` and as such a `TraversalStrategy` would enable us to get the benefits of `match()`-steps runtime query optimizer. VOTE +1 You can merge this pull request into a Git repository by running: $ git pull https://github.com/apache/tinkerpop TINKERPOP-1330 Alternatively you can review and apply these changes as the patch at: https://github.com/apache/tinkerpop/pull/417.patch To close this pull request, make a commit to your master/trunk branch with (at least) the following in the commit message: This closes #417 ---- commit 5f264db45f2f58f79ab833e900f52d66501e741a Author: Marko A. Rodriguez <okrammarko@gmail.com> Date: 2016-09-14T21:30:34Z added where().by() semantics to Gremlin and fixed a severe bug in Gremlin-Python's P object. Optimized TraversalRing for non-existent modulators. ----
2. Github user dkuppitz commented on the issue: https://github.com/apache/tinkerpop/pull/417 Regarding the `match()` translation, it's indeed very easy: ``` gremlin> g.V().as("a").outE("created").as("b"). ......1> inV().as("c").in("created").as("d"). ......2> match(......3> __.as("a").values("age").as("a_by"), ...4> __.as("b").values("weight").as("b_by"), ...5> __.as("c").in("created").values("age").min().as("c_by"), ...6> __.as("d").values("age").as("d_by"), ...7>). ......8> where("a_by", lt("b_by").or(gt("c_by")).and(neq("d_by"))). ......9> select("a", "c", "d").by("name") ==> [a:josh,c:lop,d:marko] ==>[a:josh,c:lop,d:peter] ==>[a:peter,c:lop,d:marko] ==>[a:peter,c:lop,d:josh] ``` I will make a code review and vote tomorrow.
3. Github user spmallette commented on the issue: https://github.com/apache/tinkerpop/pull/417 I think we did a pretty bad job on upgrade docs in 3.2.2 so i'm kinda on the lookout for things that we might add to them - this might be a nice thing to call attention to in "user" section imo.
4. Github user spmallette commented on the issue: https://github.com/apache/tinkerpop/pull/417 Could you please update the Traversal Induced Values recipe? https://github.com/apache/tinkerpop/blob/master/docs/src/recipes/traversal-induced-values.asciidoc
5. Github user okram commented on the issue: https://github.com/apache/tinkerpop/pull/417 @spmallette I updated `traversal-induced-values.asciidoc`. Wow -- `where().by()` is super powerful. That nested `where(select())`-clause you had in there was naaaasty. Given how much easier some things are to express, I'm realizing that people will be wanting to use `where().by()` more often than not and given that `where().by()` is a `PathProcessor` now, OLAP will not allow any `by()` beyond the element id! Eek. No fear, we have `PathProcessorStrategy` that knows how to "flatten" `select().by()` accordingly. With that, I think another ticket should provide two things: 1. `PathProcessorStrategy` should be updated to support `where().by()` flattening. 2. Create a new strategy called `WhereByMatchStrategy` which turns `where().by()` patterns into `match()`-steps. (@dkuppitz's idea) Now, if (1) above happens then (2) above

won't do anything. Why would (1) happen, but not (2). The complexity of the `where()`-step's `by()`-clause. If the `by()` clause is within the bounds of the local star graph, then (1) will trigger. Else, it won't and (2) would trigger. Next, we will have to see how these two strategies interact with `MatchPredicateStrategy` as this folds `where()`-clauses into `match()` to get the runtime optimizer benefits... Its all very complex :D

6. Github user spmallette commented on the issue: https://github.com/apache/tinkerpop/pull/417 Doc update looks good. I'd still like a short section in the upgrade docs but other than that - All tests pass with `docker/build.sh -t -n -i` VOTE +1
7. Github user dkuppitz commented on the issue: https://github.com/apache/tinkerpop/pull/417 Code looks great, although the Python-related fixes are a bit misplaced in this PR. Anyhow, VOTE: +1
8. Github user okram commented on the issue: https://github.com/apache/tinkerpop/pull/417 The Python-related fixes are necessary or else some of the `WhereTests` fail. Python has a bad ordering to the scoping variables with nested `and()`/`or()`. Thus, for the test suite to pass I had to fix a bug in Python.
9. Github user okram closed the pull request at: https://github.com/apache/tinkerpop/pull/417