
graphcore Documentation

Release 0.6

Zach Dziel

April 20, 2016

1	graphcore package	3
1.1	Submodules	3
1.2	graphcore.call_graph module	3
1.3	graphcore.call_graph_test module	4
1.4	graphcore.clause module	4
1.5	graphcore.clause_test module	5
1.6	graphcore.conftest module	5
1.7	graphcore.conftest_test module	5
1.8	graphcore.equality_mixin module	5
1.9	graphcore.equality_mixin_test module	6
1.10	graphcore.graphcore module	6
1.11	graphcore.graphcore_sql_integration_test module	7
1.12	graphcore.graphcore_test module	8
1.13	graphcore.optimize_constrain_sql_queries module	10
1.14	graphcore.optimize_reduce_like_parent_child module	10
1.15	graphcore.optimize_reduce_like_parent_child_test module	10
1.16	graphcore.optimize_reduce_like_siblings module	11
1.17	graphcore.optimize_reduce_like_siblings_test module	11
1.18	graphcore.path module	11
1.19	graphcore.path_test module	11
1.20	graphcore.query module	12
1.21	graphcore.query_plan module	12
1.22	graphcore.query_plan_test module	12
1.23	graphcore.query_planner module	12
1.24	graphcore.query_test module	13
1.25	graphcore.reflect_class module	13
1.26	graphcore.reflect_class_test module	13
1.27	graphcore.reflect_module module	13
1.28	graphcore.reflect_module_test module	13
1.29	graphcore.relation module	14
1.30	graphcore.relation_test module	14
1.31	graphcore.result_set module	14
1.32	graphcore.result_set_test module	15
1.33	graphcore.rule module	16
1.34	graphcore.rule_test module	17
1.35	graphcore.sql_query module	17
1.36	graphcore.sql_query_sqlalchemy module	17
1.37	graphcore.sql_query_test module	17

1.38	graphcore.sql_reflect module	18
1.39	graphcore.sql_reflect_test module	18
1.40	graphcore.test_harness module	18
1.41	graphcore.test_module module	18
1.42	Module contents	19
2	AST	21
2.1	Design Decisions	21
3	Indices and tables	23
	Python Module Index	25

Contents:

graphcore package

1.1 Submodules

1.2 graphcore.call_graph module

Node := [Apply|Path] Apply := { 'paths': Path|[Path]|{str: Path}, 'call': Call|None } Call := { 'function': function, 'args': Node, 'out': bool }

Apply.paths options

- *Path*: return value of *Call* is stored in path
- **[Path]: return value of the Call is should be an iterable.** The first element in the iterator will be stored in the first *Path*, and so on.
- **{str: Path} return value of the Call should be a dictionary.** The value of each *Path* will be set to the value in the dictionary at *str*

Note: only one of [Path] or {str: Path} is actually required. The other is optional sugar

FAQ:

Q: why are there multiple return paths from a call if all calls must only have a single return value?

A: The single return value ast works fine with the assumption that each call only has a single return path. This is a necessary assumption while generating the call graph, but it doesn't need to hold during the QueryOptimization phase. In some cases, you may want to replace two 'small' functions with a larger more optimal one. This is especially true when the functions are hitting an external resource and you don't want a large number of round trips.

Node: { 'incoming_edges': {Edge}, 'outgoing_edges': {Edge}, 'function': function, 'cardinality': One|Many, 'relations': Relation,

}

Relation: { 'operation': '<'|>'|...', 'value': Any,

}

This is really a glorified Path. Edge: {

 'path': Path, 'getters': {Node}, 'setter': Node|None,

}

the order of the in_paths is irrelevant (set) the order of the out paths, corresponds to the order of the returned iterable.
 TODO: out_paths: {key: (Path, Node), ...}

```
class graphcore.call_graph.CallGraph
    Bases: object

    add_node (incoming_paths, outgoing_paths, function, cardinality, relations=None)

    edge (path)

    explain ()

    nodes_dependent_on_path (path)
        return a list of nodes which depend on path.

        This is helpful for debugging when a match isn't found and you want to know where a clause came from.

    output_paths ()

    remove_node (node)

class graphcore.call_graph.Edge (path, getters, setter, out)
    Bases: object

    out: bool - True if is this path used in the final ResultSet, False if it is an intermediate value

class graphcore.call_graph.Node (call_graph, incoming_paths, outgoing_paths, function, cardinality, relations=None)
    Bases: object

    explain ()

    incoming_edges ()

    incoming_nodes ()

    input_path_by_property (property)

    name
```

1.3 graphcore.call_graph_test module

```
graphcore.call_graph_test.test_edge_hash ()
graphcore.call_graph_test.test_edge_ne ()
graphcore.call_graph_test.test_edge_not_ne ()
graphcore.call_graph_test.test_explain ()
```

1.4 graphcore.clause module

```
class graphcore.clause.Clause (key, value)
    Bases: object

    convert_to_constraint ()
        converts self from a ground clause with a value to an unground clause with a == relation

    copy ()

    merge (other)
        Combine other clause into self by mutating self

        This happens when we get additional constraints in a clause:
```


‘x?’: None, ‘x>’: 1,

or when a query search is taking place and it wants to ensure that a TempVar is marked on a path that it depends on. This may happen if there is a relational constraint on a path that another cause depends on as an input.

```
class graphcore.clause.OutVar
    Bases: graphcore.clause.Var
```

```
class graphcore.clause.TempVar
    Bases: graphcore.clause.Var
```

```
class graphcore.clause.Var
    Bases: object
```

1.5 graphcore.clause_test module

```
graphcore.clause_test.test_clause_merge_relation()
graphcore.clause_test.test_clause_merge_rhs_conflict()
graphcore.clause_test.test_repr()
graphcore.clause_test.test_str()
```

1.6 graphcore.conftest module

```
graphcore.conftest.call_graph_repr_compare(left, right)
graphcore.conftest.pytest_assertrepr_compare(op, left, right)
    pytest helper to make CallGraph comparison easier
graphcore.conftest.sql_query_repr_compare(left, right)
```

1.7 graphcore.conftest_test module

```
graphcore.conftest_test.test_call_graph_repr_compare()
graphcore.conftest_test.test_pytest_assertrepr_compare_call_graph()
graphcore.conftest_test.test_pytest_assertrepr_compare_sql_query()
graphcore.conftest_test.test_sql_query_repr_compare()
```

1.8 graphcore.equality_mixin module

```
class graphcore.equality_mixin.EqualityMixin
    Bases: object
class graphcore.equality_mixin.HashMixin
    Bases: object
graphcore.equality_mixin.freeze(o)
```

1.9 graphcore.equality_mixin_test module

```
class graphcore.equality_mixin_test.TestClass(a, b)
    Bases: graphcore.equality_mixin.EqualityMixin

graphcore.equality_mixin_test.test_equality_mixin()

graphcore.equality_mixin_test.test_equality_mixin_with_set()

graphcore.equality_mixin_test.test_equality_mixing_other()
```

1.10 graphcore.graphcore module

```
exception graphcore.graphcore.BaseTypeNotFound(subpath, path)
    Bases: graphcore.graphcore.PathNotFound

class graphcore.graphcore.DefineTypeContext(gc, type_name)
    Bases: object

    direct_map(input, output)

    module(module)

    property_type(property_name, property_type=None)

    reflect_class(cls, type_name=None)

class graphcore.graphcore.Graphcore(mapper=<built-in function map>)
    Bases: object

    available_rules_string()

    base_types()

    define_type(type_name)

    direct_map(input, output)

    explain(query)

    lookup_rule(path)
        Given a clause, return a prefix and a rule which match the clause.

        The prefix will be a list of parts of the lhs of the clause which the rule is applied to. For example if there is a rule which maps from book.id to book.name and the query has a user.book.id then this function will return ['user.book'], Rule(book.id -> book.name).

    optimize(query_search)

    property_type(base_type, property, other_type)

    query(query, limit=None, exception_handler=<function default_exception_handler>)

    register_rule(inputs, output, cardinality=<Cardinality.one: 1>, function=None)

    rule(inputs, output, cardinality=<Cardinality.one: 1>)

    search_outputs(search='', prefix='')
        return a list of outputs which contain search and/or begin with prefix

        useful for interactive exploration and debugging.

exception graphcore.graphcore.PathNotFound(path, gc)
    Bases: exceptions.Exception
```

```
class graphcore.graphcore.PropertyType (base_type, property, other_type)
    Bases: graphcore.equality_mixin.HashMixin, graphcore.equality_mixin.EqualityMixin
```

class graphcore.graphcore.**QuerySearch** (*graphcore, query*)
 Bases: object

The QuerySearch object takes a Graphcore and a Query and generates a CallGraph.

apply_rule_backwards (*output_clause, prefix, rule*)
 bind the output of rule to output_clause from the query

backward ()
 apply rules in reverse looking for the call chain that will be necessary to complete the query.

we can pick any old clause off the stack since the order that rules are resolved, at this point in the search is unimportant. We can always optimize the call graph later, one we have one.

clause_with_unbound_outvar ()
 return a clause with a variable rhs which hasnt been grounded

clauses_with_unbound_outvar ()

class graphcore.graphcore.**QuerySearchIterator** (*query*)
 Bases: object

next ()

class graphcore.graphcore.**Rules**
 Bases: object

append (*rule*)

lookup (*path, require_input*)

class graphcore.graphcore.**Schema**
 Bases: object

append (*property_type*)

resolve_type (*path, pos=-1*)
 given a full path and an index into that path, return the type of the value of the property at that index

1.11 graphcore.graphcore_sql_integration_test module

Integration testing graphcore queries on data in a sqlite in memory database

```
class graphcore.graphcore_sql_integration_test.Book (**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Base

    id

    user_id

graphcore.graphcore_sql_integration_test.SQLAlchemyQueryClass (engine)

class graphcore.graphcore_sql_integration_test.User (**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Base

    age

    id

    name
```

```
graphcore.graphcore_sql_integration_test.engine()
graphcore.graphcore_sql_integration_test.gc(engine, SQLAlchemyQueryClass)
graphcore.graphcore_sql_integration_test.session(engine)
graphcore.graphcore_sql_integration_test.test(gc, session)
graphcore.graphcore_sql_integration_test.test_query_and_filter(gc, session)
    query and filter on the same property
```

1.12 graphcore.graphcore_test module

```
class graphcore.graphcore_test.TestClause (methodName='runTest')
    Bases: unittest.case.TestCase

    test_clause_eq()

    test_convert_to_constraint()

    test_has_bound_value()

    test_has_unbound_outvar()

    test_relation()

class graphcore.graphcore_test.TestGraphcore (methodName='runTest')
    Bases: unittest.case.TestCase

    assertRetEqual (ret1, ret2)

    test_available_rules_string()

    test_basic()

    test_basic_two_step()

    test_call_filter()

    test_call_graph_ungrounded_query()

    test_call_graph_ungrounded_query_non_root()
        rules with no inputs should only be matched at the root level of the query. If we try to match book.user.id
        with user.id in this example, we'll be asserting that every book has every user.

    test_constraint_on_missing_property()
        ensure that even if a constraint, or fact isn't required to compute the output directly, it is still used to
        constrain the values.

    test_contains_query()

    test_direct_map()

    test_filter_nested_value()

    test_grounded_nested_value()

    test_has_many()

    test_has_many_and_property()

    test_long_input()

    test_long_nested_property_type()

    test_long_prefix()
```

```

test_long_rule()
test_longer_rule_first()
test_lookup_rule()
test_lookup_rule_missing()
test_lookup_rule_missing_from_node()
test_missing_rule()
test_multiple_relations()
test_nested_property_type()
test_nested_results()
test_none_result_exception_handler()
test_not_equal_dict()
test_search_outputs()
test_simple_join()
test_simple_join_and_relation()
test_simple_join_with_next_step()
test_simple_join_with_next_step_and_relation()
test_simple_join_with_next_step_and_unrelated_relation()
    the only reason to compute user.books.name is to filter on it
test_simple_nested_join()
test_simple_nested_join_multi_property()
test_three_deep_relation()
class graphcore.graphcore_test.TestQuerySearch(*args)
    Bases: unittest.case.TestCase
    test_call_graph_relation()
    test_call_graph_relation_and_outvar()
    test_call_graph_repr()
    test_call_graph_ungrounded_non_root()
    test_call_graph_ungrounded_query()
    test_call_graph_unrelated_relation()
    test_clause_with_unbound_output()
    test_clauses_with_unbound_output()
    test_explain()
    test_query_search_nested()
class graphcore.graphcore_test.hashabledict
    Bases: dict
    a hashable dict to make it easier to compare query results

```

```
graphcore.graphcore_test.make_ret_comparable(ret)
    convert lists to sets and dicts in them to frozen dicts
graphcore.graphcore_test.schema()
graphcore.graphcore_test.test_property_type_str()
graphcore.graphcore_test.test_query_nested()
graphcore.graphcore_test.test_query_nested_twice()
graphcore.graphcore_test.test_query_repr()
graphcore.graphcore_test.test_query_str()
graphcore.graphcore_test.test_schema_resolve_type_double(schema)
graphcore.graphcore_test.test_schema_resolve_type_nop(schema)
graphcore.graphcore_test.test_schema_resolve_type_simple(schema)
graphcore.graphcore_test.test_schema_resolve_type_unregistered(schema)
graphcore.graphcore_test.test_schema_str()
```

1.13 graphcore.optimize_constrain_sql_queries module

```
graphcore.optimize_constrain_sql_queries.constrain_sql_queries(call_graph)
    Move relations on SQLQuery nodes out of graphcore relations and into the where clause of the SQLQuery
```

1.14 graphcore.optimize_reduce_like_parent_child module

```
graphcore.optimize_reduce_like_parent_child.reduce_like_parent_child(call_graph,
                                                                    rule_type,
                                                                    merge_function)

    Given a call_graph, reduce parent, child nodes of rule_type using merge_function.
    Returns a modified call_graph
```

1.15 graphcore.optimize_reduce_like_parent_child_test module

```
graphcore.optimize_reduce_like_parent_child_test.list_merge(parent, child)
graphcore.optimize_reduce_like_parent_child_test.merge(parent, child, function_merge)
graphcore.optimize_reduce_like_parent_child_test.set_merge(parent, child)
graphcore.optimize_reduce_like_parent_child_test.test_reduce_like_parent_child()
graphcore.optimize_reduce_like_parent_child_test.test_reduce_like_parent_child_with_diffent
graphcore.optimize_reduce_like_parent_child_test.test_reduce_like_parent_child_with_two_ch
graphcore.optimize_reduce_like_parent_child_test.tuple_merge(parent, child)
```

1.16 graphcore.optimize_reduce_like_siblings module

```
graphcore.optimize_reduce_like_siblings.reduce_like_siblings(call_graph,
                                                             rule_type,
                                                             merge_function)
```

Given a `call_graph`, reduce sibling nodes of `rule_type` using `merge_function`.

Returns a modified `call_graph`

1.17 graphcore.optimize_reduce_like_siblings_test module

```
graphcore.optimize_reduce_like_siblings_test.reducer(nodes)
graphcore.optimize_reduce_like_siblings_test.test_reduce_like_siblings()
```

1.18 graphcore.path module

```
class graphcore.path.Path(init)
    Bases: object

    property
    relative
    subpath(root)
    subpaths()
        return all possible prefix/subpath paris.
    path = Path('a.b.c.d') path.subpaths() == [
        (('a', 'b', 'c'), Path('c.d')) (('a', 'b'), Path('b.c.d')) (('a'), Path('a.b.c.d'))
    ]
```

1.19 graphcore.path_test module

```
graphcore.path_test.test_add()
graphcore.path_test.test_init_error()
graphcore.path_test.test_init_list_element_type_error()
graphcore.path_test.test_lt()
graphcore.path_test.test_radd_fail()
graphcore.path_test.test_repr()
graphcore.path_test.test_subpaths()
```

1.20 graphcore.query module

```
class graphcore.query.Query(query)
    Bases: object

    append(clause)

    extend(query, prefix='')
        extend this Query with the query parameter type dict
        if a prefix is provided, it will be prepended to all key names

    subquery(root)
```

1.21 graphcore.query_plan module

The query plan is a sequential list of rules to apply. Other QueryPlans may in the future also handle parallel execution.

```
class graphcore.query_plan.QueryPlan(result_set, output_paths)
    Bases: object

    Execute a sequential list of nodes.

    append(node)

    execute(exception_handler=<function default_exception_handler>, limit=None)

    forward(exception_handler, limit=None)

    outputs()
```

1.22 graphcore.query_plan_test module

```
graphcore.query_plan_test.multiple_outputs(in1)
graphcore.query_plan_test.test_query_plan_multi_relation()
graphcore.query_plan_test.test_query_plan_multiple_outputs()
graphcore.query_plan_test.test_query_plan_multiple_outputs_cardinality_many()
graphcore.query_plan_test.test_query_plan_relation()
```

1.23 graphcore.query_planner module

This QueryPlanner will be very naive. Its role is to convert a CallGraph into a QueryPlan

```
class graphcore.query_planner.CallGraphIterator(call_graph)
    Bases: object

class graphcore.query_planner.QueryPlanner(call_graph, query, query_shape, mapper)
    Bases: object

    plan_query()
```


1.24 graphcore.query_test module

```
graphcore.query_test.test_contains()
```

1.25 graphcore.reflect_class module

```
graphcore.reflect_class.make_wrapped_function(name, fn)
    create a 'copy' of fn as a function with variable name obj instead of method with self
```

```
graphcore.reflect_class.reflect_class(graphcore, cls, type_name=None)
    reflect a python class cls into graphcore with name type_name
```

Assumes that elsewhere you define a rule which outputs 'type_name.obj' which will be an instance of type *cls*. It is possible that if the `__init__` is named properly, this step could be reflected as well

1.26 graphcore.reflect_class_test module

```
class graphcore.reflect_class_test.SubThing(x)
    Bases: graphcore.reflect_class_test.Thing

    foo()
```

```
class graphcore.reflect_class_test.Thing(x)
    Bases: object

    foo()
```

```
graphcore.reflect_class_test.gc()
```

```
graphcore.reflect_class_test.test_reflect_sub_thing(gc)
```

```
graphcore.reflect_class_test.test_reflect_thing(gc)
```

1.27 graphcore.reflect_module module

```
class graphcore.reflect_module.ModuleReflector(graphcore, module, type_name)
    Bases: object
```

```
graphcore.reflect_module.input_mapping_decorator(function, input_mapping)
```

1.28 graphcore.reflect_module_test module

```
graphcore.reflect_module_test.gc()
```

```
graphcore.reflect_module_test.test_double_under(gc)
```

```
graphcore.reflect_module_test.test_join(gc)
```

```
graphcore.reflect_module_test.test_join_on_arg(gc)
```

```
graphcore.reflect_module_test.test_multi_id_thing(gc)
```

```
graphcore.reflect_module_test.test_optional_arg(gc)
```

```
graphcore.reflect_module_test.test_simple_module_reflector(gc)
graphcore.reflect_module_test.test_skip_complex(gc)
graphcore.reflect_module_test.test_type_prefix(gc)
graphcore.reflect_module_test.test_verbose_arg_name(gc)
graphcore.reflect_module_test.test_verbose_function_name(gc)
```

1.29 graphcore.relation module

```
class graphcore.relation.Relation(operation, value)
    Bases: object

    merge(other)
```

1.30 graphcore.relation_test module

```
graphcore.relation_test.test_multi_relation()
graphcore.relation_test.test_multi_relation_merge()
graphcore.relation_test.test_relation_contains()
graphcore.relation_test.test_relation_eq_wrong_type()
graphcore.relation_test.test_relation_merge()
graphcore.relation_test.test_relation_ne()
graphcore.relation_test.test_relation_repr()
```

1.31 graphcore.result_set module

```
exception graphcore.result_set.NoResult
    Bases: exceptions.Exception

    rules should raise this exception if there is no result given the inputs provided. This exception will remove the
    Result which was passed in as if there were a filter applied to the ResultSet

class graphcore.result_set.NoneResult
    Bases: object

    rules or exception handlers should return this result if the value should be set to None and all clauses dependent
    on this value should be None.

class graphcore.result_set.Result(result=None, mapper=<built-in function map>)
    Bases: graphcore.equality_mixin.EqualityMixin

    apply_rule(fn, inputs, outputs, cardinality, scope, exception_handler)

    deepcopy()

    extract_json(paths)
        return the json representing paths.

        paths must already shaped like the ResultSet
```

```

get (path, default=None)
to_json ()
class graphcore.result_set.ResultSet (init=None, query_shape=None, mapper=<built-in function map>)
    Bases: graphcore.equality_mixin.EqualityMixin
    The ResultSet holds the state of the query as it is executed.
    apply_rule (fn, inputs, outputs, cardinality, scope=None, exception_handler=<function default_exception_handler>)
    deepcopy ()
    extract_json (paths)
    filter (path, relation)
    limit (limit)
        naive limit for now. won't limit sub results
    shape_path (path)
    shape_paths (paths)
    to_json ()
exception graphcore.result_set.RuleApplicationException (fn, scope, exception, traceback)
    Bases: exceptions.Exception
graphcore.result_set.default_exception_handler (result, e, fn, outputs, cardinality, scope)
graphcore.result_set.input_mapping (keys, parts=1)
    given a list of paths, return a dictionary of {path: shortened_path} where shortened_path can be used as the
    argument name when calling a function with paths inputs.

    The algorithm is to use only the right most part of the path if it is unique. If not, it uses the right 2 most parts
    seperated by an underscore. Otherwise, 3, etc.
graphcore.result_set.mapper (fn, data)
graphcore.result_set.next_sub_path (paths)
graphcore.result_set.shape_path (path, query_shape)
    return a tuple of subpaths which add together to path, but are split in the same way as the data result_set.
    shape_path([{'a.x': [{}]}], 'a.x.y.z') == ('a.x', 'y.z') shape_path([{'a.x': [{}]}], 'x.y.z') == ('x.y.z',)

```

1.32 graphcore.result_set_test module

```

graphcore.result_set_test.data ()
graphcore.result_set_test.test_apply_rule_cardinality_many (data)
graphcore.result_set_test.test_apply_rule_cardinality_many_many_outputs (data)
graphcore.result_set_test.test_apply_rule_exception (data)
graphcore.result_set_test.test_apply_rule_exception_handle ()
graphcore.result_set_test.test_apply_rule_exception_pass ()
graphcore.result_set_test.test_apply_rule_many_outputs (data)

```

```
graphcore.result_set_test.test_apply_rule_nested_none_result()
graphcore.result_set_test.test_apply_rule_none_result()
graphcore.result_set_test.test_apply_rule_none_result_exception()
graphcore.result_set_test.test_apply_rule_single_output(data)
graphcore.result_set_test.test_apply_rule_single_output_no_result(data)
graphcore.result_set_test.test_extract_json(data)
graphcore.result_set_test.test_repr()
graphcore.result_set_test.test_repr_nonbasic()
graphcore.result_set_test.test_result_eq()
graphcore.result_set_test.test_result_init()
graphcore.result_set_test.test_result_not_eq()
graphcore.result_set_test.test_result_repr()
graphcore.result_set_test.test_result_set_eq()
graphcore.result_set_test.test_result_set_extract_json_none_result()
graphcore.result_set_test.test_result_set_filter()
graphcore.result_set_test.test_result_set_init()
graphcore.result_set_test.test_result_set_nested_filter()
graphcore.result_set_test.test_result_set_to_json_none_result()
graphcore.result_set_test.test_result_to_json_none_result()
graphcore.result_set_test.test_shape_path()
graphcore.result_set_test.test_shape_path_double_dot()
graphcore.result_set_test.test_shape_path_no_match()
graphcore.result_set_test.test_shape_path_short()
graphcore.result_set_test.test_shape_paths_empty_result_set()
```

1.33 graphcore.rule module

```
class graphcore.rule.Cardinality
```

```
    Bases: enum.Enum
```

```
    static cast (init)
```

```
    many = <Cardinality.many: 2>
```

```
    one = <Cardinality.one: 1>
```

```
class graphcore.rule.Rule(function, inputs, outputs, cardinality)
```

```
    Bases: graphcore.equality_mixin.HashMixin, graphcore.equality_mixin.EqualityMixin
```

1.34 graphcore.rule_test module

```
graphcore.rule_test.test_cardinality_cast_err()
graphcore.rule_test.test_cardinality_cast_many()
graphcore.rule_test.test_rule_str()
```

1.35 graphcore.sql_query module

```
class graphcore.sql_query.SQLQuery (tables, selects, where, limit=None, one_column=False,
                                     first=False, input_mapping=None, engine=None,
                                     param_style='%s')
    Bases: graphcore.equality_mixin.HashMixin, graphcore.equality_mixin.EqualityMixin

    cleanup()
        remove clauses like 'users.id': mysql_col('users.id')

    copy()

    driver(sql, vals)

    flatten()
        merge any SQLQuery objects on the rhs of a where clause into self.

    static merge_like_siblings(nodes)

    static merge_parent_child(child, parent)
        NOTE: child and parent are switched here, it makes more sense

graphcore.sql_query.parse_comma_seperated_list(input)
graphcore.sql_query.parse_comma_seperated_set(input)
```

1.36 graphcore.sql_query_sqlalchemy module

1.37 graphcore.sql_query_test module

```
graphcore.sql_query_test.test_assert_flattenable_clause_with_no_table()
graphcore.sql_query_test.test_assert_flattenable_column_with_no_table()
graphcore.sql_query_test.test_assert_flattenable_table_alias()
graphcore.sql_query_test.test_call()
graphcore.sql_query_test.test_call_first_true()
graphcore.sql_query_test.test_call_one_column()
graphcore.sql_query_test.test_call_one_column_first_true()
graphcore.sql_query_test.test_call_with_input_mapping()
graphcore.sql_query_test.test_copy()
graphcore.sql_query_test.test_driver()
```

```
graphcore.sql_query_test.test_hash()
graphcore.sql_query_test.test_merge_parent_and_property()
graphcore.sql_query_test.test_merge_parent_and_property_multi_output()
graphcore.sql_query_test.test_merge_unbound_primary_key_and_property()
graphcore.sql_query_test.test_simple_add()
graphcore.sql_query_test.test_simple_query_merge()
graphcore.sql_query_test.test_unqualified_select()
```

1.38 graphcore.sql_reflect module

```
class graphcore.sql_reflect.SQLReflector(graphcore, engine, sql_query_class=<class 'graph-
                                         core.sql_query.SQLQuery'>, param_style='%s', ex-
                                         clude_tables=None)

    Bases: object

    sql_reflect_column(table, column_name)
```

1.39 graphcore.sql_reflect_test module

```
graphcore.sql_reflect_test.engine()
graphcore.sql_reflect_test.gc()
graphcore.sql_reflect_test.singular_table_name_engine()
graphcore.sql_reflect_test.test_sql_reflect(gc, engine)
graphcore.sql_reflect_test.test_sql_reflect_relationship(gc, singular
                                                         table_name_engine)
```

1.40 graphcore.test_harness module

```
graphcore.test_harness.book_author_id(id)
graphcore.test_harness.book_name(id)
graphcore.test_harness.user_abbreviation(name)
graphcore.test_harness.user_books_id(id)
graphcore.test_harness.user_name(id)
```

1.41 graphcore.test_module module

```
graphcore.test_module.age(user_id)
graphcore.test_module.book_ids(user_id)
graphcore.test_module.book_with_user_name(first_name, book_id)
graphcore.test_module.first_name(id)
```

```
graphcore.test_module.last_name (user_id)  
    this function takes user_id as input instead of just id  
graphcore.test_module.multi_id_thing (book_id, foo_id)  
graphcore.test_module.optionally_complex (id, const=1)  
    graphcore wont map const as an input  
graphcore.test_module.profile (first_name, location__city)  
graphcore.test_module.user_abc (user_name)  
graphcore.test_module.user_complex (id, thing, other_thing)
```

1.42 Module contents

2.1 Design Decisions

2.1.1 Option #1: QueryPlan start off with sequential list of rules (winner)

It seems a bit inefficient to build the rules first and then have to transform them to an ast later. A recursive backwards search starting from the unbound clauses does not cleanly map to an ast, since it is possible that some of the original unbound clauses will be necessary prerequisites to other clauses and so can not be assumed to be at the root of the ast. It is still possible that a recursive search will somehow make sense for some other reason, but it isnt because the ast cleanly falls out of it.

It might actually turn out to morph into the recursive ast building, but thats ok, no need to start there at the moment.

2.1.2 Option #2: QueryPlan build the ast from the get go

Does the QueryPlan even have enough information to do this? Or does it need the full list of rules first? The list of rules right now is relatively straightforward to reason about. There is no recursive search. Maybe the recursive search would be easier. The recursive search would get rid of the awkward unbound clause iterator which has to deal with the query changing underneath of it.

Indices and tables

- `genindex`
- `modindex`
- `search`

g

- graphcore, 19
- graphcore.call_graph, 3
- graphcore.call_graph_test, 4
- graphcore.clause, 4
- graphcore.clause_test, 5
- graphcore.conftest, 5
- graphcore.conftest_test, 5
- graphcore.equality_mixin, 5
- graphcore.equality_mixin_test, 6
- graphcore.graphcore, 6
- graphcore.graphcore_sql_integration_test, 7
- graphcore.graphcore_test, 8
- graphcore.optimize_constrain_sql_queries, 10
- graphcore.optimize_reduce_like_parent_child, 10
- graphcore.optimize_reduce_like_parent_child_test, 10
- graphcore.optimize_reduce_like_siblings, 11
- graphcore.optimize_reduce_like_siblings_test, 11
- graphcore.path, 11
- graphcore.path_test, 11
- graphcore.query, 12
- graphcore.query_plan, 12
- graphcore.query_plan_test, 12
- graphcore.query_planner, 12
- graphcore.query_test, 13
- graphcore.reflect_class, 13
- graphcore.reflect_class_test, 13
- graphcore.reflect_module, 13
- graphcore.reflect_module_test, 13
- graphcore.relation, 14
- graphcore.relation_test, 14
- graphcore.result_set, 14
- graphcore.result_set_test, 15
- graphcore.rule, 16
- graphcore.rule_test, 17
- graphcore.sql_query, 17
- graphcore.sql_query_test, 17
- graphcore.sql_reflect, 18
- graphcore.sql_reflect_test, 18
- graphcore.test_harness, 18
- graphcore.test_module, 18

A

add_node() (graphcore.call_graph.CallGraph method), 4
age (graphcore.graphcore_sql_integration_test.User attribute), 7
age() (in module graphcore.test_module), 18
append() (graphcore.graphcore.Rules method), 7
append() (graphcore.graphcore.Schema method), 7
append() (graphcore.query.Query method), 12
append() (graphcore.query_plan.QueryPlan method), 12
apply_rule() (graphcore.result_set.Result method), 14
apply_rule() (graphcore.result_set.ResultSet method), 15
apply_rule_backwards() (graphcore.graphcore.QuerySearch method), 7
assertRetEqual() (graphcore.graphcore_test.TestGraphcore method), 8
available_rules_string() (graphcore.graphcore.Graphcore method), 6

B

backward() (graphcore.graphcore.QuerySearch method), 7
base_types() (graphcore.graphcore.Graphcore method), 6
BaseTypeNotFound, 6
Book (class in graphcore.graphcore_sql_integration_test), 7
book_author_id() (in module graphcore.test_harness), 18
book_ids() (in module graphcore.test_module), 18
book_name() (in module graphcore.test_harness), 18
book_with_user_name() (in module graphcore.test_module), 18

C

call_graph_repr_compare() (in module graphcore.confest), 5
CallGraph (class in graphcore.call_graph), 3
CallGraphIterator (class in graphcore.query_planner), 12
Cardinality (class in graphcore.rule), 16
cast() (graphcore.rule.Cardinality static method), 16
Clause (class in graphcore.clause), 4

clause_with_unbound_outvar() (graphcore.graphcore.QuerySearch method), 7
clauses_with_unbound_outvar() (graphcore.graphcore.QuerySearch method), 7
cleanup() (graphcore.sql_query.SQLQuery method), 17
constrain_sql_queries() (in module graphcore.optimize_constrain_sql_queries), 10
convert_to_constraint() (graphcore.clause.Clause method), 4
copy() (graphcore.clause.Clause method), 4
copy() (graphcore.sql_query.SQLQuery method), 17

D

data() (in module graphcore.result_set_test), 15
deepcopy() (graphcore.result_set.Result method), 14
deepcopy() (graphcore.result_set.ResultSet method), 15
default_exception_handler() (in module graphcore.result_set), 15
define_type() (graphcore.graphcore.Graphcore method), 6
DefineTypeContext (class in graphcore.graphcore), 6
direct_map() (graphcore.graphcore.DefineTypeContext method), 6
direct_map() (graphcore.graphcore.Graphcore method), 6
driver() (graphcore.sql_query.SQLQuery method), 17

E

Edge (class in graphcore.call_graph), 4
edge() (graphcore.call_graph.CallGraph method), 4
engine() (in module graphcore.graphcore_sql_integration_test), 7
engine() (in module graphcore.sql_reflect_test), 18
EqualityMixin (class in graphcore.equality_mixin), 5
execute() (graphcore.query_plan.QueryPlan method), 12
explain() (graphcore.call_graph.CallGraph method), 4
explain() (graphcore.call_graph.Node method), 4
explain() (graphcore.graphcore.Graphcore method), 6
extend() (graphcore.query.Query method), 12
extract_json() (graphcore.result_set.Result method), 14
extract_json() (graphcore.result_set.ResultSet method), 15

F

`filter()` (`graphcore.result_set.ResultSet` method), 15
`first_name()` (in module `graphcore.test_module`), 18
`flatten()` (`graphcore.sql_query.SQLQuery` method), 17
`foo()` (`graphcore.reflect_class_test.SubThing` method), 13
`foo()` (`graphcore.reflect_class_test.Thing` method), 13
`forward()` (`graphcore.query_plan.QueryPlan` method), 12
`freeze()` (in module `graphcore.equality_mixin`), 5

G

`gc()` (in module `graphcore.graphcore_sql_integration_test`), 8
`gc()` (in module `graphcore.reflect_class_test`), 13
`gc()` (in module `graphcore.reflect_module_test`), 13
`gc()` (in module `graphcore.sql_reflect_test`), 18
`get()` (`graphcore.result_set.Result` method), 14
`Graphcore` (class in `graphcore.graphcore`), 6
`graphcore` (module), 19
`graphcore.call_graph` (module), 3
`graphcore.call_graph_test` (module), 4
`graphcore.clause` (module), 4
`graphcore.clause_test` (module), 5
`graphcore.conftest` (module), 5
`graphcore.conftest_test` (module), 5
`graphcore.equality_mixin` (module), 5
`graphcore.equality_mixin_test` (module), 6
`graphcore.graphcore` (module), 6
`graphcore.graphcore_sql_integration_test` (module), 7
`graphcore.graphcore_test` (module), 8
`graphcore.optimize_constrain_sql_queries` (module), 10
`graphcore.optimize_reduce_like_parent_child` (module), 10
`graphcore.optimize_reduce_like_parent_child_test` (module), 10
`graphcore.optimize_reduce_like_siblings` (module), 11
`graphcore.optimize_reduce_like_siblings_test` (module), 11
`graphcore.path` (module), 11
`graphcore.path_test` (module), 11
`graphcore.query` (module), 12
`graphcore.query_plan` (module), 12
`graphcore.query_plan_test` (module), 12
`graphcore.query_planner` (module), 12
`graphcore.query_test` (module), 13
`graphcore.reflect_class` (module), 13
`graphcore.reflect_class_test` (module), 13
`graphcore.reflect_module` (module), 13
`graphcore.reflect_module_test` (module), 13
`graphcore.relation` (module), 14
`graphcore.relation_test` (module), 14
`graphcore.result_set` (module), 14
`graphcore.result_set_test` (module), 15
`graphcore.rule` (module), 16
`graphcore.rule_test` (module), 17

`graphcore.sql_query` (module), 17
`graphcore.sql_query_test` (module), 17
`graphcore.sql_reflect` (module), 18
`graphcore.sql_reflect_test` (module), 18
`graphcore.test_harness` (module), 18
`graphcore.test_module` (module), 18

H

`hashabledict` (class in `graphcore.graphcore_test`), 9
`HashMixin` (class in `graphcore.equality_mixin`), 5

I

`id` (`graphcore.graphcore_sql_integration_test.Book` attribute), 7
`id` (`graphcore.graphcore_sql_integration_test.User` attribute), 7
`incoming_edges()` (`graphcore.call_graph.Node` method), 4
`incoming_nodes()` (`graphcore.call_graph.Node` method), 4
`input_mapping()` (in module `graphcore.result_set`), 15
`input_mapping_decorator()` (in module `graphcore.reflect_module`), 13
`input_path_by_property()` (`graphcore.call_graph.Node` method), 4

L

`last_name()` (in module `graphcore.test_module`), 18
`limit()` (`graphcore.result_set.ResultSet` method), 15
`list_merge()` (in module `graphcore.optimize_reduce_like_parent_child_test`), 10
`lookup()` (`graphcore.graphcore.Rules` method), 7
`lookup_rule()` (`graphcore.graphcore.Graphcore` method), 6

M

`make_ret_comparable()` (in module `graphcore.graphcore_test`), 9
`make_wrapped_function()` (in module `graphcore.reflect_class`), 13
`many` (`graphcore.rule.Cardinality` attribute), 16
`mapper()` (in module `graphcore.result_set`), 15
`merge()` (`graphcore.clause.Clause` method), 4
`merge()` (`graphcore.relation.Relation` method), 14
`merge()` (in module `graphcore.optimize_reduce_like_parent_child_test`), 10
`merge_like_siblings()` (`graphcore.sql_query.SQLQuery` static method), 17
`merge_parent_child()` (`graphcore.sql_query.SQLQuery` static method), 17
`module()` (`graphcore.graphcore.DefineTypeContext` method), 6

ModuleReflector (class in graphcore.reflect_module), 13
 multi_id_thing() (in module graphcore.test_module), 19
 multiple_outputs() (in module graphcore.query_plan_test), 12

N

name (graphcore.call_graph.Node attribute), 4
 name (graphcore.graphcore_sql_integration_test.User attribute), 7
 next() (graphcore.graphcore.QuerySearchIterator method), 7
 next_sub_path() (in module graphcore.result_set), 15
 Node (class in graphcore.call_graph), 4
 nodes_dependent_on_path() (graphcore.call_graph.CallGraph method), 4
 NoneResult (class in graphcore.result_set), 14
 NoResult, 14

O

one (graphcore.rule.Cardinality attribute), 16
 optimize() (graphcore.graphcore.Graphcore method), 6
 optionally_complex() (in module graphcore.test_module), 19
 output_paths() (graphcore.call_graph.CallGraph method), 4
 outputs() (graphcore.query_plan.QueryPlan method), 12
 OutVar (class in graphcore.clause), 5

P

parse_comma_seperated_list() (in module graphcore.sql_query), 17
 parse_comma_seperated_set() (in module graphcore.sql_query), 17
 Path (class in graphcore.path), 11
 PathNotFound, 6
 plan_query() (graphcore.query_planner.QueryPlanner method), 12
 profile() (in module graphcore.test_module), 19
 property (graphcore.path.Path attribute), 11
 property_type() (graphcore.graphcore.DefineTypeContext method), 6
 property_type() (graphcore.graphcore.Graphcore method), 6
 PropertyType (class in graphcore.graphcore), 7
 pytest_assertrepr_compare() (in module graphcore.conftest), 5

Q

Query (class in graphcore.query), 12
 query() (graphcore.graphcore.Graphcore method), 6
 QueryPlan (class in graphcore.query_plan), 12
 QueryPlanner (class in graphcore.query_planner), 12

QuerySearch (class in graphcore.graphcore), 7
 QuerySearchIterator (class in graphcore.graphcore), 7

R

reduce_like_parent_child() (in module graphcore.optimize_reduce_like_parent_child), 10
 reduce_like_siblings() (in module graphcore.optimize_reduce_like_siblings), 11
 reducer() (in module graphcore.optimize_reduce_like_siblings_test), 11
 reflect_class() (graphcore.graphcore.DefineTypeContext method), 6
 reflect_class() (in module graphcore.reflect_class), 13
 register_rule() (graphcore.graphcore.Graphcore method), 6
 Relation (class in graphcore.relation), 14
 relative (graphcore.path.Path attribute), 11
 remove_node() (graphcore.call_graph.CallGraph method), 4
 resolve_type() (graphcore.graphcore.Schema method), 7
 Result (class in graphcore.result_set), 14
 ResultSet (class in graphcore.result_set), 15
 Rule (class in graphcore.rule), 16
 rule() (graphcore.graphcore.Graphcore method), 6
 RuleApplicationException, 15
 Rules (class in graphcore.graphcore), 7

S

Schema (class in graphcore.graphcore), 7
 schema() (in module graphcore.graphcore_test), 10
 search_outputs() (graphcore.graphcore.Graphcore method), 6
 session() (in module graphcore.graphcore_sql_integration_test), 8
 set_merge() (in module graphcore.optimize_reduce_like_parent_child_test), 10
 shape_path() (graphcore.result_set.ResultSet method), 15
 shape_path() (in module graphcore.result_set), 15
 shape_paths() (graphcore.result_set.ResultSet method), 15
 singular_table_name_engine() (in module graphcore.sql_reflect_test), 18
 sql_query_repr_compare() (in module graphcore.conftest), 5
 sql_reflect_column() (graphcore.sql_reflect.SQLReflector method), 18
 SQLAlchemyQueryClass() (in module graphcore.graphcore_sql_integration_test), 7
 SQLQuery (class in graphcore.sql_query), 17
 SQLReflector (class in graphcore.sql_reflect), 18
 subpath() (graphcore.path.Path method), 11

subpaths() (graphcore.path.Path method), 11
subquery() (graphcore.query.Query method), 12
SubThing (class in graphcore.reflect_class_test), 13

T

TempVar (class in graphcore.clause), 5
test() (in module graphcore.graphcore_sql_integration_test), 8
test_add() (in module graphcore.path_test), 11
test_apply_rule_cardinality_many() (in module graphcore.result_set_test), 15
test_apply_rule_cardinality_many_many_outputs() (in module graphcore.result_set_test), 15
test_apply_rule_exception() (in module graphcore.result_set_test), 15
test_apply_rule_exception_handle() (in module graphcore.result_set_test), 15
test_apply_rule_exception_pass() (in module graphcore.result_set_test), 15
test_apply_rule_many_outputs() (in module graphcore.result_set_test), 15
test_apply_rule_nested_none_result() (in module graphcore.result_set_test), 15
test_apply_rule_none_result() (in module graphcore.result_set_test), 16
test_apply_rule_none_result_exception() (in module graphcore.result_set_test), 16
test_apply_rule_single_output() (in module graphcore.result_set_test), 16
test_apply_rule_single_output_no_result() (in module graphcore.result_set_test), 16
test_assert_flattenable_clause_with_no_table() (in module graphcore.sql_query_test), 17
test_assert_flattenable_column_with_no_table() (in module graphcore.sql_query_test), 17
test_assert_flattenable_table_alias() (in module graphcore.sql_query_test), 17
test_available_rules_string() (graphcore.graphcore_test.TestGraphcore method), 8
test_basic() (graphcore.graphcore_test.TestGraphcore method), 8
test_basic_two_step() (graphcore.graphcore_test.TestGraphcore method), 8
test_call() (in module graphcore.sql_query_test), 17
test_call_filter() (graphcore.graphcore_test.TestGraphcore method), 8
test_call_first_true() (in module graphcore.sql_query_test), 17
test_call_graph_relation() (graphcore.graphcore_test.TestQuerySearch method), 9
test_call_graph_relation_and_outvar() (graphcore.graphcore_test.TestQuerySearch method), 9
test_call_graph_repr() (graphcore.graphcore_test.TestQuerySearch method), 9
test_call_graph_repr_compare() (in module graphcore.conftest_test), 5
test_call_graph_ungrounded_non_root() (graphcore.graphcore_test.TestQuerySearch method), 9
test_call_graph_ungrounded_query() (graphcore.graphcore_test.TestGraphcore method), 8
test_call_graph_ungrounded_query() (graphcore.graphcore_test.TestQuerySearch method), 9
test_call_graph_ungrounded_query_non_root() (graphcore.graphcore_test.TestGraphcore method), 8
test_call_graph_unrelated_relation() (graphcore.graphcore_test.TestQuerySearch method), 9
test_call_one_column() (in module graphcore.sql_query_test), 17
test_call_one_column_first_true() (in module graphcore.sql_query_test), 17
test_call_with_input_mapping() (in module graphcore.sql_query_test), 17
test_cardinality_cast_err() (in module graphcore.rule_test), 17
test_cardinality_cast_many() (in module graphcore.rule_test), 17
test_clause_eq() (graphcore.graphcore_test.TestClause method), 8
test_clause_merge_relation() (in module graphcore.clause_test), 5
test_clause_merge_rhs_conflict() (in module graphcore.clause_test), 5
test_clause_with_unbound_output() (graphcore.graphcore_test.TestQuerySearch method), 9
test_clauses_with_unbound_output() (graphcore.graphcore_test.TestQuerySearch method), 9
test_constraint_on_missing_property() (graphcore.graphcore_test.TestGraphcore method), 8
test_contains() (in module graphcore.query_test), 13
test_contains_query() (graphcore.graphcore_test.TestGraphcore method), 8
test_convert_to_constraint() (graphcore.graphcore_test.TestClause method),

8	test_copy() (in module graphcore.sql_query_test), 17	8	test_long_prefix() (graphcore.graphcore_test.TestGraphcore method), 8
8	test_direct_map() (graphcore.graphcore_test.TestGraphcore method), 8	8	test_long_rule() (graphcore.graphcore_test.TestGraphcore method), 8
8	test_double_under() (in module graphcore.reflect_module_test), 13	8	test_longer_rule_first() (graphcore.graphcore_test.TestGraphcore method), 9
8	test_driver() (in module graphcore.sql_query_test), 17	9	test_lookup_rule() (graphcore.graphcore_test.TestGraphcore method), 9
4	test_edge_hash() (in module graphcore.call_graph_test), 4	9	test_lookup_rule_missing() (graphcore.graphcore_test.TestGraphcore method), 9
4	test_edge_ne() (in module graphcore.call_graph_test), 4	9	test_lookup_rule_missing_from_node() (graphcore.graphcore_test.TestGraphcore method), 9
4	test_edge_not_ne() (in module graphcore.call_graph_test), 4	11	test_lt() (in module graphcore.path_test), 11
6	test_equality_mixin() (in module graphcore.equality_mixin_test), 6	18	test_merge_parent_and_property() (in module graphcore.sql_query_test), 18
6	test_equality_mixin_with_set() (in module graphcore.equality_mixin_test), 6	18	test_merge_parent_and_property_multi_output() (in module graphcore.sql_query_test), 18
6	test_equality_mixing_other() (in module graphcore.equality_mixin_test), 6	18	test_merge_unbound_primary_key_and_property() (in module graphcore.sql_query_test), 18
9	test_explain() (graphcore.graphcore_test.TestQuerySearch method), 9	9	test_missing_rule() (graphcore.graphcore_test.TestGraphcore method), 9
4	test_explain() (in module graphcore.call_graph_test), 4	13	test_multi_id_thing() (in module graphcore.reflect_module_test), 13
16	test_extract_json() (in module graphcore.result_set_test), 16	14	test_multi_relation() (in module graphcore.relation_test), 14
8	test_filter_nested_value() (graphcore.graphcore_test.TestGraphcore method), 8	14	test_multi_relation_merge() (in module graphcore.relation_test), 14
8	test_grounded_nested_value() (graphcore.graphcore_test.TestGraphcore method), 8	9	test_multiple_relations() (graphcore.graphcore_test.TestGraphcore method), 9
8	test_has_bound_value() (graphcore.graphcore_test.TestClause method), 8	9	test_nested_property_type() (graphcore.graphcore_test.TestGraphcore method), 9
8	test_has_many() (graphcore.graphcore_test.TestGraphcore method), 8	9	test_nested_results() (graphcore.graphcore_test.TestGraphcore method), 9
8	test_has_many_and_property() (graphcore.graphcore_test.TestGraphcore method), 8	9	test_none_result_exception_handler() (graphcore.graphcore_test.TestGraphcore method), 9
8	test_has_unbound_outvar() (graphcore.graphcore_test.TestClause method), 8	9	test_not_equal_dict() (graphcore.graphcore_test.TestGraphcore method), 9
17	test_hash() (in module graphcore.sql_query_test), 17	13	test_optional_arg() (in module graphcore.reflect_module_test), 13
11	test_init_error() (in module graphcore.path_test), 11	10	test_property_type_str() (in module graphcore.graphcore_test), 10
11	test_init_list_element_type_error() (in module graphcore.path_test), 11		
13	test_join() (in module graphcore.reflect_module_test), 13		
13	test_join_on_arg() (in module graphcore.reflect_module_test), 13		
8	test_long_input() (graphcore.graphcore_test.TestGraphcore method), 8		
8	test_long_nested_property_type() (graphcore.graphcore_test.TestGraphcore method), 8		

`test_pytest_assertrepr_compare_call_graph()` (in module `graphcore.conftest_test`), 5

`test_pytest_assertrepr_compare_sql_query()` (in module `graphcore.conftest_test`), 5

`test_query_and_filter()` (in module `graphcore.graphcore_sql_integration_test`), 8

`test_query_nested()` (in module `graphcore.graphcore_test`), 10

`test_query_nested_twice()` (in module `graphcore.graphcore_test`), 10

`test_query_plan_multi_relation()` (in module `graphcore.query_plan_test`), 12

`test_query_plan_multiple_outputs()` (in module `graphcore.query_plan_test`), 12

`test_query_plan_multiple_outputs_cardinality_many()` (in module `graphcore.query_plan_test`), 12

`test_query_plan_relation()` (in module `graphcore.query_plan_test`), 12

`test_query_repr()` (in module `graphcore.graphcore_test`), 10

`test_query_search_nested()` (`graphcore.graphcore_test.TestQuerySearch` method), 9

`test_query_str()` (in module `graphcore.graphcore_test`), 10

`test_radd_fail()` (in module `graphcore.path_test`), 11

`test_reduce_like_parent_child()` (in module `graphcore.optimize_reduce_like_parent_child_test`), 10

`test_reduce_like_parent_child_with_diffent_type()` (in module `graphcore.optimize_reduce_like_parent_child_test`), 10

`test_reduce_like_parent_child_with_two_children()` (in module `graphcore.optimize_reduce_like_parent_child_test`), 10

`test_reduce_like_siblings()` (in module `graphcore.optimize_reduce_like_siblings_test`), 11

`test_reflect_sub_thing()` (in module `graphcore.reflect_class_test`), 13

`test_reflect_thing()` (in module `graphcore.reflect_class_test`), 13

`test_relation()` (`graphcore.graphcore_test.TestClause` method), 8

`test_relation_contains()` (in module `graphcore.relation_test`), 14

`test_relation_eq_wrong_type()` (in module `graphcore.relation_test`), 14

`test_relation_merge()` (in module `graphcore.relation_test`), 14

`test_relation_ne()` (in module `graphcore.relation_test`), 14

`test_relation_repr()` (in module `graphcore.relation_test`), 14

`test_repr()` (in module `graphcore.clause_test`), 5

`test_repr()` (in module `graphcore.path_test`), 11

`test_repr()` (in module `graphcore.result_set_test`), 16

`test_repr_nonbasic()` (in module `graphcore.result_set_test`), 16

`test_result_eq()` (in module `graphcore.result_set_test`), 16

`test_result_init()` (in module `graphcore.result_set_test`), 16

`test_result_not_eq()` (in module `graphcore.result_set_test`), 16

`test_result_repr()` (in module `graphcore.result_set_test`), 16

`test_result_set_eq()` (in module `graphcore.result_set_test`), 16

`test_result_set_extract_json_none_result()` (in module `graphcore.result_set_test`), 16

`test_result_set_filter()` (in module `graphcore.result_set_test`), 16

`test_result_set_init()` (in module `graphcore.result_set_test`), 16

`test_result_set_nested_filter()` (in module `graphcore.result_set_test`), 16

`test_result_set_to_json_none_result()` (in module `graphcore.result_set_test`), 16

`test_result_to_json_none_result()` (in module `graphcore.result_set_test`), 16

`test_rule_str()` (in module `graphcore.rule_test`), 17

`test_schema_resolve_type_double()` (in module `graphcore.graphcore_test`), 10

`test_schema_resolve_type_nop()` (in module `graphcore.graphcore_test`), 10

`test_schema_resolve_type_simple()` (in module `graphcore.graphcore_test`), 10

`test_schema_resolve_type_unregistered()` (in module `graphcore.graphcore_test`), 10

`test_schema_str()` (in module `graphcore.graphcore_test`), 10

`test_search_outputs()` (`graphcore.graphcore_test.TestGraphcore` method), 9

`test_shape_path()` (in module `graphcore.result_set_test`), 16

`test_shape_path_double_dot()` (in module `graphcore.result_set_test`), 16

`test_shape_path_no_match()` (in module `graphcore.result_set_test`), 16

`test_shape_path_short()` (in module `graphcore.result_set_test`), 16

`test_shape_paths_empty_result_set()` (in module `graphcore.result_set_test`), 16

`test_simple_add()` (in module `graphcore.sql_query_test`), 18

`test_simple_join()` (`graphcore.graphcore_test.TestGraphcore` method),

[9](#)
 test_simple_join_and_relation() (graphcore.graphcore_test.TestGraphcore method), [9](#)
 test_simple_join_with_next_step() (graphcore.graphcore_test.TestGraphcore method), [9](#)
 test_simple_join_with_next_step_and_relation() (graphcore.graphcore_test.TestGraphcore method), [9](#)
 test_simple_join_with_next_step_and_unrelated_relation() (graphcore.graphcore_test.TestGraphcore method), [9](#)
 test_simple_module_reflector() (in module graphcore.reflect_module_test), [13](#)
 test_simple_nested_join() (graphcore.graphcore_test.TestGraphcore method), [9](#)
 test_simple_nested_join_multi_property() (graphcore.graphcore_test.TestGraphcore method), [9](#)
 test_simple_query_merge() (in module graphcore.sql_query_test), [18](#)
 test_skip_complex() (in module graphcore.reflect_module_test), [14](#)
 test_sql_query_repr_compare() (in module graphcore.conftest_test), [5](#)
 test_sql_reflect() (in module graphcore.sql_reflect_test), [18](#)
 test_sql_reflect_relationship() (in module graphcore.sql_reflect_test), [18](#)
 test_str() (in module graphcore.clause_test), [5](#)
 test_subpaths() (in module graphcore.path_test), [11](#)
 test_three_deep_relation() (graphcore.graphcore_test.TestGraphcore method), [9](#)
 test_type_prefix() (in module graphcore.reflect_module_test), [14](#)
 test_unqualified_select() (in module graphcore.sql_query_test), [18](#)
 test_verbose_arg_name() (in module graphcore.reflect_module_test), [14](#)
 test_verbose_function_name() (in module graphcore.reflect_module_test), [14](#)
 TestClass (class in graphcore.equality_mixin_test), [6](#)
 TestClause (class in graphcore.graphcore_test), [8](#)
 TestGraphcore (class in graphcore.graphcore_test), [8](#)
 TestQuerySearch (class in graphcore.graphcore_test), [9](#)
 Thing (class in graphcore.reflect_class_test), [13](#)
 to_json() (graphcore.result_set.Result method), [15](#)
 to_json() (graphcore.result_set.ResultSet method), [15](#)
 tuple_merge() (in module graphcore.optimize_reduce_like_parent_child_test), [10](#)

U

User (class in graphcore.graphcore_sql_integration_test), [7](#)
 user_abbreviation() (in module graphcore.test_harness), [18](#)
 user_abc() (in module graphcore.test_module), [19](#)
 user_books_id() (in module graphcore.test_harness), [18](#)
 user_complex() (in module graphcore.test_module), [19](#)
 user_id (graphcore.graphcore_sql_integration_test.Book attribute), [7](#)
 user_name() (in module graphcore.test_harness), [18](#)

V

Var (class in graphcore.clause), [5](#)