Quark DB Infrastructure

The Generic Query Engine of Quark, also called the Quark DB Infrastructure, forms the core data model-independent engine of Quark that is designed to work without caring about the underlying data model implementation.

The Quark DB Infrastructure contains the set of components that can form a query engine and work together to take a graph that represents the query and return the result of its evaluation. They are:

- **YQGM**
  - An abbreviation for Yet another Query Graph Model, this is the graph used in Quark. It is independent of the data model and can support XML, relational, and other data models. YQGM can be logically divided into:
    - **Generic YQGM** (yqgm)
      - This is the implementation of the building blocks of YQGM graphs. Every entity in a YQGM graph is either an operator (a processing unit) or a quantifier (a connector that passes data between operators).
    - **Standard YQGM** (yqgm_std)
      - This is the implementation of specific YQGM operators such as the Group By, SPJ (Select-Project-Join), LOJ (Left Outer Join), Union, Unnest, and Order-By operators.

- **Rewriter** (yqgm_rules)
  - This takes a YQGM graph and applies a set of rewrite rules to simplify the graph. The objective of the rewrite rules is to apply some transformations to the graph which preserve the semantics of the query but simplify the graph by removing some operators and quantifiers. Each rewrite rule is simple in its domain of applicability and its transformation action, but when all the rewrite rules are combined, the YQGM graphs can be greatly simplified. The different rewrite rules can be applied in a round-robin or priority-wise fashion depending on whether the rules are basic simplification rules or advanced rules.

- **Storage**
  - This handles the actual storage of data by providing a set of Storage Query Units (SQUs). SQUs perform the functions of storing a piece of data and returning an ID corresponding to it, retrieving the piece of data stored when given its ID, and returning optimized evaluation plans for a part of a YQGM graph given the root of some subtree of the graph.

- **Optimizer Infrastructure** (optimizer)
  - The optimizer takes the rewritten YQGM graph and uses the Storage component to determine an optimized query evaluation plan.
    - **SQU Optimizer** (optimizer_squ)
      - This interacts with the SQUs in the Storage component to determine an optimized evaluation plan. It queries each SQU for the list of candidate evaluation plans and stitches these together to generate an optimized query execution plan.
**Evaluator Infrastructure** – The evaluator converts the YQGM graph to a physical evaluation graph and evaluates it. The evaluator infrastructure has a number of libraries:

- **Standard Evaluator Infrastructure** (evaluator_std) - This component is the concrete implementation of the generic evaluator in the Quark DB Infrastructure and converts the YQGM graph into a physical graph and then performs the evaluation on it.
- **Iterator Evaluator** (iterator_evaluator) – This is a special implementation of the standard evaluator where each operator's quantifiers are evaluated from left to right. The data is fetched from a quantifier until it doesn’t have any tuples to return, in which case the quantifier is reset, and the next tuple is fetched from the previous quantifier, and so on, until the first quantifier has no tuples left.

Suggested usage of components

---

**Component Dependency Graph**

![Component Dependency Graph](image-url)