Extending LQDAG for optimization of recursive queries

I. Introduction

We consider the problem of optimizing recursive queries over graphs. We propose to extend the logical query DAG (LQDAG) used in modern query optimizers with the support of recursion and appropriate techniques to transform recursive terms of the extended LQDAG. Our LQDAG extension is based on the fixpoint operator recently introduced in the μ-relational algebra. We show how to adapt the transformation rules for individual fixpoint terms so that they apply on a group of terms at once, leveraging the factorized representation of terms. The main benefit is to make possible a much more efficient exploration of query execution plan spaces. We report on practical experiments with UCRPQ queries over real and synthetic graphs of varying size, while comparing with state-of-the-art implementations. Experiments illustrate the benefits of exploring larger query execution plan spaces.

II. Preliminaries

A. LQDAG

Logical Query DAG (LQDAG), is a data structure used to generate the logical plan space of a given query. It was introduced in [2] and also used in [4] for detecting and unifying the common subexpressions for multi-query optimization.

B. Optimization rules in Relational Algebra (ordered by algebraic operator)

1) Filter possibilities:

Rule	Constraint
$\sigma_\theta(\varphi \bowtie \psi) \rightarrow \sigma_\theta(\varphi) \bowtie \psi$	$FC(\theta) \not\subset \tau$ with $\psi : \tau$
$\sigma_\theta(\varphi \bowtie \psi) \rightarrow (\varphi) \bowtie \sigma_\theta(\psi)$	$FC(\theta) \not\subset \tau$ with $\varphi : \tau$
$\sigma_\theta(\varphi \cup \psi) \rightarrow \sigma_\theta(\varphi) \cup \sigma_\theta(\psi)$	-
$\sigma_\theta(\widetilde{\pi}_S(\varphi)) \rightarrow \widetilde{\pi}_S(\sigma_\theta(\varphi))$	-
$\sigma_a(\sigma_b(\varphi)) \leftrightarrow \sigma_b(\sigma_a(\varphi))$	-

2) Antiprojection possibilities:

Rule	Constraint
$\widetilde{\pi}_S(\varphi \cup \psi) \rightarrow \widetilde{\pi}_S(\varphi) \cup \widetilde{\pi}_S(\psi)$	-
$\widetilde{\pi}_S(\varphi \bowtie \psi) \rightarrow \varphi \bowtie \pi_S(\psi)$	$FC(S) \not\subset \tau$ with $\varphi : \tau$
$\widetilde{\pi}_S(\varphi \bowtie \psi) \leftrightarrow \widetilde{\pi}_S(\varphi) \bowtie \psi$	$FC(S) \not\subset \tau$ with $\psi : \tau$
$\widetilde{\pi}_S(\sigma_\theta(\varphi)) \rightarrow \sigma_\theta(\widetilde{\pi}_S(\varphi))$	$FC(S) \subset \tau$ with $\psi : \tau$ *and* $S \subset \theta$
$\widetilde{\pi}_{S_1}\widetilde{\pi}_{S_2}(\varphi) \rightarrow \widetilde{\pi}_{S_2}(\widetilde{\pi}_{S_1}(\varphi))$	-

a, b $\rightarrow$ columns

φ	::=	term
	$\lvert c \rightarrow v \rvert$	constant
$\lvert$	$\sigma_\theta(\varphi)$	filter
$\lvert$	$\varphi_1 \bowtie \varphi_2$	join
$\lvert$	$\varphi_1 \triangleright \varphi_2$	antijoin
$\lvert$	$\varphi_1 \cup \varphi_2$	union
$\lvert$	$\rho_a^b(\varphi)$	rename
$\lvert$	$\widetilde{\pi}_a(\varphi)$	antiprojection
$\lvert$	X	relation variable
$\lvert$	$\mu X.\, \varphi_{const} \cup \varphi_{rec}$	fixpoint

θ definition:

θ	::=	term
	$c = v$	
$\lvert$	$\theta_1 \cap \theta_2$	
$\lvert$	$\theta_1 \cup \theta_2$	
$\lvert$	$\neg\theta$	
$\lvert$	$\theta_1\ eq\ \theta_2$	

*eq refers to all the following cases ($=$, $\neq$, $<$, $\leq$, $>$, $\geq$)

Fig. 1: Syntax of μ-RA.

3) Join possibilities:

Rule	Explanation
$\varphi \bowtie (\psi \bowtie \chi) \leftrightarrow (\varphi \bowtie \psi) \bowtie \chi$	Associativity of Join
$\varphi \bowtie \psi \leftrightarrow \psi \bowtie \varphi$	Commutativity of Join
$\varphi \bowtie (\psi \cup \gamma) \rightarrow (\varphi \bowtie \psi) \cup (\varphi \cup \gamma)$	Distributivity of Join over Union
$\varphi \bowtie \sigma_\theta(\psi) \rightarrow \sigma_\theta(\varphi \bowtie \psi)$	Pushing Join inside Filter
$\varphi \bowtie \widetilde{\pi}_S(\psi) \rightarrow \widetilde{\pi}_S(\varphi \bowtie \psi)$	Pushing Join inside Remove

4) Union possibilities:

Rule	Explanation
$\varphi \cup (\psi \cup \chi) \leftrightarrow (\varphi \cup \psi) \cup \chi$	Associativity of Union
$\varphi \cup \psi \leftrightarrow \psi \cup \varphi$	Commutativity of Union

C. μ-RA terms [3]

1) Syntax μ-RA terms: In Figure 1 is the classical Algebra introduced by Codd [1] with the addition of Fixpoints presented in [3].

2) Semantics μ-RA terms: In Figure 2 we can find the semantics of μ-RA.

Constant	$[\![\|c \to v\|]\!]_V$	$=$	$\{\{c \to v\}\}$
Relation Variable	$[\![X]\!]_V$	$=$	$V(X)$
Filter	$[\![\sigma_f(\varphi)]\!]_V$	$=$	$\{m \,\|\, m \in [\![\varphi]\!]_V \wedge f(m) = \top\}$
Join	$[\![\varphi_1 \bowtie \varphi_2]\!]_V$	$=$	$\{m_1 + m_2 \,\|\, m_1 \in [\![\varphi_1]\!]_V \wedge m_2 \in [\![\varphi_2]\!]_V \wedge m_1 \sim m_2\}$
AntiJoin	$[\![\varphi_1 \triangleright \varphi_2]\!]_V$	$=$	$\{m \in [\![\varphi_1]\!]_V \,\|\, \forall m' \in [\![\varphi_2]\!]_V \ \neg(m' \sim m)\}$
Union	$[\![\varphi_1 \cup \varphi_2]\!]_V$	$=$	$[\![\varphi_1]\!]_V \cup [\![\varphi_2]\!]_V$
Rename	$[\![\rho_a^b(\varphi)]\!]_V$	$=$	$\{\{c \to v \in m \,\|\, c \neq a\} \cup \{b \to v \,\|\, a \to v \in m\} \mid m \in [\![\varphi]\!]_V\}$
AntiProjection	$[\![\widetilde{\pi}_a(\varphi)]\!]_V$	$=$	$\{\{c \to v \in m \,\|\, c \neq a\} \,\|\, m \in [\![\varphi]\!]_V\}$
Fixpoint	$[\![\mu(X = \varphi)]\!]_V$	$=$	$[\![X]\!]_{V[X/U_\infty]}$ where $U_0 = \emptyset, U_{i+1} = U_i \cup [\![\varphi]\!]_{V[X/U_i]}$, and $U_\infty = \bigcup_{n \in \mathbb{N}} U_i$

Fig. 2: Semantics of μ-RA.

3) Recalling some important definitions and notions from [3]:

a) Some assumptions:

- The semantics of a term φ depends on an environment V which maps all free variables of φ to relations.
- We assume that for any filter f we can compute a set $FC(f)$ of column names such that the result of $f(m)$ depends only on $c \to m(c) \mid c \in FC(f)$.
- We assume that $\mathcal{F}[\Gamma]$ is : the set of all well-typed terms in the schema Γ.
- We assume $\mathfrak{C}$ as an infinite set of column names.

b) Definition 4: In a term φ, all occurrences of a variable X which appear in a subterm of the form $\mu(X = \psi)$ are bound. All other occurrences of X are free.

c) Definition 5: Given a term φ, we say that φ is constant in X when X is not a free variable of φ.

d) Definition 6: A fixpoint term $\mu(X = \varphi)$ is said:

- positive when for all the subterms $\varphi_1 \triangleright \varphi_2$ of φ, φ_2 is constant in X;
- linear when for all subterms of φ of the form $\varphi_1 \bowtie \varphi_2$ or $\varphi_1 \triangleright \varphi_2$ is constant in X;
- mutually recursive when there exists a subterm $\mu(Y = \psi)$ of φ with X free in ψ.

e) Definition 9: The set of derivations $d(\psi, X)$ is:

$$
\begin{array}{lcl}
d(\psi_1 \cup \psi_2, X) & = & d(\psi_1, X) \cup d(\psi_2, X) \\
d(\psi_1 \triangleright \psi_2, X) & = & d(\psi_1, X) \\
d(\psi_1 \bowtie \psi_2, X) & = & d(\psi_1, X) \cup d(\psi_2, X) \\
d(\rho_a^b(\psi), X) & = & \{p \circ (b \to a, a \to \bot) \mid p \in d(\psi, X)\} \\
d(\widetilde{\pi}_a(\psi), X) & = & \{p \circ (a \to \bot) \mid p \in d(\psi, X)\} \\
d(\sigma_f(\psi), X) & = & d(\psi, X) \\
d(\mu(Y = \psi), X) & = & \emptyset \\
d(X, X) & = & \{()\} \ \text{(a singleton identity)} \\
d(X, R) & = & \emptyset \\
d(|c \to c|, X) & = & \emptyset
\end{array}
$$

Where $\circ$ represents the composition and $(a_1 \to b_1, ..., a_n \to b_n)$ represents the function that maps each a_i to its b_i and every other column name to itself. Note that this definition manipulates functions with an infinite domain, but the domain where they do not coincide with the identity is finite and they are thus computable.

f) Definition 10: Given a term φ linear and positive in a variable X, we define the stabilizer of X in ψ as the following set of column names: $stab(\psi, X) = \{c \in \mathfrak{C} | \ \forall p \in d(\psi, X) \ p(c) = c\}$.

g) Definition 11: We say that a column $c \in \mathfrak{C}$ can be added to or removed from a term $\psi \in \mathcal{F}[\Gamma]$ recursive in X when $add(\psi, X, c) = \top$ holds, with add defined as:

$$
\begin{array}{lcl}
add(\psi_1 \cup \psi_2, X, c) & = & add(\psi_1, X, c) \wedge add(\psi_2, X, c) \\
add(\psi_1 \bowtie \psi_2, X, c) & = & add(\psi_1, X, c) \wedge add(\psi_2, X, c) \\
add(\psi_1 \triangleright \psi_2, X, c) & = & add(\psi_1, X, c) \wedge add(\psi_2, X, c) \\
add(\rho_a^b(\psi), X, c) & = & add(\psi, X, c) \wedge c \notin \{a, b\} \\
add(\widetilde{\pi}_a(\psi), X, c) & = & add(\psi, X, c) \text{ when } c \neq a \\
add(\widetilde{\pi}_c(\psi), X, c) & = & X \notin free(\psi) \\
add(\sigma_f(\psi), X, c) & = & add(\psi, X, c) \wedge c \notin FC(f) \\
add(\mu(Y = \psi), X, c) & = & add(\psi, X, c) \\
add(R, X, c) & = & c \notin \Gamma(R) \text{ when } X \neq R \\
add(X, X, c) & = & \top \\
add(|c' \rightarrow v|, X, c) & = & c \neq c'
\end{array}
$$

h) LEMMA 4: Let $\mu(X = \kappa \cup \psi) \in \mathcal{F}[\Gamma]$ be a decomposed fixpoint of type t, let $c \in (\mathfrak{C} \backslash t)$ that can be added to ψ, and w a mapping of type t. We note $w(v) = w \cup \{c \rightarrow w\}$.
If $\forall R \in R, c \notin \Gamma(R)$, then we have:

- $c \in stab\{\psi, X\}$
- $\Gamma \cup \{X \rightarrow t \cup \{c\}\} \vdash \psi : t \cup \{c\}$
- $[\![\psi \bowtie |c \rightarrow v|]\!]_{V[X/\{w\}]} = [\![\psi]\!]_{V[X/\{w(v)\}]}$

4) Rewrite rules considering μ-RA terms [3]: In this sections we will recall the 5 rewrite rules:

a) Pushing Filters inside a fixpoint. [Theorem 1 from [3]]:
Let $\mu(X. \kappa \cup \psi)$ be a decomposed fixpoint term, v an environment and f a filter condition with $FC(f) \subseteq stab(\psi, X)$. then we have: $[\![\sigma_f(\mu(X. \kappa \cup \psi))]\!]_v = [\![\mu(X. \sigma_f(\kappa) \cup \psi)]\!]_v$.

b) Pushing AntiJoins inside a fixpoint. [Theorem 2 from [3]]:
Let $\mu(X. \kappa \cup \psi)$ be a decomposed fixpoint term, v an environment and φ a term of type $t \subseteq stab(\psi, X)$ (we suppose that X is not a free variable of φ). Then we have: $[\![\mu(X. \kappa \cup \psi) \triangleright \varphi]\!]_v = [\![\mu(X. \kappa \triangleright \varphi \cup \psi)]\!]_v$.

c) Pushing Joins inside a fixpoint. [Theorem 3 from [3]]:

Let $\mu(X. \kappa \cup \psi) \in \mathcal{F}[\Gamma]$ be a decomposed fixpoint term of type t_k and $\varphi \in \mathcal{F}[\Gamma]$ (with $X \notin free(\varphi)$) a term of type t_φ such that:
(1) $t_\varphi \subseteq stab(\psi, X)$
(2) $\forall c \in t_\varphi \backslash t_k \; add(\psi, X, c)$
Then we have $\Gamma \vdash \mu(X. \kappa \bowtie \varphi \cup \psi) : t_\varphi \cup t_k$ with $\forall v$ compatible with Γ:
$[\![\mu(X. \kappa \cup \psi) \bowtie \varphi]\!]_v = [\![\mu(X. \kappa \bowtie \varphi \cup \psi)]\!]_v$.

d) Merging fixpoints. [Theorem 4 from [3]]:
Given two decomposed fixpoints $\mu(X. \kappa_1 \cup \psi_1)$ and $\mu(X. \kappa_2 \cup \psi_2)$ of types t_1 and t_2 such that:
(1) $t_1 \cap t_2 \subseteq stab(\psi_2, X, C_2) \cap stab(\psi_1, X, C_1)$
(2) $\forall c \in t_1 \backslash t_2 \; add(\psi_2, X, c)$
(3) $\forall c \in t_2 \backslash t_1 \; add(\psi_1, X, c)$
then we have: $[\![\mu(X. \kappa_1 \cup \psi_1) \bowtie \mu(X. \kappa_2 \cup \psi_2)]\!]_v = [\![\mu(X. \kappa_1 \bowtie \kappa_2 \cup \psi_1 \cup \psi_2)]\!]_v$.

e) Pushing Antiprojections inside a fixpoint. [Theorem 5 from [3]]:
Let $\mu(X. \kappa \cup \psi) \in \mathcal{F}[\Gamma]$ be a decomposed fixpoint term of type t_k. Let $c \in \mathfrak{C}$ be such that $add(\psi, X, c)$. Then:
$[\![\widetilde{\pi}_c(\mu(X. \kappa \cup \psi))]\!]_v = [\![\mu(X. \widetilde{\pi}_c(\kappa) \cup \psi)]\!]_v$

III. Does stab change after applying the 5 SIGMOD'20 theorems concerning fixpoints?

A. Stab in Pushing Filters in a Fixpoint

Theorem 1: Let's consider a term t of the form $\sigma_f(\mu X.\kappa \cup \psi)$ with $FC(f) \subseteq stab(\psi, X)$; a term t' of the form $\mu X_1.\sigma_f(\kappa) \cup \psi_1$ where $\psi_1 = \psi[X/X_1]$ obtained by Theorem 1 of [3]. Then we have $stab(\psi, X) = stab(\psi_1, X_1)$.
Proof:
For term t, $stab(\psi, X)$ is calculated (based on Definition 10 of [3]) in the recursive part of the decomposed fixpoint.
In term t', the filter was pushed on the constant side of the fixpoint, leaving the recursive part ψ_1 unchanged. Based on Definition 10 of [3], the $stab(\psi_1, X_1)$ of the term t' does not depend on the constant part $\sigma_f(\kappa)$ of the fixpoint, but is calculated on its recursive part ψ_1. So, we have $stab(\psi, X) = stab(\psi_1, X_1)$.

B. Property of stab

typ_ψ: The set of column names of a term ψ.

a) Property: Let's consider the recursive parts ψ and ψ_1 two different μ-RA terms; the derivation $d(X, X)$ from the set of derivations of Definition 9. Then, we have $stab(\psi, X) = stab(\psi_1, X_1)$.

Proof:
Definition 9, is the set of derivations of $d(\psi, X)$.
$d(X, X) = \{()\}$, so considering this particular case, $d(\psi, X)$ is (X, X), which is $\{()\}$ (a singleton identity). Continuing with this derivation, for ψ_1: $d(\psi_1, X_1) = d(X_1, X_1) = \{()\}$. This shows that $d(\psi, X) = d(\psi_1, X_1)$ where $\psi_1 = \psi[X/X_1]$. So, we show that $stab(\psi, X) = stab(\psi_1, X_1)$.

C. Stab in Pushing AntiJoin in a Fixpoint

Theorem 2: Let's consider a term t of the form $\mu(X. \kappa \cup \psi) \triangleright \varphi$ with $typ_\varphi \subseteq stab(\psi, X)$ (we suppose that X is not a free variable of φ); a term t' of the form $\mu(X_1. \kappa \triangleright \varphi \cup \psi_1)$ obtained by Theorem 2 of [3]. Then we have $stab(\psi, X) = stab(\psi_1, X_1)$.

Proof:
For term t, $stab(\psi, X)$ is calculated (based on Definition 10 of [3]) in the recursive part of the decomposed fixpoint.
In term t', the antijoin with the term φ happened on the constant side of the fixpoint, leaving the recursive part ψ_1 unchanged. Based on Definition 10 of [3], the $stab(\psi_1, X_1)$ of the term t' does not depend on the constant part of the fixpoint, but is calculated on its recursive part ψ_1, and based

on Property III-B, the calculation of stabilizer is invariant. So, we have $stab(\psi, X) = stab(\psi_1, X_1)$.

D. Stab in Pushing Join in a Fixpoint

Theorem 3: Let's consider a term t of the form $\mu(X.\ \kappa \cup \psi) \bowtie \varphi$ with φ of type $typ \subseteq stab(\psi, X)$ (with $X \notin free(\varphi)$) a term of type typ_φ such that:
(1) $typ_\varphi \subseteq stab(\psi, X)$
(2) $\forall c \in typ_\varphi \backslash typ\ \ add(\psi, X, c)$; a term t' of the form $\mu(X_1.\ (\kappa \bowtie \varphi) \cup \psi_1)$ obtained by Theorem 3 of [3]. Then we have $stab(\psi, X) = stab(\psi_1, X_1)$.

Proof:
For term t, $stab(\psi, X)$ is calculated (based on Definition 10 of [3]) in the recursive part of the decomposed fixpoint.
In term t', the join with the term φ happened on the constant side of the fixpoint, leaving the recursive part ψ_1 unchanged. Based on Definition 10 of [3], the $stab(\psi_1, X_1)$ of the term t' does not depend on the constant part of the fixpoint, but is calculated on its recursive part ψ_1, and based on Property III-B, the calculation of stabilizer is invariant. So, we have $stab(\psi, X) = stab(\psi_1, X_1)$.

E. Stab in Merging Fixpoints

Theorem 4: Let's consider a term t of the form $\mu(X_1.\ \kappa_1 \cup \psi_1)$ with type t_1; a term t' of the form $\mu(X_2.\ \kappa_2 \cup \psi_2)$ with type t_2 such that:
(1) $t_1 \cap t_2 \subseteq stab(\psi_2, X, C_2) \cap stab(\psi_1, X, C_1)$
(2) $\forall c \in t_1 \backslash t_2\ add(\psi_2, X, c)$
(3) $\forall c \in t_2 \backslash t_1\ add(\psi_1, X, c)$
; a term t" obtained by $t \bowtie t'$ which by Theorem 4 of [3] is $\mu(X.\ (\kappa_1 \bowtie \kappa_2) \cup \psi_1 \cup \psi_2)$. So, we have $stab((\psi_1 \cup \psi_2), X) = stab(\psi_1, X_1) \cup stab(\psi_2, X_2)$.

Proof: For term t, $stab(\psi_1, X_1)$ is calculated (based on Definition 10 of [3]) in the recursive part ψ_1 of the decomposed fixpoint.
For term t', $stab(\psi_2, X_2)$ is calculated (based on Definition 10 of [3]) in the recursive part ψ_2 of the decomposed fixpoint.
The new term t", has a new stab $stab((\psi_1 \cup \psi_2), X)$. Based on the set of derivations found in Definition 9, $d(\psi_1 \cup \psi_2, X) = d(\psi_1, X) \cup d(\psi_2, X)$ which allows us to calculate the new stab as the union of the previous stabilizers of the two terms $stab(\psi_1, X_1) \cup stab(\psi_2, X_2)$.

F. Stab in Pushing AntiProjection in a Fixpoint

Theorem 5: Let's consider a term t of the form $\widetilde{\pi}_c(\mu(X.\ \kappa \cup \psi))$ of type t_k and $c \in \mathfrak{C}$ be such that $add(\psi, X, c)$; a term t' of the form $\mu(X.\ \widetilde{\pi}_c(\kappa) \cup \psi)$. Then we have $stab(\psi, X) = stab(\psi_1, X_1)$.
Proof:
For term t, $stab(\psi, X)$ is calculated (based on Definition 10 of [3]) in the recursive part of the decomposed fixpoint.
In term t', the antiprojection happened on the constant side of the fixpoint, leaving the recursive part ψ_1 unchanged. Based on Definition 10 of [3], the $stab(\psi_1, X_1)$ of the term t' does not depend on the constant part $\widetilde{\pi}_c(\kappa)$ of the fixpoint, but is calculated on its recursive part ψ_1, and based on Property III-B, the calculation of stabilizer is invariant. So, we have $stab(\psi, X) = stab(\psi_1, X_1)$.

G. Stab when applying Relational Algebra rewrite rules [1]

Note: All the terms considered in μ-RA are linear, positive and non mutually recursive.

We consider the subset of μ-terms produced by the translation of UCRPQ queries into the μ-algebra, as found in [3].
The General Form of such a term: $\mu(X.\ \kappa \cup \psi_1 \cup \psi_2 \ldots \cup \psi_n)$.
How this form is calculated:
Let's take the terms: $\mu(X_1.\ R_1 \cup \widetilde{\pi}_m(\rho^m_{trg}(R_1) \bowtie \rho^m_{src}(X_1))) \bowtie \mu(X_2.\ R_2 \cup \widetilde{\pi}_m(\rho^m_{trg}(R_2) \bowtie \rho^m_{src}(X_2))) \ldots \bowtie \mu(X_n.\ R_n \cup \widetilde{\pi}_m(\rho^m_{trg}(R_n) \bowtie \rho^m_{src}(X_n)))$.
This term can be written:
$\mu(X.\ (R_1 \bowtie R_2 \ldots \bowtie R_n) \cup \widetilde{\pi}_m(\rho^m_{trg}(R_1) \bowtie \rho^m_{src}(X_1)) \cup \widetilde{\pi}_m(\rho^m_{trg}(R_1) \bowtie \rho^m_{src}(X)) \cup \widetilde{\pi}_m(\rho^m_{trg}(R_2) \bowtie \rho^m_{src}(X)) \ldots \cup \widetilde{\pi}_m(\rho^m_{trg}(R_n) \bowtie \rho^m_{src}(X)))$, where $(R_1 \bowtie R_2 \ldots \bowtie R_n)$ is the constant part denoted as κ and the other parts in between $\cup$ are the recursive parts of the terms $\psi_1, \psi_2 \ldots, \psi_n$.
This term in a simplified manner is: $\mu(X.\ \kappa \cup \psi_1 \cup \psi_2 \ldots \cup \psi_n)$.

Theorem 6: Let's consider a μ-RA term t of the general form:
$\mu(X.\ \kappa \cup \psi_1 \cup \psi_2 \ldots \cup \psi_n)$; r as the set of optimization (rewrite) rules of Relational Algebra; a term t' of the form $\mu(X_1.\ r(\kappa) \cup r(\psi_1) \cup r(\psi_2) \ldots \cup r(\psi_n))$. Then we have $d(\psi, X) = d(r(\psi), X)\ \forall \psi$ and $r \in R$ (where R is the set of transformation rules of Codd algebra, introduced in II-B.)
Proof:
Let r be a transformation rule (of a given set of rules found in [1]).
$r(\psi)$ denotes the μ-term produced by the application of the rule r on the term ψ. (nb: if r cannot apply because ψ does not match the required form for r to apply then $r(\psi) = \psi$)
These optimization rules are applied in the constant part κ and the recursive part $\psi_1 \cup \psi_2 \ldots \cup \psi_n$. We will only comment on the changes happening in the recursive part, because X is present only there. For the constant part of the decomposed fixpoint, there is no X present, so we do not have a stab calculation.
For UCRPQ terms, in each of the ψ_i during the iteration step, if new calculations for X are found, they are joined to the other chunk of the recursive part. Considering that in the syntax of Relational Algebra presented in [1], there does not exist a relation variable X that implicates the recursion so they will not change the X.

Everytime, a rewrite rule is applied, there is a new ordering of the algebraic operators while preserving the semantics. The $stab$ of term t is calculated based on the set of derivations of

Definition 9. Zooming in an example:
Before applying rewrite rules r we have a term that contains the following in the recursive part $\sigma_f(\psi_1 \bowtie \psi_2)$ and the stab is calculated as $d(\psi_1 \bowtie \psi_2, X)$ which is $d(\psi_1, X) \cup d(\psi_2, X)$. When the rewrite rules are applied, the recursive part we had before became $\sigma_f(\psi_1) \bowtie \psi_2$ and the stab is as follows: $d(\sigma_f(\psi_1), X) \cup d(\psi_2, X)$, which is $d(\psi_1, X) \cup d(\psi_2, X)$. No matter when the algebraic operator is found, the stab is not depended in the order of this operator. The term ψ_i does not change, hence the calculation of derivations and the stab stay unchanged.

As we said before, when rewrite (optimization) rules of [1] are applied, they just change the order of algebraic operator, they cannot replace them. We go through each "block" of the rewrite possibilities presented in II-B for each algebraic operator.

- Join possibilities II-B3
  Considering the table of all Join possibilities, if the join is pushed the new term will be $\psi_j \bowtie \psi_k$. The derivation is calculated in this "new term" created, it will be $d(\psi_j \bowtie \psi_k, X)$. By Definition 9, $d(\psi_1 \bowtie \psi_2, X) = d(\psi_1, X) \cup d(\psi_2, X)$, the re-ordering does not cause loss of information.
- Filter possibilities II-B1
  Considering the table of all Filter possibilities, if the filter is pushed the new term will be $\sigma_f(\psi_j)$. The derivation is calculated in this "new term" created, it will be $d(\sigma_f(\psi_j), X)$. By Definition 9, $d(\sigma_f(\psi), X) = d(\psi, X)$, so the filter pushed in the term does not change the derivation.
- AntiProjection possibilities II-B2
  Applying r in a query, it cannot change an $\widetilde{\pi}_a(\psi)$ to $\widetilde{\pi}_b(\psi)$, where a is different from b. Considering the table of all AntiProjection possibilities, if the AntiProjection is pushed the new term will be $\widetilde{\pi}_a(\psi_j)$. By Definition 9, $d(\widetilde{\pi}_a(\psi), X) = \{p \circ (a \rightarrow \perp) \mid p \in d(\psi, X)\}$, when removing a column the stab is unchanged and this set of rules r does not modify the calculations of derivations.
- Union possibilities II-B4
  Considering the table of all Union possibilities, if the union is pushed the new term will be $\psi_j \cup \psi_k$. The derivation is calculated in this "new term" created, it will be $d(\psi_j \cup \psi_k, X)$. By Definition 9, $d(\psi_1 \cup \psi_2, X) = d(\psi_1, X) \cup d(\psi_2, X)$, the re-ordering does not cause loss of information.
- Rename operator
  Applying r in a query, it cannot change $\rho_a^b(\psi)$ to $\rho_c^d(\psi)$, where a is different to b and c is different to d. By Definition 9, $(\rho_a^b(\psi), X) = \{p \circ (b \rightarrow a, a \rightarrow \perp) \mid p \in d(\psi, X)\}$, when after the composition the columns are renamed, the stab is unchanged. When these rules are applied, the information is preserved.
- Recursion variable (X)
  Applying the set of rules r cannot change the a recursion variable X to X'. Codd algebra rules, cannot consider the recursion variable, hence they cannot change it and the information is preserved.
- AntiJoin possibilities(not implemented, but we can write for it, since it is on the set of derivations)
  In the case of AntiJoin, $d(\psi_1 \triangleright \psi_2, X) = d(\psi_1, X)$, and if Codd Algebra rules are applied, the stab is always checked in the term ψ_1, thus preserving the information.

IV. Does add change after applying the 5 SIGMOD'20 theorems concerning fixpoints?

A. Add in Pushing Filters in a Fixpoint

Theorem 7: Let's consider a term t of the form $\sigma_f(\mu(X.\kappa \cup \psi))$; a term t' of the form $\mu(X_1.\sigma_f(\kappa) \cup \psi_1)$ obtained by Theorem 1 of [3]. Then we have $add(\psi, X, c)$ = $add(\psi_1, X_1, c)$.
Proof:
For term t, $add(\psi, X, c)$ is calculated (based on Definition 11 of [3]) in the recursive part of the decomposed fixpoint.
In term t', the filter was pushed on the constant side of the fixpoint, leaving the recursive part ψ_1 unchanged. Based on Definition 11 of [3], the $add(\psi_1, X_1, c)$ of the term t' does not depend on the constant part $\sigma_f(\kappa)$ of the fixpoint, but is calculated on its recursive part ψ_1. So, we have $add(\psi, X, c)$ = $add(\psi_1, X_1, c)$.

B. Property of add

a) Property: $add(\psi, X, c) = add(\psi_1, X_1, c)$ where ψ_1 = $\psi[X/X_1]$.
In ψ_1 the typ changes, but based on Definition 11, $add(X, X, c)$ = $\top$, making the add invariant. So, we have $add(\psi, X, c) = add(\psi_1, X_1, c)$

C. Add in Pushing AntiJoin in a Fixpoint

Theorem 8: Let's consider a term t of the form $\mu(X.\ \kappa \cup \psi) \triangleright \varphi$ with φ of type $typ \subseteq stab(\psi, X)$ (we suppose that X is not a free variable of φ); a term t' of the form $\mu(X_1.\ (\kappa_1 \triangleright \varphi) \cup \psi_1)$ obtained by Theorem 2 of [3]. Then we have $add(\psi, X, c)$ = $add(\psi_1, X_1, c)$.

Proof:
For term t, $add(\psi, X, c)$ is calculated (based on Definition 11 of [3]) in the recursive part of the decomposed fixpoint.
In term t', the antijoin with the term φ happened on the constant side of the fixpoint, leaving the recursive part ψ_1 unchanged. Based on Definition 11 of [3], the $add(\psi_1, X_1, c)$ of the term t' does not depend on the constant part of the fixpoint, but is calculated on its recursive part ψ_1, and based on Property III-B, the calculation of add is invariant. So, we have $add(\psi, X)$ = $add(\psi_1, X_1, c)$.

D. Add in Pushing Join in a Fixpoint

Theorem 9: Let's consider a term t of the form $\mu(X.\ \kappa \cup \psi) \bowtie \varphi$ with φ of type $typ \subseteq stab(\psi, X)$ (with $X \notin free(\varphi)$) a term of type typ_φ such that:

(1) $typ_{\varphi} \subseteq stab(\psi, X)$
(2) $\forall c \in typ_{\varphi} \backslash typ$ $add(\psi, X, c)$; a term t' of the form $\mu(X_1.\ (\kappa \bowtie \varphi) \cup \psi_1)$ obtained by Theorem 3 of [3]. Then we have $stab(\psi, X) = stab(\psi_1, X_1)$.
Then we have $add(\psi, X, c) = add(\psi_1, X_1, c)$.

Proof:
For term t, $add(\psi, X, c)$ is calculated (based on Definition 11 of [3]) in the recursive part of the decomposed fixpoint.
In term t', the join with the term φ happened on the constant side of the fixpoint, leaving the recursive part ψ_1 unchanged. Based on Definition 11 of [3], the $add(\psi_1, X_1, c)$ of the term t' does not depend on the constant part of the fixpoint, but is calculated on its recursive part ψ_1, and based on Property III-B, the calculation of add is invariant. So, we have $add(\psi, X, c) = add(\psi_1, X_1, c)$.

E. Add in Merging Fixpoints

Theorem 10: Let's consider a term t of the form $\mu(X_1.\ \kappa_1 \cup \psi_1)$ with type t_1; a term t' of the form $\mu(X_2.\ \kappa_2 \cup \psi_2)$ with type t_2 such that:
(1) $t_1 \cap t_2 \subseteq stab(\psi_2, X, C_2) \cap stab(\psi_1, X, C_1)$
(2) $\forall c \in t_1 \backslash t_2$ $add(\psi_2, X, c)$
(3) $\forall c \in t_2 \backslash t_1$ $add(\psi_1, X, c)$
; a term t" obtained by $t \bowtie t'$ which by Theorem 4 of [3] is $\mu(X.\ (\kappa_1 \bowtie \kappa_2) \cup \psi_1 \cup \psi_2)$. So, we have $add((\psi_1 \cup \psi_2), X, c) = add(\psi_1, X_1, c) \cup add(\psi_2, X_2, c)$.

Proof: For term t, $add(\psi_1, X_1, c)$ is calculated (based on Definition 11 of [3]) in the recursive part ψ_1 of the decomposed fixpoint.
For term t', $add(\psi_2, X_2, c)$ is calculated (based on Definition 11 of [3]) in the recursive part ψ_2 of the decomposed fixpoint.
The new term t", has a new stab $add((\psi_1 \cup \psi_2), X, c)$. Based on Definition 11, $add(\psi_1 \cup \psi_2, X, c) = add(\psi_1, X, c) \cup add(\psi_2, X, c)$ which allows us to calculate the new add as the union of the previous add of the two terms $add(\psi_1, X_1, c) \cup add(\psi_2, X_2, c)$.

F. Add in Pushing AntiProjection in a Fixpoint

Theorem 11: Let's consider a term t of the form $\widetilde{\pi}_c(\mu(X.\ \kappa \cup \psi))$ of type t_k and $c \in \mathfrak{C}$ be such that $add(\psi, X, c)$; a term t' of the form $\mu(X.\ \widetilde{\pi}_c(\kappa) \cup \psi)$. Then we have $add(\psi, X) = add(\psi_1, X_1, c)$.
Proof:
For term t, $add(\psi, X, c)$ is calculated (based on Definition 11 of [3]) in the recursive part of the decomposed fixpoint.
In term t', the antiprojection happened on the constant side of the fixpoint, leaving the recursive part ψ_1 unchanged. Based on Definition 11 of [3], the $add(\psi_1, X_1, c)$ of the term t' does not depend on the constant part $\widetilde{\pi}_c(\kappa)$ of the fixpoint, but is calculated on its recursive part ψ_1, and based on Property III-B, the calculation of add is invariant. So, we have $add(\psi, X, c) = add(\psi_1, X_1, c)$.

G. Add when applying Relational Algebra rewrite rules [1]

Note: All the terms considered in μ-RA are linear, positive and non mutually recursive.

We consider the subset of μ-terms produced by the translation of UCRPQ queries into the μ-algebra, as found in [3].
The General Form of such a term: $\mu(X.\ \kappa \cup \psi_1 \cup \psi_2 \ldots \cup \psi_n)$.
How this form is calculated:
Let's take the terms: $\mu(X_1.\ R_1 \cup \widetilde{\pi}_m(\rho^m_{trg}(R_1) \bowtie \rho^m_{src}(X_1))) \bowtie \mu(X_2.\ R_2 \cup \widetilde{\pi}_m(\rho^m_{trg}(R_2) \bowtie \rho^m_{src}(X_2))) \ldots \bowtie \mu(X_n.\ R_n \cup \widetilde{\pi}_m(\rho^m_{trg}(R_n) \bowtie \rho^m_{src}(X_n)))$.
This term can be written:
$\mu(X.\ (R_1 \bowtie R_2 \ldots \bowtie R_n) \cup \widetilde{\pi}_m(\rho^m_{trg}(R_1) \bowtie \rho^m_{src}(X_1)) \cup \widetilde{\pi}_m(\rho^m_{trg}(R_1) \bowtie \rho^m_{src}(X)) \cup \widetilde{\pi}_m(\rho^m_{trg}(R_2) \bowtie \rho^m_{src}(X)) \ldots \cup \widetilde{\pi}_m(\rho^m_{trg}(R_n) \bowtie \rho^m_{src}(X)))$, where $(R_1 \bowtie R_2 \ldots \bowtie R_n)$ is the constant part denoted as κ and the other parts in between $\cup$ are the recursive parts of the terms $\psi_1, \psi_2 \ldots, \psi_n$.
This term in a simplified manner is: $\mu(X.\ \kappa \cup \psi_1 \cup \psi_2 \ldots \cup \psi_n)$.

Theorem 12: Let's consider a μ-RA term t of the general form:
$\mu(X.\ \kappa \cup \psi_1 \cup \psi_2 \ldots \cup \psi_n)$; r as the set of optimization (rewrite) rules of Relational Algebra; a term t' of the form $\mu(X_1.\ r(\kappa) \cup r(\psi_1) \cup r(\psi_2) \ldots \cup r(\psi_n))$. Then we have $add(\psi, X, c) = add(r(\psi), X, c)$ $\forall \psi$, c the column to be added or removed, and $r \in R$ (where R is the set of transformation rules of Codd algebra, introduced in II-B.)
Proof:
Let r be a transformation rule (of a given set of rules found in [1]).
$r(\psi)$ denotes the μ-term produced by the application of the rule r on the term ψ. (nb: if r cannot apply because ψ does not match the required form for r to apply then $r(\psi) = \psi$)
These optimization rules are applied in the constant part κ and the recursive part $\psi_1 \cup \psi_2 \ldots \cup \psi_n$. We will only comment on the changes happening in the recursive part, because X is present only there. For the constant part of the decomposed fixpoint, there is no X present, so we do not have a add calculation.
For UCRPQ terms, in each of the ψ_i during the iteration step, if new calculations for X are found, they are joined to the other chunk of the recursive part. Considering that in the syntax of Relational Algebra presented in [1], there does not exist a relation variable X that implicates the recursion so they will not change the X.

Everytime, a rewrite rule is applied, there is a new ordering of the algebraic operators while preserving the semantics.

As we said before, when rewrite (optimization) rules of [1] are applied, they just change the order of algebraic operator, they cannot replace them. We go through each "block" of the rewrite possibilities presented in II-B for each algebraic operator.

- Join possibilities II-B3
  Considering the table of all Join possibilities, if the join is pushed the new term will be $\psi_j \bowtie \psi_k$. The add is calculated in this "new term" created, it will be $add(\psi_j \bowtie \psi_k, X, c)$. By Definition 11, $add(\psi_1 \bowtie \psi_2, X, c) = add(\psi_1, X, c) \wedge add(\psi_2, X, c)$, the re-ordering does not cause loss of information.
- Filter possibilities II-B1
  Considering the table of all Filter possibilities, if the filter is pushed the new term will be $\sigma_f(\psi_j)$. The derivation is calculated in this "new term" created, it will be $add(\sigma_f(\psi_j), X, c)$. By Definition 11, $add(\sigma_f(\psi), X, c) = add(\psi, X, c)$, so the filter pushed in the term does not change the derivation.
- AntiProjection possibilities II-B2
  Applying r in a query, it cannot change an $\widetilde{\pi}_a(\psi)$ to $\widetilde{\pi}_b(\psi)$, where a is different from b. Considering the table of all AntiProjection possibilities, if the AntiProjection is pushed the new term will be $\widetilde{\pi}_a(\psi_j)$. By Definition 11, $add(\widetilde{\pi}_a(\psi), X, c) = add(\psi, X, c)$ when $c \neq a$, when removing a column the stab is unchanged and this set of rules r does not modify the calculations of derivations.
- Union possibilities II-B4
  Considering the table of all Union possibilities, if the union is pushed the new term will be $\psi_j \cup \psi_k$. The derivation is calculated in this "new term" created, it will be $add(\psi_j \cup \psi_k, X, c)$. By Definition 9, $add(\psi_1 \cup \psi_2, X, c) = add(\psi_1, X, c) \cup add(\psi_2, X, c)$, the re-ordering does not cause loss of information.
- Rename operator
  Applying r in a query, it cannot change $\rho_a^b(\psi)$ to $\rho_c^d(\psi)$, where a is different to b and c is different to d. By Definition 11, $add(\rho_a^b(\psi), X, c) = add(\psi, X, c) \ \wedge \ c \notin \{a, b\}$, when after the composition the columns are renamed, the add is unchanged. When these rules are applied, the information is preserved.
- Recursion variable (X)
  Applying the set of rules r cannot change the recursion variable X to X'. Codd algebra rules, cannot consider the recursion variable, hence they cannot change it and the information is preserved.
- AntiJoin possibilities(not implemented, but we can write for it, since it is on the set of derivations)
  In the case of AntiJoin, $add(\psi_1 \triangleright \psi_2, X,) = add(\psi_1, X)$, and if Codd Algebra rules are applied, the add is always checked in the term ψ_1, thus preserving the information.

V. μ-LQDAG APPROACH

We propose an extension of the LQDAG to support the recursive algebraic terms and rewrite rules for these recursive terms. Throughout this paper we will refer to this extended LQDAG, as μ-LQDAG.

μ-LQDAG nodes are of two types: "alternative" nodes and "operation" nodes. Alternative nodes can only have operation nodes as children and vice versa, operation nodes can only have alternative nodes as children. What we call alternative, is a Set[μ-LQDAG] and what we call operation is the algebraic operation like: join ($\bowtie$), filter (σ_θ) etc. What is powerful with this representation, is that there will not exist two alternative nodes that represent the same result set and all μ-LQDAGs in the same alternative are equivalent.

We focus on the logical part of query optimization. After a query is given as an input, we generate all the semantically equivalent rewritings of it. These equivalent transformations are possible because of the application of optimization rules. There is not a fixed order in rule application, the rule is triggered when there is needed and all the conditions are verified.

A. Syntax and semantics of μ-LQDAG

In Figure 3, we can find the syntax of μ-LQDAG.

a, b $\rightarrow$ columns

d	::=		term
		$\lvert c \rightarrow v \rvert$	constant
		$\sigma_\theta(\alpha)$	filter
		$\alpha_1 \bowtie \alpha_2$	join
		$\alpha_1 \triangleright \alpha_2$	antijoin
		$\alpha_1 \cup \alpha_2$	union
		$\rho_a^b(\alpha)$	rename
		$\widetilde{\pi}_a(\alpha)$	antiprojection
		$\mu X.\, \alpha_{const} \cup \alpha_{rec}$	fixpoint

α ::= $\{d_1, d_2, ..., d_n\}$

Fig. 3: Syntax of μ-LQDAG.

We present a function $[\![S]\!]$ that translate the μ-LQDAG to a set of μ-RA terms. In Figure 4, we present the semantics of μ-LQDAG.

B. Stab and add are the same for every μ-LQDAG in an alternative (Suppose we can use stab and add of an Alternative)[Just an idea, maybe too early to define this]

Hypotheses: All μ-LQDAG-s semantically equivalent, sharing the same type are part of one Alternative node.

Let's take a μ-LQDAG m_1 of the form $\mu X.\alpha_{const_1} \cup \alpha_{rec_1}(X)$ with $stab_1(\alpha_{rec_1}, X)$; a μ-LQDAG m_2 of the form $\mu X.\alpha_{const_2} \cup \alpha_{rec_2}(X)$ with $stab_2(\alpha_{rec_2}, X)$ where m_1 & m_2 are semantically equivalent and are part of the same alternative node. Then we have, $stab_1(\alpha_{rec_1}, X) = stab_2(\alpha_{rec_2}, X)$.

Proof(try):

Let's denote with r the set of rewrite rules of [1] and the rewrite rules for fixpoint of [3]. We suppose that m_1 and m_2 are equivalent and part of the same alternative, but $stab_1(\alpha_{rec_1}, X) \neq stab_2(\alpha_{rec_2}, X)$.

After applying the set r of rewrite rules, new μ-LQDAG-s &

Constant	$S[\![\|c \to v\|]\!]$	$=$	$\{\|c \to v\|\}$
Filter	$S[\![\sigma_f(\alpha)]\!]$	$=$	$\{\sigma_f(d) \mid d \in S[\![\alpha]\!]\}$
Join	$S[\![\alpha_1 \bowtie \alpha_2]\!]$	$=$	$\{d_1 \bowtie d_2 \mid d_1 \in S[\![\alpha_1]\!] \wedge d_2 \in S[\![\alpha_2]\!]\}$
AntiJoin	$S[\![\alpha_1 \triangleright \alpha_2]\!]$	$=$	$\{d_1 \triangleright d_2 \mid d_1 \in S[\![\alpha_1]\!] \wedge d_2 \in S[\![\alpha_2]\!]\}$
Union	$S[\![\alpha_1 \cup \alpha_2]\!]$	$=$	$\{d_1 \cup d_2 \mid d_1 \in S[\![\alpha_1]\!] \wedge d_2 \in S[\![\alpha_2]\!]\}$
Rename	$S[\![\rho_a^b(\alpha)]\!]$	$=$	$\{\rho_a^b(d) \mid d \in S[\![\alpha]\!]\}$
AntiProjection	$S[\![\widetilde{\pi}_a(\alpha)]\!]$	$=$	$\{\widetilde{\pi}_a(d) \mid d \in S[\![\alpha]\!]\}$
Fixpoint	$S[\![\mu X.\, \alpha_{const} \cup \alpha_{rec}(X)]\!]$	$=$	$\{\mu X.\, d_{const} \cup d_{rec}(X) \mid d_{const} \in S[\![\alpha_{const}]\!] \wedge d_{rec} \in S[\![\alpha_{rec}]\!]\}$
Alternative	$S[\![\{d_1, d_2, ..., d_n\}]\!]$	$=$	$\{S[\![d_1]\!] \cup S[\![d_2]\!] ... \cup ... S[\![d_n]\!]\}$

Fig. 4: Semantics of μ-LQDAG.

Alternatives will be created recursively until new ones cannot be created anymore. When traced back to find the μ-LQDAGs m_1 & m_2, from the new expanded μ-LQDAG, we will end up in different alternatives from them, because $stab_1 \neq stab_2$, which is in opposition to the first supposition where they were part of the same Alternative.

C. Transformation rules on fixpoint of μ-LQDAG, are the transformation rules correct? (Proofs)

Transformation rules of μ-LQDAG of 5 theorems presented in [3] and their adaptation to work on a set (factorized representation).

a) Theorem 1 - Pushing Filters in a Fixpoint:

Theorem 13: Let's consider the following $\mu - LQDAG$ $\mu X.\alpha_{const} \cup \alpha_{rec}(X)$, v an environment and f a filter condition with $FC(f) \subseteq stab(\alpha_{rec}, X)$.
$\forall \alpha_{const}, \forall \alpha_{rec}, \forall f, \forall v$ and with $FC(f) \subseteq stab(\psi, X)$
we have $\forall t \in S[\![\sigma_f(\mu X.\alpha_{const} \cup \alpha_{rec}(X)))]\!]$
$\exists\, t' \in S[\![\mu X.\sigma_f(\alpha_{const}) \cup \alpha_{rec}(X)]\!]$
and $[\![t]\!]_v = [\![t']\!]_v$

b) Theorem 2 - Pushing AntiJoins in a Fixpoint:

Theorem 14: Let's consider the following $\mu - LQDAG$ $\mu X.\alpha_{const} \cup \alpha_{rec}(X)$, v an environment and α_φ an Alternative of type $typ \subseteq stab(\alpha_{rec}, X)$ (we suppose that X is not a free variable of φ).
$\forall \alpha_{const}, \forall \alpha_{rec}, \forall f, \forall v$ and $\forall \alpha_\varphi$ of type $typ \subseteq stab(\alpha_{rec}, X)$
we have $\forall t \in S[\![(\mu X.\, \alpha_{const} \cup \alpha_{rec}(X)) \triangleright \alpha_\varphi]\!]$
$\exists\, t' \in S[\![\mu X.\, (\alpha_{const} \triangleright \alpha_\varphi) \cup \alpha_{rec}(X)]\!]$
and $[\![t]\!]_v = [\![t']\!]_v$

c) Theorem 3 - Pushing Joins in a Fixpoint:

Theorem 15: Let's consider the following $\mu - LQDAG$ $\mu X.\alpha_{const} \cup \alpha_{rec}(X) \in \mathcal{F}[\Gamma]$ of type t_k, and $\alpha_\varphi \in \mathcal{F}[\Gamma]$ (with $X \notin free(\alpha_\varphi)$) an Alternative of type t_φ such that:
1) $t_\varphi \subseteq stab(\alpha_{rec}, X)$
(2) $\forall c \in t_\varphi \backslash t_k$ $add(\alpha_{rec}, X, c)$
Then we have $\Gamma \vdash \mu X.\alpha_{const} \cup \alpha_{rec}(X) : t_\varphi \cup t_k$ with $\forall v$ compatible with Γ:

$\forall \alpha_{const}, \forall \alpha_{rec}, \forall \alpha_\varphi, stab(\alpha_{rec}, X), add(\alpha_{rec}, X, c)$
we have $\forall t \in S[\![\alpha_\varphi \bowtie (\mu X.\alpha_{const} \cup \alpha_{rec}(X))]\!]$
$\exists\, t' \in S[\![\mu X.(\alpha_\varphi \bowtie \alpha_{const}) \cup \alpha_{rec}(X)]\!]$
and $[\![t]\!]_v = [\![t']\!]_v$

d) Theorem 4 - Merging Fixpoints:

Theorem 16: Let's consider the two following $\mu - LQDAG$s $\mu X_1.\ \alpha_{const_1} \cup \alpha_{rec_1}(X_1)$ and $\mu(X_2.\ \alpha_{const_2} \cup \alpha_{rec_2}(X_2))$ of types t_1 and t_2 such that:
(1) $t_1 \cap t_2 \subseteq stab(\alpha_{rec_2}, X_2, C_2) \cap stab(\alpha_{rec_1}, X_1, C_1)$
(2) $\forall c \in t_1 \backslash t_2$ $add(\alpha_{rec_2}, X_2, c)$
(3) $\forall c \in t_2 \backslash t_1$ $add(\alpha rec_1, X_1, c)$

$\forall \alpha_{const_1}, \forall \alpha_{const_2}, \forall \alpha_{rec_1}, \forall \alpha_{rec_2}, \forall \alpha_\varphi, stab(\psi, X), add(\psi, X, c)$
we have $\forall t \in S[\![\mu X_1.\, \alpha_{const_1} \cup \alpha_{rec_1}(X_1) \bowtie \mu X_2.\, \alpha_{const_2} \cup \alpha_{rec_2}(X_2)]\!]$
$\exists\, t' \in S[\![\mu X.\, (\alpha_{const_1} \bowtie \alpha_{const_2}) \cup \alpha_{rec_1}(X_1) \cup \alpha_{rec_2}(X_2)]\!]$
and $[\![t]\!]_v = [\![t']\!]_v$

e) Theorem 5 - Pushing AntiProjections in a Fixpoint:

Theorem 17: Let's consider the following $\mu - LQDAG$ $\mu X.\alpha_{const} \cup \alpha_{rec}(X) \in \mathcal{F}[\Gamma]$ of type t_k. Let $a \in \mathfrak{C}$ be such that $add(\alpha_{rec}, X, a)$.
$\forall \alpha_{const}, \forall \alpha_{rec}, \forall v$ and $add(\alpha_{rec}, X, a)$
we have $\forall t \in S[\![\widetilde{\pi}_a(\mu X.\alpha_{const} \cup \alpha_{rec}(X)))]\!]$
$\exists\, t' \in S[\![\mu X.\widetilde{\pi}_a(\alpha_{const}) \cup \alpha_{rec}(X)]\!]$
and $[\![t]\!]_v = [\![t']\!]_v$

f) Expansion: We start from an Alternative, which itself is a Set[μ-LQDAG]. After applying the rules, new transformations are possible, and new μ-LQDAGs are created, thus creating new Alternatives. We are able to ensure unicity of μ-LQDAG and alternatives Everytime new transformations are possible, we make sure that μ-LQDAG and alternatives were not previously created. The method that performs expansion is called recursively and everytime a new μ-LQDAG is created, it is added in an Alternative, until no new transformations are available anymore. At the end, we have this set of μ-LQDAGs created during the expansion, along with the starting alternative. Then, it is the cost estimation function that helps choosing between all these equivalent plans for the most efficient one.

REFERENCES

[1] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377387, June 1970.

[2] G. Graefe and W. J. McKenna. The volcano optimizer generator: Extensibility and efficient search. In *Proceedings of the Ninth International Conference on Data Engineering*, pages 209–218, Washington, DC, USA, 1993. IEEE Computer Society.

[3] L. Jachiet, P. Genevès, N. Gesbert, and N. Layaïda. On the optimization of recursive relational queries: Application to graph queries. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2020 (to appear). https://hal.inria.fr/hal-01673025v5/document.

[4] P. Roy, S. Seshadri, S. Sudarshan, and S. Bhobe. Efficient and extensible algorithms for multi query optimization. *SIGMOD Rec.*, 29(2):249260, May 2000.