Paper Review - Group 2

Hao Liu
Yunju Wang
Zhiliang Zhang
Zhengfeng Chen

2023 Fall

1 Introduction

This report is a review after reading 'Data dependencies for query optimization: a survey', including an Introduction, a brief summary, each section of approaches, application, and a conclusion.

2 Summary

In the field of database optimization query research, especially improving query execution efficiency through data dependencies, this survey covers the most popular types of data dependencies(Unique column combinations (UCCs), Functional dependencies (FDs), Order dependencies (ODs), and Inclusion dependencies (INDs)) and their application in operations of relational databases, such as collection, join, grouping and aggregation, distinctness, subquery, set operations, selection, and sorting. In summary, this survey provides an overview and classification of query optimization and execution strategies supported by data dependencies.

3 Query Optimization

This paper has classified the optimization activities of the query optimization process into three categories. These activities are (i)cost-independent transformation, (ii)cardinality & cost estimation, and (iii)cost-based transformation. The summaries of these activities in the paper are concise and explanatory. However, when the optimization techniques are divided into these three activities, the lack of relation between these activities and the state-of-the-art industry cases makes it difficult for the reader to relate and cross-reference. Thus, we research the optimization activities used in PostgreSQL. Dombrovskaya et al. stated that the optimizer in PostgreSQL uses transformation rules, heuristics, and cost-based optimization algorithms to build an execution plan[1]. Transformation rules are a type of cost-independent transformation where operations are reordered to optimize the query plan without cost estimation. Heuristics provides guidelines to evaluate and compare query plans based on their estimated costs. These activities are in line with the activities stated in the paper.

4 Data Dependencies

This paper has given a brief introduction to four types of data dependencies, which are **unique column combinations**, **functional dependencies**, **order dependencies,** and **inclusion dependencies**. In this section, we will give a concise explanation of each data dependency and summarize their characteristics

4.1 Unique column combinations

Unique column combinations are sets of the relational database's columns that uniquely identify a record in a table. For example, in a table of students, the combination of student ID and student name

is a unique column combination. The set of attributes in unique column combinations is also known as the candidate key. In most relational database management systems, a table has a primary key. If a table does not have a primary key, a default attribute ID will be the primary key. Since the primary key denotes unique column combinations and makes it the most common type of data dependencies. UCC is also created from the result uniqueness required in SQL queries, such as DISTINCT statement or EXCEPT, UNION, INTERSECT, or GROUP BY clauses.

4.2 Functional dependencies

A functional dependency statement between two sets of columns X and Y indicating that for any two rows of data in a relational instance, if they have the same values for columns in X, they must have the same values for columns in Y. For example, city and state are functionally dependent on zip code. FD is a popular property in relational databases because it supports the standard of database schema and checking data consistency. However, Not every place has function dependencies, and discovering it needs extra effort. Also, maintaining FD is expensive and time-consuming, as it requires updating every operation, such as insertion, deletion, and update.

4.3 Order dependencies

An order dependency is a statement that indicates $X \mapsto Y$, where X and Y are sets of columns in R, if R is ordered by X, then it is also ordered by Y. However, the reverse is not true necessarily. Most of the time, OD is created from ORDER By clause specified by a user, ordered results created from database operations or naturally ordered columns.

4.4 Inclusion dependencies

An inclusion dependency is a statement in which some relation columns are contained in other columns. The most common example of inclusion dependency is the foreign key. Discovering such dependency requires going through all the relations and attributes, as we know it is hard.

4.5 Data Dependency Properties

This section discussed the properties of data dependencies and how they might affect the query process. The first is *up-to-dateness,* which focuses on whether the data dependency still holds after the operations, such as insertion, deletion, and update. The second is *Minimality*, state-of-the-art algorithms mostly mine for only minimal (or maximal) data dependencies. Query optimizer might require non-minimal (or maximal) data dependencies. We will infer the Optimal data dependencies from minimal data dependencies. However, *Completeness* matters as well. Without a complete set of minimal/maximal dependencies, it might be necessary to calculate the closure. Next is *Conditions*, data dependencies that only hold under certain conditions that limit their scope. Last is Null-semantics, so we focus on the semantics of null values in relational database systems dependencies. Whether comparing null will result in True, False, or unknown needs to be accurately interpreted for data dependencies.

5 Classification of Dependency-driven Query Optimization

This paper has classified the application areas of dependency-driven query optimization techniques into seven categories. They are Join, Selection, Grouping & aggregation functions, Project and distinctness, Sorting, Set Operations, and Other. We can observe that most optimization techniques focus on basic operations rather as most advanced ones derive from basic operations such as rank and window functions. Furthermore, this paper has provided an exhaustive list of optimization techniques are based on data dependencies with their query optimization activities and application areas. Furthermore, this paper has identified further research directions, which are Efficient implementation and integration, Incremental discovery and maintenance, and Empirical impact evaluation.

6 Unique Column Combinations

6.1 UCCs and Joins

This paper listed four optimization techniques on how UCC works with joins.
The first two optimization techniques happen in cost-based plan transformations. One is *the pipeline with grouping*. This technique suggested that aggregations and asymmetrically implemented joins could be pipelined if the grouping attributes contain the primary key. The example this paper gave was not very clear. In most cases, the database will sort the two relations before joining and grouping. The sorted relations will be pipelined to the join operator and then merged on condition. This method is already very efficient and works for non-UCC cases. The other is *invisible join*. However, this method is for column-oriented database engines. It is not very suitable for this paper.

The other two optimization techniques happen in cost-independent transformation. The first is back join, when a derived relation from R joined with another relation derived from R to retrieve more attributes. Utilizing joining on UCC, the spurious tuples from the back join can be avoided. Second is semijoin, when inner joining two relations R and S on attribute X, if X is a UCC on S and S does not need to supply an additional column to the result, then rewriting inner join to semijoin can potentially reduce data transmission cost overhead.

6.2 UCCs and Grouping and Aggregation

This paper did not elaborate on UCC on grouping and aggregation in this subsection but chose to discuss it with FD-based optimizations in a later section. This is due to UCC being a special form of functional dependencies.

6.3 UCCs and Distinctness

This paper mentions that using DISTINCT operations that result in certain combinations with UCC is redundant as UCC already guarantees the uniqueness of the result. A straightforward optimization technique is to remove the DISTINCT operation from the query plan.

6.4 UCCs and Subqueries

This paper mentions an optimization technique that can be used to embed subqueries into joins. `SELECT R.A, R.B FROM R WHERE EXISTS (SELECT * FROM S WHERE S.A = R.A)`, if S.A is a UCC can be rewritten to `SELECT R.A, R.B FROM R, S WHERE S.A = R.A`. Due to the UCC on S.A, the subquery will only return one or no result. Thus rewriting it into join will not affect the result. Then query planner can utilize the techniques used in joins to find a more efficient query plan. PostgreSQL has already implemented this optimization technique. By using `EXPLAIN` in the above example, the same execution plan will be returned.

6.5 UCCs and Set Operations

In SQL's set operations, such as UNION, INTERSECT, and EXCEPT, unless provided the ALL keyword, otherwise the result will be distinct. UCC is used to reduce the effort in duplication removal in set operations. As stated in this paper, INTERSECT and INTERSECT ALL are equal if there is at least one UCC on one of the involved tables. By writing the query from INTERSECT to INTERSECT ALL, we will avoid spending the costly duplication removal.

6.6 Further Optimization Opportunities with UCCs

SELECT clause over a set of attributes that contain UCCs, rather than scanning through the whole table, the query can be aborted after the first matching tuple. In real-world cases, unique indexes such as B-Tree are built on UCCs to speed up the query. However, the developer needs to be careful in writing queries as the execution is subjected to the leftmost prefix rule. This is the same for *sorting*. If X is a UCC on R, then the clause ORDER BY X, Y can be reduced to ORDER BY X.

7 Functional Dependencies

We talk about the functional dependencies and how to optimize queries from five aspects: grouping, join, selection, sorting, and further optimization.

7.1 FDs and grouping

If there is a functional dependency relationship between $B, C \rightarrow A$, there is a grouping operation performed on A, B, and C. So A is not necessary because all elements belong to the group of B and C, and it must also have the same value as A. Therefore, we can remove the grouping attributes A, focusing solely on B and C.. For example, `SELECT C, SUM(D) FROM R WHERE B = 17 GROUP BY C, A` can be rewritten to `GROUP BY C`. This can also be applied to SQL DISTINCT clauses.

7.2 FDs and joins

This section discusses how we will optimize queries using the Functional Dependencies that involve join operations.
Firstly, Use functional dependencies to achieve early aggregation before joining, and group-by operations that are pushed below joins. For example, use functional dependencies to prune connection trees during query plan generation.
Secondly, apply FDs to simplify queries that involve joins. Assume a natural join $R \bowtie S$ that joins the relations R and S on the attributes Z with $S[Z] \subseteq [Z]$ and vice versa. If the FD $X \rightarrow A$ and $X \subseteq Z$, then the join attributes Z can be reduced to Z/A, because matching values in X guarantees matching values in A.
Finally, it shows that an equality self-join can be avoided in cases, where the join attribute functionally determines all other attributes in the distinct projection of the query. For example, if $A \rightarrow B$ and $B \rightarrow C$ hold, then the join predicate $R.A = S.B$ implies $R.B = S.C$ and $R.B = S.C$, where $S.C$ is a copy of $R.C$.

7.3 FDs and selection

Review several techniques that use FDs to optimize selection predicates, such as predicate rewriting, predicate inference, predicate pushdown, and predicate elimination.
For example, if $A \rightarrow B$ holds, then the predicate $A = x \ AND \ B = y$ can be rewritten as $A = x$ if y is the value of B, and that corresponds to x. For example, two, if $A \rightarrow B$, and $B \rightarrow C$ hold, then the predicate $A = x$ implies $B = y$ and $C = z$, where y and z are the values of B and C that correspond to x.

7.4 FDs and Sorting

We can reduce the attribute lists in `ORDER BY` clauses with the use of known FDs. For example, the clause `ORDER BY X, A` can be reduced to `ORDER BY X` if the `FD` $X \rightarrow A$ holds, because for a certain value of `X`, there is only one value in `A`.

7.5 Further Optimization Opportunities

1. FDs can improve cardinality estimations and lead to better query plans.
2. FDs let cardinality information propagate from one attribute to another.
3. FDs can estimate the size of projections.
4. FDs can be used for query rewriting in combination with horizontal table partitioning.

8 Order Dependencies

In this section, order and Order dependencies are used to optimize queries from four aspects: grouping, join, sorting, and further optimization.

8.1 ODs and Sorting

1. ODs can reduce the number of attributes in the order clause, for example: If $X \overset{<}{\mapsto} Y$ holds on to a relation R, the clause `ORDER BY X, Y` can be reduced to `ORDER BY X`.
2. ODs can be utilized to substitute sorting attributes, for example, there is an example where the costly string-sort was replaced by the cheaper integer-sort: A (integer), B (string), and C (integer), and the OD $A \overset{<}{\mapsto} B$ to hold. Hence, a statement `ORDER BY B, C` could be replaced by `ORDER BY A, C`.

8.2 ODs and Joins

1. Order information can be used to simplify joins in the plan optimization phase, to improve the join operator execution phase.
2. ODs can be used for query rewriting to avoid expensive joins between fact and dimension tables.
3. Order dependencies can also be used to optimize certain theta joins via query rewriting if combined with UCCs.

8.3 ODs and Grouping

ODs can be used for grouping data and aggregating data, hashing, and sorting both operators' profit from pre-ordered inputs.

8.4 Further Optimization Opportunities with ODs

The order depends on offering various further optimizations:

1. Selection
2. Aggregate functions
3. Leveraging indexes
4. Generating distinctness
5. Cost estimation

9 Inclusion Dependencies

Because Inclusion dependencies may span across multiple data sources, they often serve data linkage and data integration scenarios. In this section, how they can be utilized for query optimization from joins and further optimization.

9.1 INDs and Joins

1. Given the IND $R.X \subseteq S.X$ with its formal definition $\forall t_r \in R, \exists t_s \in S : t_r[X] = t_s[X]$ all tuples from R match tuples in S so that the semi-join is redundant; hence, it can be removed.
2. Semi-join reduces the number of tuples considered by join operators before the actual join is conducted.
3. The IND $R.X \subseteq S.X$ and a UCC on S.X on the relations R and S, then $R \bowtie_X S$ can be avoided if there are no further selections or projections on any attributes of S.
4. If there are two joins $R \bowtie_X S$ and $S \bowtie_X T$ with the INDs $R.X \subseteq S.X$ and $S.X \subseteq T.X$ and a UCC on S.X, the two joins can be reduced to a single join following from the transitivity of INDs.

9.2 INDs Offer Various Further Optimizations

1. Query folding
2. Exists
3. EXCEPT
4. Cardinality estimation

10 Further Optimizations

This survey mainly includes the following content: 1. Data dependencies are powerful tools to improve query efficiency. 2. How to use data dependencies for most all query optimization tasks, such as plan selection, execution optimization, and query rewriting. 3. Depend on data dependency queries can improve the correctness, performance, and robustness. of query processing in various applications. 4. Data dependencies are not only related to relational database queries but also to non-relational database queries, such as XML queries, graph queries, or RDF queries. 5. This survey proposes future research directions in using data dependency for query optimization and provides some recommendations for practitioners. From this article, we can learn: 1. The current status of using data dependency to achieve data query optimization, knowing which technologies have been applied, which are being studied, future research and application directions, and even which materials should be queried. 2. How to use four common data dependencies to improve query efficiency. The significance of this paper for reality is that it can help database practitioners and researchers understand and apply data dependencies for query optimization. This is a relevant and timely topic for database research and practice. Overall, we think this survey is a valuable contribution to the field of database management systems, and we have also learned a lot from this survey.

11 Conclusion

This survey mainly includes the following content:

1. Data dependencies are powerful tools to improve query efficiency.
2. How to use data dependencies for most all query optimization tasks, such as plan selection, execution optimization, and query rewriting.
3. Depend on data dependency queries can improve the correctness, performance, and robustness of query processing in various applications.
4. Data dependencies are not only related to relational database queries, but also to non-relational database queries, such as XML queries, graph queries, or RDF queries.
5. This survey proposes future research directions in using data dependency for query optimization and provides some recommendations for practitioners.

From this article, we can learn:

1. The current status of using data dependency to achieve data query optimization, knowing which technologies have been applied, which are being studied, and future research and application directions, and even which materials should be queried.
2. How to use four commonly data dependencies to improve query efficiency.

The significance of this paper for reality is that it can help database practitioners and researchers to understand and apply data dependencies for query optimization. This is a relevant and timely topic for database research and practice.
Overall, we think this survey is a valuable contribution to the field of database management systems, and we have also learned a lot from this survey.

References

[1] Jan Kossmann et al. Data dependencies for query optimization: a survey. *The VLDB Journal*, 31:1–22, 2021.

[2] Boris Novikov Henrietta Dombrovskaya and Anna Bailliekova. *PostgreSQL Query Optimization: The Ultimate Guide to Building Efficient Queries*. Apress, 2021.

[3] Microsoft. Using cross joins. 2012.

[1] [2] [3]