实践中的可扩展查询优化器

作者: Microsoft 译者: 渠继涵

目录

第一章 介绍

SQL 是一个用于查询关系型数据的高级声明式语言。它已经成为了关系数据查询的事实标准，并且被大部分的关系型数据库所支持。此外，在一些大数据系统中也逐渐开始流行。

SQL 允许对关系型数据进行声明式的查询，包括选择、连接、分组、聚合以及嵌套子查询，这对各种决策的支持非常重要，尤其是企业中的商业智能场景。

以下面的 Query 1 为例：

Query 1

```
SELECT *
FROM R, S, T
WHERE R.a = S.a AND S.c = T.d AND T.e = 10
```

在图 1-1 中展示了在一个数据库管理系统中，处理一个 Query 的 SQL 的主要流程。Query 处理的三个流程将会在下面依次介绍。

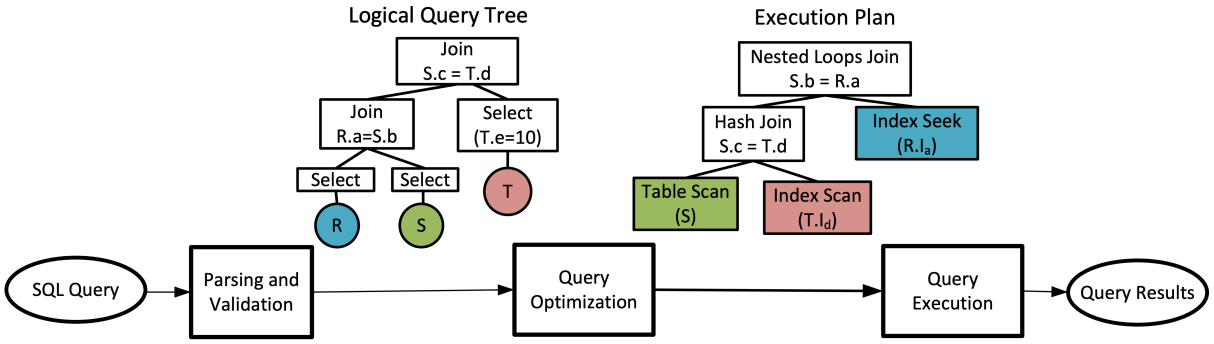


图 1-1　查询处理流程

解析以及验证（Parsing and Validation） 解析与验证这一步是将一个 SQL 查询转换成内部的表现形式，这一步确保查询符合 SQL 的语法规则并且引用了已经存在的数据库的对象，例如：表或者列等。这一步的输出结果为逻辑查询树，逻辑查询树是一个使用代数表示的 Query，其中每一个树的节点是一个关系算子（例如：Select、Join 等）。在图 1-1 展示了 Query 1 在经过解析以及验证后的输出的逻辑查询树。

查询优化（Query optimizer） 查询优化（*query optimizer*）将逻辑查询树作为输入，并负责生成由查询执行引擎解释或者编译的高效的执行计划。执行计划（*execution plan*）（也被成为计划（*plan*））是一个由物理算子所组成的树，树的边表示为两个操作之间的数据流动。例如在图 1-1 中展示了 Query 1 在经过查询优化后输出的执行计划。对于一个给定的查询来说，生成的执行计划的数量可能会随着引用的表的数量而呈现指数增长的趋势。并且不同的执行计划会有着不同的执行效率。因此，查询的性能很大程度上取决于优化器是否能从大量的执行计划中选择出一个效率高的执行计划的能力。有关关系型数据库的查询优化的概述，可以参考[1]。

查询执行（Query execution） 查询执行引擎从查询优化中获取执行的计划，然后生成查询的结果。查询执行引擎实现了一系列的物理算子，物理算子负责为查询计划构建数据块。一个物

理算子将一个或者多个数据记录作为其输入，称之为行（*rows*），并输出一组行。常见的物理算子包括 `Table Scan` 、 `Index Scan` 、 `Index Seek` （详见 第 8 章 ）、 `Hash Join` 、 `Nested Loops Join` 、 `Merge Join` 、 `Sort` 等。有关各种物理算子算法的描述，建议参考[2]。

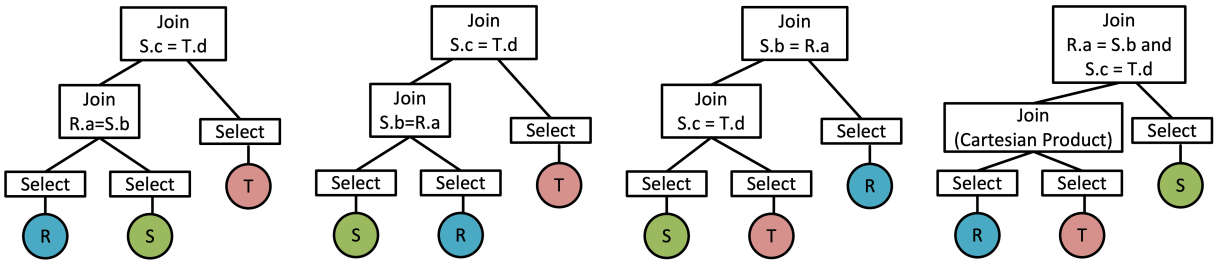
大多数关系型数据库中，执行引擎都使用迭代器模型，这个模型的每一个物理算子都实现 `Open` 、 `GetNext` 、 `Close` 这三个方法。每一个迭代器都包含了记录其状态（包括大小、哈希表的位置等信息）。在 `Open` 中，算子初始化并且开始准备处理数据。当 `GetNext` 被调用的时候，算子会生成下一个输出的行，或者返回没有行（没有结果，意味着处理的结束）。这不难想到，要生成输出的行，执行计划中的非叶节点必须要多次对其子算子调用 `GetNext` 。例如在图 1-1 中， `Nested Loop Join` 算子在 `Hash Join` 算子上调用 `GetNext` ，而 `Hash Join` 算子又调用了 `Table Scan` 的 `GetNext` 方法。当一个算子完成了输出（也就是没有更多的行了），父算子会对其调用 `Close` 方法，允许其清理状态信息。上述迭代器模型可以方便的添加新的算子，由于每一个运算符都是一个迭代器，对数据执行的是“拉”取操作。所以也被成为拉取模型（*pull module*）。这里建议通过阅读[3]来了解更多的关于执行引擎中 *pull module* 的相关知识。

但是迭代器模型存在较高的函数调用开销，每一次调用 `GetNext` 仅处理了一行数据，在现代 CPU 上的性能表现并不理想。向量化（*Vectorization*）支持批量处理，也就是说，每一次物理算子调用 `GetNext` 都可以处理一批数据行，并且利用现代 CPU 的 SIMD（）[4]指令集。结合列式存储[5]，向量化显著的提高了哪些用于决策的查询的执行效率。此外，代码生成（*code generation*）是一个可以从执行计划生成高效代码的技术。例如使用 C 语言，生成的代码随后被编译执行。或者利用 LLVM 这样的工具直接生成机器码。在[6]讨论了向量化与编译的一些取舍，感兴趣可以看一下。

1.1 查询优化的关键挑战

为了在众多可选的执行计划中选择最为高效的那一个，查询优化器必须明确需要搜索的空间（*Search space*），然后通过代价估计（*Cost estimation*）来比较不同的执行计划，最后通过搜索算法（*Search Algorithm*）遍历搜索空间，找到执行执行效率比较低（理想情况下是最低）的那个执行计划。下面我们来简单的介绍以下查询优化器的这几个方面。

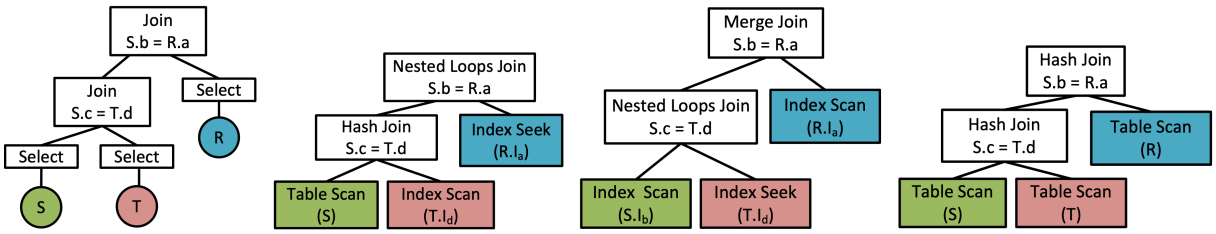
搜索空间（Search space）对于一个查询来说，搜索空间包含了大量的等价的执行计划。如果这个查询非常复杂，那么这个搜索空间也会非常的大。首先，对于一个给定的查询的代数表示来说，它可以等价的转换成不同的代数表示。这种等价转换的主要原因是关系代数的一些性质。例如$Join(Join(R,\ S),\ T) \iff Join(Join(S,\ T),\ R)$就是利用了 Join 操作的交换律与结合率[7]。在图 1-2 中展示了同一个查询语句的四种等价而关系代数的表示方式。



(a) Logical plan L_1 **(b)** Logical plan L_2 **(c)** Logical plan L_3 **(d)** Logical plan L_4

图 1-2 等价的逻辑查询树

其次，对于一个给定的逻辑算子他也有众多不同的实现方式。因此一个给定的逻辑查询树会有众多不同的执行计划。例如在图 1-3 中，对于给定的(a)图中的逻辑查询树来说，我们在众多执行计划中，还给出了等价的三种执行计划(b)、(c)、(d)。尽管这三种执行计划的 `Join` 算子的求值顺序是相同的，但是用于实现 `Join` 的物理算子却有所不同。在(a)中的 `Select` 逻辑算子可以使用 `Table Scan` 、`Index Scan` 或者 `Index Seek` 来实现。并且 `Join` 算子也可以使用 `Nested Loops Join` 、`Hash Join` 、 `Merge Join` 来实现。当S与T之间的连接大小（*Join size*）比较小，且在列$R.a$上才在索引I_a的时候，(b)是效率最高的执行计划。当$S.b$以及$R.a$分别存在索引$I.b$、$I.a$时， `Merge Join` 的效率是最高的（也就是索引提供 `Merge Join` 所需的字段的排序）。相比之下，当S、R之间的连接规模比较大的时候， `Hash Join` 可能是首选。因此，除非优化器在其搜索空间中考虑到了每一个这样的执行计划，并且比较它们的资源的使用情况以及预期的相对性能，否则根本无法生成一个好的执行计划。



(a) Logical plan L **(b)** Execution plan P_1 **(c)** Execution plan P_2 **(d)** Execution plan P_3

图 1-3 相同的逻辑计划会有不同的执行计划

代价估计（Cost estimation） 相同查询的不同执行计划会有不同的效率，我们通过消耗的时间以及资源（例如 CPU、内存、I/O）来度量。如图 1-3 所示，对于大型的数据库的复杂查询来说，一个好的执行计划与一个差的执行计划之间的在时间消耗的差距可能是几个数量级的差距。因此，为了为一个查询在搜索空间里选择一个好的执行计划，查询优化器通常使用代价模型（*Cost Module*）来准确的估算执行查询所需的工作，以便对执行计划进行相对比较。具体来说，一个物理算子必须通过查询的连接关系的大小以及统计信息来估算用于实现该操作的算法所需的代价。最后，尽管在成本的估算中有至少三个维度的信息（CPU、内存、I/O），但是最后代价模型会将多个维度的结果变成一个数字，用于在不同的两个执行计划之间进行比较。

搜索算法（Search Algorithm） 原则上，我们可以通过枚举每一个搜索空间中的执行计划，并且调用代价模型来确定每一个计划的成本，最终找到代价最低的计划。但是不同的执

行计划可能会共享相同的逻辑/逻辑查询树（子树），例如图 1-2 中的 `Select` 以及在图 1-3 中的 `Table Scan` 。所以在枚举的时候需要避免重复的探索。但是即便如此，在实际的应用中，枚举的成本依旧是极其高昂的。因此一个优秀的查询优化器会在不显著影响执行计划的选择的情况下，尽量降低枚举的成本。

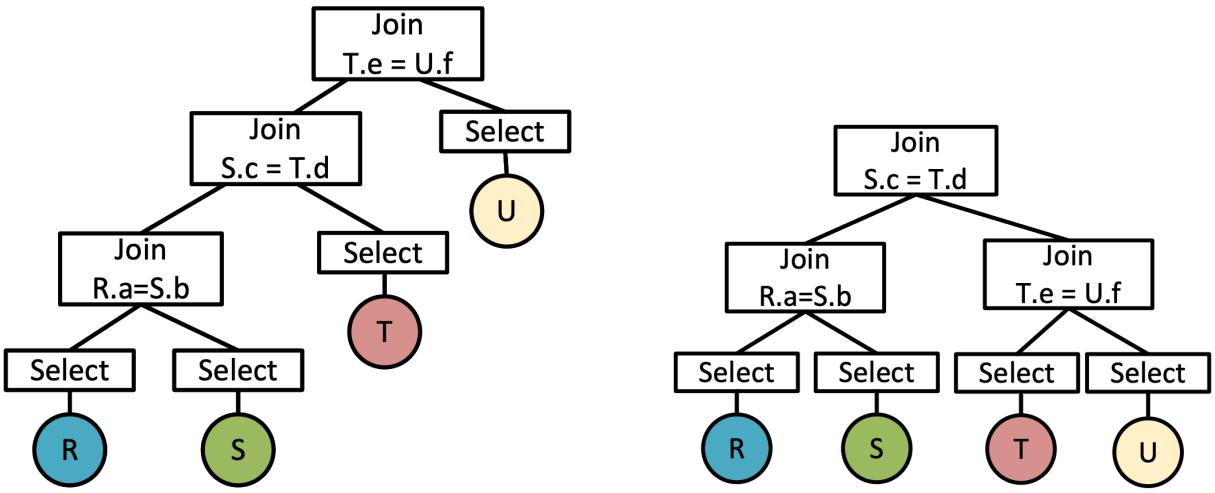
总的来说，一个优秀的查询优化器应该具备以下几个特点：

1. 考虑足够大的潜在的计划的搜索空间。
2. 对执行计划的成本能够准备的建模，以便能够在不同的计划中找到相对来说更高效的计划。
3. 提供一种搜索算法可以高效的找到低成本的执行计划。

1.2 System R 查询优化器

IBM 的 System R 项目是查询优化领域的先驱者。我们简单的回顾一下 System R 的查询优化器是如何解决在第 1.1 节中提到的这些挑战的。该项目中使用的技术对所有后来者都产生了深远的影响（包括可扩展的查询优化器）。

搜索空间 System R 基于成本的查询优化的计划选择主要聚焦于选择-投影-连接（Select-Project-Join，SPJ）这几类查询。对于 Select 这个逻辑算子的物理算子实现包含 `Table Scan` 以及 `Index Scan` 。对于 Join，System R 提供了两个物理算子： `Nested Loops Join` 以及 `Merge Join` （需要两个连接的列都是有序的）。对于在 Query 1 的例子，就像图 1-2 以及图 1-3 中展示的一样，对于 SPJ 查询来说，有着众多逻辑查询树以及执行计划。这主要是因为 Join 操作具有结合率以及交换律，并且 Scan 以及 Join 存在多种物理算子的实现。System R 针对 SPJ 查询的搜索空间局限于二叉连接操作的线性序列（*linear squence*）空间。例如 $Join(Join(R, S), T), U)$。在图 1-4 左侧展示了一个线性连接的序列的例子。而图右侧的逻辑查询树并不在 System R 的搜索空间内。此外，优化器还提供了基于程序分析而并非基于代价的提高嵌套查询效率的技术。



(a) Linear plan　　**(b)** Bushy plan

图 1-4　线性序列空间（Linear）以及浓密（Bushy）空间

代价估计 System R 使用公式来估算执行计划中每一个算子的 CPU 以及 I/O 成本。与现在的查询优化器不同，System R 中并没有将内存纳入到成本的估算中。System R 的优化器维护了一系列关于表以及索引的统计信息。例如：表的行数、表的数据页的数量、索引的页数、每一列中不同值的数量。System R 提供了一组公式来计算单个谓词或者 Join 谓词的被选择的可能性。对于包含多个 Select 的 `WHERE` 语句来说，被选择的可能性由所有谓词的选择的可能性的乘积来决定（这也是假设了每个谓词都是相互独立的）。因此 Join 输出的基数估计为：两个输入关系的基数的乘积×所有谓词的选择性。成本模型公式与基于表与索引的统计信息相结合，就能使得 System R 可以估算出每一个执行计划的 CPU 以及 I/O 成本。

搜索算法 System R 的优化器使用动态规划（*dynamic projramming*）这种搜索算法来找到“最优”的 Join 顺序。这种算法核心依赖于代价模型是是满足最优性原理（*principle of optimality*）的。换句话说，它假设在一个线性序列的 Join 中，n 个 Join 的最优计划可以通过将 n-1 个最优计划扩展一个额外的 Join 来获得。例如：R、S、T的最优连接计划是P_{RST}，那么P_{RST}可以通过三种方式来获得：将R与P_{ST}连接、将S与P_{RT}连接、将T与P_{RS}连接。其中，P_{ST}、P_{RT}、P_{RS}分别是S与T、R与T、R与S的最优 Join 计划。与枚举所有 Join 的朴素方式（时间复杂度为$O(n!)$）不同，动态规划仅枚举了$O(\mathrm{n}2^{n-1})$个计划，因此速度更快。尽管时间复杂度依旧是随着连接的数量而指数增长的。

System R 的搜索算法中还有一个比较重要的方面：有趣的排序（*interesting orders*）[1]。以一个 Join 三张表 R、S、T 的查询 Q 为例，其连接谓词$R.a = S.a$和$S.a = T.a$。假设使用 `Index Seek` 在 S 表执行 `Nested Loops Join` 的成本低于使用 `Merge Join` 的成本。此时优化器在考虑 R、S、T 的连接计划时，会剪枝掉使用 `Merge Join` 的 R 和 S 的计划。然而使用 `Merge Join` 会使得连接结果按照 `a` 排序，这可能会显著降低后续与 T 表的 `Merge Join` 的成本。因此这种剪枝方式会导致查询的次优解。

这里的核心问题在于，算子输出的有序性（也就是算子具有有趣性）可能会降低父算子亦或者祖先算子的成本。为了在动态规划的框架下处理因有趣性导致的与最优性原理冲突，并且保留动态规划的优势，搜索算法在枚举每一个表达式的时候都会考虑有趣性。具体来说，当且仅当表达式的有趣性相同时，才对其计划进行代价的比较[2]，并且为不同的有序序列保留一个最优的计划。

1.3 可扩展的查询优化器架构

System R 引用的重要的概念几乎被所有主流查询优化器采用，包括使用统计数据以及代价模型确定执行计划、基于动态规划的 Join 顺序以及考虑有趣性的必要性。然而这种框架无法以基于成本的方式灵活的、高效的扩展到关系代数中的其他等价变换以及新形态的数据库中，这会错失更低成本的查询计划。随着关系型数据库以及 SQL 在决策中愈发重要，这些额外的代数等

[1]译者注：指合并的结果是有序的。

[2]译者注：同一个逻辑表达式的不同物理计划，若有趣性不同（例如一个有序一个无序），将视为不同的子问题，分别保留最优解。例如$R \bowtie S$会同时保留 `Merge Join` 以及 `Nested Loops Join` 的结果，即使其中一个在当前步骤中成本较高。

价变换在生成更加高效的执行计划中也变得日益重要。这些代数等价变化包括：将 Group-by 下推到 Join 下面执行以降低 Join 的成本、优化非关联写非交换的 Outer Joins 以及去除嵌套查询的关联性。此外数据库系统引入的一些新的架构可以显著的降低执行的成本，例如物化视图（materialized views）[8,9]通过预计算并存储查询子表达式的结果来显著降低查询执行的成本，这对 OLAP 以及其他分析型工作至关重要。此外优化器还需要为了支持更加高效的执行 SQL 查询而引入新的逻辑/物理算子，例如 Apply[10]。

幸运的是，随着 SQL 查询优化器的实际要求在不断地扩展，关于可以扩展的数据库系统的研究为扩展 System R 架构提高来替代方案。可扩展的数据库系统被能够根据应用的需求定制数据库系统。具体的来说，Exodus[11]以及后来的 Volcano[3]（可以支持用户指定的查询执行算子）都是在这种背景下出现的。由于需要支持自定义的算子，所以提供可扩展的查询优化器框架成为了必然。因此，优化器的可扩展性一开始就是 Volcano 的特性。并且 Volcano 最初被设想为“用于多种目的的实验工具”[3]。它允许架构师从专家系统（产生式系统）中获取灵感，设计可插拔一个新的规则（*rules*），从而扩展优化器的功能。后来，Volcano/Cascades[12,13]和 Starburst[14-16]等可扩展的优化器框架将 SQL 查询优化作为了核心的应用场景，这满足了 SQL 查询优化对新架构的迫切要求。

在本文的大部分内容中，我们都将聚焦于 Volcano/Cascades 的可扩展优化器。这些可扩展的优化框架都围绕规则（rules）这个概念展开。逻辑转换规则（*logical transformation rule*）表示的是 SQL（或者是关系代数）的等价关系。例如前面提到的 Join 的交换律以及结合率所蕴含的等价关系就可以通过规则来表达。类似，一条规则可以定义将 Group-by 下推到 Join 下的等价关系。对一个查询树应用逻辑转换规则可以生成一个等价的查询树。实现规则（*implementation rule*）定义了逻辑算子（例如 Join）到物理算子（例如 Hash Join）的映射。要为查询生成一个执行计划，就需要用到实现规则。合理的选择一系列规则的应用方式，就可以将查询树转换一个高效的执行计划。有一点需要注意，在这种架构下，每次引入一个新的算子、逻辑转换以及实现规则的时候，无须修改优化器的搜索算法。但是必须注意到，转换并不一定会降低成本，因此搜索算法必须以基于成本的方式在各种替代方案中选择。

SQL 是一个声明式的查询语言，这允许查询优化器为 SQL 查询创建一个高效的执行计划。这些执行计划既保持了语义等价的逻辑转换，又能聪明的为逻辑算子选择高效的实现方式。查询优化的终极目标就是生成语义等价的、与用户/应用无关的，高效的，执行计划。可扩展的优化器通过引入规则，对查询树依次应用规则，并且在基于成本的搜索算法的驱动下，达成这个目标。

1.4 大纲

在本章节，我们聚焦来可扩展的查询优化器的技术，并且以 Microsoft SQL Server 为例阐述了其中的关键概念。与本文的另一个作者的撰写的关于查询优化的概述文章[1]相比，本书详细介绍了优化器的框架，以及在实际中常用的额外的转换规则。我们通过伪代码以及示例深入讲解了可扩展性框架以及规则。

第 2 章：我们回顾了以下可扩展的查询优化器框架 Volcano 以及其极具影响力的后来者，Cascades 框架。我们描述了查询算法以及这两种框架所必须的一些关键数据结构，以及能有效提高查询效率的其他技术。此外，我们通过几个例子展示了 Microsoft SQL Server 的查询优化器是如何利用 Cascades 框架的。最后，我们还说了一下查询优化器是如何应对并行以及分布式查询流程。

第 3 章：在这里我们简单的回顾了一下其他可扩展的查询优化器，包括 IBM DB2 使用的 Starburst、Greenplum DB 使用的 Orca、Apache Hive 使用的 Calcite、SparkSQL 使用的 Catalyst。尽管 PostgreSQL 的查询优化器并不具备 Volcano 以及 Cascades 等框架的可扩展性，但是鉴于其受欢迎的程度，我们也对其查询优化器进行一些简要的概述。

第 4 章：可扩展的优化器的有效性来自于其应用的规则。在这一章节，我们会回顾一些关键的逻辑转换规则实现规则。这些规则包括：基表的访问路径（access paths to base tables）、内外连接（inner/outer joins）、分组（group-by）、聚合（aggregation）以及嵌套查询的去相关化（decorrelation of nested queries）。我们还会精选一些“高级”的规则，例如用于优化常见于数仓的星形以及雪花型查询的规则、侧向信息传递规则（sideways information passing）、用户定义函数（UDFS, user-defined functions）以及物化视图（materialized views）。

第 5 章：优化器框架严重依赖代价模型以及基数估计，在本章节，我们将概述模型以及基数估计，聚焦在业界的实现上。我们将会讨论优化器使用的统计汇总方法，例如直方图，以及它们是如何利用在复杂的查询中。此外，我们还将探讨数据库系统中对采样（Sampling）和草图（Sketches）技术的应用。最后，我们将以 Microsoft SQL Server 为例来说明这些概念以及技术。

第 6 章：大多数的查询优化的文章都忽略了数据库的整个生命周期内对优化器生成的计划进行管理的问题。而计划的管理会对整体的工作负载的性能起到关键性的影响。在这样的背景下，我们讨论了几个重要的挑战：（a）计划的缓存以及失效（b）利用执行的反馈改进次优计划（c）查询提示，它允许用户影响优化器的计划选择（d）优化参数化查询。

第 7 章：虽然本书主要围绕着实践中的可扩展的查询优化器展开，但是我们在本章节提及一些尚未解决的问题以及正在探索的几个研究方向。

勘误与更新：我们提供了勘误与更新[3]，我们鼓励读者发现错误后通过邮件向我们报告错误。

1.5 建议阅读

- Access Path Selection in a Relational Database Management System[17]
- Query Evaluation Techniques for Large Databases[2]
- An Overview of Query Optimization in Relational Systems[1]

[3]https://www.microsoft.com/en-us/research/project/extensible-query-optimizers-in-practice-errata-and-updates/

第二章 可扩展的查询优化器

构建一个可以扩展的查询优化器的方式之一是拥有一组可以扩展的规则（*rule*），这个规则用于定义所有等效的计划的空间。正如第 1 章中所提及的，这种方法以逻辑算子、物理算子以及一系列的转换（*transformation*）和实现规则（*implementation rules*）的概念为中心。优化器在搜索策略（*search strategy*）的指导下，按照一定的顺序应用规则（rules），在等效的计划空间内探索，并且在众多备选计划空间中选择一个高效的计划。

在这一章中，我们首先介绍一个扩展的优化器的概念（第 2.1 节）。然后我们深入讨论两个可以扩展的优化器框架：Volcano 框架以及 Cascades 框架。我们从介绍 Volcano 以及其搜索开始（第 2.2 节），然后我们简单的看下 Volcano 的局限性。正因其局限性，催生了其后来者：Cascades 框架（第 2.3 节）。我们介绍了在实践中用于提高 Volcano 和 Cascades 效率的其他优化以及启发式方法（第 2.4 节）。为了说明可扩展的查询优化器如何轻松的将新功能合并到查询处理中，我们介绍了 Microsoft SQL Server 的优化器是如何利用 Cascades 框架的可扩展性来实现列存的（第 2.5 节）。在本章的最后，我们将介绍可扩展的查询优化器是如何生成多核并行以及分布式的查询（第 2.6 节）。

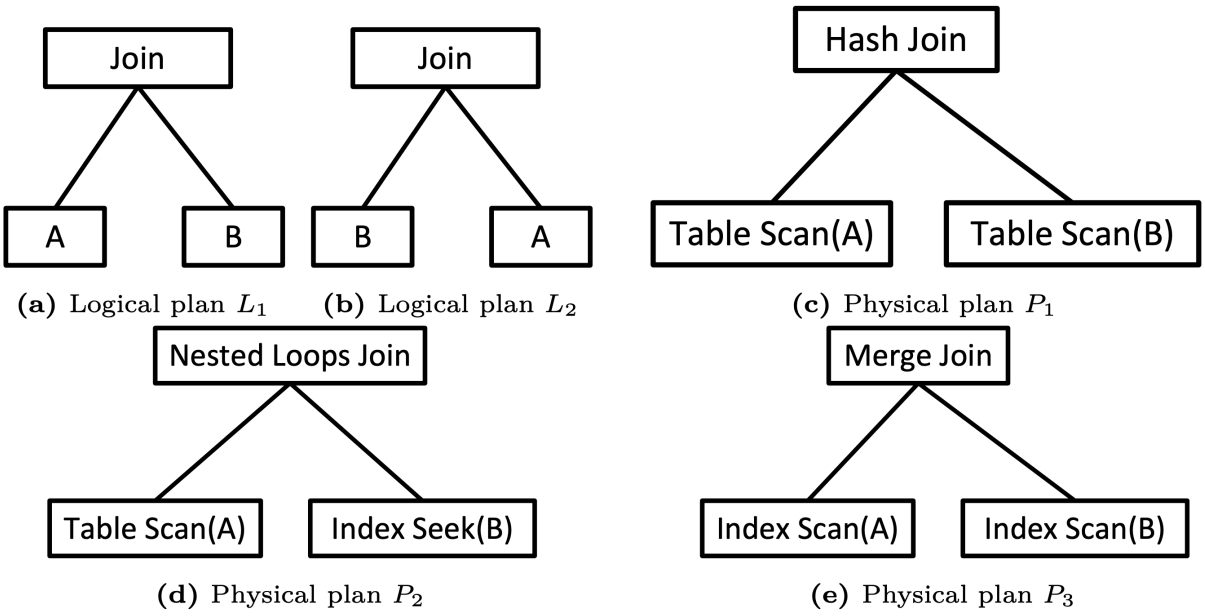
2.1 基本概念

我们在基于规则的（rule-based）的可扩展优化器（Volcano/Cascades）中介绍了一些重要的概念。但是需要注意，这些概念（例如：算子 operators、属性 properties）并不是 Volcano/Cascades 中所独有的概念，它们也被用于 System R、Starburst 以及 EXODUS 等系统的查询优化器中。

看一下 Query 2，其相应的逻辑以及物理计划在图 2-1 中。

Query 2

```
SELECT *
FROM A, B
WHERE A.k = B.k
```



(a) Logical plan L_1 **(b)** Logical plan L_2 **(c)** Physical plan P_1

(d) Physical plan P_2 **(e)** Physical plan P_3

图 2-1

逻辑以及物理算子 逻辑算子定义了一个或者多个关系的关系操作，例如在图 2-1 中 A 和 B 之间的 Join 运算符。这里有一点需要注意，在查询优化器中，可能会引入非关系的逻辑算子，例如 `Apply` 。 `Apply` 用来处理子查询（第 4.5.3 节）以及用于处理并行性的交换操作（第 2.6.1 节）。因此使用逻辑算子的集合以及由此产生的优化器的搜索空间，超过了 SQL 查询中所呈现的范围。物理算子是一种算法的实现，用于执行在查询执行中所需的操作。物理算子的一个例子是 `Hash Join` （图 2-1）。注意，一个逻辑算子可以由不同的物理算子来实现，反之亦然。例如，逻辑算子 `Join` 可以由 `Hash Join` 和 `Nested Loops Join` 来实现。类似，物理算子 `Hash Join` 可以用于实现多种逻辑算子，例如 `Join` 、 `Left Outer Join` 以及 `Union` 。此外，一个逻辑算子可以由多个物理算子来实现。例如我们在 第 4.2 节将会介绍的逻辑算子 `Inner Join` ，该算子一个通过 `Sort` 以及 `Merge Join` 算子来实现。

逻辑以及物理表达式 一个逻辑表达式是由逻辑算子构成的树状结构。它代表着一个关系代数表达式。例如 Query 2 中的连接操作就可以使用$L_1 : A \bowtie B$表示，也正如图 2-1所表示的一样。一个物理表达式是一个由物理算子构成的树状结构，它也被成为物理计划（*physical plan*）或者简称为计划（*plan*）。例如表达式$P_1 : HashJoin(TableScan(A), TableScan(B))$表示的就是图 2-1 中的$L_1$实现。

逻辑以及物理属性 举一个表达式的逻辑属性的例子，就如表达式的关系代数表达式以及表达式的基数（*cardinality*）。例如$A \bowtie B$和$B \bowtie A$都产生了 A 与 B join 的结果，且它们拥有相同的逻辑数据（例如基数）

表达式的等价性 如果两个表达式的属性是相同的，那么就认为这两个表达式是等价的。例如图 2-1 表达式$L_1 : A \bowtie B$与$L_2 : B \bowtie A$是等价的。如果两个物理表达式的逻辑以及物理属性是相同的，那么也认为这两个物理表达式是等价的。例如下面两个表达式就认为是等价的。

P_1 : HashJoin(TableScan(A), TableScan(B))

P_2 : NestedLoopsJoin(TableScan(A), TableScan(B))

规则 Rules

2.2 Volcano

2.2.1 简介

2.2.2 查询

算法 1: 在 Volcano 优化器中查询，在 `GenerateLogicalExpr` 、 `MatchTransRule` 、 `UpdatePlan` 这几个操作中，备忘录（Mono）中的组和表达式会被更新。随着搜索结果的推进，表达式的成本的限制也会被更新。在 `FindBestPlan` 操作结束以后，搜索得到的缓存结果会被添加到 Mono 中。

```
1: function GenerateLogicalExpr(LogExpr, Rules) ▷
2:     for Child in inputs of LogExpr
3:         if Group(Child) ∉ Memo then
4:             GererateLogicalExpr(Child)
5:     MatchTransRules(LogExpr, Rules)
6: function MatchTransRuleLogExpr, Rules
7:     for rule in Rules do
8:         if rule matches LogExpr then
9:             NewLogExpr ← Transformm(LogExpr, rule) ▷ 更新 memo 并且记录其邻居
10:            GenerateLogicalExpr(NewLogExpr)
11:            ▷ 只会在NewLogExpr在 memo 中不存在的时候才会执行
```

2.2.3 自定义查询策略

2.2.4 添加新的规则以及运算符

2.3 Cascades

2.3.1 Cascades 的主要改进

2.3.2 查询简介

2.3.3 查询算法

2.3.4 Cascades 中的查询优化示例

2.4 提高查询效率的技术

2.5 Microsoft SQL Server 的扩展性示例

2.6 并行分布式查询流程

2.6.1 多核并行

2.6.2 分布式查询优化

2.7 建议阅读

第三章 业界其他可扩展的查询优化器

3.1 Starburst

3.2 Orca

3.3 Calcite

3.4 PostgreSQL

3.5 建议阅读

第四章 执行计划的关键转化

4.1 访问方式转换

4.2 Inner Join 转换

4.2.1 Join 的交换律以及结合率

4.2.2 将过滤下推（上拉）到 Join 之下（之上）

4.2.3 Inner Join 的物理转换规则

4.3 Outer Join 转换

4.3.1 Join 的交换律以及结合率

4.3.2 冗余规则

4.4 Group By 转换

4.4.1 完成 Group By 的下推

4.5 去相关化

4.5.1 没有相关变量的嵌套子查询

4.5.2 有相关变量的嵌套子查询

4.5.3 使用 Apply 代数表示子查询

4.5.4 具有相关变量的子查询的查询的其他优化

4.6 其他关键转化规则

4.6.1 星形以及雪花

4.6.2 横向信息传递

4.6.3 用户自定义函数（UDFs）

4.6.4 物化视图

4.7 建议阅读

第五章 代价估计

5.1 代价估计简介

5.2 代价模型

5.2.1 代价模型校准

5.3 统计学

5.3.1 直方图

5.3.2 草图

5.3.3 采样

5.3.4 统计管理

5.4 基数估计

5.4.1 要求与挑战

5.4.2 关键技术

5.4.3 状态与限制

5.5 例子：Microsoft SQL Server 中的代价估计

5.5.1 代价模型

5.5.2 统计学

5.5.3 基数估计

5.6 建议阅读

第六章 执行计划的管理

6.1 计划缓存（Plan Caching）与失效

6.2 使用执行的反馈改进次优执行计划

6.3 使用提示（Hints）影响执行计划

6.3.1 Microsoft SQL Server 中的提示

6.3.2 提示的实现

6.4 优化参数化查询

6.4.1 参数查询优化中的挑战

6.4.2 确定需要缓存的计划

6.4.3 为查询示例选择需要执行的计划

6.5 建议阅读

第七章 开放性问题

7.1 健壮的查询处理

7.2 查询结果缓存

7.3 结果反馈驱动的统计

7.4 利用机器学习进行查询优化

7.5 查询优化中的其他研究主题

7.6 最大的问题

第八章 附录

第九章 致谢

略

第十章 参考文献

[1] CHAUDHURI S. An overview of query optimization in relational systems[C/OL]//Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. Seattle, Washington, USA: Association for Computing Machinery, 1998: 34-43. https://doi.org/10.1145/275487.275492. DOI:10.1145/275487.275492.

[2] GRAEFE G. Query Evaluation Techniques for Large Databases[J/OL]. ACM Comput. Surv., 1993, 25(2): 73-170. https://doi.org/10.1145/152610.152611. DOI:10.1145/152610.152611.

[3] GRAEFE G. Volcano - An Extensible and Parallel Query Evaluation System[J/OL]. IEEE Trans. Knowl. Data Eng., 1994, 6(1): 120-135. https://doi.org/10.1109/69.273032. DOI:10.1109/69.273032.

[4] BONCZ P A, ZUKOWSKI M, NES N. MonetDB/X100: Hyper-Pipelining Query Execution[C/OL]//Second Biennial Conference on Innovative Data Systems Research, CIDR 2005, Asilomar, CA, USA, January 4-7, 2005, Online Proceedings. www.cidrdb.org, 2005: 225-237. http://cidrdb.org/cidr2005/papers/P19.pdf.

[5] STONEBRAKER M, ABADI D J, BATKIN A, et al. C-store: a column-oriented DBMS[Z/OL]//BRODIE M L. Making Databases Work: the Pragmatic Wisdom of Michael Stonebraker: Vol. 22. ACM / Morgan & Claypool, 2019: 491-518. https://doi.org/10.1145/3226595.3226638. DOI:10.1145/3226595.3226638.

[6] KERSTEN T, LEIS V, KEMPER A, et al. Everything You Always Wanted to Know About Compiled and Vectorized Queries But Were Afraid to Ask[J/OL]. Proc. VLDB Endow., 2018, 11(13): 2209-2222. http://www.vldb.org/pvldb/vol11/p2209-kersten.pdf. DOI:10.14778/3275366.3275370.

[7] FAGIN R. Normal Forms and Relational Database Operators[C/OL]//BERNSTEIN P A. Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data, Boston, Massachusetts, USA, May 30 - June 1. ACM, 1979: 153-160. https://doi.org/10.1145/582095.582120. DOI:10.1145/582095.582120.

[8] CHIRKOVA R, YANG J. Materialized Views[J/OL]. Found. Trends Databases, 2012, 4(4): 295-405. https://doi.org/10.1561/1900000020. DOI:10.1561/1900000020.

[9] Gupta A, Mumick I S. Materialized views: techniques, implementations, and applications[M]. Cambridge, MA, USA: MIT Press, 1999.

[10] GALINDO-LEGARIA C A, JOSHI M. Orthogonal Optimization of Subqueries and Aggregation[C/OL]//MEHROTRA S, SELLIS T K. Proceedings of the 2001 ACM

SIGMOD international conference on Management of data, Santa Barbara, CA, USA, May 21-24, 2001. ACM, 2001: 571-581. https://doi.org/10.1145/375663.375748. DOI:10.1145/375663.375748.

[11] CAREY M J, DEWITT D J, GRAEFE G, et al. The EXODUS extensible DBMS project: an overview[M]//Readings in Object-Oriented Database Systems. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989: 474-499.

[12] GRAEFE G. The Cascades Framework for Query Optimization[J/OL]. IEEE Data Eng. Bull., 1995, 18(3): 19-29. http://sites.computer.org/debull/95SEP-CD.pdf.

[13] GRAEFE G, MCKENNA W J. The Volcano Optimizer Generator: Extensibility and Efficient Search[C/OL]//Proceedings of the Ninth International Conference on Data Engineering, April 19-23, 1993, Vienna, Austria. IEEE Computer Society, 1993: 209-218. https://doi.org/10.1109/ICDE.1993.344061. DOI:10.1109/ICDE.1993.344061.

[14] LOHMAN G M. Grammar-like functional rules for representing query optimization alternatives[J/OL]. SIGMOD Rec., 1988, 17(3): 18-27. https://doi.org/10.1145/971701.50204. DOI:10.1145/971701.50204.

[15] LOHMAN G M. Grammar-like functional rules for representing query optimization alternatives[C/OL]//Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data. Chicago, Illinois, USA: Association for Computing Machinery, 1988: 18-27. https://doi.org/10.1145/50202.50204. DOI:10.1145/50202.50204.

[16] PIRAHESH H, LEUNG T Y C, HASAN W. A Rule Engine for Query Transformation in Starburst and IBM DB2 C/S DBMS[C/OL]//GRAY W A, LARSON P Å. Proceedings of the Thirteenth International Conference on Data Engineering, April 7-11, 1997, Birmingham, UK. IEEE Computer Society, 1997: 391-400. https://doi.org/10.1109/ICDE.1997.581945. DOI:10.1109/ICDE.1997.581945.

[17] SELINGER P G, ASTRAHAN M M, CHAMBERLIN D D, et al. Access Path Selection in a Relational Database Management System[C/OL]//BERNSTEIN P A. Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data, Boston, Massachusetts, USA, May 30 - June 1. ACM, 1979: 23-34. https://doi.org/10.1145/582095.582099. DOI:10.1145/582095.582099.

[18] WU W, CHI Y, ZHU S, et al. Predicting query execution time: Are optimizer cost models really unusable[C/OL]//JENSEN C S, JERMAINE C M, ZHOU X. 29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013. IEEE Computer Society, 2013: 1081-1092. https://doi.org/10.1109/ICDE.2013.6544899. DOI:10.1109/ICDE.2013.6544899.

[19] CHEN T, GUESTRIN C. XGBoost: A Scalable Tree Boosting System[C/OL]//KRISHNAPURAM B, SHAH M, SMOLA A J, et al. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016. ACM, 2016: 785-794. https://doi.org/10.1145/2939672.2939785. DOI:10.1145/2939672.2939785.

[20] CHEUNG A, SOLAR-LEZAMA A, MADDEN S. Optimizing database-backed applications with query synthesis[C/OL]//BOEHM H J, FLANAGAN C. ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '13, Seattle, WA, USA, June 16-19, 2013. ACM, 2013: 3-14. https://doi.org/10.1145/2491956.2462180. DOI:10.1145/2491956.2462180.

[21] AGARWAL P K, CORMODE G, HUANG Z, et al. Mergeable summaries[C/OL]//BENEDIKT M, KRÖTZSCH M, LENZERINI M. Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2012, Scottsdale, AZ, USA, May 20-24, 2012. ACM, 2012: 23-34. https://doi.org/10.1145/2213556.2213562. DOI:10.1145/2213556.2213562.

[22] CHU F C, HALPERN J Y, SESHADRI P. Least Expected Cost Query Optimization: An Exercise in Utility[C/OL]//VIANU V, PAPADIMITRIOU C H. Proceedings of the Eighteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 31 - June 2, 1999, Philadelphia, Pennsylvania, USA. ACM Press, 1999: 138-147. https://doi.org/10.1145/303976.303990. DOI:10.1145/303976.303990.

[23] ABOULNAGA A, CHAUDHURI S. Self-tuning Histograms: Building Histograms Without Looking at Data[C/OL]//DELIS A, FALOUTSOS C, GHANDEHARIZADEH S. SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA. ACM Press, 1999: 181-192. https://doi.org/10.1145/304182.304198. DOI:10.1145/304182.304198.

[24] ARMBRUST M, XIN R S, LIAN C, et al. Spark SQL: Relational Data Processing in Spark[C/OL]//SELLIS T K, DAVIDSON S B, IVES Z G. Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015. ACM, 2015: 1383-1394. https://doi.org/10.1145/2723372.2742797. DOI:10.1145/2723372.2742797.

[25] ARMENATZOGLOU N, BASU S, BHANOORI N, et al. Amazon Redshift Re-invented[C/OL]//IVES Z G, BONIFATI A, ABBADI A E. SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022. ACM, 2022: 2205-2217. https://doi.org/10.1145/3514221.3526045. DOI:10.1145/3514221.3526045.

[26] BABCOCK B, CHAUDHURI S. Towards a Robust Query Optimizer: A Principled and Practical Approach[C/OL]//ÖZCAN F. Proceedings of the ACM SIGMOD

International Conference on Management of Data, Baltimore, Maryland, USA, June 14-16, 2005. ACM, 2005: 119-130. https://doi.org/10.1145/1066157.1066172. DOI:10.1145/1066157.1066172.

[27] KOCSIS L, SZEPESVÁRI C. Bandit Based Monte-Carlo Planning[C/OL]// FÜRNKRANZ J, SCHEFFER T, SPILIOPOULOU M. Machine Learning: ECML 2006, 17th European Conference on Machine Learning, Berlin, Germany, September 18-22, 2006, Proceedings: Vol. 4212. Springer, 2006: 282-293. https://doi.org/10.1007/11871842/_29. DOI:10.1007/11871842_29.

[28] BEGOLI E, CAMACHO-RODRÍGUEZ J, HYDE J, et al. Apache Calcite: A Foundational Framework for Optimized Query Processing Over Heterogeneous Data Sources[C/OL]//DAS G, JERMAINE C M, BERNSTEIN P A. Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018. ACM, 2018: 221-230. https://doi.org/10.1145/3183713.3190662. DOI:10.1145/3183713.3190662.

[29] BRUNO N, CHAUDHURI S. Exploiting statistics on query expressions for optimization[C/OL]//FRANKLIN M J, MOON B, AILAMAKI A. Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, USA, June 3-6, 2002. ACM, 2002: 263-274. https://doi.org/10.1145/564691.564722. DOI:10.1145/564691.564722.

[30] BRUNO N, CHAUDHURI S, GRAVANO L. STHoles: A Multidimensional Workload-Aware Histogram[C/OL]//MEHROTRA S, SELLIS T K. Proceedings of the 2001 ACM SIGMOD international conference on Management of data, Santa Barbara, CA, USA, May 21-24, 2001. ACM, 2001: 211-222. https://doi.org/10.1145/375663.375686. DOI:10.1145/375663.375686.

[31] BRUNO N, GALINDO-LEGARIA C A, JOSHI M, et al. Unified Query Optimization in the Fabric Data Warehouse[C/OL]//BARCELÓ P, SÁNCHEZ-PI N, MELIOU A, et al. Companion of the 2024 International Conference on Management of Data, SIGMOD/PODS 2024, Santiago AA, Chile, June 9-15, 2024. ACM, 2024: 18-30. https://doi.org/10.1145/3626246.3653369. DOI:10.1145/3626246.3653369.

[32] CHAUDHURI S, DAS G, SRIVASTAVA U. Effective Use of Block-Level Sampling in Statistics Estimation[C/OL]//WEIKUM G, KÖNIG A C, DESSLOCH S. Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004. ACM, 2004: 287-298. https://doi.org/10.1145/1007568.1007602. DOI:10.1145/1007568.1007602.

[33] CHAUDHURI S, MOTWANI R, NARASAYYA V R. Random Sampling for Histogram Construction: How much is enough?[C/OL]//HAAS L M, TIWARY A. SIGMOD 1998,

Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA. ACM Press, 1998: 436-447. https://doi.org/10.1145/276304.276343. DOI:10.1145/276304.276343.

[34] CHAUDHURI S, MOTWANI R, NARASAYYA V R. On Random Sampling over Joins[C/OL]//DELIS A, FALOUTSOS C, GHANDEHARIZADEH S. SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA. ACM Press, 1999: 263-274. https://doi.org/10.1145/304182.304206. DOI:10.1145/304182.304206.

[35] CHEN C M, ROUSSOPOULOS N. Adaptive Selectivity Estimation Using Query Feedback[C/OL]//SNODGRASS R T, WINSLETT M. Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, Minneapolis, Minnesota, USA, May 24-27, 1994. ACM Press, 1994: 161-172. https://doi.org/10.1145/191839.191874. DOI:10.1145/191839.191874.

[36] COLE R L, GRAEFE G. Optimization of Dynamic Query Evaluation Plans[C/OL]//SNODGRASS R T, WINSLETT M. Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, Minneapolis, Minnesota, USA, May 24-27, 1994. ACM Press, 1994: 150-160. https://doi.org/10.1145/191839.191872. DOI:10.1145/191839.191872.

[37] ANTOSHENKOV G. Dynamic Query Optimization in Rdb/VMS[C/OL]//Proceedings of the Ninth International Conference on Data Engineering, April 19-23, 1993, Vienna, Austria. IEEE Computer Society, 1993: 538-547. https://doi.org/10.1109/ICDE.1993.344026. DOI:10.1109/ICDE.1993.344026.

[38] GOLDSTEIN J, LARSON P Å. Optimizing Queries Using Materialized Views: A practical, scalable solution[C/OL]//MEHROTRA S, SELLIS T K. Proceedings of the 2001 ACM SIGMOD international conference on Management of data, Santa Barbara, CA, USA, May 21-24, 2001. ACM, 2001: 331-342. https://doi.org/10.1145/375663.375706. DOI:10.1145/375663.375706.

[39] HAAS L M, FREYTAG J C, LOHMAN G M, et al. Extensible Query Processing in Starburst[C/OL]//CLIFFORD J, LINDSAY B G, MAIER D. Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data, Portland, Oregon, USA, May 31 - June 2, 1989. ACM Press, 1989: 377-388. https://doi.org/10.1145/67544.66962. DOI:10.1145/67544.66962.

[40] AVNUR R, HELLERSTEIN J M. Eddies: Continuously Adaptive Query Processing[C/OL]//CHEN W, NAUGHTON J F, BERNSTEIN P A. Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000,

Dallas, Texas, USA. ACM, 2000: 261-272. https://doi.org/10.1145/342009.335420. DOI:10.1145/342009.335420.

[41] HELLERSTEIN J M, STONEBRAKER M. Predicate Migration: Optimizing Queries with Expensive Predicates[C/OL]//BUNEMAN P, JAJODIA S. Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, DC, USA, May 26-28, 1993. ACM Press, 1993: 267-276. https://doi.org/10.1145/170035.170078. DOI:10.1145/170035.170078.

[42] HUAI Y, CHAUHAN A, GATES A, et al. Major technical advancements in apache hive[C/OL]//DYRESON C E, LI F, ÖZSU M T. International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014. ACM, 2014: 1235-1246. https://doi.org/10.1145/2588555.2595630. DOI:10.1145/2588555.2595630.

[43] IOANNIDIS Y E, CHRISTODOULAKIS S. On the Propagation of Errors in the Size of Join Results[C/OL]//CLIFFORD J, KING R. Proceedings of the 1991 ACM SIGMOD International Conference on Management of Data, Denver, Colorado, USA, May 29-31, 1991. ACM Press, 1991: 268-277. https://doi.org/10.1145/115790.115835. DOI:10.1145/115790.115835.

[44] LI B, LU Y, KANDULA S. Warper: Efficiently Adapting Learned Cardinality Estimators to Data and Workload Drifts[C/OL]//IVES Z G, BONIFATI A, ABBADI A E. SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022. ACM, 2022: 1920-1933. https://doi.org/10.1145/3514221.3526179. DOI:10.1145/3514221.3526179.

[45] MACKERT L F, LOHMAN G M. R* Optimizer Validation and Performance Evaluation for Local Queries[C/OL]//ZANIOLO C. Proceedings of the 1986 ACM SIGMOD International Conference on Management of Data, Washington, DC, USA, May 28-30, 1986. ACM Press, 1986: 84-95. https://doi.org/10.1145/16894.16863. DOI:10.1145/16894.16863.

[46] MARKL V, RAMAN V, SIMMEN D E, et al. Robust Query Processing through Progressive Optimization[C/OL]//WEIKUM G, KÖNIG A C, DESSLOCH S. Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004. ACM, 2004: 659-670. https://doi.org/10.1145/1007568.1007642. DOI:10.1145/1007568.1007642.

[47] BOROVICA-GAJIC R, IDREOS S, AILAMAKI A, et al. Smooth Scan: Statistics-oblivious access paths[C/OL]//GEHRKE J, LEHNER W, SHIM K, et al. 31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015.

IEEE Computer Society, 2015: 315-326. https://doi.org/10.1109/ICDE.2015.7113294. DOI:10.1109/ICDE.2015.7113294.

[48] SCHMIDT T, KIPF A, HORN D, et al. Predicate Caching: Query-Driven Secondary Indexing for Cloud Data Warehouses[C/OL]//BARCELÓ P, SÁNCHEZ-PI N, MELIOU A, et al. Companion of the 2024 International Conference on Management of Data, SIGMOD/PODS 2024, Santiago AA, Chile, June 9-15, 2024. ACM, 2024: 347-359. https://doi.org/10.1145/3626246.3653395. DOI:10.1145/3626246.3653395.

[49] SHANKAR S, NEHME R V, AGUILAR-SABORIT J, et al. Query optimization in microsoft SQL server PDW[C/OL]//CANDAN K S, CHEN Y, SNODGRASS R T, et al. Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20-24, 2012. ACM, 2012: 767-776. https://doi.org/10.1145/2213836.2213953. DOI:10.1145/2213836.2213953.

[50] STONEBRAKER M, ROWE L A. The Design of Postgres[C/OL]//ZANIOLO C. Proceedings of the 1986 ACM SIGMOD International Conference on Management of Data, Washington, DC, USA, May 28-30, 1986. ACM Press, 1986: 340-355. https://doi.org/10.1145/16894.16888. DOI:10.1145/16894.16888.

[51] GALINDO-LEGARIA C A, JOSHI M, WAAS F, et al. Statistics on Views[C/OL]//FREYTAG J C, LOCKEMANN P C, ABITEBOUL S, et al. Proceedings of 29th International Conference on Very Large Data Bases, VLDB 2003, Berlin, Germany, September 9-12, 2003. Morgan Kaufmann, 2003: 952-962. http://www.vldb.org/conf/2003/papers/S28P03.pdf. DOI:10.1016/B978-012722442-8/50089-6.

[52] IOANNIDIS Y E. The History of Histograms (abridged)[C/OL]//FREYTAG J C, LOCKEMANN P C, ABITEBOUL S, et al. Proceedings of 29th International Conference on Very Large Data Bases, VLDB 2003, Berlin, Germany, September 9-12, 2003. Morgan Kaufmann, 2003: 19-30. http://www.vldb.org/conf/2003/papers/S02P01.pdf. DOI:10.1016/B978-012722442-8/50011-2.

[53] CHAUDHURI S, DAYAL U, NARASAYYA V R. An overview of business intelligence technology[J/OL]. Commun. ACM, 2011, 54(8): 88-98. https://doi.org/10.1145/1978542.1978562. DOI:10.1145/1978542.1978562.

[54] HIRZEL M, SOULÉ R, SCHNEIDER S, et al. A catalog of stream processing optimizations[J/OL]. ACM Comput. Surv., 2013, 46(4): 1-34. https://doi.org/10.1145/2528412. DOI:10.1145/2528412.

[55] BRUNO N, CHAUDHURI S, RAMAMURTHY R. Power Hints for Query Optimization[C/OL]//IOANNIDIS Y E, LEE D L, NG R T. Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shang-

hai, China. IEEE Computer Society, 2009: 469-480. https://doi.org/10.1109/ICDE.2009.68. DOI:10.1109/ICDE.2009.68.

[56] HARITSA J R. Robust Query Processing: A Survey[J/OL]. Found. Trends Databases, 2024, 15(1): 1-114. https://doi.org/10.1561/1900000089. DOI:10.1561/1900000089.

[57] NARASAYYA V R, CHAUDHURI S. Cloud Data Services: Workloads, Architectures and Multi-Tenancy[J/OL]. Found. Trends Databases, 2021, 10(1): 1-107. https://doi.org/10.1561/1900000060. DOI:10.1561/1900000060.

[58] GRAEFE G. New algorithms for join and grouping operations[J/OL]. Comput. Sci. Res. Dev., 2012, 27(1): 3-27. https://doi.org/10.1007/s00450-011-0186-9. DOI:10.1007/S00450-011-0186-9.

[59] BERNSTEIN P A, CHIU D M W. Using Semi-Joins to Solve Relational Queries[J/OL]. J. ACM, 1981, 28(1): 25-40. https://doi.org/10.1145/322234.322238. DOI:10.1145/322234.322238.

[60] FLAJOLET P, MARTIN G N. Probabilistic Counting Algorithms for Data Base Applications[J/OL]. J. Comput. Syst. Sci., 1985, 31(2): 182-209. https://doi.org/10.1016/0022-0000(85)90041-8. DOI:10.1016/0022-0000(85)90041-8.

[61] CHAUDHURI S, NARASAYYA V R, RAMAMURTHY R. A pay-as-you-go framework for query execution feedback[J/OL]. Proc. VLDB Endow., 2008, 1(1): 1141-1152. http://www.vldb.org/pvldb/vol1/1453977.pdf. DOI:10.14778/1453856.1453977.

[62] HARITSA J R. Robust Query Processing: Mission Possible[J/OL]. Proc. VLDB Endow., 2020, 13(12): 3425-3428. http://www.vldb.org/pvldb/vol13/p3425-haritsa.pdf. DOI:10.14778/3415478.3415561.

[63] HARMOUCH H, NAUMANN F. Cardinality Estimation: An Experimental Survey[J/OL]. Proc. VLDB Endow., 2017, 11(4): 499-512. http://www.vldb.org/pvldb/vol11/p499-harmouch.pdf. DOI:10.1145/3186728.3164145.

[64] HILPRECHT B, BINNIG C. Zero-Shot Cost Models for Out-of-the-box Learned Cost Prediction[J/OL]. Proc. VLDB Endow., 2022, 15(11): 2361-2374. https://www.vldb.org/pvldb/vol15/p2361-hilprecht.pdf. DOI:10.14778/3551793.3551799.

[65] LEE K, DUTT A, NARASAYYA V R, et al. Analyzing the Impact of Cardinality Estimation on Execution Plans in Microsoft SQL Server[J/OL]. Proc. VLDB Endow., 2023, 16(11): 2871-2883. https://www.vldb.org/pvldb/vol16/p2871-dutt.pdf. DOI:10.14778/3611479.3611494.

[66] CHAUDHURI S, NARASAYYA V R. Automating Statistics Management for Query Optimizers[C/OL]//LOMET D B, WEIKUM G. Proceedings of the 16th International Conference on Data Engineering, San Diego, California, USA, February 28 - March 3,

2000. IEEE Computer Society, 2000: 339-348. https://doi.org/10.1109/ICDE.2000.839433. DOI:10.1109/ICDE.2000.839433.

[67] LEE A W, ZAÏT M. Closing the query processing loop in Oracle 11g[J/OL]. Proc. VLDB Endow., 2008, 1(2): 1368-1378. http://www.vldb.org/pvldb/vol1/1454178.pdf. DOI:10.14778/1454159.1454178.

[68] LU Y, KANDULA S, KÖNIG A C, et al. Pre-training Summarization Models of Structured Datasets for Cardinality Estimation[J/OL]. Proc. VLDB Endow., 2021, 15(3): 414-426. http://www.vldb.org/pvldb/vol15/p414-lu.pdf. DOI:10.14778/3494124.3494127.

[69] NEGI P, WU Z, KIPF A, et al. Robust Query Driven Cardinality Estimation under Changing Workloads[J/OL]. Proc. VLDB Endow., 2023, 16(6): 1520-1533. https://www.vldb.org/pvldb/vol16/p1520-negi.pdf. DOI:10.14778/3583140.3583164.

[70] RAMACHANDRA K, PARK K, EMANI K V, et al. Froid: Optimization of Imperative Programs in a Relational Database[J/OL]. Proc. VLDB Endow., 2017, 11(4): 432-444. http://www.vldb.org/pvldb/vol11/p432-ramachandra.pdf. DOI:10.1145/3186728.3164140.

[71] ATSERIAS A, GROHE M, MARX D. Size Bounds and Query Plans for Relational Joins[J/OL]. SIAM J. Comput., 2013, 42(4): 1737-1767. https://doi.org/10.1137/110859440. DOI:10.1137/110859440.

[72] AOKI P M. Implementation of Extended Indexes in POSTGRES[J/OL]. SIGIR Forum, 1991, 25(1): 2-9. https://doi.org/10.1145/122642.122643. DOI:10.1145/122642.122643.

[73] BIZARRO P, BRUNO N, DEWITT D J. Progressive Parametric Query Optimization[J/OL]. IEEE Trans. Knowl. Data Eng., 2009, 21(4): 582-594. https://doi.org/10.1109/TKDE.2008.160. DOI:10.1109/TKDE.2008.160.

[74] GALINDO-LEGARIA C A, ROSENTHAL A. Outerjoin Simplification and Reordering for Query Optimization[J/OL]. ACM Trans. Database Syst., 1997, 22(1): 43-73. https://doi.org/10.1145/244810.244812. DOI:10.1145/244810.244812.

[75] GIBBONS P B, MATIAS Y, POOSALA V. Fast incremental maintenance of approximate histograms[J/OL]. ACM Trans. Database Syst., 2002, 27(3): 261-298. https://doi.org/10.1145/581751.581753. DOI:10.1145/581751.581753.

[76] GALINDO-LEGARIA C A, GRABS T, GUKAL S, et al. Optimizing Star Join Queries for Data Warehousing in Microsoft SQL Server[C/OL]//ALONSO G, BLAKELEY J A, CHEN A L P. Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7-12, 2008, Cancún, Mexico. IEEE Computer Society, 2008: 1190-1199. https://doi.org/10.1109/ICDE.2008.4497528. DOI:10.1109/ICDE.2008.4497528.

[77] KIM W. On Optimizing an SQL-like Nested Query[J/OL]. ACM Trans. Database Syst., 1982, 7(3): 443-469. https://doi.org/10.1145/319732.319745. DOI:10.1145/319732.319745.

[78] HALEVY A Y. Answering queries using views: A survey[J/OL]. VLDB J., 2001, 10(4): 270-294. https://doi.org/10.1007/s007780100054. DOI:10.1007/S007780100054.

[79] STEINBRUNN M, MOERKOTTE G, KEMPER A. Heuristic and Randomized Optimization for the Join Ordering Problem[J/OL]. VLDB J., 1997, 6(3): 191-208. https://doi.org/10.1007/s007780050040. DOI:10.1007/S007780050040.

[80] GALINDO-LEGARIA C A, ROSENTHAL A. How to Extend a Conventional Optimizer to Handle One- and Two-Sided Outerjoin[C/OL]//GOLSHANI F. Proceedings of the Eighth International Conference on Data Engineering, February 3-7, 1992, Tempe, Arizona, USA. IEEE Computer Society, 1992: 402-409. https://doi.org/10.1109/ICDE.1992.213169. DOI:10.1109/ICDE.1992.213169.

[81] DORAISWAMY H, DARERA P N, HARITSA J R. On the Production of Anorexic Plan Diagrams[C/OL]//KOCH C, GEHRKE J, GAROFALAKIS M N, et al. Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, September 23-27, 2007. ACM, 2007: 1081-1092. http://www.vldb.org/conf/2007/papers/research/p1081-d.pdf.

[82] PHILIPPE FLAJOLET, ÉRIC FUSY, OLIVIER GANDOUET, et al. Hyperloglog: The analysis of a near-optimal cardinality estimation algorithm[J]. 2007.

[83] NAMBIAR R O, POESS M. The Making of TPC-DS[C/OL]//DAYAL U, WHANG K Y, LOMET D B, et al. Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006. ACM, 2006: 1049-1058. http://dl.acm.org/citation.cfm?id=1164217.

[84] GUPTA A, MUMICK I S. Maintenance of Materialized Views: Problems, Techniques, and Applications[J/OL]. IEEE Data Eng. Bull., 1995, 18(2): 3-18. http://sites.computer.org/debull/95JUN-CD.pdf.

[85] PELLENKOFT A, GALINDO-LEGARIA C A, KERSTEN M L. The Complexity of Transformation-Based Join Enumeration[C/OL]//JARKE M, CAREY M J, DITTRICH K R, et al. VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases, August 25-29, 1997, Athens, Greece. Morgan Kaufmann, 1997: 306-315. http://www.vldb.org/conf/1997/P306.PDF.

[86] CHAUDHURI S, SHIM K. Including Group-By in Query Optimization[C/OL]//BOCCA J B, JARKE M, ZANIOLO C. VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile. Morgan Kaufmann, 1994: 354-366. http://www.vldb.org/conf/1994/P354.PDF.

[87] DU W, KRISHNAMURTHY R, SHAN M C. Query Optimization in a Heterogeneous DBMS[C/OL]//YUAN L Y. 18th International Conference on Very Large Data Bases, August 23-27, 1992, Vancouver, Canada, Proceedings. Morgan Kaufmann, 1992: 277-291. http://www.vldb.org/conf/1992/P277.PDF.

[88] DEWITT D J, NAUGHTON J F, SCHNEIDER D A. An Evaluation of Non-Equijoin Algorithms[C/OL]//LOHMAN G M, SERNADAS A, CAMPS R. 17th International Conference on Very Large Data Bases, September 3-6, 1991, Barcelona, Catalonia, Spain, Proceedings. Morgan Kaufmann, 1991: 443-452. http://www.vldb.org/conf/1991/P443.PDF.

[89] MELNIK S, GUBAREV A, LONG J J, et al. Dremel: A Decade of Interactive SQL Analysis at Web Scale[J/OL]. Proc. VLDB Endow., 2020, 13(12): 3461-3472. http://www.vldb.org/pvldb/vol13/p3461-melnik.pdf. DOI:10.14778/3415478.3415568.

[90] FREITAG M J, NEUMANN T. Every Row Counts: Combining Sketches and Sampling for Accurate Group-By Result Estimates[C/OL]//9th Biennial Conference on Innovative Data Systems Research, CIDR 2019, Asilomar, CA, USA, January 13-16, 2019, Online Proceedings. www.cidrdb.org, 2019. http://cidrdb.org/cidr2019/papers/p23-freitag-cidr19.pdf.

[91] HILPRECHT B, SCHMIDT A, KULESSA M, et al. DeepDB: Learn from Data, not from Queries![J/OL]. Proc. VLDB Endow., 2020, 13(7): 992-1005. http://www.vldb.org/pvldb/vol13/p992-hilprecht.pdf. DOI:10.14778/3384345.3384349.

[92] SUN J, LI G. An End-to-End Learning-based Cost Estimator[EB/OL]. (2019). https://arxiv.org/abs/1906.02560.

[93] AGUILAR-SABORIT J, RAMAKRISHNAN R. POLARIS: The Distributed SQL Engine in Azure Synapse[J/OL]. Proc. VLDB Endow., 2020, 13(12): 3204-3216. http://www.vldb.org/pvldb/vol13/p3204-saborit.pdf. DOI:10.14778/3415478.3415545.

[94] HEWITT E. Cassandra - The Definitive Guide: Distributed Data at Web Scale[M/OL]. Springer, 2011. http://www.oreilly.de/catalog/9781449390419/index.html.

[95] RAMAMURTHY R, DEWITT D J. Buffer-pool Aware Query Optimization[C/OL]//Second Biennial Conference on Innovative Data Systems Research, CIDR 2005, Asilomar, CA, USA, January 4-7, 2005, Online Proceedings. www.cidrdb.org, 2005: 250-261. http://cidrdb.org/cidr2005/papers/P21.pdf.

[96] MELTON J, SIMON A. SQL:1999: understanding relational language components[M]. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001.

[97] SESHADRI P, PIRAHESH H, LEUNG T. Complex query decorrelation[C/OL]//Proceedings of the Twelfth International Conference on Data Engineering: Vol. 0. 1996: 450-458. DOI:10.1109/ICDE.1996.492194.

[98] OLKEN F. Random Sampling from Databases[D]. 1993.

[99] KOOI R. The Optimization of Queries in Relational Databases[D]. 1980.

[100] KRISHNAN P, VITTER J S, IYER B R. Estimating Alphanumeric Selectivity in the Presence of Wildcards[C/OL]//JAGADISH H V, MUMICK I S. Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996. ACM Press, 1996: 282-293. https://doi.org/10.1145/233269.233341. DOI:10.1145/233269.233341.

[101] XUE M, BU Y, SOMANI A, et al. Adaptive and Robust Query Execution for Lakehouses at Scale[J/OL]. Proc. VLDB Endow., 2024, 17(12): 3947-3959. https://doi.org/10.14778/3685800.3685818. DOI:10.14778/3685800.3685818.