MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 33, NO. 2, 2026
journal homepage: www.mais-journal.ru

ARTIFICIAL INTELLIGENCE

A Method for Ranking SQL Query Execution Plans Based on Textual Descriptions and Vector Representation Models

N. K. Vasilenko[1]

DOI: 10.18255/1818-1015-2026-2-266-280

[1]Ershov Institute of Informatics Systems SB RAS, Novosibirsk, Russia

MSC2020: 68T05
Research article
Full text in Russian

Received April 29, 2026
Revised May 14, 2026
Accepted May 19, 2026

This paper studies the problem of ranking SQL query execution plans by execution time. We propose a method in which structural encoding of the plan tree is replaced with a textual description of the plan, which is then converted into a vector representation using a vector representation model. A compact prediction model is trained on top of this representation and used to rank plans. Three approaches are compared: the prediction model from Bao, which relies on structural encoding of the plan tree, the cost estimate produced by the PostgreSQL optimizer, and the proposed method based on textual plan descriptions and vector representation models. In addition, several vector representation models and two variants of textual plan description are investigated: the raw plan text and a normalized description. Experiments are conducted on the CEB benchmark built on IMDb data under two evaluation settings: random splits and query-template splits. Quality is evaluated using pairwise accuracy and Spearman correlation. The results show that under random splits, the best configuration based on jina-code-embeddings-0.5b outperforms both the Bao model and the optimizer cost estimate on both ranking metrics. For all considered models, the raw textual plan description is more informative than the normalization scheme used in this work. Under query-template splits, the quality of all learned approaches decreases; in this setting, the best model based on vector representations and the Bao model remain comparable to each other, but both are outperformed by the optimizer cost estimate. These results indicate that textual plan descriptions and vector representation models can serve as a basis for predictive plan-ranking methods, although robust generalization to previously unseen query templates remains an open problem.

Keywords: SQL query optimization; execution plans; plan ranking; execution time prediction; embeddings

INFORMATION ABOUT THE AUTHORS

Vasilenko, Nikita K. (corresponding author) | ORCID iD: 0009-0003-8727-3000. E-mail: vasilenko.nikita.research@gmail.com
Postgraduate Student

For citation: N. K. Vasilenko, "A method for ranking SQL query execution plans based on textual descriptions and vector representation models", *Modeling and Analysis of Information Systems*, vol. 33, no. 2, pp. 266–280, 2026.
DOI: 10.18255/1818-1015-2026-2-266-280.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 33, № 2, 2026
сайт журнала: www.mais-journal.ru

Метод ранжирования планов выполнения SQL-запросов на основе текстового описания и моделей векторных представлений

Н. К. Василенко[1]

DOI: 10.18255/1818-1015-2026-2-266-280

[1]Институт систем информатики им. А.П. Ершова СО РАН, Новосибирск, Россия

УДК 004.65+004.8
Научная статья
Полный текст на русском языке

Получена 29 апреля 2026 г.
После доработки 14 мая 2026 г.
Принята к публикации 19 мая 2026 г.

В статье рассматривается задача ранжирования планов выполнения SQL-запросов по времени выполнения. Такая задача представляет интерес для интеллектуальной оптимизации запросов, поскольку позволяет оценить, насколько выбранное описание плана сохраняет информацию, связанную с фактической стоимостью его исполнения. Предлагается метод, в котором вместо структурной кодировки дерева плана используется его текстовое описание, для которого затем строится векторное представление с помощью модели векторных представлений. Поверх полученного векторного представления обучается компактная предсказательная модель, используемая для ранжирования планов. В работе сравниваются три подхода: предсказательная модель из подхода Bao, использующая структурную кодировку дерева плана, стоимостная оценка оптимизатора PostgreSQL и предложенный метод на основе текстового описания плана и моделей векторных представлений. Дополнительно исследуются несколько моделей векторных представлений и два варианта текстового описания плана — исходный текст и нормализованное описание. Эксперименты проведены на наборе CEB, построенном на данных IMDb, в двух режимах тестирования: при случайном разбиении и при разбиении по шаблонам запросов. Качество оценивалось по попарной точности и коэффициенту Спирмена. Показано, что при случайном разбиении выборки лучшая конфигурация на основе модели jina-code-embeddings-0.5b превосходит модель Bao и стоимостную оценку оптимизатора по обеим метрикам ранжирования. Установлено также, что исходное текстовое описание плана для всех рассмотренных моделей оказывается более информативным, чем использованная схема нормализации. При разбиении выборки по шаблонам запросов качество всех обучаемых подходов снижается; при этом лучшая модель на основе векторных представлений и модель Bao остаются на сопоставимом уровне между собой, однако по обеим метрикам уступают стоимостной оценке оптимизатора. Полученные результаты показывают, что текстовое описание плана и модели векторных представлений могут использоваться как основа для построения предсказательных моделей ранжирования, хотя проблема устойчивой генерализации на ранее не наблюдавшиеся шаблоны запросов остаётся открытой.

Ключевые слова: оптимизация SQL-запросов; планы выполнения; ранжирование планов; предсказание времени выполнения; векторные представления

ИНФОРМАЦИЯ ОБ АВТОРАХ

Василенко, Никита Константинович (автор для корреспонденции) | ORCID iD: 0009-0003-8727-3000. E-mail: vasilenko.nikita.research@gmail.com
Аспирант

Для цитирования: N. K. Vasilenko, “A method for ranking SQL query execution plans based on textual descriptions and vector representation models”, *Modeling and Analysis of Information Systems*, vol. 33, no. 2, pp. 266–280, 2026.
DOI: 10.18255/1818-1015-2026-2-266-280.

Введение

Качество работы реляционной СУБД во многом определяется тем, насколько удачно планировщик выбирает план выполнения запроса из множества допустимых альтернатив. Ошибки в оценке времени выполнения могут приводить к заметным потерям производительности, особенно на сложных аналитических нагрузках [1—3]. Значительная часть работ сосредоточена на уточнении оценок кардинальностей и стоимостных моделей [4, 5].

К ранним представителям этого направления относится работа M. Akdere и соавторов, где предсказание времени рассматривается как задача обучения по признакам планов выполнения [6]. Одним из наиболее известных представителей этого подхода является метод Bao [7], в котором предсказательная модель оценивает качество планов и используется внутри механизма выбора управляющих воздействий на оптимизатор. В дальнейшем под моделью Bao понимается именно предсказательная часть этого подхода, использующая структурную кодировку дерева плана и свёрточную сеть по дереву. Этот подход получил развитие в ряде последующих работ [8—12], однако во всех них требуется специально сконструированное описание запроса или плана. Так, в AutoSteer сохраняется общая схема Bao и используется модель оценки планов, опирающаяся на дерево плана. В Lero задача формулируется как попарное сравнение планов, а каждый план кодируется как дерево векторов признаков узлов с последующей свёрточной обработкой по дереву. В QueryFormer предлагается ещё более сложная схема на основе tree-transformer архитектуры с дополнительными механизмами кодирования структуры дерева и статистик. Даже в FASTgres, где дерево плана не выступает основным объектом кодирования, вводится специально разработанное признаковое описание запроса и его контекста. Таким образом, хотя конкретные способы кодирования различаются, для всех этих подходов характерна необходимость разработки входной формы данных или специализированной архитектуры для работы со структурой запроса или плана.

С появлением больших языковых моделей исследователи стали рассматривать их и в контексте оптимизации запросов. Появились работы по переписыванию запросов и по использованию языковых моделей в качестве внешнего интеллектуального компонента оптимизатора [13—16]. Хотя такие модели показывают многообещающие результаты, основной проблемой остается их высокая вычислительная стоимость и задержка вывода, что для такой области, как планирование запросов критически важно.

В отличие от генеративного применения языковых моделей, использование их как кодировщиков не требует пошаговой генерации текста и позволяет получать фиксированное представление плана за один проход модели. Параллельно с этим в области обработки текста и программного кода сформировался большой класс предобученных моделей векторных представлений [17]. В связи с этим представляет интерес вопрос о том, можно ли отказаться от ручного построения структурной кодировки дерева плана и заменить его текстовым представлением плана, преобразованным в вектор с помощью предобученной языковой модели. Если качество такого подхода окажется сопоставимым с ручной кодировкой, это позволит существенно упростить построение предсказательных моделей для оптимизации запросов.

Настоящая статья посвящена именно этой задаче. Рассматривается ранжирование планов по времени выполнения как самостоятельная предсказательная задача. Сравниваются модель Bao, использующая структурную кодировку дерева плана, модели на основе векторных представлений планов и стоимостная оценка стандартного планировщика запросов в PostgreSQL. В отличие от работ, где большая языковая модель используется для генерации переписываний или подсказок, в данной работе она используется только как источник векторного представления плана. Такой подход может быть полезен как более простой по представлению предсказательный сигнал для последующего сравнения планов, поскольку не требует последовательной генерации языковой модели.

Основной вклад работы состоит в следующем:

- предложен подход к ранжированию планов выполнения SQL-запросов по времени исполнения, в котором вместо ручной структурной кодировки дерева плана используется текстовое представление плана и его векторное представление, полученное с помощью предобученной языковой модели;
- проведено сравнение нескольких моделей векторных представлений и двух способов текстового представления плана, включая исходный текст и нормализованное текстовое описание;
- выполнено экспериментальное сравнение предложенного подхода с моделью Bao, использующей ручную структурную кодировку дерева плана, а также со стоимостной оценкой стандартного планировщика PostgreSQL;
- экспериментально показано, что на исследуемой нагрузке CEB при случайном разбиении лучшая конфигурация на основе моделей векторных представлений превосходит модель Bao по обеим метрикам ранжирования.

1. Постановка задачи и метрики оценки

Пусть задан набор планов выполнения $\mathcal{P} = \{P_i\}_{i=1}^{N}$, полученных для набора запросов, и соответствующие измеренные времена выполнения T_i. Требуется построить функцию оценки

$$F(P) \in \mathbb{R}, \tag{1}$$

которая задаёт порядок планов, согласованный с фактическим временем выполнения. Для большинства пар планов должно выполняться соотношение

$$T_i > T_j \Rightarrow F(P_i) > F(P_j). \tag{2}$$

В данной работе рассматривается задача межобъектного ранжирования планов из набора запросов, а не непосредственный выбор лучшего плана среди альтернатив одного и того же запроса. Такая постановка позволяет оценить, насколько выбранное представление плана содержит сигнал, связанный с фактическим временем выполнения, и может использоваться как прокси-задача при исследовании методов офлайн-анализа и ранжирования кандидатных планов.

В работе рассматриваются три модели $F(P)$, используемые для ранжирования планов выполнения P.

1. **Стоимостная оценка оптимизатора.** В качестве базовой модели используется стоимость плана, вычисляемая стандартным планировщиком PostgreSQL во время этапа планирования. Эта величина не является прямой оценкой времени выполнения и выражается в условных единицах стоимостной модели. Следует отметить, что в самой СУБД она предназначена прежде всего для сравнения альтернативных планов одного и того же запроса, тогда как в данной работе рассматривается её использование для глобального ранжирования планов, относящихся к разным запросам. Тем не менее именно по ней оптимизатор сравнивает альтернативные планы и выбирает наиболее предпочтительный с собственной точки зрения.
2. **Модель Bao со структурной кодировкой дерева плана.** Во втором подходе используется модель Bao, описанная в разделе 2.
3. **Модель на основе векторного представления текстового плана.** В третьем подходе используется предложенная в работе модель, описанная в разделе 3.

Поскольку в данной работе модель рассматривается прежде всего как средство ранжирования планов по ожидаемому времени выполнения, качество оценивается не по абсолютной ошибке прогноза, а по тому, насколько хорошо сохраняется относительный порядок планов. Для этого используются две метрики.

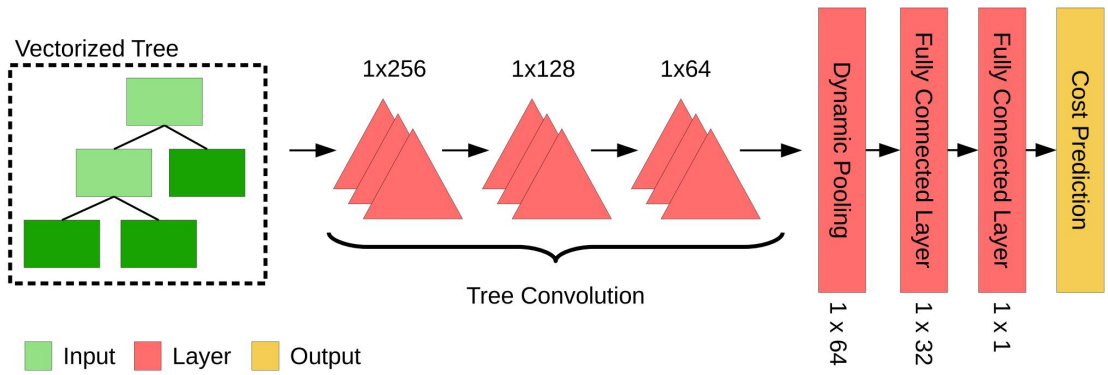

Fig. 1. Bao prediction model architecture [7]

Рис. 1. Архитектура предсказательной модели Bao [7]

Первая метрика, **попарная точность** (*pairwise accuracy*),

$$\mathrm{PA} = \frac{1}{|\mathcal{K}|} \sum_{(i,j)\in\mathcal{K}} \mathbb{I}\Big[(F(P_i) - F(P_j))(T_i - T_j) > 0\Big], \tag{3}$$

где $\mathcal{K}$ — множество сравниваемых пар планов, а $\mathbb{I}[\cdot]$ — индикаторная функция, показывает долю пар, для которых модель правильно определяет относительный порядок по времени выполнения. Такая метрика показывает, насколько хорошо модель сохраняет относительный порядок планов по времени выполнения, и может использоваться как косвенный показатель пригодности представления плана для последующих задач сравнения альтернативных планов.

Вторая метрика, **коэффициент ранговой корреляции Спирмена**, определяется как корреляция между истинными и предсказанными рангами:

$$\rho = \frac{\sum_{i=1}^{n} (r_i - \bar{r})(\hat{r}_i - \bar{\hat{r}})}{\sqrt{\sum_{i=1}^{n} (r_i - \bar{r})^2} \sqrt{\sum_{i=1}^{n} (\hat{r}_i - \bar{\hat{r}})^2}}, \tag{4}$$

где r_i — ранг i-го плана по измеренному времени выполнения, $\hat{r}_i$ — его ранг по предсказанию модели, а $\bar{r}$ и $\bar{\hat{r}}$ — средние значения соответствующих рангов. В отличие от попарной точности, коэффициент Спирмена оценивает согласованность глобального порядка на всём наборе планов. Он показывает, насколько близко предсказанное ранжирование к истинному, и при этом не зависит от монотонных преобразований шкалы предсказаний. В отличие от попарной точности, коэффициент Спирмена более чувствителен к систематическим ошибкам глобального ранжирования, в частности к несогласованности оценок между различными группами запросов.

2. Предсказательная часть Bao как метод сравнения

В качестве метода сравнения в работе используется предсказательная модель из подхода Bao. Сам подход представляет собой систему оптимизации запросов на основе многорукого бандита, в которой выбор управляющих воздействий на оптимизатор опирается на отдельную модель оценки планов. В данной статье рассматривается именно эта предсказательная часть Bao, используемая как модель ранжирования планов по времени выполнения.

Модель включает три основных компонента. Архитектура представлена на рисунке 1.

Первый компонент — **структурная кодировка дерева плана**. Исходное дерево плана приводится к бинарной форме: узлы с числом потомков больше двух раскладываются в бинарную

структуру, а отсутствующие потомки заменяются фиктивными узлами. После этого каждый узел кодируется фиксированным вектором признаков, включающим тип физического оператора, оценку кардинальности и оценку стоимости. В результате план представляется как дерево векторов фиксированной длины.

Второй компонент — **свёрточные слои по дереву**. Они применяются к полученному дереву признаков и позволяют извлекать локальные структурные шаблоны в поддеревьях плана, учитывая взаимное расположение родительских и дочерних узлов. Последовательное применение нескольких таких слоёв формирует внутреннее представление плана, чувствительное к его структуре.

Третий компонент — **полносвязная часть сети**. После свёрточных слоёв представление всего дерева агрегируется в вектор фиксированной длины, который затем подаётся на несколько полносвязных слоёв. Выходом модели является скалярная оценка, используемая для ранжирования планов по ожидаемому времени выполнения.

3. Подход на основе векторных представлений планов выполнения

В предлагаемом подходе ручная структурная кодировка дерева плана заменяется текстовым представлением плана запроса, полученным от СУБД в человекочитаемой форме. Для такого текста строится векторное представление с помощью предобученной модели векторных представлений, после чего поверх полученного вектора обучается полносвязная предсказательная модель, аналогичная выходной части модели Bao. Далее эти три этапа рассматриваются отдельно: построение текстового представления плана, получение его векторного представления и обучение предсказательной модели.

3.1. Текстовые представления плана

В экспериментах используются два текстовых представления планов выполнения — исходное и нормализованное. Их сравнение позволяет оценить, насколько для задачи ранжирования по времени выполнения важны, с одной стороны, полнота исходного описания плана, а с другой стороны, устойчивость текстового представления к вариативности форматирования и лексики. В обоих случаях для построения входного текста использовались только те поля плана, которые доступны до выполнения запроса.

Исходное представление строится непосредственно по тексту *EXPLAIN*, возвращаемому СУБД в человекочитаемой форме, без дополнительных преобразований. Такой вариант максимально близок к реальному тексту плана, с которым работает пользователь или администратор базы данных. Его достоинство состоит в том, что он сохраняет всю доступную текстовую информацию, включая названия операторов, условия и числовые оценки. Однако вместе с этим он содержит значительную вариативность форматирования, неоднородность лексики и множество деталей, не всегда существенных для задачи ранжирования планов по времени выполнения. Сокращенный пример текста представлен на рисунке 2.

Нормализованное представление предназначено для получения более устойчивой формы плана. Преобразование выполняется как прямой обход дерева плана с записью одного узла на строку. Каждая строка начинается с индикатора глубины L_d, за которым следуют тип физического оператора и фиксированный набор атрибутов узла.

Для числовых атрибутов *Total Cost* и *Plan Rows* используется логарифмическое бакетирование по основанию 10 с шагом 0.1. В результате числовые значения представляются токенами вида 10^k, что уменьшает чувствительность модели к малым вариациям абсолютных величин. Дополнительно вычисляется относительный признак *ct_frac*, равный отношению *Total Cost* текущего узла к *Total Cost* корневого узла. Этот признак также записывается в виде логарифмического бакета и позволяет отразить масштаб узла относительно всего плана.

```
Aggregate  (cost=12345.67..12345.68 rows=1 width=8)
  ->  Nested Loop  (cost=12.34..12340.12 rows=1500 width=16)
        Join Filter: (t.id = mi.movie_id)
        ->  Index Scan using title_pkey on~title t (cost=0.42..8.45 rows=1
        width=8)
              Index Cond: (id = 1234)
              Filter: (production_year > 2000)
        ->  Bitmap Heap Scan on~movie_info mi (cost=11.90..12200.00 rows=
        20000 width=8)
              Recheck Cond: (info_type_id = 3)
              Filter: (info = 'USA')
              ->  Bitmap Index Scan on~info_type_id_movie_info(cost=0.00..
              6.50 rows=20000 width=0)
                    Index Cond: (info_type_id = 3)
```

Fig. 2. Example of the plain plan representation

Рис. 2. Пример исходного представления плана

```
L0 Aggregate cost=10^4.1..10^4.1 rows=10^0.0 width=8 ct_frac=10^0.0
type=Aggregate
L1 Nested Loop cost=10^1.1..10^4.1 rows=10^3.2 width=16 ct_frac=10^0.0
type=Nested Loop join=Inner cond_joinfilter: (t.id = mi.movie_id)
L2 Index Scan cost=10^-0.4..10^0.9 rows=10^0.0 width=8 ct_frac=10^-3.2
type=Index Scan rel=title idx=title_pkey cond_idx: (id = N:r25=1225)
filter: (production_year > N:r25=2000)
L2 Bitmap Heap Scan cost=10^1.1..10^4.1 rows=10^4.3 width=8 ct_frac=10^0.0
type=Bitmap Heap Scan rel=movie_info recheck: (info_type_id = N:small)
filter: (info = S)
L3 Bitmap Index Scan cost=0..10^0.8 rows=10^4.3 width=0 ct_frac=10^-3.3
type=Bitmap Index Scan idx=info_type_id_movie_info
cond_idx: (info_type_id = N:small)
```

Fig. 3. Example of the normalized plan representation

Рис. 3. Пример нормализованного представления плана

Предикатные выражения, в частности *Filter*, *Hash Cond* и *Index Cond*, нормализуются отдельно. Из них удаляются явные приведения типов, строковые литералы заменяются на специальный маркер *S*, а числовые литералы переводятся в бакеты, отдельно для малых и больших значений. Конструкции вида *IN (. . .)* и массивы приводятся к компактной форме *LIST:len=k*. Такая нормализация позволяет уменьшить размер словаря токенов, сохраняя при этом информацию о структуре условий и грубом масштабе числовых значений.

Признаки параллелизма и служебные атрибуты узлов намеренно не включаются в текстовое описание. Это сделано для уменьшения шума и повышения устойчивости представления. Пример нормализованного плана представлен на рисунке 3. В результате нормализованный вариант можно рассматривать как каноническую сериализацию дерева плана, ориентированную на стабильность токенов.

3.2. Модели векторных представлений

Выбор моделей векторных представлений был организован так, чтобы охватить несколько различных классов современных моделей, а не ограничиваться близкими вариантами одной архитектуры. В итоговый набор были включены три модели:

1. Qwen3-Embedding-0.6B[1]. Эта модель рассматривается как современный универсальный эмбеддер общего назначения. Для неё заявлены контекст длиной до 32 000 токенов и настраиваемая размерность выходного вектора до 1024. Она включена в сравнение как сильная модель общего назначения.
2. jina-code-embeddings-0.5b[2]. Эта модель выбрана как представитель специализированных моделей для кода и технических текстов. Она поддерживает задачи сопоставления кода и текста, работает с 15+ языками программирования, использует вектор размерности 896 и допускает длину входа до 32 768 токенов. Она включена в набор как пример специализированной модели для формализованных входных данных.
3. BAAI/bge-m3[3]. Эта модель выбрана как представитель универсальных моделей, ориентированных на сильное семантическое сопоставление. Её ключевые особенности — многофункциональность, многоязычность и поддержка входов разной длины, от коротких текстов до документов объёмом до 8192 токенов. Она включена в набор как пример мощной многоязычной модели векторных представлений широкого профиля.

Таким образом, выбранный набор покрывает три разных класса предобученных моделей векторных представлений: универсальные модели общего назначения, специализированные модели для кода и технических текстов и модели, ориентированные на семантическое сопоставление.

3.3. Модель ранжирования поверх векторных представлений

После построения векторного представления $x \in \mathbb{R}^d$ выполняются L2-нормализация и стандартизация по статистикам обучающей выборки. Поверх этого представления используется компактный многослойный перцептрон, функционально аналогичный полносвязной выходной части BAO-подобной модели. Он состоит из трёх скрытых полносвязных слоёв с нелинейностью GELU и прореживанием и отображает входной вектор в скалярную оценку, используемую для ранжирования планов по ожидаемому времени выполнения.

4. Экспериментальная методика

Эксперименты выполнены на наборе CEB [18], построенном на данных IMDb. Этот набор представляет собой расширение классического бенчмарка JOB и содержит множество экземпляров запросов, сгруппированных по шаблонам. Под шаблоном в наборе CEB понимается параметризованная форма SQL-запроса, задающая его общую структуру. Шаблон фиксирует набор участвующих таблиц, условия соединения между ними и расположение предикатов.

Разные шаблоны отличаются друг от друга именно этой структурой: составом отношений, графом соединений, числом и типом фильтров. Поэтому разные шаблоны задают разные классы запросов и, как правило, приводят к различным пространствам допустимых планов выполнения.

Запросы внутри одного шаблона сохраняют общую структурную форму, но различаются конкретными значениями параметров, прежде всего константами в предикатах. Эти различия изменяют селективность условий, промежуточные кардинальности и фактическое время выполнения, так что даже в пределах одного шаблона могут наблюдаться разные оптимальные планы и заметный разброс по времени выполнения. На рисунке 4 представлены распределения времени выполнения по группам запросов, где каждая группа соответствует одному шаблону CEB. Всего набор содержит 3133 запроса, разделенных на 16 шаблонов. Для удобства из анализа были исключены запросы с временем выполнения выше 200000 мс, в результате чего в итоговом наборе осталось 3129 запросов. Стоимостные оценки планов, вычисляемые стандартным оптимизатором, а также измеренные времена выполнения запросов были получены в PostgreSQL 18.

[1] https://huggingface.co/Qwen/Qwen3-Embedding-0.6B
[2] https://huggingface.co/jinaai/jina-code-embeddings-0.5b
[3] https://huggingface.co/BAAI/bge-m3

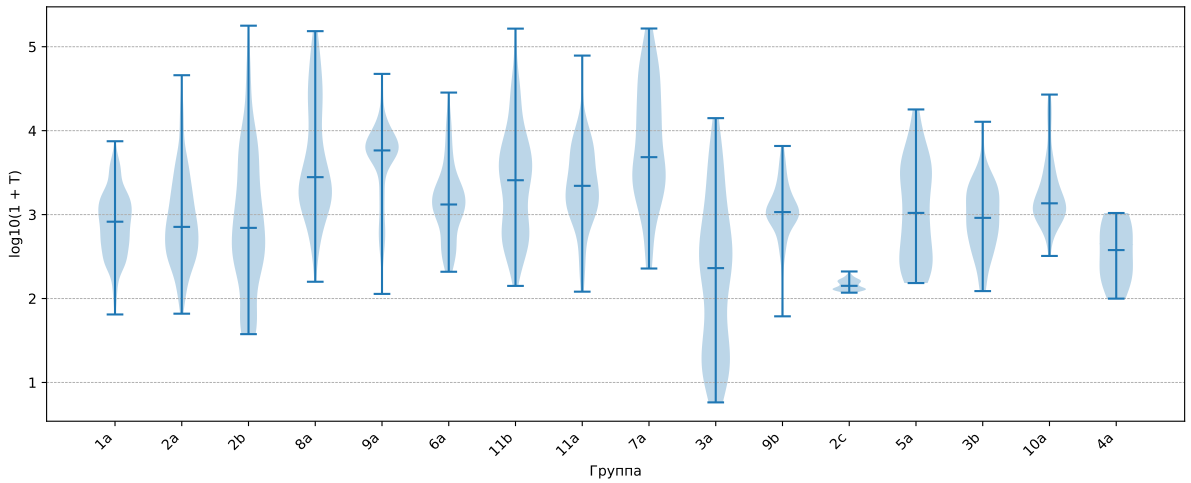

Fig. 4. Execution time distribution by group for CEB

Рис. 4. Распределение времени выполнения по группам для CEB

Обучение предсказательных моделей проводилось по единой схеме. Для каждого плана строилось входное представление соответствующего типа, после чего модель обучалась предсказывать логарифмированное время выполнения

$$y = \log(1 + T), \tag{5}$$

где T обозначает измеренное время выполнения в миллисекундах. Использование логарифмической шкалы обусловлено тем, что для рассматриваемого набора характерно распределение времени выполнения с длинным правым хвостом, то есть наличие относительно небольшого числа очень медленных планов. Такое преобразование уменьшает влияние выбросов и делает обучение более устойчивым. Хотя обучение формулируется как регрессия, в дальнейшем выход модели используется только как ранжирующая оценка.

Для моделей на векторных представлениях перед обучением выполнялись L2-нормализация векторов и стандартизация по статистикам обучающей выборки. В качестве предсказательной части использовался компактный многослойный перцептрон, а обучение формулировалось как задача регрессии по среднеквадратичной ошибке с использованием оптимизатора AdamW и ранней остановки по качеству на валидационной части обучающей выборки. Тем самым все сравниваемые модели помещались в максимально близкую экспериментальную среду и различались прежде всего способом представления плана. После обучения полученные модели использовались как ранжировщики планов.

Для оценки качества использовались два режима тестирования:

1. **Случайное разбиение.** Использовалась пятикратная кросс-валидация: на каждом разбиении 20% запросов выделялись в тестовую выборку, а остальные использовались для обучения и валидации.
2. **Разбиение по шаблонам.** В этом режиме в тестовую выборку целиком выделялась одна группа запросов, соответствующая одному шаблону, а обучение проводилось на остальных группах. Распределение числа запросов по шаблонам существенно неоднородно: часть групп содержит лишь несколько десятков наблюдений. Поэтому для этого тестирования использовались только те группы, в которых число запросов превышает 100. Таких групп оказалось 12.

Использование двух режимов тестирования позволяет разделить два принципиально разных сценария применения. Случайное разбиение отражает работу на известной стационарной нагрузке,

Table 1. Model performance under random split

Таблица 1. Результаты моделей при случайном разбиении

Модель	**Входное описание плана**	**Попарная точность,** $\mu \pm \sigma$	**Спирмен** ρ**,** $\mu \pm \sigma$
Bao	структурная кодировка	0.742 ± 0.012	0.663 ± 0.032
Qwen3-Embedding-0.6B	нормализованный текст	0.718 ± 0.033	0.602 ± 0.076
Qwen3-Embedding-0.6B	исходный текст	0.750 ± 0.018	0.683 ± 0.035
bge-m3	нормализованный текст	0.688 ± 0.016	0.537 ± 0.043
bge-m3	исходный текст	0.725 ± 0.020	0.624 ± 0.049
jina-code-embeddings-0.5b	нормализованный текст	0.733 ± 0.016	0.637 ± 0.036
jina-code-embeddings-0.5b	исходный текст	**0.765 ± 0.014**	**0.715 ± 0.030**
Оценка оптимизатора	стоимость плана	0.537 ± 0.009	0.085 ± 0.031

где в обучающей и тестовой выборках присутствуют запросы одних и тех же шаблонов. Разбиение по шаблонам моделирует более строгую ситуацию переноса на ранее не наблюдавшийся тип запроса и тем самым позволяет оценить способность моделей к генерализации. Такая проверка важна, поскольку недостаточная способность к адаптации на новых шаблонах является одним из основных ограничений большинства обучаемых методов в задаче оптимизации запросов.

5. Результаты

5.1. Случайное разбиение

Результаты экспериментов при случайном разбиении приведены в таблице 1. Для каждой модели в таблице указаны среднее значение метрики μ и стандартное отклонение σ по всем разбиениям.

При тестировании со случайным разбиением все обучаемые модели превосходят стоимостную оценку оптимизатора по обеим метрикам, как по попарной точности, так и по коэффициенту Спирмена. Это наблюдение сохраняется для всех рассмотренных моделей и обоих вариантов текстового представления плана.

Наилучшие результаты по обеим метрикам показывает модель jina-code-embeddings-0.5b, использующая исходный текст плана. Для неё попарная точность составляет 0.765 ± 0.014, а коэффициент Спирмена — 0.715 ± 0.030. Второе место по обеим метрикам занимает Qwen3-Embedding-0.6B на исходном тексте, а модель Bao со структурной кодировкой дерева плана располагается на следующей позиции.

Для всех трёх моделей векторных представлений использование исходного текста плана даёт более высокие результаты, чем использование нормализованного представления. Это наблюдается как по попарной точности, так и по коэффициенту Спирмена. Следовательно, исходное текстовое представление оказывается более информативным для задачи ранжирования, чем его нормализованный вариант.

Для лучшей модели на основе векторных представлений выполнялась парная бутстреп-оценка разности метрик относительно модели Bao. Среднее преимущество составило $\Delta = 0.0228$ по попарной точности с 95 % доверительным интервалом $[0.0131; 0.0322]$ и $\Delta = 0.0518$ по коэффициенту Спирмена с 95 % доверительным интервалом $[0.0292; 0.0742]$.

5.2. Разбиение по группам

Результаты экспериментов при разбиении по группам приведены в таблице 2. В тестовую выборку каждый раз целиком выделялась одна группа запросов, соответствующая одному шаблону, а обучение проводилось на остальных группах.

При разбиении по группам значения обеих метрик для всех обучаемых моделей заметно ниже, чем при случайном разбиении. Это наблюдение относится как к моделям на основе векторных

Table 2. Model performance under group split

Таблица 2. Результаты моделей при разбиении по группам

Модель	**Входное описание плана**	**Попарная точность,** $\mu \pm \sigma$	**Спирмен** ρ**,** $\mu \pm \sigma$
Bao	структурная кодировка	0.595 ± 0.068	0.271 ± 0.197
Qwen3-Embedding-0.6B	нормализованный текст	0.545 ± 0.085	0.130 ± 0.248
Qwen3-Embedding-0.6B	исходный текст	0.583 ± 0.055	0.241 ± 0.163
bge-m3	нормализованный текст	0.547 ± 0.065	0.138 ± 0.188
bge-m3	исходный текст	0.557 ± 0.046	0.168 ± 0.135
jina-code-embeddings-0.5b	нормализованный текст	0.569 ± 0.064	0.203 ± 0.189
jina-code-embeddings-0.5b	исходный текст	0.597 ± 0.087	0.283 ± 0.248
Оценка оптимизатора	стоимость плана	**0.614 ± 0.079**	**0.316 ± 0.222**

представлений, так и к модели Bao со структурной кодировкой дерева плана. Одновременно стандартные отклонения оказываются существенно выше, чем при случайном разбиении, что указывает на сильную неоднородность качества между различными шаблонами запросов. Наилучшие средние результаты, как и в режиме случайного разбиения, показывает jina-code-embeddings-0.5b на исходном тексте плана. Таким образом, относительное упорядочивание обучаемых моделей в целом сохраняется: jina-code-embeddings-0.5b формально имеет более высокие средние значения, однако с учётом больших стандартных отклонений корректнее говорить о сопоставимом качестве с моделью Bao.

Как и в предыдущем эксперименте, для всех трёх моделей векторных представлений использование исходного текста плана даёт более высокие значения обеих метрик, чем использование нормализованного представления.

На ранее не наблюдавшихся шаблонах стоимостная оценка оптимизатора показывает наилучшие результаты по обеим метрикам. Для неё попарная точность составляет 0.614 ± 0.079, а коэффициент Спирмена — 0.316 ± 0.222. Наконец, можно отметить, что по сравнению с режимом случайного разбиения стоимостная оценка оптимизатора улучшает свои показатели по обеим метрикам, тогда как для обучаемых моделей наблюдается снижение качества.

5.3. Сравнение вычислительных затрат

Показатели, приведённые в таблице 3, получены как средние времена обработки одного плана по всему множеству CEB. Для всех обучаемых подходов измерялось время работы на этапе инференса, без учёта обучения моделей. В колонке «Время построения описания, мс» для Bao указано время структурной кодировки дерева плана, а для моделей векторных представлений — время построения векторного представления по тексту плана. Для стоимостной оценки оптимизатора дополнительное построение представления не требуется. Колонка «Суммарное время получения оценки, мс» включает полное среднее время получения оценки плана, то есть время построения самого плана в PostgreSQL и последующее время вычисления оценки соответствующим методом. Таким образом, для моделей векторных представлений суммарное время включает планирование запроса, построение векторного представления и проход предсказательной модели, а для Bao — планирование запроса, структурную кодировку дерева и предсказание модели. Среднее время получения самого плана на всём множестве CEB составило около 24 мс. Измерения для моделей векторных представлений и для Bao выполнялись на одной вычислительной системе; инференс выполнялся на GPU NVIDIA GeForce RTX 5070 Ti.

Как видно из таблицы 3, время обработки нормализованного текста для всех моделей векторных представлений оказывается на 20–30 % ниже, чем время обработки исходного текста плана.

Table 3. Comparison of average plan scoring time across different approaches

Таблица 3. Сравнение среднего времени получения оценки плана различными подходами

Модель	Входное описание плана	Время построения описания, мс	Суммарное время получения оценки, мс
Bao	структурная кодировка	0.3	24.4
Qwen3-Embedding-0.6B	нормализованный текст	98.5	122.6
Qwen3-Embedding-0.6B	исходный текст	135.8	159.9
bge-m3	нормализованный текст	48.1	72.2
bge-m3	исходный текст	71.6	95.7
jina-code-embeddings-0.5b	нормализованный текст	64.2	88.3
jina-code-embeddings-0.5b	исходный текст	81.2	105.3
Оценка оптимизатора	стоимость плана	0.0	24.1

Это связано с тем, что исходное текстовое описание в среднем имеет больший объём, чем нормализованное.

Среди моделей векторных представлений, которые превосходят Bao по метрикам качества на случайном разбиении, наиболее быстрой для исходного текста является jina-code-embeddings-0.5b, для которой суммарное время составляет 105.3 мс. Модель Qwen3-Embedding-0.6B, также превосходящая Bao по метрикам качества, напротив, оказывается самой медленной среди рассмотренных моделей и требует 159.9 мс на один план при использовании исходного текста.

Среднее время на построение самого плана во всех экспериментах составляет около 24 мс на всём датасете. По этой причине для стоимостной оценки оптимизатора суммарное время практически совпадает со временем планирования, тогда как для остальных подходов к нему добавляется стоимость построения описания плана и вычисления оценки.

Все модели векторных представлений требуют значительно большего времени на построение описания плана по сравнению с Bao и со стоимостной оценкой оптимизатора. По суммарному времени модель Bao превосходит подходы на основе моделей векторных представлений по скорости примерно в 3–6 раз в зависимости от выбранной модели и текстового описания плана.

6. Обсуждение

В обоих экспериментах исходный текст плана показывает более высокие результаты, чем нормализованное представление. Использованное нормализованное представление рассматривалось как пример возможной сериализации плана, а не как результат отдельной ручной оптимизации текстового формата. Имена таблиц и полей сохранялись исходя из предположения, что модель векторных представлений может извлекать из них полезный сигнал. Наблюдаемое превосходство исходного текста можно объяснить тем, что выбранная схема нормализации оказалась неудачной именно для данной задачи, а также тем, что исходный формат плана может быть ближе к тем типам технических текстов, на которых обучались используемые модели. Альтернативные схемы нормализации в работе специально не исследовались, поскольку это уже означало бы переход к ручной инженерии входного описания. При этом не исключается, что другие варианты текстового кодирования могли бы дать лучший результат.

Результаты разбиения по группам показывают, что генерализация на ранее не наблюдавшиеся шаблоны остаётся основным ограничением всех обучаемых подходов. Существенное снижение качества наблюдается как для модели Bao, так и для моделей на основе векторных представлений. Это указывает на то, что главным практическим сценарием применения таких моделей остаётся работа на стационарной или частично повторяющейся нагрузке, где в истории уже присутствуют запросы близких шаблонов.

Поведение стоимостной оценки оптимизатора в двух режимах тестирования также имеет естественное объяснение. Стоимость изначально разрабатывалась как внутренний сигнал для сравнения близких альтернативных планов одного и того же запроса, а не как средство глобального ранжирования планов из разных классов запросов. Поэтому при случайном разбиении, где сравниваются планы разных шаблонов, возможны систематические смещения между группами запросов, из-за чего коэффициент Спирмена оказывается низким. В то же время внутри отдельных групп стоимость может сохранять корректный относительный порядок, что поддерживает сравнительно более высокую попарную точность. Этим же объясняется и рост качества стоимостной оценки при разбиении по шаблонам: когда тестовая часть состоит из запросов одного шаблона, разброс структур планов уменьшается, и стоимость оптимизатора оказывается более согласованной с временем выполнения внутри группы. Вместе с тем даже в этом режиме наблюдается заметный разброс значений метрик между группами, что указывает на нестабильность такого сигнала, особенно для более сложных запросов.

На запросах тех же шаблонов, что присутствуют в обучении, лучшая модель на основе векторных представлений превосходит модель Bao по обеим метрикам качества. Это указывает на то, что модели векторных представлений способны захватывать структурные особенности плана без явного ручного кодирования дерева и потому могут рассматриваться как альтернатива структурной кодировке в предсказательных моделях ранжирования.

Сравнение по скорости показывает, что основным ограничением подходов на основе моделей векторных представлений является стоимость построения представления плана. В отличие от метода Bao, где после получения плана дополнительная обработка занимает пренебрежимо малое время, в предлагаемом подходе основная задержка связана именно с пропуском текстового описания через модель векторных представлений. Это означает, что выигрыш в простоте представления плана и отказе от специализированной структурной кодировки дерева достигается ценой более высокой вычислительной стоимости. Таким образом, между методом Bao и подходом на основе моделей векторных представлений возникает явный компромисс: первый вариант лучше подходит для сценариев, чувствительных к задержке, тогда как второй интересен в ситуациях, где важнее упростить построение признакового представления и сохранить высокое качество ранжирования на известной нагрузке.

При этом различия по времени между самими моделями векторных представлений показывают, что качество и вычислительная стоимость не совпадают напрямую. Наиболее сильная по качеству модель jina-code-embeddings-0.5b оказывается заметно быстрее Qwen3-Embedding-0.6B, которая также даёт высокие метрики, но требует наибольшего времени обработки. Кроме того, нормализованное представление плана стабильно уменьшает время обработки по сравнению с исходным текстом, однако это ускорение сопровождается ухудшением качества. Следовательно, в данной постановке нормализованный текст даёт выигрыш по скорости, но не обеспечивает лучшего компромисса по совокупности точности и времени, тогда как исходный текст оказывается более дорогим, но более информативным представлением плана.

Заключение

В работе исследована задача ранжирования планов SQL-запросов по времени выполнения и проведено сравнение трёх подходов: предсказательной модели Bao со структурной кодировкой дерева плана, стоимостной оценки стандартного оптимизатора и предложенного подхода на основе моделей векторных представлений текстового описания плана. Показано, что при случайном разбиении выборки лучшая конфигурация на основе jina-code-embeddings-0.5b и исходного текста плана превосходит модель Bao по обеим метрикам ранжирования. Также установлено, что для всех рассмотренных моделей исходное текстовое представление плана оказывается более информативным, чем использованная в работе схема нормализации, хотя нормализованный текст обеспечивает мень-

шее время обработки. При разбиении по шаблонам запросов качество всех обучаемых подходов снижается; при этом лучшая модель на основе векторных представлений и модель Bao показывают сопоставимые результаты между собой, но уступают стоимостной оценке оптимизатора, что указывает на сохраняющуюся проблему генерализации на ранее не наблюдавшиеся шаблоны запросов. Сравнение по скорости показывает, что подходы на основе моделей векторных представлений заметно уступают Bao по времени получения оценки плана, поскольку основная вычислительная стоимость связана с построением векторного представления. Поэтому такие методы наиболее естественно рассматривать в сценариях офлайн-оптимизации, где дополнительные вычислительные затраты допустимы ради повышения качества ранжирования и упрощения представления плана. Дальнейшая работа может быть связана с построением специализированных кодировщиков планов и с повышением устойчивости моделей к новым типам запросов.

References

[1] V. Leis, A. Gubichev, A. Mirchev, P. Boncz, A. Kemper, and T. Neumann, “How good are query optimizers, really?”, *Proceedings of the VLDB Endowment*, vol. 9, no. 3, pp. 204–215, 2015. DOI: 10.14778/2850583.2850594

[2] S. Chaudhuri, “An overview of query optimization in relational systems”, in *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Association for Computing Machinery, 1998, pp. 34–43. DOI: 10.1145/275487.275492

[3] H. Lan, Z. Bao, and Y. Peng, “A survey on advancing the DBMS query optimizer: Cardinality estimation, cost model, and plan enumeration”, *Data Science and Engineering*, vol. 6, no. 1, pp. 86–101, 2021. DOI: 10.1007/s41019-020-00149-7

[4] K. Kim, J. Jung, I. Seo, W.-S. Han, K. Choi, and J. Chong, “Learned cardinality estimation: An in-depth study”, Association for Computing Machinery, 2022, pp. 1214–1227. DOI: 10.1145/3514221.3526154

[5] T. Siddiqui, A. Jindal, S. Qiao, H. Patel, and W. Le, “Cost models for big data query processing: Learning, retrofitting, and our findings”, in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, Association for Computing Machinery, 2020, pp. 99–113. DOI: 10.1145/3318464.3380584

[6] M. Akdere, U. Çetintemel, M. Riondato, E. Upfal, and S. B. Zdonik, “Learning-based query performance modeling and prediction”, in *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, IEEE Computer Society, 2012, pp. 390–401. DOI: 10.1109/ICDE.2012.64

[7] R. Marcus, P. Negi, H. Mao, N. Tatbul, M. Alizadeh, and T. Kraska, “Bao: Making learned query optimization practical”, in *Proceedings of the 2021 International Conference on Management of Data*, Association for Computing Machinery, 2021, pp. 1275–1288. DOI: 10.1145/3448016.3452838

[8] C. Anneser et al., “AutoSteer: Learned query optimization for any SQL database”, *Proceedings of the VLDB Endowment*, vol. 16, no. 12, pp. 3515–3527, 2023. DOI: 10.14778/3611540.3611544

[9] L. Woltmann, J. Thiessat, C. Hartmann, D. Habich, and W. Lehner, “FASTgres: Making learned query optimizer hinting effective”, *Proceedings of the VLDB Endowment*, vol. 16, no. 11, pp. 3310–3322, 2023. DOI: 10.14778/3611479.3611528

[10] R. Zhu et al., “Lero: A learning-to-rank query optimizer”, *Proceedings of the VLDB Endowment*, vol. 16, no. 6, pp. 1466–1479, 2023. DOI: 10.14778/3583140.3583160

[11] Z. Yang, W.-L. Chiang, S. Luan, G. Mittal, M. Luo, and I. Stoica, “Balsa: Learning a query optimizer without expert demonstrations”, in *Proceedings of the 2022 International Conference on Management of Data*, ACM, 2022, pp. 931–944. DOI: 10.1145/3514221.3517885

[12] Y. Zhao, G. Cong, J. Shi, and C. Miao, “QueryFormer: A tree transformer model for query plan representation”, *Proceedings of the VLDB Endowment*, vol. 15, no. 8, pp. 1658–1670, 2022. DOI: 10.14778/3529337.3529349

[13] Z. Li, H. Yuan, H. Wang, G. Cong, and L. Bing, “LLM-R2: A large language model enhanced rule-based rewrite system for boosting query efficiency”, *Proceedings of the VLDB Endowment*, vol. 18, no. 1, pp. 53–65, 2024. DOI: 10.14778/3696435.3696440

[14] J. Liu and B. Mozafari, “GenRewrite: Query rewriting via large language models”, *Proceedings of the ACM on Management of Data*, vol. 4, no. 1, pp. 1–26, 2026. DOI: 10.1145/3786684

[15] S. Dharwada, H. Devrani, J. Haritsa, and H. Doraiswamy, *Query rewriting via LLMs*, 2025. arXiv: 2502.12918 [cs.DB].

[16] Y. Song et al., *QUITE: A query rewrite system beyond rules with LLM agents*, 2026. arXiv: 2506.07675 [cs.DB].

[17] P. Siebers, C. Janiesch, and P. Zschech, “A survey of text representation methods and their genealogy”, in *IEEE Access*, vol. 10, Institute of Electrical and Electronics Engineers (IEEE), 2022, pp. 96 492–96 513. DOI: 10.1109/access.2022.3205719

[18] P. Negi et al., “Flow-loss: Learning cardinality estimates that matter”, *Proceedings of the VLDB Endowment*, vol. 14, no. 11, pp. 2019–2032, 2021. DOI: 10.14778/3476249.3476259