논문 2026-2-15 http://dx.doi.org/10.29056/jsf.2026.06.15

XAI 기반 설명가능한 SQL 쿼리 성능 최적화 추천 프레임워크

최옥주*† , 신원선**

An Explainable AI-based Framework for SQL Query Performance Optimization Recommendation

Okjoo Choi*† , Wonsun Shin**

요 약

최근 머신러닝 기반의 SQL 쿼리 성능 예측을 활용한 최적화 방법들이 활발히 연구되고 있다. 그러나 대부분의 연구는 예측 정확도 향상에 초점을 두고 있어서 블랙박스 형태의 모델이 도출한 결과에 대한 해석과 설명이 어렵다는 한계를 가진다. 본 연구에서는 SQL 쿼리 성능 예측과 최적화 과정의 설명가능성을 제공하기 위해 XAI 기반의 SQL 쿼리 성능 최적화 추천 프레임워크을 제안한다. 제안한 프레임워크의 유효성을 검증하기 위해 PostgreSQL 환경에서 TPC-H Benchmark의 22개의 표준 쿼리를 대상으로 실험을 수행하였다. 실험 결과, 제안한 모델은 테스트 데이터 셋에서 MAE 359.846ms, RMSE 466.056ms, R^2 Score 0.9563을 달성하여 높은 예측 성능을 보였다. 또한 SHAP 분석을 통해 실행시간의 통계적 특징과 실행계획의 구조적 특징이 주요 성능 영향 요인으로 나타났으며 LIME 분석을 통해 개별 SQL 쿼리 수준에서 예측 결과를 설명할 수 있음을 확인하였다.

Abstract

Although recent machine learning-based approaches have shown promise in predicting SQL query performance, most studies primarily focus on improving prediction accuracy, leaving the interpretability of black-box models largely unaddressed. To address this limitation, this study proposes an explainable AI-based SQL query performance optimization recommendation framework that integrates performance prediction with post-hoc explanation techniques. The proposed framework was evaluated in a PostgreSQL environment using the 22 standard TPC-H benchmark queries. Experimental results demonstrated strong prediction performance on the test set, achieving a MAE of 359.846 ms, an RMSE of 466.056 ms, and an R^2 score of 0.9563. Furthermore, SHAP analysis revealed that statistical features of execution time and structural features of the execution plan were the major factors affecting performance, while LIME analysis demonstrated that the prediction results could be explained at the level of individual SQL queries.

한글키워드 : 설명가능 인공지능(XAI), SQL 쿼리 최적화, 쿼리 실행계획, SHAP, LIME

keywords : Explainable AI(XAI), SQL Query Optimization, Query Execution Plan, SHAP, LIME

* 배재대학교 AI.소프트웨어공학부
** 배재대학교 스마트ICT융합학과
† 교신저자:최옥주(email: okjoo.choi@pcu.ac.kr)

접수일자: 2026.06.02. 심사완료: 2026.06.07.
게재확정: 2026.06.20.

1. 서 론

빅데이터, 클라우드 컴퓨팅, 인공지능 기반 서비스의 확산으로 인해 데이터베이스 시스템(Database System, DBS)이 처리해야 하는 데이터의 양과 쿼리의 복잡성이 지속적으로 증가하고 있다. SQL 쿼리 성능 최적화는 사용자가 작성한 논리적인 쿼리를 실제로 실행할 수 있는 실행계획(Execution Plan)으로 변환하고 실행 비용을 낮추기 위한 최적의 경로를 찾는 것이다[1]. 실행 비용은 동일한 SQL 쿼리라도 연산자의 순서, 인덱스 사용 유무, 조인 순서 및 알고리즘에 따라 수백 배 차이가 발생한다[2].

데이터의 규모가 커지고 쿼리 구조가 복잡해지면서 전통적인 규칙 또는 비용 기반 쿼리 최적화 기법은 최적의 실행계획을 찾는 데 한계가 존재한다. 이러한 방법들은 미리 정해진 경험적인 규칙에 의존하거나 실행 통계 자료에 기반하고 있어서 비용 모델이 정확하지 않거나 오래된 통계를 사용할 경우 비효율적인 실행계획이 발생한다[3].

이를 해결하기 위해 최근에 머신러닝(Machine Learning, ML) 기법을 활용한 SQL 성능 예측 연구가 활발히 수행되고 있다[4]. 이들 연구는 기존의 비용 기반 모델보다 우수한 예측 성능을 달성하였으나 대부분의 모델이 블랙박스(Black box) 형태로 동작하기 때문에 예측 결과에 대한 설명이 어렵다는 한계를 가진다.

설명가능한 인공지능(Explainable Artificial Intelligence, XAI)은 머신러닝 모델의 의사결정 과정을 인간이 이해할 수 있도록 설명하는 기술이다. XAI 알고리즘인 SHAP과 LIME은 금융, 의료, 제조 등 다양한 분야에서 활용되고 있으며 복잡한 모델의 예측 근거를 정량적으로 설명하는 장점을 가진다. 그러나 데이터베이스 분야에서는 XAI를 SQL 실행계획 분석과 성능 최적화에 적용한 연구가 많지 않은 상황이다.

본 연구에서는 XAI 기법을 사용하여 SQL 쿼리 성능 최적화 추천 프레임워크(eXplainable SQL Tuning, xSQLtune)를 제안한다.

xSQLTune 프레임워크는 SQL 워크로드 수집, 실행계획 추출, 특징 추출, 성능 예측, XAI 기반 설명, 최적화 추천의 전체 과정을 통합적으로 지원한다. XGBoost 기반 성능 예측 모델과 SHAP 및 LIME 기반 설명 모델을 결합하여 SQL 성능 예측의 설명가능성을 확보하고, 실행계획 구조를 고려한 최적화 추천안을 생성한다. 이를 검증하기 위해 PostgreSQL 환경에서 TPC-H 데이터셋과 22개 표준 쿼리를 사용하여 워크로드 수집, 튜닝 추천, 시각적 설명 과정을 통합한 실험을 수행하였다.

2. 관련 연구

2.1 전통적인 쿼리 최적화

1) 규칙 기반 쿼리 최적화

규칙 기반 쿼리 옵티마이저(Rule-based Query Optimizer, RBO)는 쿼리의 논리 형태를 정해진 규칙 및 휴리스틱 형태로 변환한다[5]. RBO는 쿼리의 구조적 규칙에 따라 최적의 접근 경로 및 조인 순서를 미리 정해 놓는다. 쿼리가 입력되면 쿼리를 실행하기 전에 다양한 변환 규칙을 적용하여 쿼리 표현을 개선한 후에 최적화 단계를 진행한다. 구현이 용이하고 이해하기 쉬우나 복잡한 쿼리나 조인 순서에서 한계가 발생한다.

2) 비용 기반 쿼리 최적화

비용 기반 쿼리 옵티마이저(Cost-based Query Optimizer, CBO)는 쿼리의 다양한 실행계획의 비용을 비교 분석하여 가장 효율적인 실행계획을 선택한다[6]. CBO는 테이블의 크기 인덱스 유무, 조인 카디널리티 등의 통계정보를 사용하는 비용

모델이다. 데이터 정보 반영과 좋은 실행계획을 얻을 수 있는 반면 통계정보의 정확성이나 매우 복잡한 쿼리의 카디널리티 추정 오류에 따라 실행계획의 품질이 달라질 수 있다.

2.2 머신러닝 기반 쿼리 최적화

학습형 쿼리 옵티마이저(Learned Query Optimizer, LQO)는 머신러닝(Machine Learning, ML)과 강화학습(Reinforcement Learning, RL) 기법을 활용하여 쿼리를 최적화한다. 학습형 쿼리 최적화는 쿼리 워크로드와 성능 데이터를 학습하여 최적의 구성 값을 예측하거나 실행계획을 개선하는데 활용되고 있다. Neo나 Bao는 전통적인 옵티마이저를 기반으로 딥러닝 모델을 사용하여 쿼리 실행계획의 비용을 예측한다[7][8]. Balsa나 QTune는 강화학습을 기반으로 쿼리 최적화된 설정을 학습하는 기법이다[9][10]. CDBTune은 쿼리 이력과 워크로드를 기반으로 튜닝 전략을 추천하는 기법이다[11]

그러나 머신러닝 기반 최적화 기법은 시스템 성능 향상에 기여해 왔지만 대부분 블랙박스 모델에 의존하고 있어서 그 결과를 해석하고 근거를 제시하는데 어려움이 있다. 이러한 문제를 해결하기 위해 설명 가능성(Explainability)이 중요한 이슈로 부각되고 있다.

2.3 설명가능한 인공지능 기반 데이터베이스

설명가능한 인공지능(eXplainable AI, XAI)은 머신러닝 모델의 예측 결과나 의사결정 과정을 사람이 이해할 수 있는 방식으로 설명하는 기술이다. 주요 XAI 알고리즘에는 SHAP와 LIME이 있다[12][13].

최근 데이터베이스 분야에서도 XAI를 활용하려는 연구가 진행되고 있다. Explainable Gradient Boosting 기반 SQL Performance Anomaly Detection 연구는 실행계획 특징을 기반으로 SQL 성능 이상을 탐지하고 XGBoost 모델의 설명가능성을 제공하였다[14]. xDBTune은 데이터베이스 튜닝 과정에 SHAP를 적용하여 추천된 튜닝 파라미터가 성능에 미치는 영향을 설명하는 XAI 기반 데이터베이스 튜닝 프레임워크를 제안하였다[15]. 그러나 기존 XAI 기반 연구들은 주로 DBMS 파라미터 튜닝 또는 이상 탐지에 집중되어 있으며 SQL 쿼리 실행계획 자체를 분석하여 성능을 예측하고 설명가능한 최적화 추천안을 생성하는 통합 프레임워크는 아직 충분히 연구되지 않았다.

3. 설명가능한 SQL 쿼리 성능 최적화 추천 프레임워크

본 연구에서는 SQL 쿼리의 실행 성능을 예측하고 예측 결과에 대한 설명가능성을 제공하는 XAI 기반의 SQL 쿼리 성능 최적화 추천 프레임워크(eXplainable SQL Tuning, xSQLTune)를 제안한다. xSQLTune 프레임워크는 그림 1과 같이 6개 모듈로 구성된다. (1) Query Workload Collector, (2) Query Plan Extractor, (3) Feature Extractor, (4) Performance Predictor, (5) XAI-based Explainer, (6) Optimization Recommender

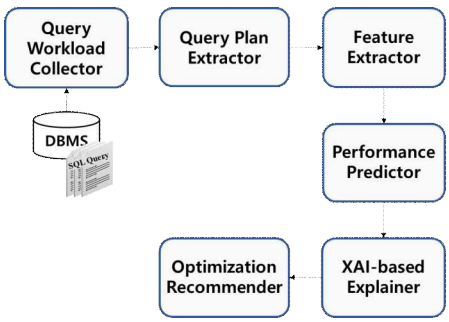

그림 1. xSQLTune 프레임워크
Fig. 1. xSQLTune Framework

3.1 Query Workload Collector

쿼리 워크로드 수집기(Query Workload Collector, QWC)는 SQL 쿼리를 반복 실행해서 성능 분석 대상이 되는 실행 시간, 반환 행 수, 로그기록, 실행 상태, 쿼리 시간대별 트래픽 등 쿼리 워크로드를 수집한다. 수집된 정보는 각 쿼리에 대하여 평균 실행시간, 중앙값, 최소 및 최대 실행시간 등을 계산한다. 또한 쿼리 식별자, SQL 해시값, 실행 횟수 등의 메타데이터를 함께 저장하여 분석에 활용한다. QWC 모듈은 다양한 SQL 쿼리의 실제 수행 성능 데이터를 확보하여 ML 모델 학습에 필요한 레이블을 생성하기 위해 실행된다.

3.2 Query Plan Extractor

쿼리 실행 계획 추출기(Query Plan Extractor, QPE)는 DBMS의 기능을 이용하여 SQL 쿼리의 실행 계획을 추출한다. 실행 계획은 DBMS Query Optimizer가 선택한 물리적 연산자들의 트리구조로 표현되며 쿼리 성능에 직접적인 영향을 미친다. 추출된 실행 계획에는 Scan, Join, Sort, Aggregate 등의 연산자 정보와 실제 실행 시간, 예상 행 수, 실제 행 수, shared read blocks, shared hit blocks 등의 통계 정보를 포함한다.

QPE 모듈은 쿼리 실행 결과뿐 아니라 쿼리 수행 과정에서 발생하는 내부 동작 정보를 확보함으로써 성능 병목 원인을 분석할 수 있는 기반 데이터를 제공한다.

3.3 Feature Extractor

특징 추출기(Feature Extractor, FE)는 QPE모듈이 생성한 실행 계획으로부터 ML 모델 입력에 사용할 특징(Feature)을 추출한다. 실행 계획 트리를 순회하면서 쿼리 구조적 특징, 통계적 특징, 자원 사용 특징을 계산한다. 특징들은 SQL 쿼리의 복잡도와 시스템 자원 사용량을 정량적으로 표현하며 성능 예측 모델의 입력 데이터로 활용된다.

표 1. 특징
Table 1. Features

Structural Features	Statistical Features	Resource Usage Features
Node Count Scan Count Join Count Sort Count Aggregate Count Plan Depth Nested Loop Count Hash Join Count Merge Join Count	Actual Rows Sum Plan Rows Sum Average Cardinality Error Loop Count	Shared Hit Blocks Shared Read Blocks Temp Read Blocks Temp Written Blocks

3.4 Performance Predictor

성능 예측기(Performance Predictor, PP)는 추출된 특징을 기반으로 SQL 쿼리의 실행 성능을 예측한다. 본 연구에서는 비선형 관계를 효과적으로 학습할 수 있는 XGBoost(eXtreme Gradient Boosting) 회귀 모델을 사용하였다.

입력 데이터는 FE모듈이 생성한 특징 벡터이며, 목표 변수는 QWC모듈이 수집한 평균 실행 시간이다. XGBoost 모델은 다수의 결정트리(Decision Tree)를 순차적으로 학습하여 예측 오차를 최소화한다. 학습된 모델은 SQL 쿼리가 실제로 수행되기 전에 예상 실행 시간을 추정할 수 있으며, 향후 최적화 효과를 사전에 평가하는 데 활용된다.

3.5 XAI-based Explainer

XAI 기반 설명기(XAI-based Explainer, Explainer)는 예측 결과의 근거를 제시한다. 본 연구에서는 SHAP와 LIME을 결합하여 SHAP는 전체 데이터셋 수준의 전역적(Global) 설명을 제공하고 LIME은 특정 쿼리 인스턴스 주변에 국소적(Local) 선형 모델을 생성하여 개별 예측 결

과에 영향을 미친 주요 특징을 식별한다. 이를 통해 다음 정보를 제공한다.

- 성능에 가장 큰 영향을 미치는 특징 식별
- 특징의 긍정적·부정적 기여도 분석
- 개별 쿼리 수준의 영향 요인 설명
- 전역 및 국소 설명 결과 비교

이를 통해 데이터베이스 관리자는 단순한 성능 예측 결과뿐 아니라 예측의 근거를 직관적으로 이해할 수 있다.

3.6 Optimization Recommender

최적화 추천기(Optimization Recommender, OR)은 SHAP 및 LIME 분석 결과와 실행계획 정보를 종합하여 SQL 튜닝 추천안을 생성한다. SHAP 중요도와 LIME 중요도를 기반으로 성능에 큰 영향을 미치는 특징을 식별한다. 이후 실행계획 분석 결과와 결합하여 규칙 기반(Rule-based) 추론을 수행한다.

예를 들어 seq scan 비중이 높고 index scan 비중이 낮은 경우에는 인덱스 생성 또는 인덱스 재설계를 추천한다. nested loop join이 과도하게 사용되는 경우에는 조인 순서 변경 또는 SQL 재작성을 제안한다. OR 모듈은 다음과 같은 최적화 방안을 제공한다.

- 인덱스 생성 및 재설계
- 조인 순서 최적화
- SQL Query Rewrite
- 통계정보 갱신
- 메모리 파라미터 튜닝(work_mem 등)

4. 실험

본 장에서는 xSQLTune 프레임워크의 성능 예측 정확성과 설명가능성을 검증한다. 이를 위하여 PostgreSQL 환경에서 TPC-H 벤치마크 데이터셋을 수행하고, 실행계획으로부터 추출된 특징을 이용하여 SQL 실행시간 예측 모델을 구축하였다. 또한 SHAP 및 LIME 기반 설명 모델을 적용하여 성능에 영향을 미치는 주요 요인을 분석하고 최종적으로 SQL 최적화 추천안의 타당성을 평가하였다.

표 2. xSQLTune 수도코드
Table 2. xSQLTune Pseudocode

Algorithm: xSQLTune Framework

Input:
  Q: SQL workload
  DBMS: PostgreSQL
  f: trained performance predictor
  Dbg: background dataset for SHAP and LIME
  τshap: SHAP threshold
  α1, α2, α3: node scoring weight
Output:
  Ŷ: prediction execution time set
  E: explanation set
  A: optimization recommendation set

Ŷ ← ∅, E ← ∅, A ← ∅
for each query q in Q do
    P ← EXPLAIN_ANALYZE_JSON(q, DBMS)
    Z ← ExtractNodeFeatures(P)
    x ← AggregateFeatures(Z)
    ŷ ← f(x)
    Ŷ ← Ŷ ∪ {ŷ}
    ϕ ← SHAP(f, x, Dbg)
    S ← { j | |ϕ_j| ≥ τshap }
    g ← LIME(f, x, Dbg)
    β ← Coefficients(g)
  for each node p in P do
    $s_p \leftarrow \Sigma_{j \in F(p)} |\phi_j|$
    $b_p \leftarrow \Sigma_{j \in F(p)} |\beta_j|$
    $m_p \leftarrow |\log((1 + \text{actual\_rows}_p)/(1 + \text{plan\_rows}_p))|$
    $B_p \leftarrow \alpha1 \cdot s_p + \alpha2 \cdot b_p + \alpha3 \cdot m_p$
  end for
  r ← RankNodesByScore(B)
  a ← GenerateOptimizationRecommendation(r, P)
  e ← CombineExplanation(ϕ, β, B)
  E ← E ∪ {e}
  A ← A ∪ {a}
end for
return Ŷ, E, A

4.1 실험 데이터

4.1.1 TPC-H 데이터 셋

본 실험에서는 TPC-H 의사결정 지원 벤치마크 데이터 셋을 사용한다[16]. TPC-H 데이터셋은 customer, orders, lineitem, part, partsupp. supplier, nation, region의 기본 테이블로 구성되어 있고 본 실험에서는 8개 테이블 모두 사용한다.

4.1.2 TPC-H 쿼리

실험에 사용된 SQL 워크로드는 TPC-H 표준 쿼리 22개를 사용한다. 각 쿼리는 다중 조인, 집계 함수(aggregation function), 정렬, 서브쿼리, 그룹화 등 다양한 연산으로 구성된다.

4.2 실험 절차

실험은 xSQLTune 프레임워크의 각 구성요소를 순차적으로 실행한다.

4.2.1 Query Workload Collection

각 SQL 쿼리는 PostgreSQL 환경에서 반복 수행한다. 실행 결과로부터 Query ID, Query Hash, Start Time, End Time, Elapsed Time. Row Count, Execution Status 수집하였다.

4.2.2 Query Plan Extraction

각 SQL 쿼리의 실행 계획은 PostgreSQL의 EXPLAIN ANALYZE 명령을 이용하여 수집하였다. 수집된 실행계획은 JSON 형태로 저장되며, Feature Extractor의 입력 데이터로 사용된다.

4.2.3 Feature Extraction

실행 계획을 순회하면서 전체 노드수, 조인 연산수, 정렬 연산 수 등의 쿼리 구조적 특징과 sequential scan수, index scan 수 등의 실행 연산 특징과 버퍼 히트 블록, 디스크 읽기 블록, 임시 읽기/쓰기 블록 등의 자원 사용 특징을 추출한다.

4.2.4 Performance Prediction

XGBoost 회귀모델을 사용하여 SQL 실행 시간을 예측하였다. 입력 변수는 Feature Extractor가 생성한 특징 벡터이며, 출력 변수는 평균 실행 시간이다. 성능 예측 정확도는 MAE(Mean Absolute Error), RMSE(Root Mean Squared Error), R^2 Score를 사용하였다.

4.2.5 XAI-based Explanation

설명가능성 분석을 위하여 SHAP와 LIME을 적용하였다. SHAP는 각 특징이 예측 결과에 미치는 기여도를 계산하고 SHAP Summary Plot, SHAP Waterfall Plot을 사용하여 시각화하였다. LIME은 특정 쿼리 인스턴스 주변에서 선형 근사 모델을 생성하여 설명을 수행한다. LIME Local Explanation, LIME Feature Importance Plot을 생성하여 개별 SQL 쿼리의 성능 결정 요인을 분석하였다.

두 XAI 기법의 설명 결과 일관성을 평가하기 위하여 상관분석을 수행하였다. SHAP 중요도 순위와 LIME 중요도 순위 간의 상관계수를 계산하여 설명 결과의 신뢰성을 검증하였다.

4.2.6 Optimization Recommendation

최적화 추천을 위해 SHAP 중요도, LIME 중요도, 실행 계획 정보를 통합하여 SQL 튜닝 추천안을 생성한다. 주요 추천 유형은 인덱스 생성 및 재설계, 조인 순서 변경, 통계 정보 갱신 등이 있으며 타당성은 PostgreSQL 실행 계획과 DBA의 일반적인 SQL 튜닝 가이드라인을 기준으로 평가하였다.

4.3 실험 결과

4.3.1 성능 예측 분석

Query Workload Collector와 Query Plan Extractor를 통해 수집한 실행 계획 특징 및 실행 통계 정보를 기반으로 XGBoost 회귀모델을 구축하여 SQL 실행 시간을 예측하였다.

표 3. 성능 예측 결과
Table 3. Performance Prediction Result

Metric	Training Set	Test Set
MAE (ms)	0.013 ms	359.846 ms
RMSE (ms)	0.025 ms	466.056 ms
R^2 Score	0.999999	0.9563

실험 결과 학습 데이터에 대한 MAE는 0.013ms, RMSE는 0.025ms로 나타났으며 R^2 Score는 0.999999를 측정되었다. 이는 학습 데이터에 대해 거의 정확하게 실행 시간을 예측하였음을 의미한다. 테스트 데이터에 대해서는 MAE가 359.846ms, RMSE가 466.056ms로 측정되었으며, 결정계수(R^2)는 0.9563으로 나타났다. 일반적으로 회귀모델에서 R^2 값이 0.9 이상이면 매우 높은 설명력을 갖는 것으로 평가하는데 본 연구의 결과는 전체 SQL 실행시간 변동성의 약 95.63%를 설명할 수 있음을 의미한다. 특히 테스트 데이터에서 R^2 값이 0.95 이상으로 나타난 것은 Query Plan 기반 특징과 실행 통계 특징이 SQL 성능을 효과적으로 설명할 수 있음을 보여준다. 반면 학습 데이터와 테스트 데이터 간 오차 차이가 비교적 크게 나타난 것은 실험 데이터 규모가 제한적이기 때문으로 판단된다.

4.3.2 설명가능성 분석

(1) SHAP 분석

SHAP 분석은 각 특징이 모델 예측 결과에 기여한 정도를 전역적 관점에서 설명한다. 본 연구에서는 SHAP Summary Plot과 SHAP Waterfall Plot을 이용하여 SQL 실행 시간 예측에 영향을 미치는 주요 특징을 분석하였다.

SHAP Summary Plot에서 x축의 SHAP 값이 양수 방향이면 해당 특징이 예측 실행 시간을 증가시키고 음수 방향이면 실행 시간을 감소시키는 것을 의미한다. 각 점은 하나의 SQL 쿼리 인스턴스를 의미하며 색상은 해당 특징값의 크기를 나타낸다.

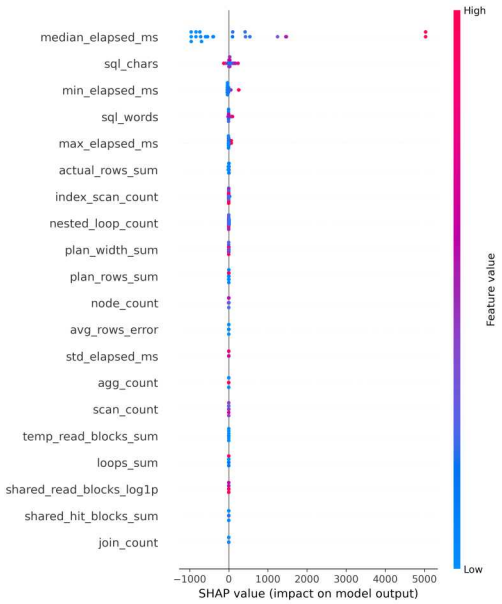

그림 2. SHAP Summary Plot
Fig. 2. SHAP Summary Plot

분석 결과, 그림 2와 같이 매우 넓은 범위의 SHAP 값으로 나타난 median_elapsed_ms가 예측 모델의 출력값에 가장 큰 영향을 미치는 변수라는 것을 알 수 있다. 반면, 대부분의 특징은 SHAP 값이 0 근처에 집중되어 있고 실행 계획 기반 특징인 join_count, shared_hit_blocks_sum, shared_read_blocks_log1p 등의 중요도는 상대적으로 낮게 나타났다. 이는 현재 학습 데이터에서

실행 시간의 통계 특징이 모델 예측에 직접적으로 사용되고 있기 때문으로 해석할 수 있다. 본 연구에서는 동일 SQL의 반복 수행 이력을 활용한다. 따라서 median_elapsed_ms, min_elapsed_ms, max_elapsed_ms, std_elapsed_ms 특징들은 과거 실행 이력을 나타내는 runtime history feature로 활용하였다. 향후에는 실행 이력 정보를 제외한 쿼리 계획 기반 특징만을 사용하여 설명 가능성을 검증할 예정이다.

SHAP Waterfall Plot은 특정 SQL 쿼리에 대한 예측 결과를 증가 또는 감소 방향을 보여준다. 기준값(E[f(X)])은 전체 학습 데이터의 평균 예측값을 의미하고, f(x)는 개별 특징의 SHAP 값을 기준값에 순차적으로 반영하여 최종 예측값(f(x))을 도출한다. Q1 쿼리의 경우, 그림 3에서 보는 바와 같이 예측 실행 시간은 1343.13ms로 측정되었다. 또한 median_elapsed_ms(+100.83)와 sql_chars(+25.25)가 예측에 가장 영향을 주는 것을 알 수 있다. 반면 min_elapsed_ms(-7.58)와 plan_width_sum(-1.68)은 실행 시간 감소 방향으로 작용하였다.

(2) LIME 분석

LIME에서도 SHAP 결과와 유사한 특징 중요도 순위가 분석되었다. 특히 median_elapsed_ms의 평균 절대 LIME 가중치는 약 1,080 이상으로 나타났으며 다른 모든 특징보다 압도적으로 높은 값을 보였다. 이는 SHAP 결과와 동일하게 모델이 실행 시간 통계 변수에 크게 의존하고 있음을 보여준다. 반면 Query Plan 관련 특징인 merge_join_count, hash_join_count, agg_count 등의 중요도가 상대적으로 높게 나타난 것은 LIME이 개별 쿼리 수준에서 연산자 구조의 일부 영향이 있음을 의미한다.

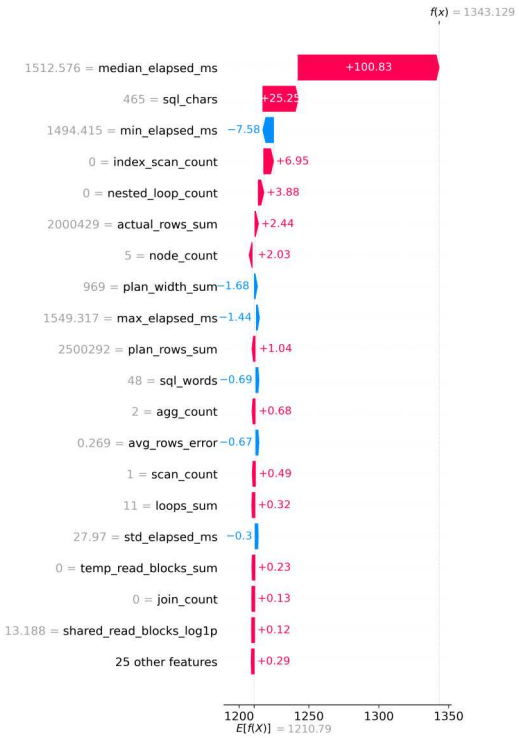

그림 3 SHAP Waterfall Plot
Fig. 3. SHAP Waterfall Plot

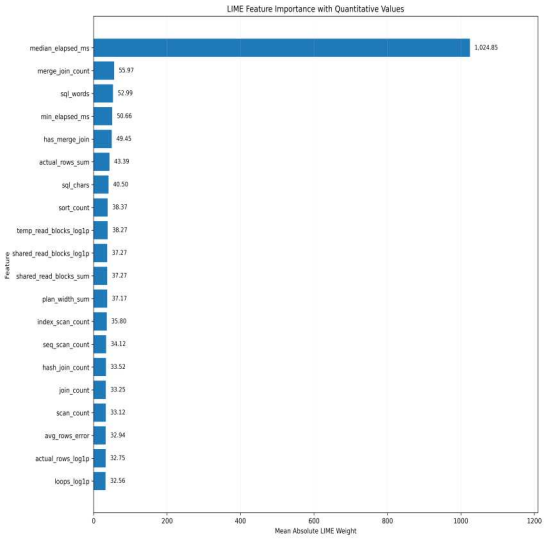

그림 4. LIME Importance Plot
Fig. 4. LIME Importance Plot

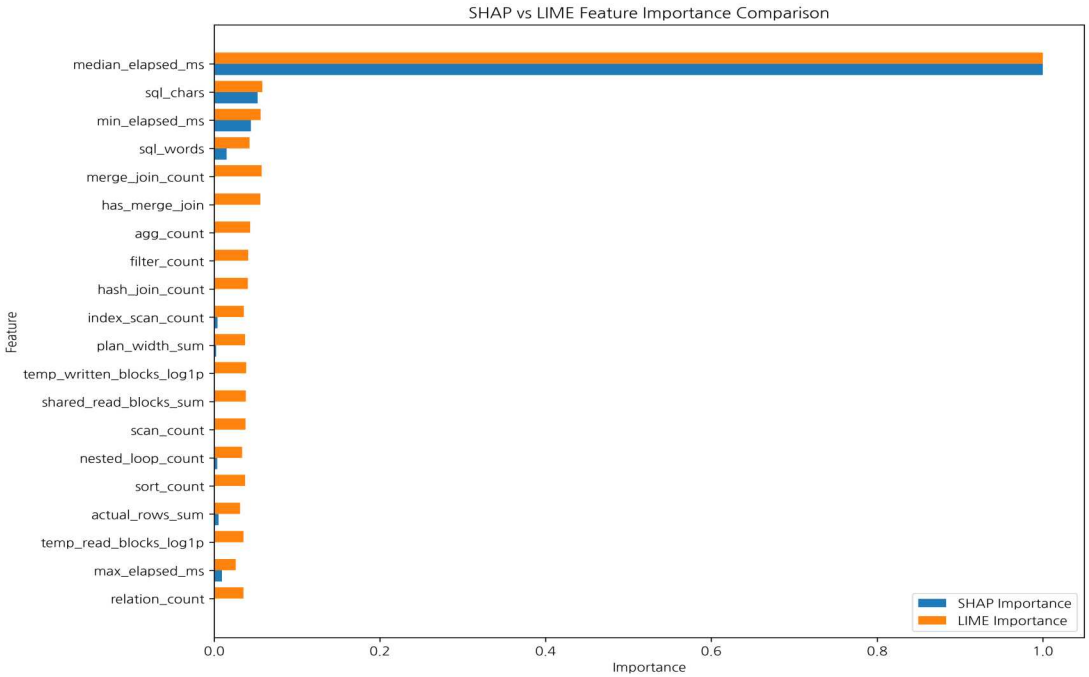

그림 5. SHAP vs LIME 특징 중요도 비교
Fig. 5. SHAP vs LIME Feature Importance Comparison

(3) SHAP-LIME 중요도 비교

그림 5에서 보는 바와 같이 SHAP와 LIME의 중요도를 비교한 결과, 두 기법 모두 median_elapsed_ms를 가장 중요한 특징으로 식별하였다. 또한 sql_words, sql_chars, min_elapsed_ms, merge_join_count 등의 특징들도 공통적으로 상위 중요 변수로 나타났다. 이는 두 설명 기법이 모델의 주요 의사결정 요인을 유사하게 식별하고 있음을 의미한다.

그러나 LIME은 Join 및 Aggregate 관련 특징이 더 높은 중요도가 부여된 반면 SHAP는 실행 시간 통계 변수에 상대적으로 더 집중하는 경향을 보였다. 이는 SHAP가 전역적 관점의 설명을 제공하는 반면 LIME은 개별 쿼리 주변의 국지적 설명을 수행하기 때문으로 해석된다.

(6) SHAP-LIME 상관관계

그림 6은 SHAP와 LIME 중요도 순위 간 상관관계를 나타낸다.

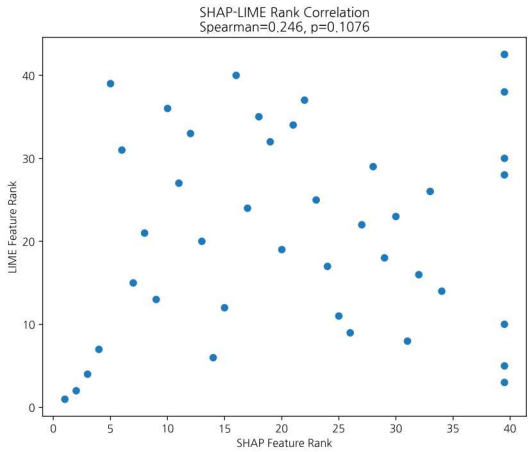

그림 6. SHAP-LIME 순위 상관관계
Fig. 6. SHAP-LIME Rank Correlation

Spearman 순위 상관계수는 다음과 같다.

- Spearman Correlation: 0.246

- p-value: 0.1076

Spearman 상관계수는 0.246으로 낮은 양의 상관관계를 나타내며, p-value는 0.1076으로 통계적 유의수준 0.05를 충족하지 못하였다. 이는 SHAP와 LIME이 일부 주요 특징에 대해서는 유사한 경향을 보였으나 전체 특징 순위에서는 상당한 차이가 존재함을 의미한다. 이러한 결과는 두 알고리즘의 중요도 비교 결과와 같이 SHAP가 전체 데이터 셋에 대한 전역적 설명을 제공하는 반면, LIME은 개별 쿼리 인스턴스 주변에서 국소적 설명을 제공하기 때문이라 할 수 있다. 따라서 동일한 예측 모델에 대해서도 중요도 순위 차이가 발생할 수 있다.

4.3.3 최적화 추천 평가 분석

실험 결과 표 4와 같이 22개 모든 쿼리에서 최소 2개 이상의 최적화 추천안이 생성되었다.

표 4. 추천 유형별 발생 빈도
Table 4. Frequency by Recommendation Type

Recommendation Type	Frequency	Rate
Index Recommendation	22	100.0%
Memory Tuning Recommendation	22	100.0%
Sort/Aggregation Recommendation	17	77.3%
Query Simplification Recommendation	14	63.6%
Statistics Recommendation	4	18.2%
Join Rewrite Recommendation	2	9.1%

Index Recommendation과 Memory Tuning Recommendation은 모든 쿼리에서 발생하였는데 이는 대부분의 쿼리에서 Sequential Scan과 대규모 I/O가 관찰되었고 Sort 또는 Aggregation 연산이 빈번하게 실행되었기 때문이다.

특히 복잡한 쿼리(Q02, Q05, Q08, Q09, Q21 등)의 경우에는 높은 Temp Block I/O, 큰 Cardinality Estimation Error, 깊은 Execution Plan, 다수의 Nested Loop Join 등의 특징이 발견되었다. 예를 들어, 쿼리 Q05의 경우 표 5와 같이 인덱스, 조인 재작성 등 5개의 추천안이 생성되었다.

표 5. 쿼리 Q05에 대한 추천 유형
Table 5. Recommendation Type for Query Q05

Query Q05	Recommendation Type	Priority
select n_name, , sum(l_extendedprice * (1-l_discount)) as revenue from customer, orders , lineitem, supplier , nation, region where c_custkey = o_custkey and o_orderkey = l_orderkey and s_suppkey = l_suppkey and c_nationkey = s_nationkey and s_nationkey = n_nationkey and n_regionkey = r_regionkey and r_name = 'asia' and o_orderdate >= to_date('1994-01-01', 'yyyy-mm-dd') and o_orderdate < to_date('1994-01-01', 'yyyy-mm-dd') + interval '1 year' group by n_name order by revenue desc	Index Recommendation	high
	Join Rewrite Recommendation	high
	Memory Tuning Recommendation	high
	Sort/Aggregation Recommendation	medium
	Query Simplification Recommendation	medium

본 연구에서는 추천안 생성까지 검증하였으며 실제 튜닝 적용 후 성능 개선 효과에 대한 정량적 검증은 향후 연구로 남긴다.

4.4 토의

본 연구에서 제안한 쿼리 예측 결과에 대한 설명 가능성과 실질적인 최적화 추천 기능을 제공하는 프레임워크는 기존 연구와 다음과 같은 차별성이 있다.

(1) PostgreSQL의 EXPLAIN ANALYZE 결과를 기반으로 Query Plan Feature를 추출한다.

(2) XGBoost 기반 성능 예측 모델을 구축하여 SQL 실행 시간을 예측한다.

(3) SHAP 및 LIME을 동시에 적용하여 전역(Global) 및 국소(Local) 설명을 제공한다.

(4) 실행 계획 구조와 XAI 결과를 결합하여 설명 가능한 최적화 추천안을 생성한다.

(5) SQL 성능 예측, 설명, 최적화 추천을 하나의 통합 프레임워크로 제공한다.

표 6. 기존 연구와 비교
Table 6. Comparison with Existing studies

	ML-based Prediction	SHAP	LIME	Recommendation
NEO	O	X	X	X
BAO	O	X	X	X
CDBTune	O	X	X	O
xDBTune	O	O	X	O
xSQLTune (our study)	O	O	O	O

본 연구에 존재하는 다음과 같은 한계점은 추후 연구로 남기고자 한다.

(1) 데이터 셋의 제한성

본 연구에서 사용한 TPC-H는 의사결정지원 시스템 워크로드를 대표하지만 실제 운영 환경의 다양한 쿼리 패턴을 모두 반영하지는 못한다.

(2) DBMS 의존성

본 연구는 PostgreSQL 실행계획 구조를 기반으로 특성을 추출하였다. Oracle, SQL Server, MySQL 등은 실행 계획 구조와 비용 모델이 다르므로 직접적인 적용에는 추가적인 특성 설계가 요구된다.

(3) 추천안 효과의 정량적 검증 부족

본 연구는 생성된 추천안의 타당성을 분석하였으나 실제 추천안을 적용한 후의 성능 향상률을 체계적으로 측정되지 않았다.

5. 결 론

본 연구에서는 SQL 성능 예측의 정확성뿐 아니라 설명 가능성을 제공하기 위한 XAI 기반 설명 가능한 SQL 쿼리 성능 최적화 프레임워크를 제안하였다. 제안 프레임워크는 PostgreSQL 실행 계획과 실행 통계 정보를 기반으로 SQL 성능 예측 모델을 구축하고, SHAP 및 LIME 기법을 활용하여 모델의 의사결정 과정을 설명하였다. 또한 설명 결과와 실행 계획 특성을 결합하여 자동 최적화 추천안을 생성함으로써 성능 예측에서 실제 튜닝 의사결정까지 지원할 수 있도록 설계하였다.

TPC-H 벤치마크의 22개 표준 쿼리를 대상으로 실험한 결과 다음과 같은 결론을 도출하였다.

(1) SHAP 분석 결과 실행 시간 통계 변수와 실행 계획 구조 변수들이 SQL 성능 예측에 가장 중요한 영향을 미치는 것으로 나타났다.

(2) LIME 분석을 통해 개별 쿼리 수준에서 성능 저하 원인을 직관적으로 설명할 수 있었으며, 모델의 설명 가능성을 확보할 수 있었다.

(3) 실행 계획 특성과 SHAP 설명 결과를 결합한 자동 최적화 추천 시스템은 인덱스 생성, 통계 갱신, 메모리 튜닝, 조인 재작성, 쿼리 단순화 등 실제 데이터베이스 튜닝에 활용 가능한 추천안을 제공하였다.

본 연구는 기존의 블랙박스 머신러닝 모델 기반의 SQL 성능 예측 연구의 한계를 보완하고, 예측 결과에 대한 신뢰성과 해석 가능성을 제공하였다. 또한 데이터베이스 관리자와 개발자가 SQL 성능 병목 원인을 이해하고 효율적인 튜닝 전략을 수립할 수 있도록 지원함으로써 실제 데이터베이스 운영 환경에서 활용할 수 있는 설명 가능한 SQL 쿼리 성능 최적화 프레임워크의 가

능성을 확인하였다.

향후에는 본 연구의 한계점을 극복하기 위해 실제 운영 데이터와 다양한 DBMS 환경에서의 검증을 통해 범용성을 확보하고 자동 튜닝 시스템과의 통합을 통해 완전한 자율형 데이터베이스 관리 환경으로 확장할 계획이다.

이 논문은 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원-지역지능화혁신인재양성사업의 지원을 받아 수행된 연구임 (IITP-2026-RS-2022-00156334)

참 고 문 헌

[1] Y. E. Ioannidis, “Query Optimization,” ACM Computing Surveys, Vol. 28, No. 1, pp. 121-123, March, 1996. https://doi.org/10.1145/234313.234367

[2] S. Chaudhuri, “An Overview of Query Optimization in Relational Systems,” in Proceedings of the 17th ACM Symposium on Principles of Database Systems, Seattle, Washington, pp. 34-43, May 1998. https://doi.org/10.1145/275487.275492

[3] V. Leis, A. Gubichev, A. Mirchev, P. Boncz, A. Kemper, and T. Neumann, “Still Asking: How Good Are Query Optimizers, Really?,” Proceedings of the VLDB Endowment, Vol. 18, Issue 12, pp. 5531-5536, August 2025. https://doi.org/10.14778/3750601.3760521

[4] R. Zhu, L. Weng, B. Ding, and J. Zhou, “Learned Query Optimizer: What is New and What is Next”, in Proceedings of the Companion of the 2024 International Conference on Management of Data, Santiago, Chile, pp. 561-569, 2024. https://doi.org/10.1145/3626246.3654692

[5] H. Pirahesh, J. M. Hellerstein, and W. Hasan, “Extensible/Rule Based Query Rewrite Optimization in Starburst,” ACM SIGMOD Record, Vol.21, No. 2, pp 39-48, June 1992. https://doi.org/10.1145/141484.130294

[6] P. Griffiths Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price, “Access Path Selection in a Relational Database Management System,” in Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data, Boston MA, pp. 23-34, 1979. https://doi.org/10.1145/582095.582099

[7] R. Marcus, P. Negi, H. Mao, C. Zhang, M.Alizadeh, T. Kraska, ... and N. Tatbul, “Neo: A Learned Query Optimizer,” Proceedings of the VLDB Endowment, Vol. 12, No. 11, pp. 1705-1718, 2019. https://doi.org/10.14778/3342263.3342644

[8] R. Marcus and O. Papaemmanouil, “Plan-Structured Deep Neural Network Models for Query Performance Prediction”, in Proceedings of the VLDB Endowment, Vol. 12, No. 11, pp. 1733-1746, 2019,VLDB. https://doi.org/10.14778/3342263.3342646

[9] 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, Philadelphia: PA, pp. 931-944, 2022. https://doi.org/10.1145/3514221.35178

[10] Guoliang Li et al., “QTune: A Query-Aware Database Tuning System with Deep Reinforcement Learning,” In Proc. VLDB Endowment, Vol. 12, pp.2118-2130, 2019

[11] J. Zhang et al. “CDBTune+: An Efficient Deep Reinforcement Learning-based Automatic Cloud Database Tuning System,” The VLDB Journal, Vol.30, pp.959-987, 2021

[12] S. Lundberg and S. Lee, “A Unified Approach to Interpreting Model Predictions”,

In Proc. NeurIPS, pp.4768-4777, 2017
[13] Ribeiro et al. "Why Should I Trust You?", Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp1135 - 1144, 2016, https://doi.org/10.1145/2939672.2939778
[14] S. Gorle, J. Christadoss, and S. Sethuraman , "Explainable Gradient-Boosting Classifier for SQL Query Performance Anomaly Detection", American Journal of Cognitive Computing and AI Systems, Vol 9 2025, https://ajccai.org/index.php/publication/article/view/36
[15] O. Choi and W. Shin, "xDBTune: eXplainable Database Tuning Framework," The Transactions of the Korea Information Processing Society, vol. 14, no. 9, pp. 704-712, 2025. DOI: https://doi.org/10.3745/TKIPS.2025.14.9.704.
[16] TPC-H. Dataset [Internet]. Available: https://www.tpc.org/tpc_documents_current_versions/current_specifications5.asp.

저 자 소 개

최옥주(Okjoo Choi)

2008.02 숙명여자대학교 컴퓨터과학과 박사
1990.8-1996.3 LG생산기술원 주임원구원
1996.7-2009.8 한국오라클 수석컨설턴트
2009.9-2022.12 카이스트 전산학부 연구교수
2023.10-2024.2 강원대학교 산학협력중점교수
2024.3-현재 배재대학교 AI·SW공학부 조교수
<주관심분야> XAI, AI신뢰성, 데이터베이스, 빅데이터 분석, 데이터품질, 소프트웨어품질

신원선(Wonsun Shin)

1997년 숙명여자대학교 전산학과 석사
2022년 충북대학교 전파통신공학과 박사수료
2022년~현재 배재대학교 스마트ICT융합학과 박사과정
1996년 숙명여자대학교 아태여성정보통신원 연구원
2015년 공룡컴 평생교육원 이사
2020년~현재 주식회사 비전21테크 대표
관심분야: 자동음성인식, 감정인식, 데이터 품질