**Db2 AI Query Optimizer: Expanded Predicate Support and Improved Model Training**

A network connection structure with blue dots and lines

AI-generated content may be incorrect.

**Introduction**

In our previous blog posts ([1](#References), [2](#References), [3](#References), [4](#References)), we introduced the Db2 AI Query Optimizer, a revolutionary approach to database query optimization using neural networks. We demonstrated how our system learns from table column correlations to make more intelligent decisions about query execution plans, outperforming traditional optimizers in many scenarios.

Today, we're excited to share 2 significant enhancements to the Db2 AI Query Optimizer that further improve its performance and capabilities:

1. Leveraging of Transfer Learning to reduce model training time and improve model robustness
2. Expanded support for the 'IS NULL' predicate

These new features represent our ongoing commitment to making database query optimization smarter, faster, and more efficient.

The best part? You don't need to do anything to take advantage of these improvements. If you are already using the Db2 AI Query Optimizer, these improvements are automatically enabled as soon as you upgrade to Db2 version 12.1.2 or later.

**Transfer Learning: Accelerating Model Training**

One of the most exciting new features in the AI Query Optimizer is the implementation of transfer learning for model training. If you recall from our previous blog ([3](#References)), neural network training for the Db2 AI Query Optimizer was already impressively fast - "Done in 60 Seconds" - but we've managed to make it even faster, while also improving model accuracy and robustness.

**What is Transfer Learning?**

Transfer learning is a machine learning technique where a model developed for one task is reused as the starting point for a model on a second task. In the context of the Db2 AI Query Optimizer, this means we can leverage knowledge from previously trained models to accelerate the training of new models.

**Understanding Neural Network Weight Distributions**

To understand why transfer learning is so effective, let's first examine what happens during neural network training from scratch. Recall the structure of the neural network used in the Db2 AI Query Optimizer:

A diagram of a network structure

AI-generated content may be incorrect.

**Neural Network Training from Random Initialization**

When we train a neural network from scratch, we start with randomly initialized weights following a normal distribution centered around 0.0. The image below shows a sample random distribution of weights for the first 2 layers of an AI Query Optimizer neural network for the TPCDS schema INVENTORY table, where 4 columns are included in the model:

A graph of a weight distribution

AI-generated content may be incorrect.

As training progresses, the network adjusts these weights to minimize prediction errors. By the end of training, the weights still maintain a roughly normal distribution, but with some important differences.

A comparison of a diagram

AI-generated content may be incorrect.

Notice that while the distribution remains centered near zero, it has:

1. Spread out slightly, with a wider variance
2. Developed some outliers - weights with significantly larger positive or negative values

Let's look at a different visualization of the neural network weights, a heatmap. This visualization shows the magnitude of the weights, with bright yellow indicating larger positive values and dark blue indicating larger negative values. At the start of training we can see that the colours are randomly distributed, with no clear structure or patterns:

A comparison of weights and weights

AI-generated content may be incorrect.

During training, some interesting patterns start to emerge in the heatmap representation, and by the end of training the heatmaps for the first 2 layers are as follows:

A comparison of weights and weights

AI-generated content may be incorrect.

We can see that the first layer weights have formed specific patterns that capture the underlying data relationships. In this case, we observed one very interesting pattern in particular (highlighted in red) in the first layer weights where one neuron would develop a larger positive weight value, while an adjacent neuron would develop a larger negative value. This pattern emerged consistently across trained models and represents the network learning to detect important features in the underlying data. Recall from our neural network architecture that each 'pair' of inputs ([0,1], [2,3], [4,5], [6,7]) represents 'min' and 'max' values for a single table column, and from the heatmap we can see that the trained network has learned about this relationship.

For the second layer, it's hard (at least for humans!) to discern any specific patterns in the layer weights. However, we can see that the weights are no longer random, suggesting the trained network has learned something from the training data. Most values are clustered around zero, and there are a select few larger positive and negative values. Subsequent layers showed similar patterns, with most weights clustered around zero, and a few larger positive and negative values.

These "learned patterns" in the weight distribution (even if we can't always detect them with the naked eye) are what make the neural network effective at its task.

**How Transfer Learning Works in the AI Query Optimizer**

Previously, each table's neural network model was trained from scratch, with random weight initializations. While this approach worked well, it required the full training time for each model to learn the underlying data patterns and correlations.

With our new transfer learning approach:

1. During development, we trained many models of various sizes (e.g. various numbers of input columns for the model) and determined ideal starting weights for each model size.
2. When training a new model for a specific table, rather than initializing the model weights to random values, we instead initilize them with the appropriate pre-computed model weights.
3. We then fine-tune the model on the specific table data

The model starts with a better understanding of general table selectivity data patterns and only needs to learn the specific nuances and column correlations of the table in question. This approach allowed us to reduce the number of training epochs (the number of times the model iterates over the training data), which means faster training times and less resource consumption.

**Weight Distributions in Transfer Learning**

When training from scratch, we start with random weights that have no meaningful pattern, and through training, the network develops specific patterns that represent learned features.

With transfer learning, we start with weights that already have a more structured distribution and contain meaningful patterns:

A comparison of weights and weights

AI-generated content may be incorrect.

The fine-tuning process preserves most of the useful patterns from the pre-trained model while adjusting specific weights to better fit the current table's data characteristics:

A comparison of weights and weights

AI-generated content may be incorrect.

Notice how the fine-tuned model preserves much of the structure from the pre-trained model while making targeted adjustments. This visual representation helps explain why transfer learning is so efficient - it's building upon existing knowledge rather than learning everything from scratch. This is why transfer learning can achieve similar or better accuracy with significantly less training time.

**Training Time and Accuracy Improvements**

Our internal benchmarks show that transfer learning reduces model training time by an average of 35%.

While faster training time is the most obvious benefit, transfer learning also provides other advantages. For example, another (arguably more) important benefit is that it makes model training more robust. Starting from random weight initializations is more risky since a 'bad' initial random weight distribution can lead to longer training times and sub-optimal model accuracy. Starting from a consistently-good initial state (from transfer learning) makes the training process more predictable and robust.

**'IS NULL' Predicate Support**

The IS NULL predicate is commonly used in SQL queries when dealing with data that may have missing values. Traditional optimizers often assume independence between NULL values in one column and (NULL or non-NULL) values from other columns, which can lead to suboptimal query plans. The initial Db2 AI Query Optimizer neural networks were already trained on data that may have contained NULL values, depending on whether the associated table contained NULL value for one or more columns included in the model. After extensive testing and validation, we've confirmed that our Db2 AI Query Optimizer neural networks effectively learn correlations between NULL values and values from other columns. This means that the Db2 AI Query Optimizer can safely and effectively handle IS NULL predicates without requiring any additional configuration or tuning.

**Example Query with 'IS NULL' Predicate**

Consider a query like:

SELECT \* FROM

SAMPLE.PROJECT PROJECT

WHERE

PROJECT.PRENDATE BETWEEN '12/31/2001' AND '12/31/2003' AND

PROJECT.PRSTAFF < 8 AND

PROJECT.RESPEMP > '000019' AND

PROJECT.MAJPROJ IS NULL

The Db2 AI Query Optimizer can now accurately predict the combined selectivity of this set of predicates, taking into account any correlations between the values in columns PRENDATE, PRSTAFF, RESPEMP, and whether MAJPROJ values are NULL. This leads to more efficient query plans, as the optimizer can better estimate the number of rows that will satisfy the query conditions.

**Explain Output for 'IS NULL' Predicates**

The explain facility has been extended to show when NULL predicates are handled by the Db2 AI Query Optimizer:

Table Cardinality Model Predicates:

-----------------------------------

2) Q1.MAJPROJ IS NULL

3) ('000019' < Q1.RESPEMP)

4) (Q1.PRSTAFF < 8)

5) (Q1.PRENDATE <= '2003-12-31')

6) ('2001-12-31' <= Q1.PRENDATE)

For this particular query, when run against the SAMPLE database (generated using the db2sampl command, refer to blog [4](#References) for more details), the actual number of rows returned is 4. The Db2 AI Query Optimizer estimates 5.2 rows satisfy the combined predicates, whereas the optimizer estimate without using the Db2 AI Query Optimizer is 1.6 rows, so the Db2 AI Query Optimizer estimate is closer to the actual number of rows returned.

**Continuous Improvements**

The combination of transfer learning and enhanced NULL predicate support has led to significant improvements in the Db2 AI Query Optimizer:

1. **Faster model training**: As mentioned, training time is reduced by 35% on average
2. **More accurate selectivity estimates**: Especially for queries involving IS NULL predicates
3. **Better query plans**: The improved selectivity estimates lead to better access plan choices

**How to Take Advantage of These New Features**

The good news is that you don't need to do anything special to take advantage of these new features. They are automatically enabled for the Db2 AI Query Optimizer when you upgrade to the latest version of Db2.

Any new models will leverage transfer learning for faster and more robust training. And any queries that use IS NULL predicates will automatically benefit from the enhanced NULL predicate support.

**Summary**

The Db2 AI Query Optimizer continues to evolve with new features that make it even more powerful and efficient. The addition of transfer learning significantly reduces model training time, while support for IS NULL predicates improves selectivity estimates for queries involving NULL values.

These enhancements represent our ongoing commitment to making database query optimization smarter, faster, and more efficient. By continuously improving the AI Query Optimizer, we're helping you get the best possible performance from your database queries.

Try out these new features today and experience the benefits of AI-powered query optimization!

What's next on our journey? We're putting the finishing touches on a major new enhancement to the Db2 AI Query Optimizer that will yet again revolutionize query optimization, while further reducing the burden of manual performance tuning. Can you guess what it is? Stay tuned for more details!

**References**

1. Zuzarte, Calisto. [The AI Query Optimizer in Db2](https://www.idug.org/news/the-ai-query-optimizer-in-db2). IDUG. 2024-05-23.
2. Zuzarte, Calisto & Corvinelli, Vincent. [The AI Query Optimizer Features in Db2 Version 12.1](https://www.idug.org/news/the-ai-query-optimizer-features-in-db2-version121). IDUG. 2024-08-12.
3. Finnie, Liam. [The Neural Networks Powering the Db2 AI Query Optimizer](https://www.idug.org/news/the-neural-networks-powering-the-db2-ai-query-optimizer). IDUG. 2024-10-17.
4. Frendo, Brandon. [Simplify Query Performance Tuning with the Db2 AI Query Optimizer](https://www.idug.org/news/simplify-query-performance-tuning-with-the-db2-ai-query-optimizer). IDUG. 2024-12-15.
5. Pan, Sinno Jialin, and Qiang Yang. [A Survey on Transfer Learning](https://ieeexplore.ieee.org/document/5288526). IEEE Transactions on Knowledge and Data Engineering 22, no. 10 (2010): 1345-1359.
6. Brownlee, Jason. [A Gentle Introduction to Transfer Learning for Deep Learning](https://machinelearningmastery.com/transfer-learning-for-deep-learning/). Machine Learning Mastery. 2019-09-16.