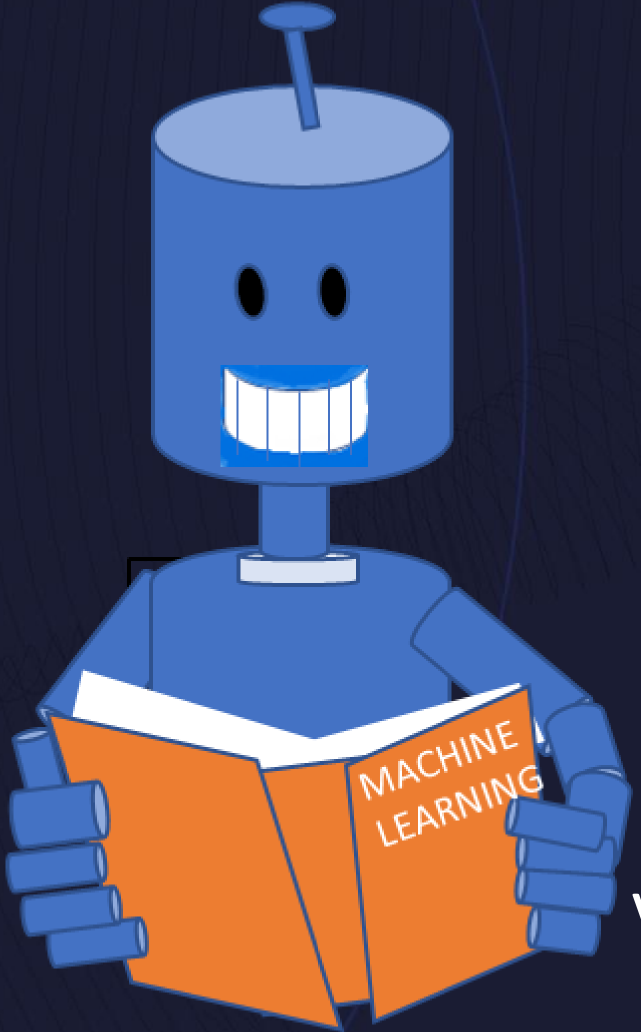
Using Machine Learning in the Db2 Optimizer



Calisto Zuzarte, IBM

WDUG

Wisconsin Db2 Users Group

Virtual Conference | March 09, 2022

Agenda

- Motivation
- Cardinality Estimation
- Db2 11.5.6 ML Optimizer Tech Preview
    - Architecture
    - Experimental Results

Motivation

Evolution Of the Database Optimizer

```
IF <condition 1>
THEN
   {
      <action 1>
   }
ELSE IF <condition 2>
THEN
   {
      <action 2>
   }
ELSE <action 3>
```

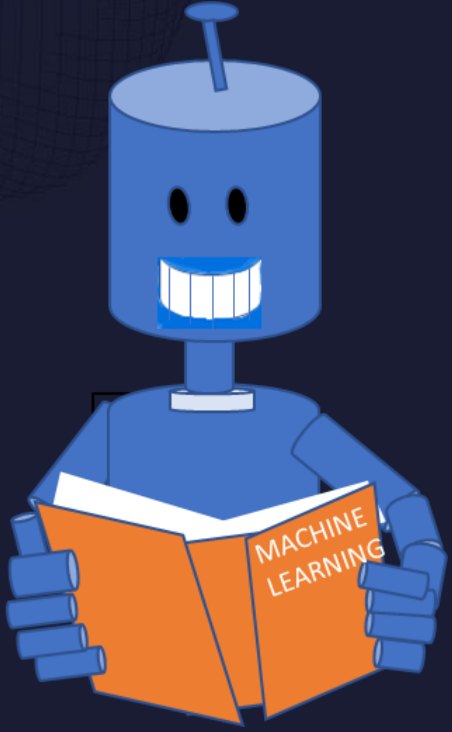
RULE BASED CODE

1980 →

$$\sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{N - 1}}$$

STATISTICS BASED COST MODEL

2020 →

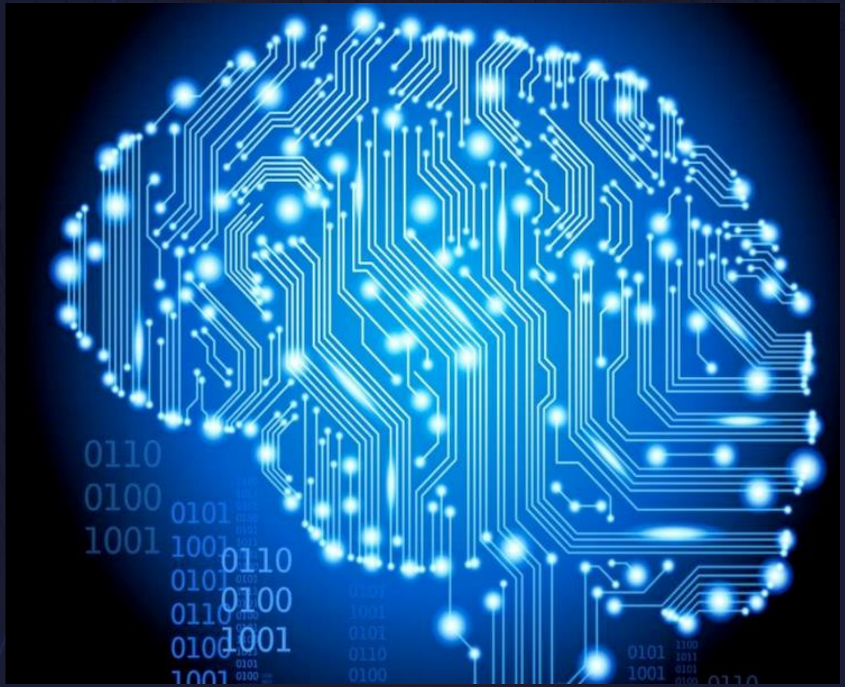


MACHINE LEARNING MODEL

Optimizer Challenges

Performance Stability	Query complexity, higher data volumes and demanding user expectations require an easily adaptable and stable solution
Tuning Effort	Minimum customer tuning needed to adapt to specific characteristics of user data, workloads and environment
Development Effort	Minimum effort needed to the optimizer with new features, configuration changes and hardware upgrades

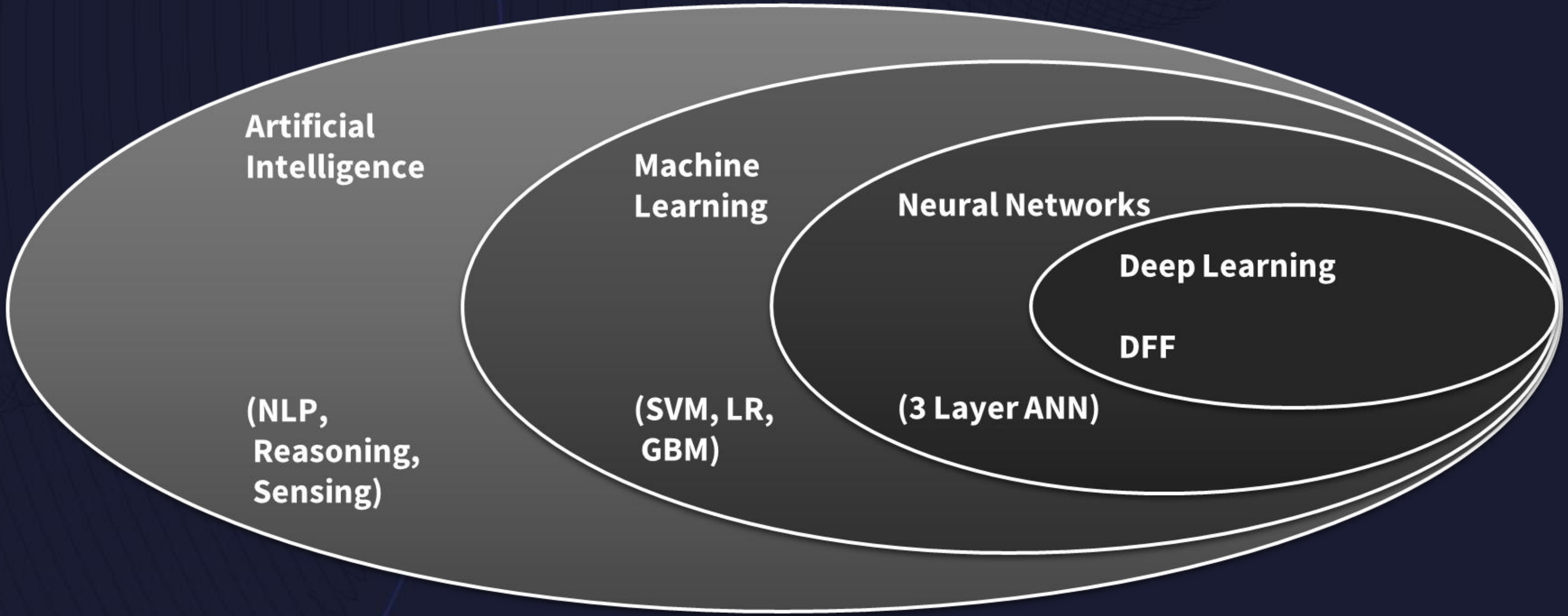
Artificial Intelligence (AI) is the simulation of human intelligence in machines that are programmed to think like humans.



Machine Learning provides AI systems the ability to automatically learn and improve from experience without being explicitly programmed.

A Neural Network is a series of algorithms that tries to recognize underlying relationships in a set of data using interconnected nodes much like neurons in a human brain

Infusing AI in Db2



Benefits of Machine Learning

01

Adapt to specific user data characteristics

02

Adapt to specific user query workloads

03

Learn from optimizer and run-time feedback

Machine Learning Goals

Automate Everything	Make performance tuning simple with automation
Achieve Reliable Performance	By constantly learning and improving the model
Simplify Optimizer Development	By training the model in the specific user environment
Infuse ML Gradually	Gradually replace traditional optimizer techniques

A Phased Approach

Phase 1	Cardinality Estimation
Phase 2	Join Planning
Phase 3	Other Aspects

Cardinality Estimation

Cardinality Estimation

Cardinality Estimation is the number of rows input to or output from an operator

Cost based optimizers rely on reasonably accurate cardinality

Bad cardinality estimation is often the primary source of query performance problem tickets from customers

Tuning For Good Cardinality Estimates

Actual : 10,113,972

```
                1136.92
                HSJOIN
                (   4)
                 90013
                 67956
        /----------+----------\
    2e+06                   1136.92
   TBSCAN                   ^HSJOIN
   (   5)                   (   6)
   469.792                   89537
     344                     67612
      |               /-------+-------\
    2e+06        2.87997e+08        0.288374
CO-TABLE: DB2INST1  TBSCAN            TBSCAN
  CUSTOMER          (   7)            (   8)
     Q1             88366.6           274.207
                     67383              229
                       |                 |
                 2.87997e+08           73049
             CO-TABLE: DB2INST1      DB2INST1
               STORE_SALES           DATE_DIM
                    Q3                  Q2
```

Default Statistics

```
                457723
                HSJOIN
                (   4)
                 90013
                 67956
        /----------+----------\
    2e+06                    457723
   TBSCAN                   ^HSJOIN
   (   5)                   (   6)
   469.792                   89537
     344                     67612
      |               /-------+-------\
    2e+06        2.87997e+08        116.099
CO-TABLE: DB2INST1  TBSCAN            TBSCAN
  CUSTOMER          (   7)            (   8)
     Q1             88366.6           274.207
                     67383              229
                       |                 |
                 2.87997e+08           73049
             CO-TABLE: DB2INST1      DB2INST1
               STORE_SALES           DATE_DIM
                    Q3                  Q2
```

With additional Column Group Statistics

```
                       9.05383e+06
                         ^HSJOIN
                         (   4)
                         90083.4
                          67956
                 /---------+---------\
          9.05383e+06               2e+06
            ^HSJOIN                 TBSCAN
            (   5)                  (   8)
            89564.4                 469.792
             67612                    344
      /-------+-------\                |
2.87997e+08        116.099           2e+06
  TBSCAN            TBSCAN      CO-TABLE: DB2INST1
  (   6)            (   7)         CUSTOMER
  88366.6           274.214           Q1
   67383              229
     |                 |
2.87997e+08          73049
CO-TABLE: DB2INST1  CO-TABLE: DB2INST1
 STORE_SALES         DATE_DIM
     Q3                 Q2
```

With additional Statistical Views

ML To The Rescue

Can ML avoid the need for the tuning by experts? YES!

Are there areas not currently adequately covered by the traditional optimizer? YES!

Predicate Support (1|2)

Predicates supported:

- Local Predicates with Equality, Range, Between , IN, OR
- Single-column equality pairwise join predicates over base tables.

Predicates not supported:

- multi-column and non-equality join predicates
- predicates with host variables or parameter markers not using REOPT
- predicates with expressions around the columns
- These will be evaluated by the traditional Db2 optimizer.

Predicate Support (2|2)

SELECT * FROM T1, T2
WHERE
T1.C0 = T2.C0 AND -- Pair-Wise Join Predicates ★
T1.C6 IN (5, 3, 205) AND -- IN Predicates ★
T1.C1 = 'abc' AND -- Equality Predicates ★
T1.C2 BETWEEN 5 AND 10 AND -- BETWEEN Predicates ★
T2.C3 <= 120 AND -- Range Predicates ★
(T1.C4 > 5 AND T1.C5 < 20 OR T1.C4 < 2 AND T1.C5 = 100) AND -- OR Predicates ★
T1.C3 = ? AND -- Predicates With Parameter Markers ★
MOD(T1.C4, 10) = 1; -- Predicates With Expressions ★

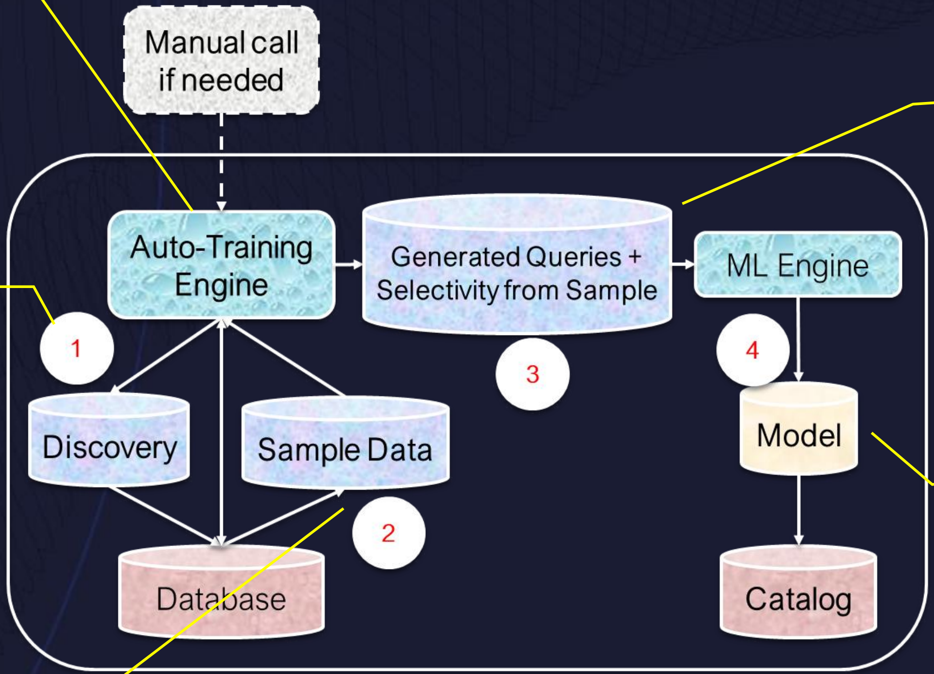
Db2 11.5.6 ML Optimizer Tech Preview Architecture

Automatic Training

The **Auto-Training Engine** looks for tables without a model

If no model exists, a **Discovery Engine** mines the data to help training

Sample data is retrieved from the table.



Training Queries are automatically generated

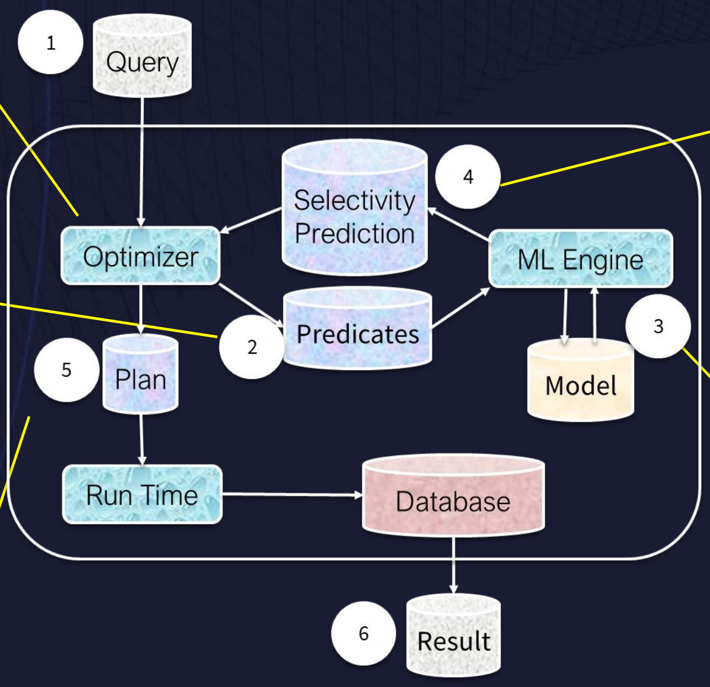
The **ML model** is built using the training queries and sample data

Cardinality Prediction Using ML

Queries processed normally except for card estimation

Eligible Predicates are encoded as inputs to the ML Engine

The ML estimates are integrated in the optimizer to get the execution plan



The cardinality estimation is sent to the optimizer

The **ML model** gives a cardinality estimate for the predicate set

Automatic Feedback and Retraining

Automatic Feedback of table data changes is used.

Future: Optimizer and run time feedback will be added

Automatic Retraining is currently triggered based on table modification activity not unlike how Auto-RUNSTATS is triggered for a table

Db2 11.5.6 ML Optimizer Tech Preview
Experimental Results

Model Size and Training Time

NN Model Size is significantly better than with LGBM

NN Model size is 1000X better !
30KB versus 30MB

Accuracy, (not shown here) is a little better with LGBM than with NN

TABLENAME	MODEL SIZE (MiB)		TRAINING TIME (S)	
	NN	LGBM	NN	LGBM
CALL_CENTER	0.021	0.003	0	2
CATALOG_PAGE	0.022	33.401	60	94
CATALOG_RETURNS	0.037	32.742	67	358
CATALOG_SALES	0.037	32.745	103	376
CUSTOMER	0.024	33.147	37	358
CUSTOMER_ADDRESS	0.023	33.717	34	89
DATE_DIM	0.037	33.176	43	362
INCOME_BAND	0.021	0.066	1	2
ITEM	0.030	6.432	68	307
PROMOTION	0.022	13.707	480	14
REASON	0.021	0.146	9	1
SHIP_MODE	0.021	0.182	28	2
STORE	0.022	0.422	46	2
STORE_RETURNS	0.024	32.763	47	361
STORE_SALES	0.037	32.865	68	342
TIME_DIM	0.022	1.861	34	80
WAREHOUSE	0.021	0.003	0	1
WEB_PAGE	0.022	7.889	40	3
WEB_RETURNS	0.037	32.767	82	347
WEB_SALES	0.037	32.757	82	368
WEB_SITE	0.024	2.650	46	6

Training Time is also better with NN compared to LGBM

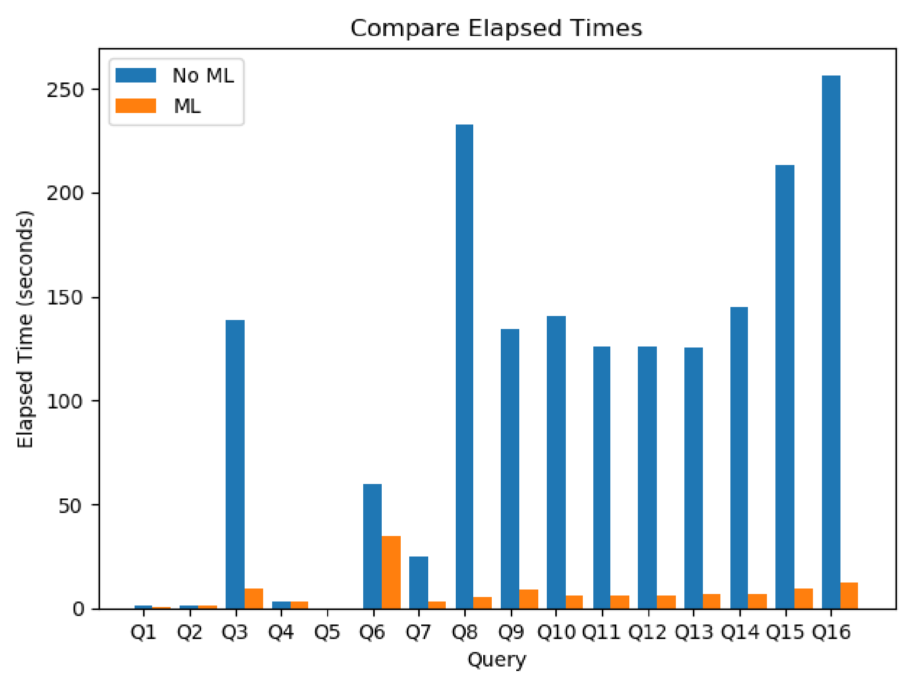
Training time is 5X less than LGBM
5 m versus 1 m

Real World Problematic Queries

10X benefit in some of these scenarios simulated in-house

In practice the average benefit will be less

The goal is to get more reliable performance.



Query Example

An example of one of the queries (Q10) in the benchmark

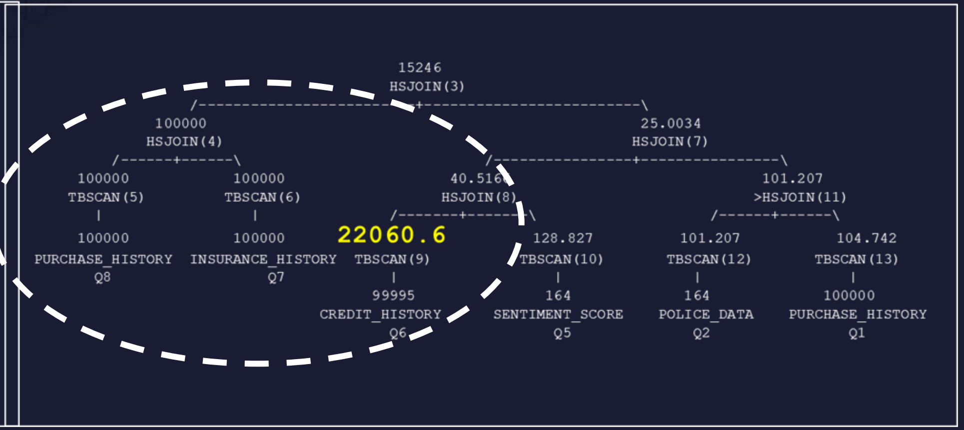
The key benefit with ML was a better cardinality estimate with the set of highly correlated BETWEEN predicates

```
SELECT
    IH.AMOUNT,
    CHD.COMMENTS
FROM
    DEMO.PURCHASE_HISTORY    PH,
    DEMO.INSURANCE_HISTORY    IH,
    DEMO.CREDIT_HISTORY_DATA    CHD,
    DEMO.SENTIMENT_SCORE_DATA    SSD,
    DEMO.POLICE_DATA   PD
    LEFT OUTER JOIN
       (SELECT EMAILID
        FROM DEMO.PURCHASE_HISTORY   PH1
       WHERE PH1.PURCHASE_DATE BETWEEN '2018-12-30' and '2018-12-31' ) X
     ON PD.EMAILID = X.EMAILID
WHERE
    PH.INSURANCE_ID = IH.INSURANCE_ID AND
    PH.PURCHASE_DATE BETWEEN '2014-01-01' AND '2019-12-31' AND
    PD.EMAILID = PH.EMAILID AND
    PD.CRIMINAL_RANK > .4 AND
    PD.EMAILID = SSD.EMAILID AND
    SSD.SCORE < .7 AND
    PH.EMAILID = CHD.EMAILID AND
    CHD.PAY_0 BETWEEN 0 AND 2 AND
    CHD.PAY_2 BETWEEN 0 AND 2 AND
    CHD.PAY_3 BETWEEN 0 AND 2 AND
    CHD.PAY_5 BETWEEN 0 AND 2 AND
    CHD.PAY_6 BETWEEN 0 AND 2 AND
    CHD.PAY_4 BETWEEN 0 AND 2 AND
    CHD.BILL_AMT1 BETWEEN 150 AND 746814 AND
    CHD.BILL_AMT2 BETWEEN 0 AND 743970 AND
    CHD.BILL_AMT3 BETWEEN 0 AND 689643 AND
    CHD.BILL_AMT4 BETWEEN 0 AND 706864
```

Q10 Cardinality / Plan Change with ML

No ML

ML



Join Cardinality – Single Table Model

For both plots : (1) Closer to 0 is better (2) Thinner box is better

ML | No ML, Basic Statistics + CGS | No ML, Basic Statistics

N:1 JOIN - ONE JOIN PREDICATE

M:M JOIN - THREE JOIN PREDICATES

Tech Preview Automation Switches

- Enabling the ML Optimizer
  - db2set DB2_ML_OPT="ENABLE:ON"
  - db2 –tf MLOptimizerCreateTables.ddl

- Disabling the ML Optimizer
  - db2set DB2_ML_OPT="ENABLE:OFF"

Manual Steps If Necessary

- Defining a Model:
  CALL SYSTOOLS.DEFINE_MODEL('MYSCHEMA', 'MYTABLE', 'C1,C2,C3', OUT_TEXT)

- Toggle to use the traditional Optimizer:
  db2set -im DB2_SELECTIVITY="ML_PRED_SEL OFF"

- Deleting a model:
  DELETE FROM SYSTOOLS.TABLE_MODELS
  WHERE SCHEMANAME = 'MYSCHEMA' AND TABLENAME = 'MYTABLE';

Summary

The initial Db2 ML Optimizer goal is to improve **cardinality estimation**

This addresses the leading cause of performance issues

Reducing tuning needs will improve the out-of-the-box experiences

Infusing AI in the Db2 Optimizer is strategic

Speaker: Calisto Zuzarte
Company: IBM
Email Address: calisto@ca.ibm.com