AI Query Optimizer and Query Tuner

Calisto Zuzarte
2025-03-13

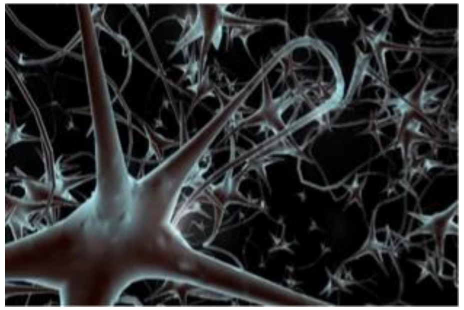
calisto@ca.ibm.com
Tridex NY

Agenda

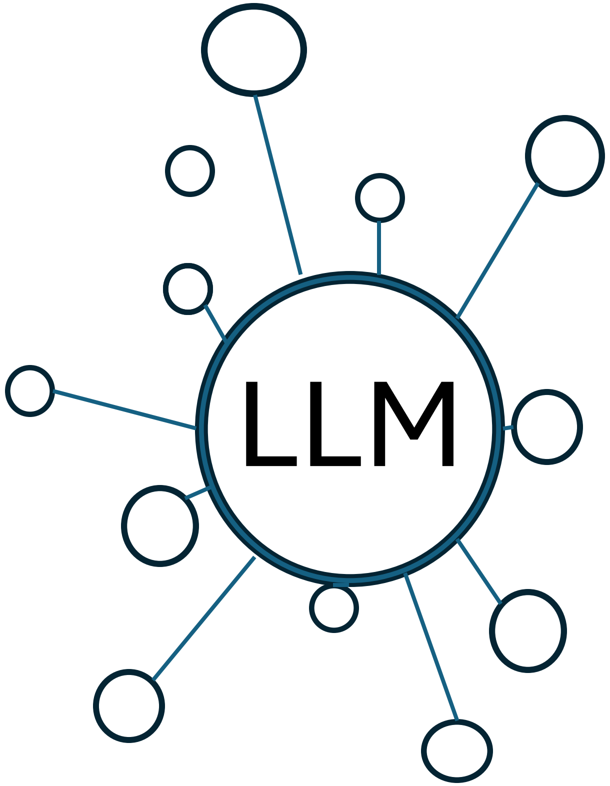
- Motivation
- AI Query Optimizer (Db2 v12.1)
- (AI) Query Tuner (Coming Soon)

AI

ML

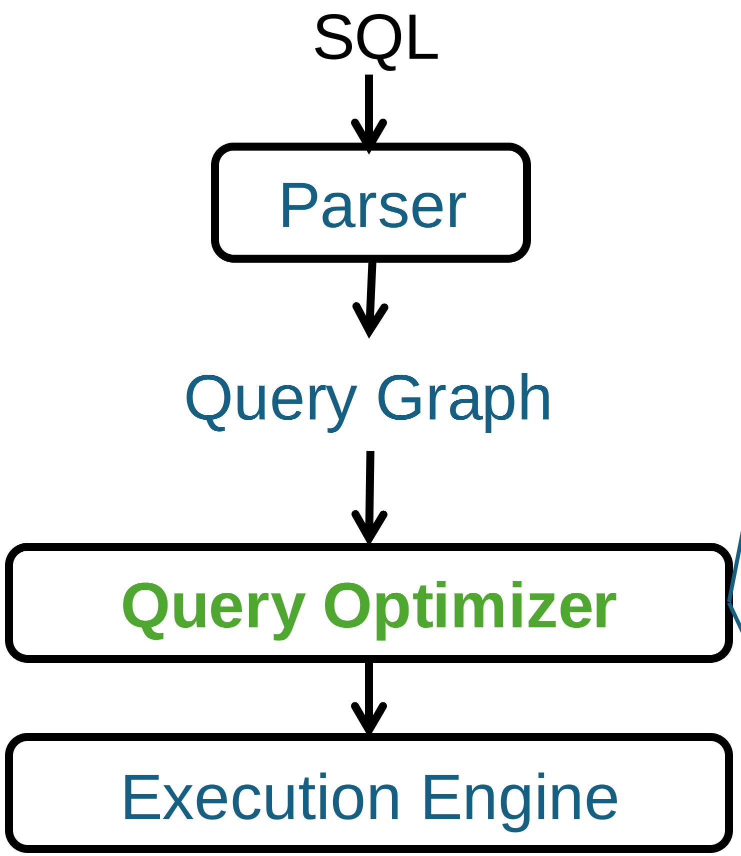


NN



Motivation

The Query Optimizer



- Rewrites the query graph for performance
- Estimates the number of rows for each operator
- Estimates the costs of each operator
- Generates alternate subplans
- Selects the cheapest overall plan
- Sends it to the execution engine

Cardinality Estimation

- Cardinality is the number of rows input to or output from an operator
- Generally reduced by predicates (increased with expanding joins)
- Traditionally estimated using statistics
- Predicate columns are generally assumed to be independent
- Errors of many orders of magnitude can occur due to skew and correlation
- How can we improve cardinality estimates?

Improving Cardinality Estimates

Actual : 10,113,972

10,000X off !

```
            1136
           HSJOIN
           (   4)
             |
    /--------+--------\
2,000,000            1136
 TBSCAN             HSJOIN
 (   5)             (   6)
   |            /------+------\
2,000,000  287,997,000      0.288374
CUSTOMER     TBSCAN          TBSCAN
   Q1        (   7)          (   8)
               |               |
          287,997,000        73049
          STORE_SALES       DATE_DIM
              Q3               Q2
```

Default Statistics

20X off !

```
            457723
            HSJOIN
            (   4)
              |
    /---------+---------\
2,000,000             457723
 TBSCAN               HSJOIN
 (   5)               (   6)
   |              /------+------\
2,000,000    287,997,000      116.099
CUSTOMER       TBSCAN          TBSCAN
   Q1          (   7)          (   8)
                 |               |
             287,997,00        73049
            STORE_SALES       DATE_DIM
                Q3               Q2
```

With additional Column Group Statistics

Close

```
                      9,053,830
                        HSJOIN
                        (   4)
                          |
              /-----------+-----------\
         9,053,830                 2,000,00
           HSJOIN                   TBSCAN
           (   5)                   (   8)
      /------+------\                 |
287,997,000      116.099          2,000,000
  TBSCAN          TBSCAN           CUSTOMER
  (   6)          (   7)              Q1
    |               |
287,997,000       73049
STORE_SALES      DATE_DIM
    Q3              Q2
```

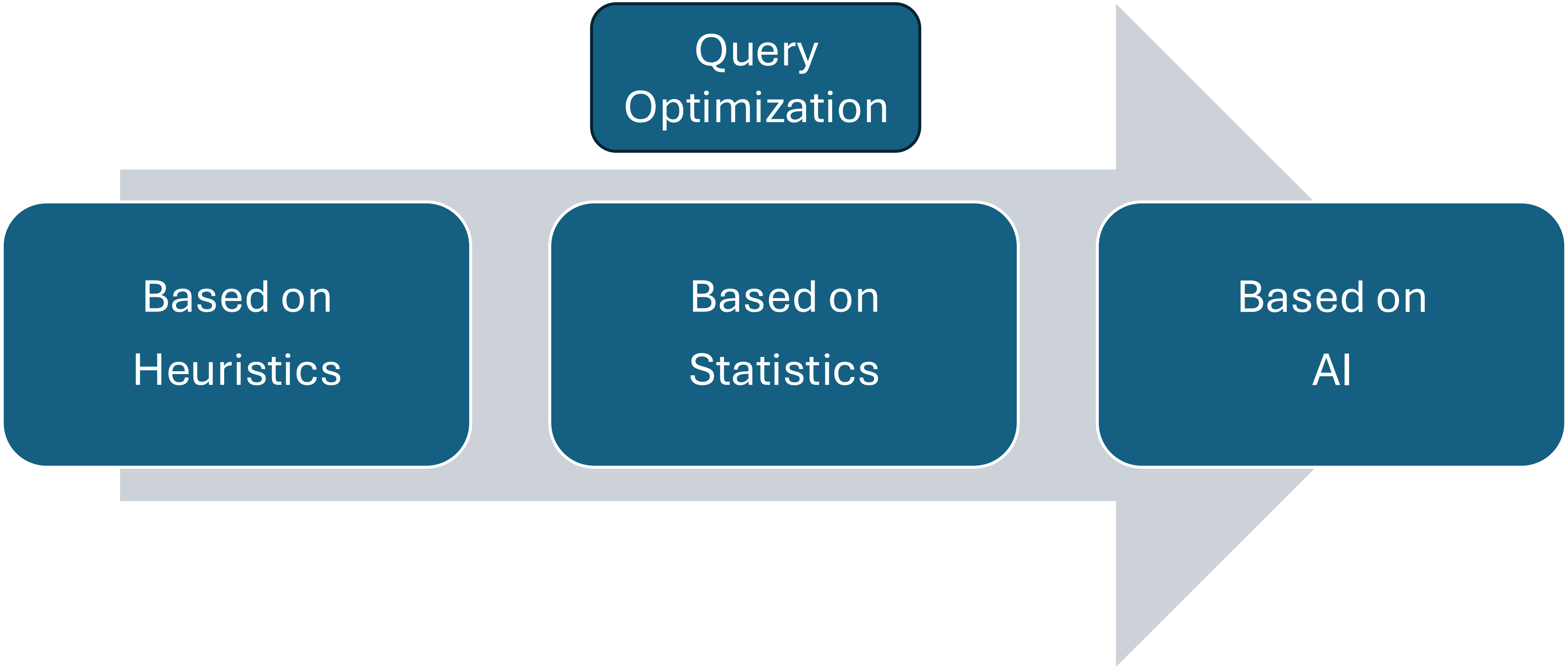
With additional Statistical Views

Tuning is Difficult

- What Column Group Statistics should one collect?
- What are Statistical Views and how does one create one that will improve performance for the query?
- Would an index improve performance and what columns should one define that index on?
- These are tasks for the AI Query Tuner .

AI Query Optimizer

Evolution of Query Optimizer Model



Can AI do Better?

Optimizer Challenges

- Performance Stability
- Development Effort
- Tuning Effort

AI Query Optimizer Goals

- Automate Everything
- Achieve Reliable Performance
- Simplify Optimizer Development

Customization Benefits

- Adapt to User Data
- Adapt to User Workloads
- Learn from Optimizer and Runtime feedback

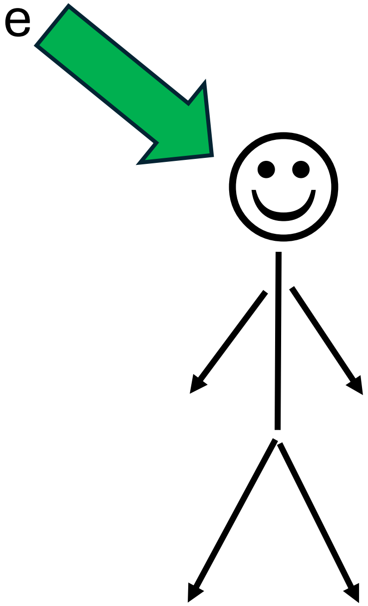
Infuse AI Gradually

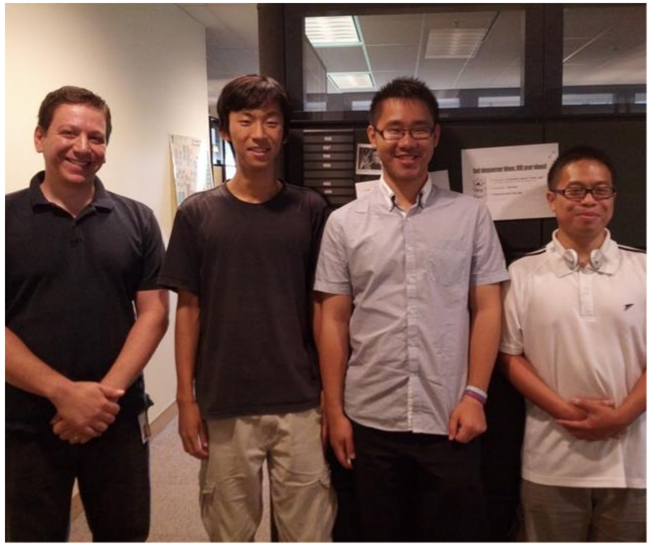
- Local Predicate Cardinality Estimation
- Join Cardinality Estimation
- Query Rewrite, Tuning, Other aspects …

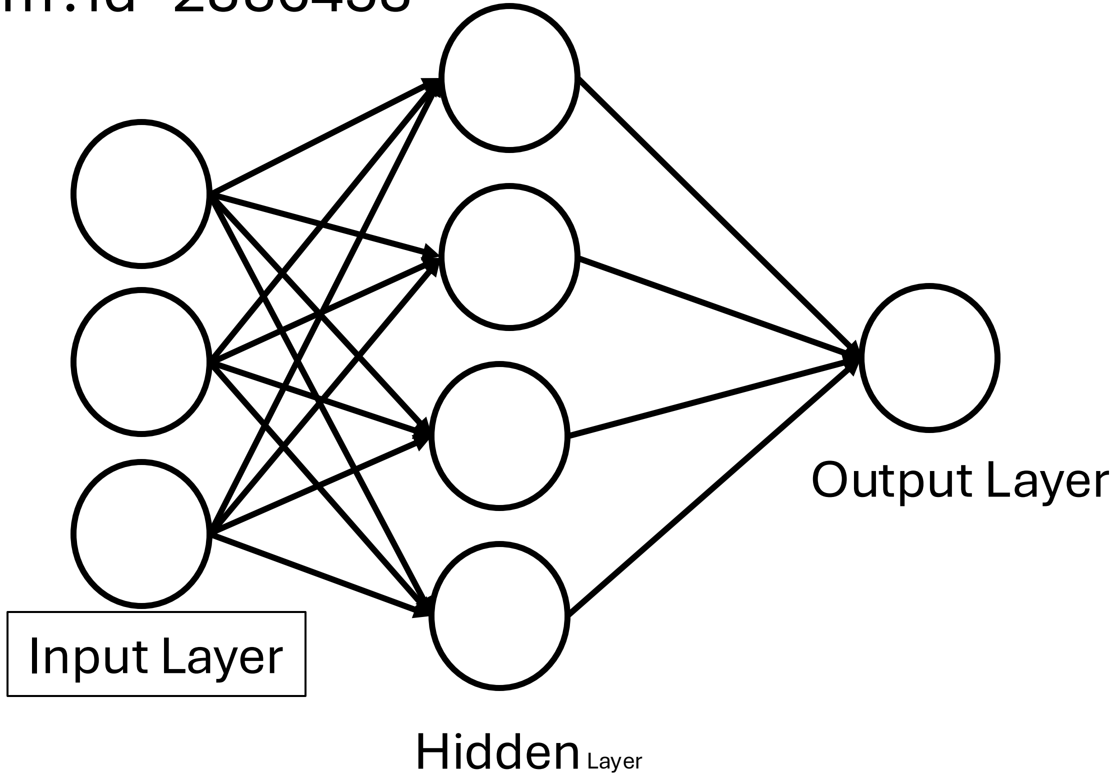
Our First Prototype in 2013
Research Paper Published in 2015

- "Cardinality Estimation Using Neural Networks" CASCON 2015: 53-59. Henry Liu, Mingbin Xu, Ziting Yu, Vincent Corvinelli, Calisto Zuzarte
  - https://dl.acm.org/citation.cfm?id=2886453

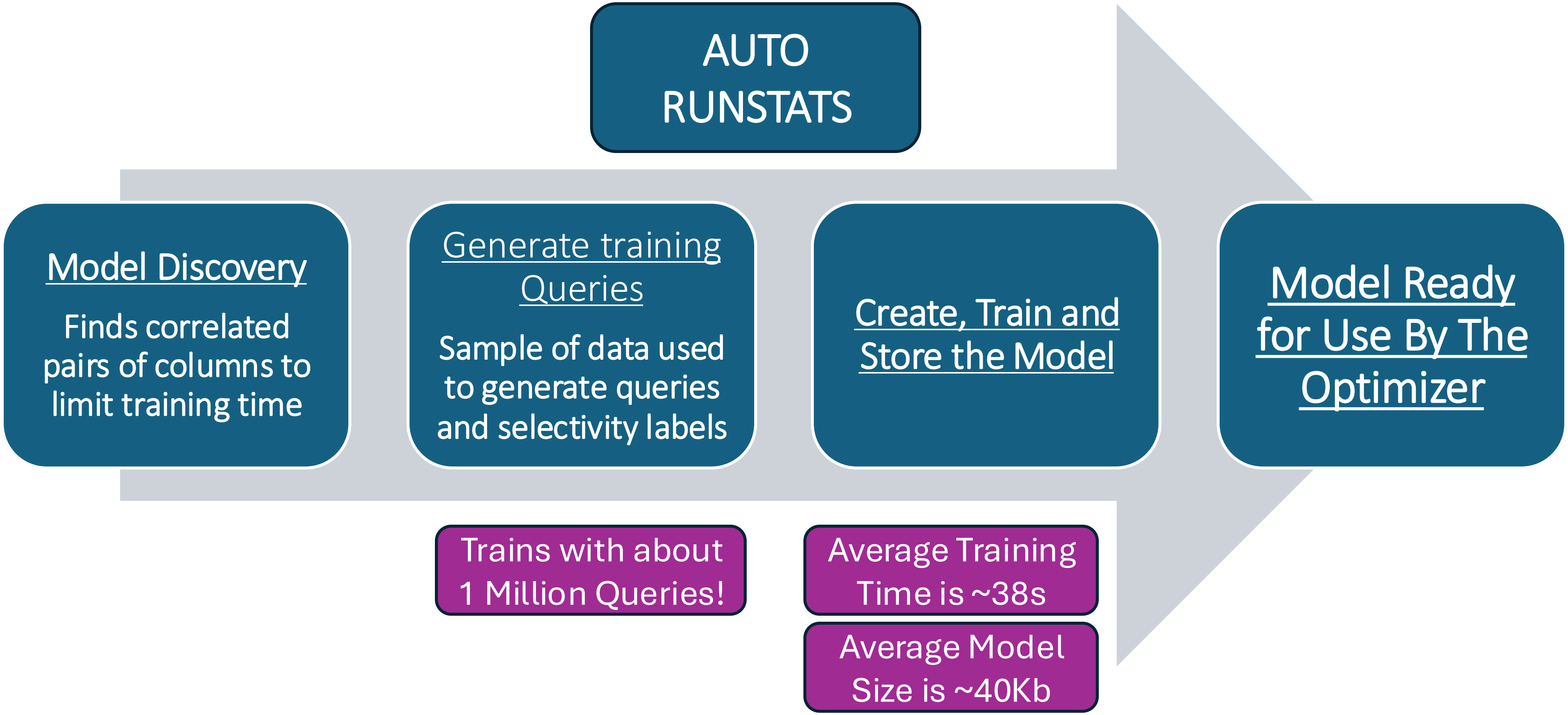
Me



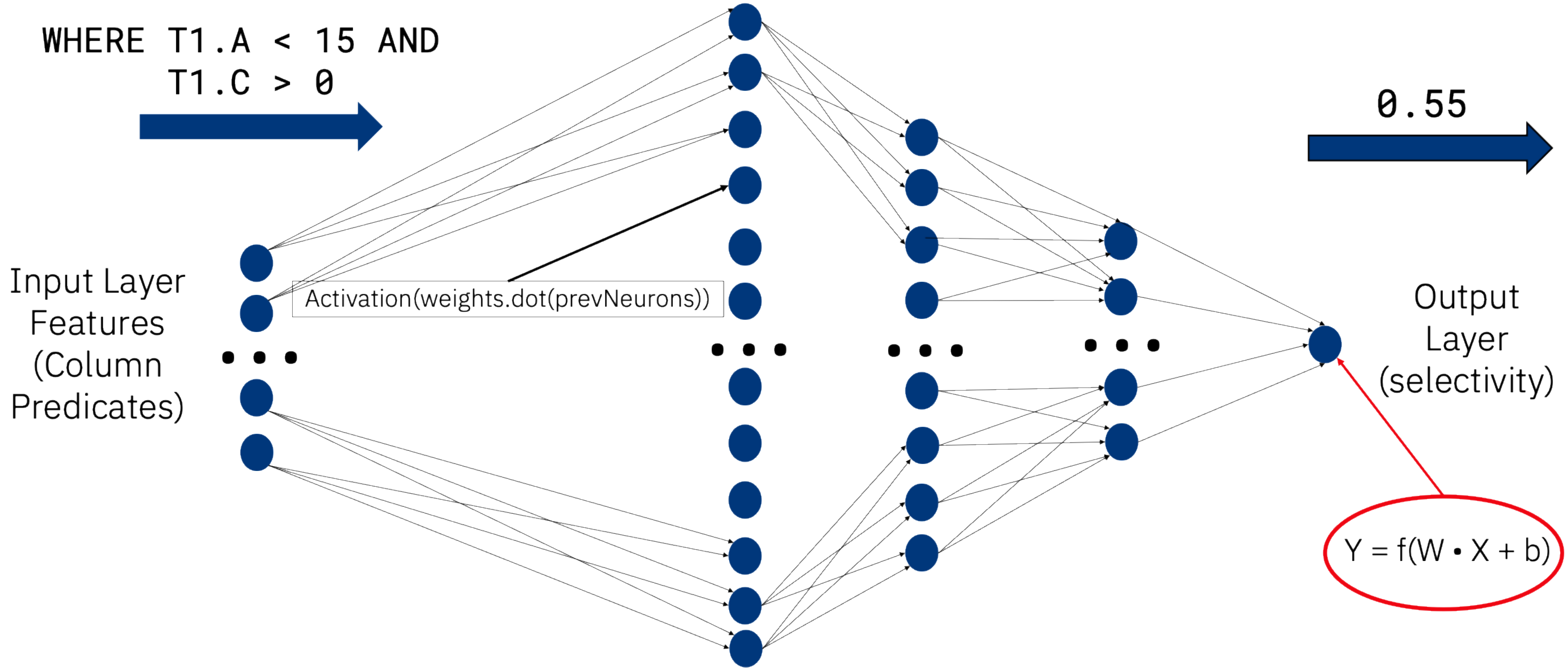




Training the Model



How Does a Neural Network Machine Learning Cardinality Estimation Model Work?



Retraining a Model

- WHEN
  - With enough data changes, Statistics and Model retraining is triggered.

- HOW
  - Drive model discovery/training again
  - Create a brand-new model instead of fine-tuning an existing model
  - Previously discovered correlated columns are preserved
  - New correlations are added
  - Retrained model is stored as a new record in the catalog
  - Old model is still present, we always keep two records for REVERT usage

Using the Model

Query Compiler

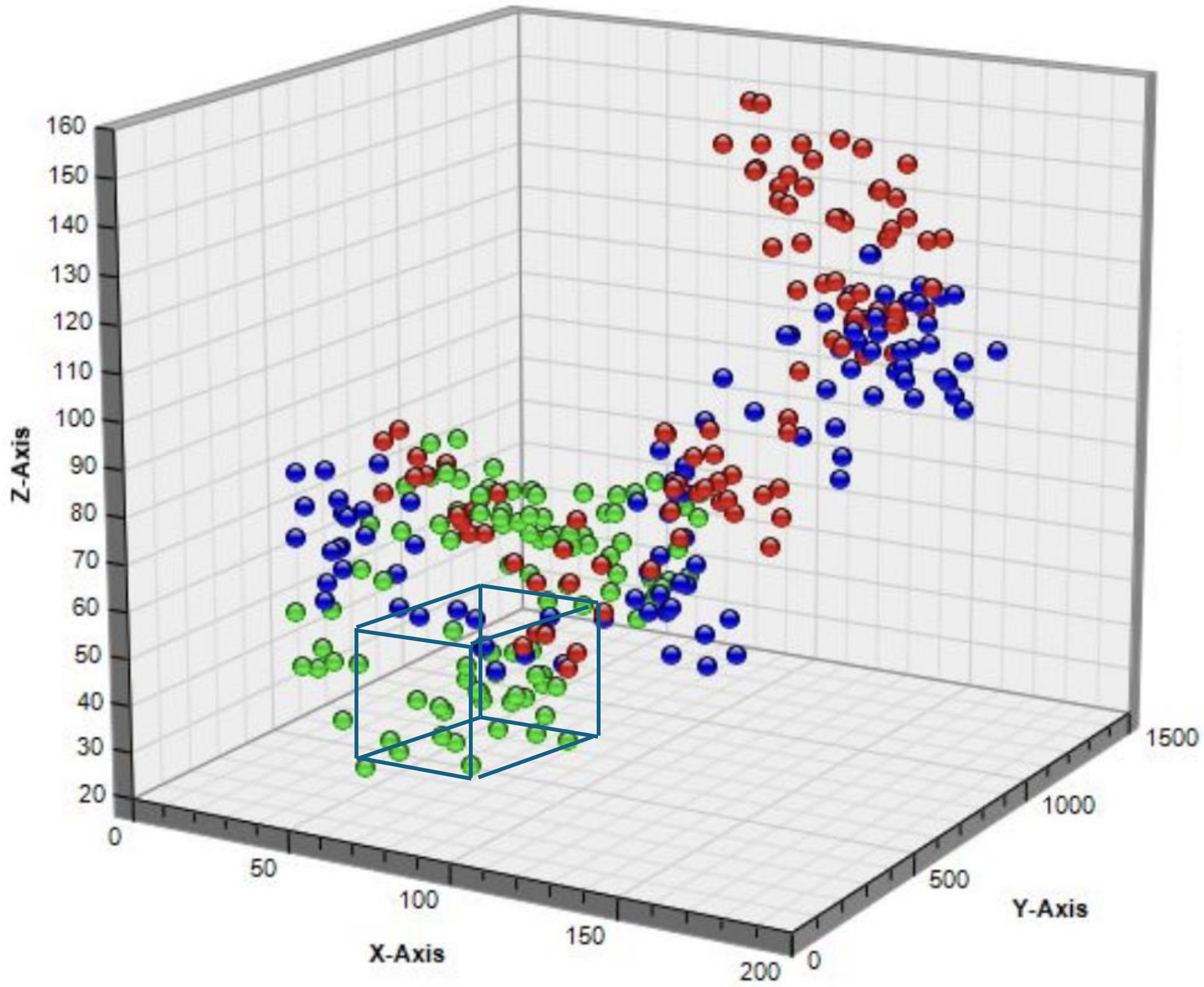
Model Loaded in the Cache

Optimizer Calls Model API

Model Predicts Selectivity

Optimizer Uses the Selectivity

Model Visualization

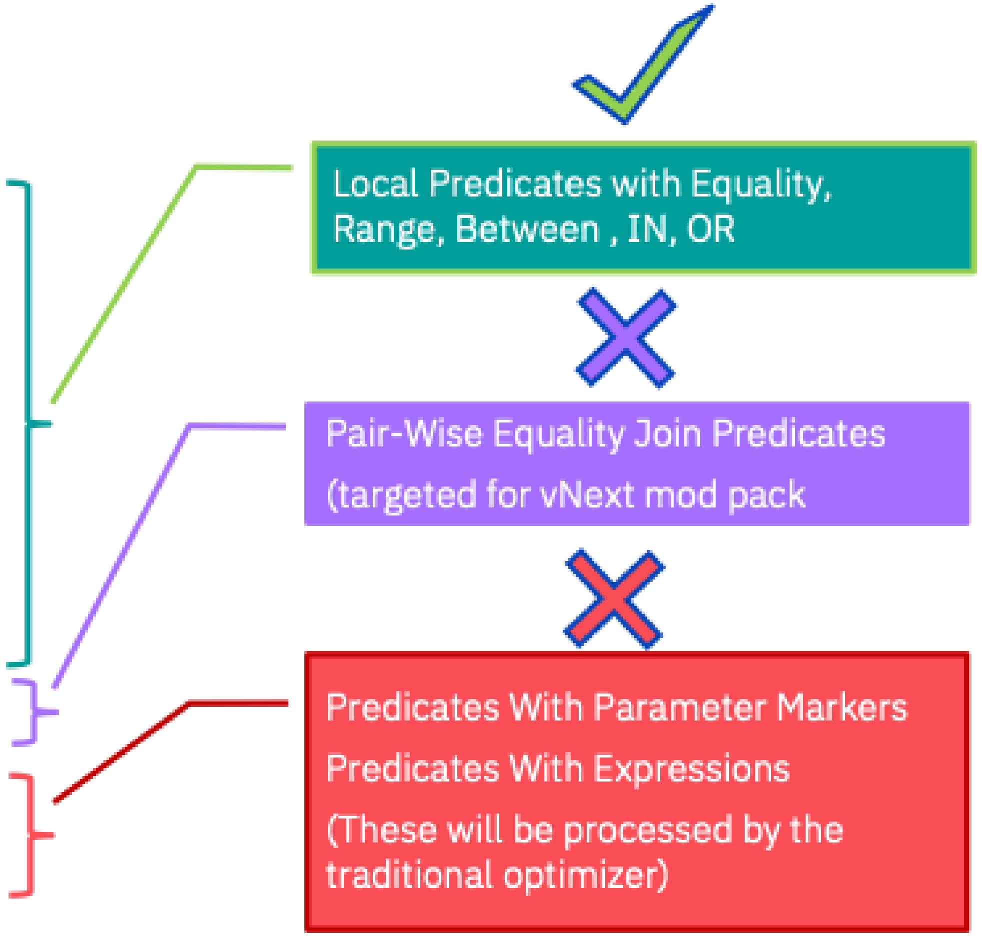


Predicate Support

- **Supported: Local predicates with**
  - Equality
  - Range
  - BETWEEN
  - IN
  - OR
  - LIKE with supported patterns such as no wildcards (=) or a trailing wildcard only
- **Not yet supported**
  - Equality join predicates
  - Multi-column and non-equality **join** predicates
  - Predicates with host variables or parameter markers not using REOPT
  - Predicates with expressions around the columns

Predicate Examples

```
SELECT * FROM T1, T2
WHERE
  T1.C1 = ‘abc’ AND
  T1.C6 IN (5, 3, 205) AND
  T1.C2 BETWEEN 5 AND 10 AND
  T2.C3 <= 120 AND
  ((T1.C4 > 5 AND T1.C5 < 20) OR
   (T1.C4 < 2 AND T1.C5 = 100))
AND
  T1.C5 LIKE ‘string%’ AND
  T1.C0 = T2.C0 AND
  T1.C3 = ? AND
  MOD(T1.C4, 10) = 1
```



Where the Model Does Exceptionally Well

```
SELECT
    GUEST_LAST_NAME,
    ARRIVAL_DATE,
    DEPARTURE_DATE
FROM
    HOTEL_DB
WHERE
    (ARRIVAL_DATE <= '2019-12-25' and
     DEPARTURE_DATE >= '2019-12-25') OR
    (ARRIVAL_DATE <= '2018-12-25' and
     DEPARTURE_DATE >= '2018-12-25') OR
    (ARRIVAL_DATE <= '2017-12-25' and
     DEPARTURE_DATE >= '2017-12-25')
```

Correlation between columns involved in **multiple range predicates**

```
SELECT
    GUEST_LAST_NAME,
    ARRIVAL_DATE,
    DEPARTURE_DATE
FROM
    HOTEL_DB
WHERE
    DATE_C BETWEEN
      '2019-08-01' and '2019-08-31' AND
    COMPANY = 'IBM'
```

Correlation between **equality predicates and range predicates**

Storage, Retrieval and Model Information

- New catalog table SYSIBM.SYSAIMODELS
- Catalog cache. Only most recent version of each model is cached
- SYSIBM.SYSDEPENDENCIES. Useful for looking up models based on the table name and vice versa
- Looking up details of the model:

```
SELECT MODELSCHEMA, MODELNAME, CREATE_TIME, TABCOLUMNS, ISENABLED, VERSION
FROM SYSCAT.AIOPT_TABLECARDMODELS
WHERE TABNAME = ‘T1’;
```

MODELSCHEMA	MODELNAME	CREATE_TIME	TABCOLUMNS	ISENABLED	VERSION
SYSIBM	SQL240506160304427566	2024-05-06-16.08.53.301767	C1,C2	1	0
SYSIBM	SQL240506160304427566	2024-05-06-16.03.04.427599	C1,C2	1	1

Turning on the AI Optimizer

- The AI Optimizer is automatically turned on for newly created databases

- For existing databases, the AI optimizer can be turned on as follows:
  - New settings under AUTO_MAINT
    - Automatic maintenance (AUTO_MAINT) = ON
    - Automatic AI maintenance (AUTO_AI_MAINT) = ON
    - AI Optimizer (AUTO_AI_OPTIMIZER) = OFF
    - Automatic Model Discovery (AUTO_MODEL_DISCOVER) = ON
  - Turning on the AI Optimizer
    - db2 update db cfg for <dbname> using AUTO_AI_OPTIMIZER ON

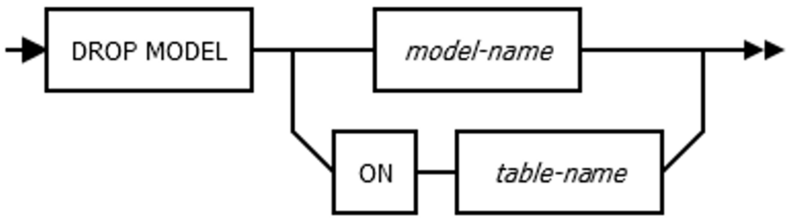
Comparing Estimates with the Traditional Optimizer

- A switch is available to see the difference in the estimates using the model versus the estimates in the traditional optimizer
  - db2set DB2_SELECTIVITY=MODEL_PRED_SEL ON
  - db2set DB2_SELECTIVITY=MODEL_PRED_SEL OFF
- Can be embedded as a guideline or profile to control the use of models on a per query basis
- This is a good way of comparing estimates without dropping a model

DDL : In Case of an Emergency

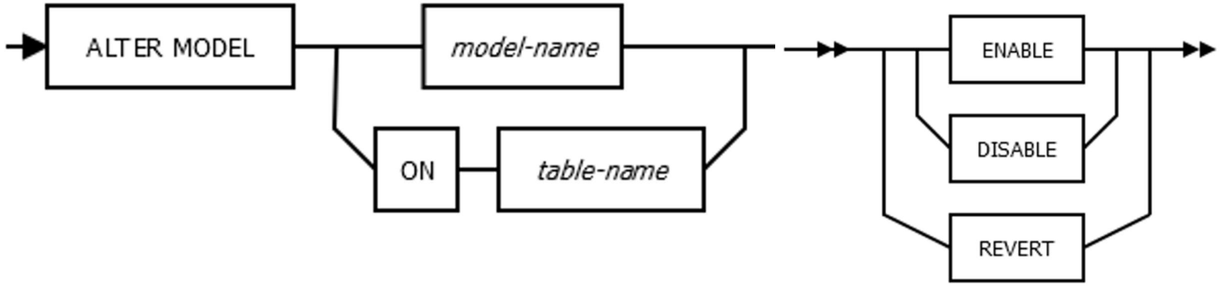
DROP MODEL

- Will drop models



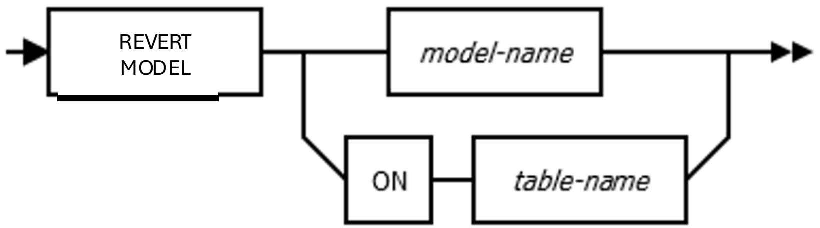
ALTER MODEL

- Will alter the model
- ENABLE/DISABLE controls model training and usage



REVERT MODEL

- Swaps the most recent model with an older model



Entries Added to the Statistics Log

```
2022-03-11-12.06.49.326064-480 I532207E727          LEVEL: Event
…
DISCOVER: TABLE CARDINALITY MODEL : Object name with schema : AT "2022-03-11-12.06.49.325975" : BY
"Asynchronous" : start
OBJECT  : Object name with schema, 34 bytes
MLO_DBCFG_ENG_RANGE.MIXEDDATA_AUTO
IMPACT  : None
DATA #1 : String, 18 bytes
Automatic Runstats

2022-03-11-12.06.49.328033-480 I532935E871          LEVEL: Event
…
DISCOVER: TABLE CARDINALITY MODEL : Object name with schema : AT "2022-03-11-12.06.49.327990" : BY
"Asynchronous" : success
OBJECT  : Object name with schema, 34 bytes
MLO_DBCFG_ENG_RANGE.MIXEDDATA_AUTO
IMPACT  : None
DATA #1 : String, 18 bytes
Automatic Runstats
DATA #2 : String, 113 bytes
TABLE CARDINALITY MODEL ON "MLO_DBCFG_ENG_RANGE"."MIXEDDATA_AUTO" ON COLUMNS ("DISTCOL", "INTCOL1", "INTCOL2")
```

Entries Added to the Statistics Log (continued)

```
2022-03-11-12.06.49.329270-480 I534521E882          LEVEL: Event
…
TRAIN   : TABLE CARDINALITY MODEL : Object name with schema : AT "2022-03-11-12.06.49.329230" : BY "Asynchronous" : start
OBJECT  : Object name with schema, 34 bytes
MLO_DBCFG_ENG_RANGE.MIXEDDATA_AUTO
IMPACT  : None
DATA #1 : String, 18 bytes
Automatic Runstats
DATA #2 : String, 113 bytes
TABLE CARDINALITY MODEL ON "MLO_DBCFG_ENG_RANGE"."MIXEDDATA_AUTO" ON COLUMNS ("DISTCOL", "INTCOL1", "INTCOL2")

2022-03-11-12.06.54.367094-480 I535404E742          LEVEL: Event
…
TRAIN   : TABLE CARDINALITY MODEL : Object name with schema : AT "2022-03-11-12.06.54.367035" : BY "Asynchronous" :
success
OBJECT  : Object name with schema, 34 bytes
MLO_DBCFG_ENG_RANGE.MIXEDDATA_AUTO
IMPACT  : None
DATA #1 : String, 18 bytes
Automatic Runstats
DATA #2 : String, 1174 bytes
Model metrics: Rating: 3 (Very good), Table samples: 33 (33), Flags: 0x0, Training time: 5059 (1/20/11/0), Validation MSE:
0.000424, Accuracy bucket counts: 0,791,4665,1213,0, Accuracy bucket means: 0.000000,-1.244713,-0.080033,1.228198,0.000000
Table column cardinalities: 10,10,10
Sample column cardinalities: 10,10,10
Sample column mappings: 10,10,10
Column flags: 00000000,00000000,00000000
Base algorithm metrics: Training metric: 0.000413, Validation metric: 0.000426, Previous validation metric: 0.000428, Pre-
training validation metric: 0.001477, Used training iterations: 21, Configured training iterations: 39, Training set size:
66695, Pre-training time: 430, Training time: 2544, Accuracy bucket counts: 0,878,4578,1213,0, Accuracy bucket means:
0.000000,-1.232078,-0.063045,1.228198,0.000000
Low selectivity algorithm metrics: Training metric: 0.000000, Validation metric: 0.000020, Previous validation metric:
0.000000, Pre-training validation metric: 0.000002, Used training iterations: 36, Configured training iterations: 44,
Training set size: 38031, Pre-training time: 163, Training time: 2483, Accuracy bucket counts: 2,5,2910,0,0, Accuracy
bucket means: -2.000233,-1.999801,0.058431,0.000000,0.000000
```

Model Policies

- Configure which tables can have models
- Model policies will still allow automatic statistics collection
- Model policies do not affect model retraining
- Auto-runstats policies will impact model discovery and training

```
<Db2AutoAiOptPolicy>
  <ModelDiscoveryTableScope modelType='TableCardModel'>
    <FilterCondition>
       WHERE (TABSCHEMA,TABNAME) NOT IN (VALUES ’TPCDS’,’STORE_SALES’))
    </FilterCondition>
  </ModelDiscoveryTableScope>
</Db2AutoAiOptPolicy>
```

EXPLAIN (db2exfmt)

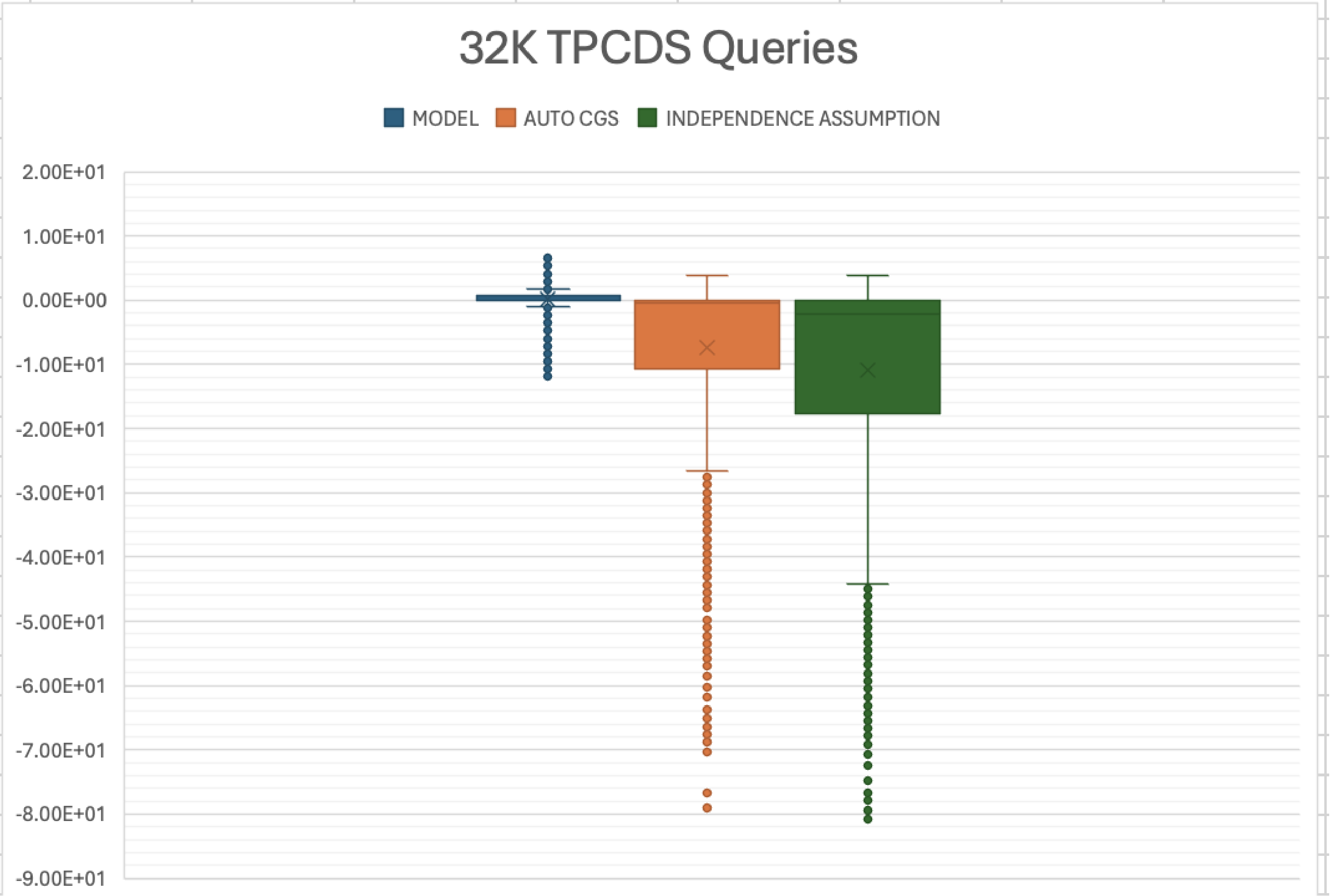
- Source for the cardinality estimation is a model
- the list of predicates the model computed the combined selectivity for
- Model information will also be listed in the “objects used” and includes the columns the model was trained on
- Each area will also show the model schema and name

```
4) TBSCAN: (Table Scan)
           Predicates:
           ----------
           8) Sargable Predicate,
                      Comparison Operator: Less Than or Equal (<=)
                      Subquery Input Required: No
                      Filter Factor: 0.934924
                      Filter Factor Source: SYSIBM. SQL240913170855940498

                      Predicate Text:
                      --------------
                      …

           Table Cardinality Model Predicates:
           ----------------------------------
           Model:     SYSIBM.SQL240913170855940498
           Predicates:
                      1) (Q3.BILL_AMT1 <= 746814)
                      2) (150 <= Q3.BILL_AMT1)
                      3) (Q3.PAY_2 <= 2)
                      4) (0 <= Q3.PAY_2)
                      …
Objects Used in Access Plan:
---------------------------
           Schema:   DEMO
           Name:     CREDIT_HISTORY_DATA
           Type:     Table
                      ...
                      Model Schema: SYSIBM
                      Model Name: SQL240913170855940498
                      Columns in model:
                                 BILL_AMT1
                                 PAY_2
                                 …
```

Cardinality Estimation Accuracy

Closer to 0
Is better

Thinner Box
Plot is better

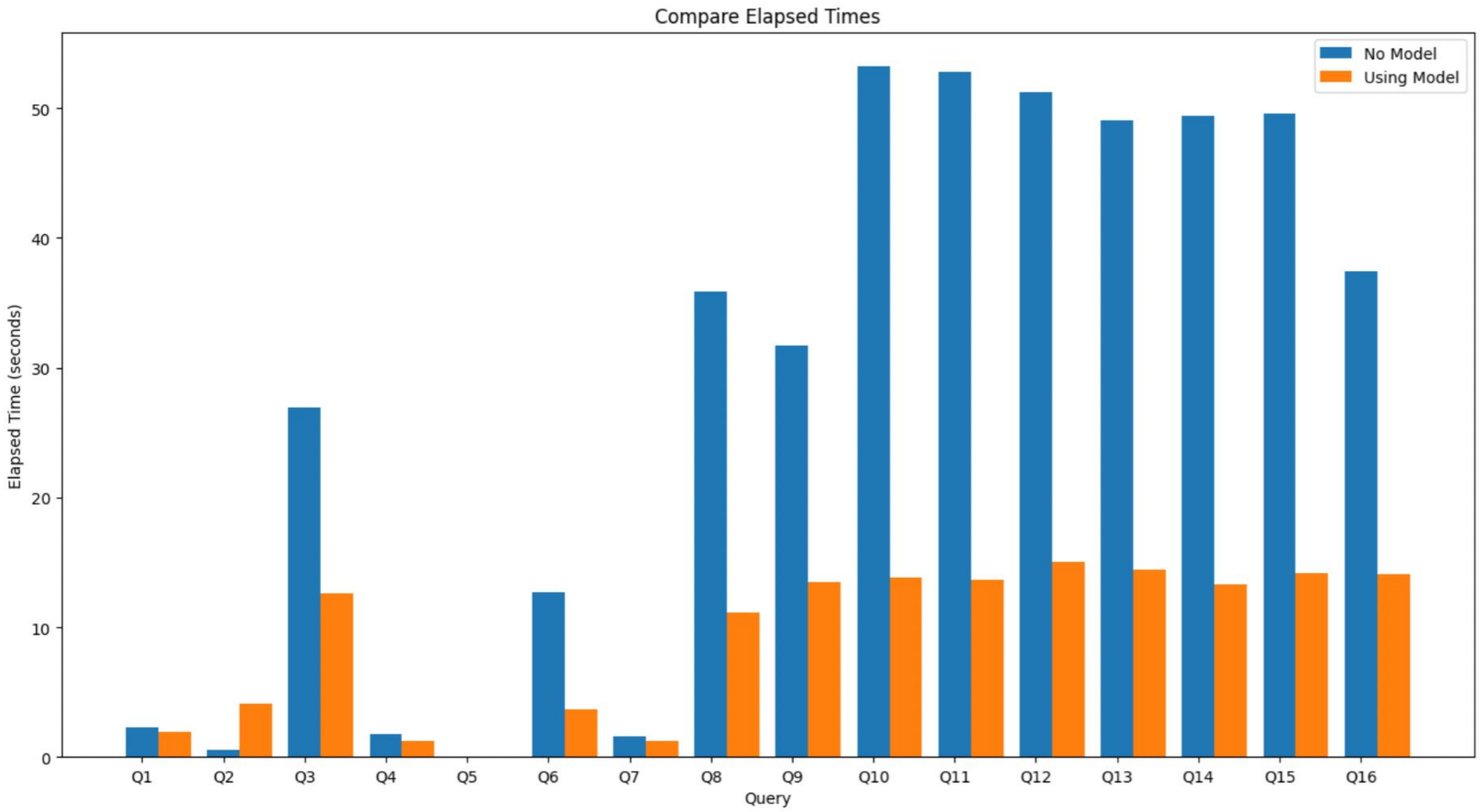


Real world Problem Queries

3X faster in scenarios simulated in-house

The average benefit will depend on the workload and prior tuning

The goal is to get reliable performance

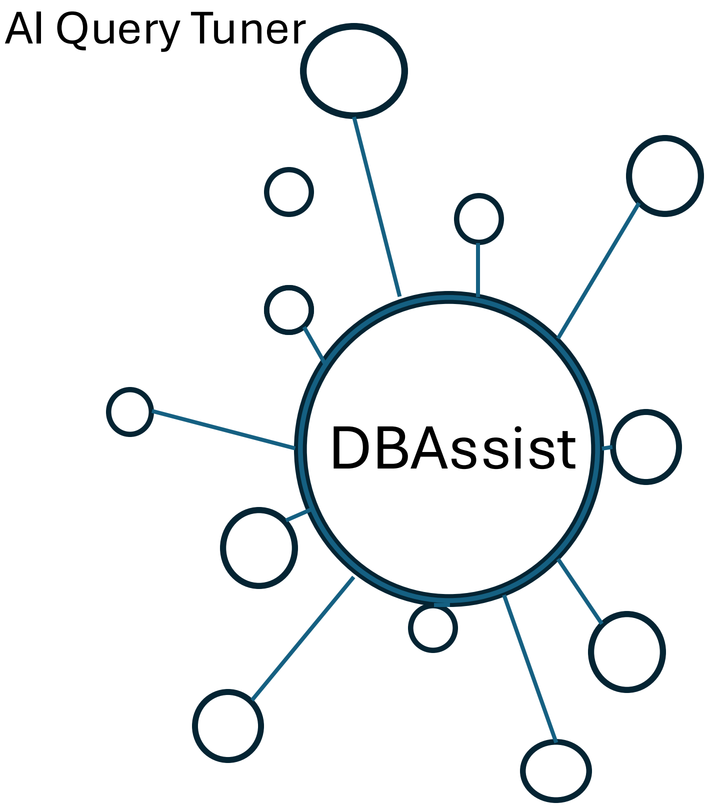


(AI) Query Tuner

What is the AI Query Tuner?

- Simplifying database and query performance tuning is critical with a shortage of highly skilled DBAs increasing database sizes query complexity
- The DBA Assistant (DBAssist) is an AI-powered tool designed for DBAs that provides insights and smart recommendations through a natural language chat interface.
- DBAssist is trained on a wide knowledge base and database telemetry, to streamline information retrieval and quickly help answer questions and troubleshoot problems on your database systems.
- The Query Tuner is a recommendation agent within the engine available for DBAssist to consume.

DBAssist



Tuning for Performance

- Plan to develop AI models to generate recommendations for
  - Single query analysis
  - Workload analysis

- Near term – Explain Analyzer Model
  - Current workload recommendation functionality available through the db2 advisor, such as with index recommendations, will be leveraged as a first step.