Tridex Db2 LUW
September 19th
2024

The AI Query Optimizer Features in Db2 Version 12.1

Calisto Zuzarte

IBM

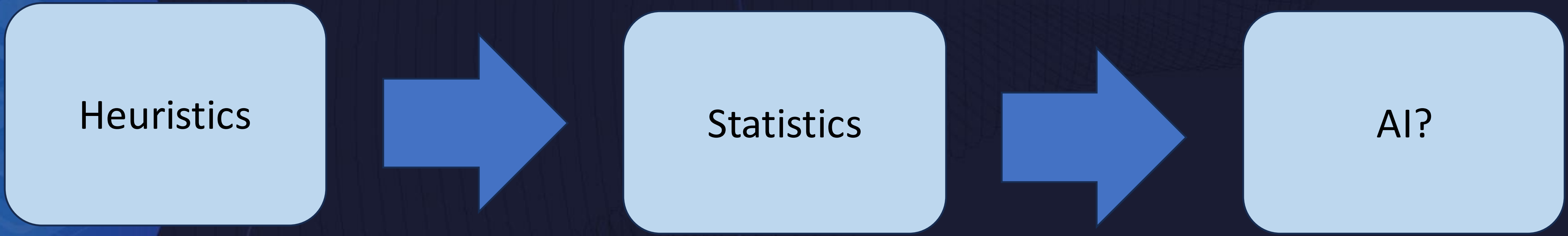
Platform: LUW

Agenda

- Motivation
  - Db2 Query Optimization
  - AI Optimizer versus the “Legendary” Optimizer
  - AI Optimizer Goals
- V12 Phase 1 Functionality
  - Model training
  - Model prediction
  - Model storage and retrieval
  - Model externals
- Results

Motivation

Evolution of Query Optimization

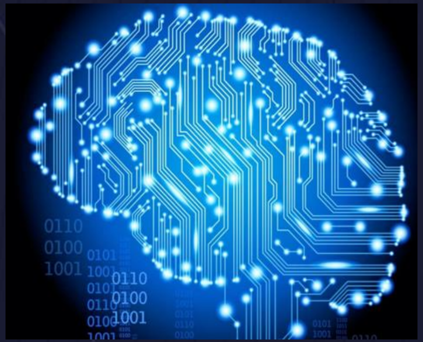


Heuristics

Statistics

AI?

Artificial Intelligence (AI), Machine Learning (ML), Neural Networks (NN)



Artificial Intelligence 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

The Db2 Optimizer

- Generates alternative access subplans
- Estimates cost of each subplan
- Selects the cheapest overall plan
- Costing depends heavily on cardinality estimates

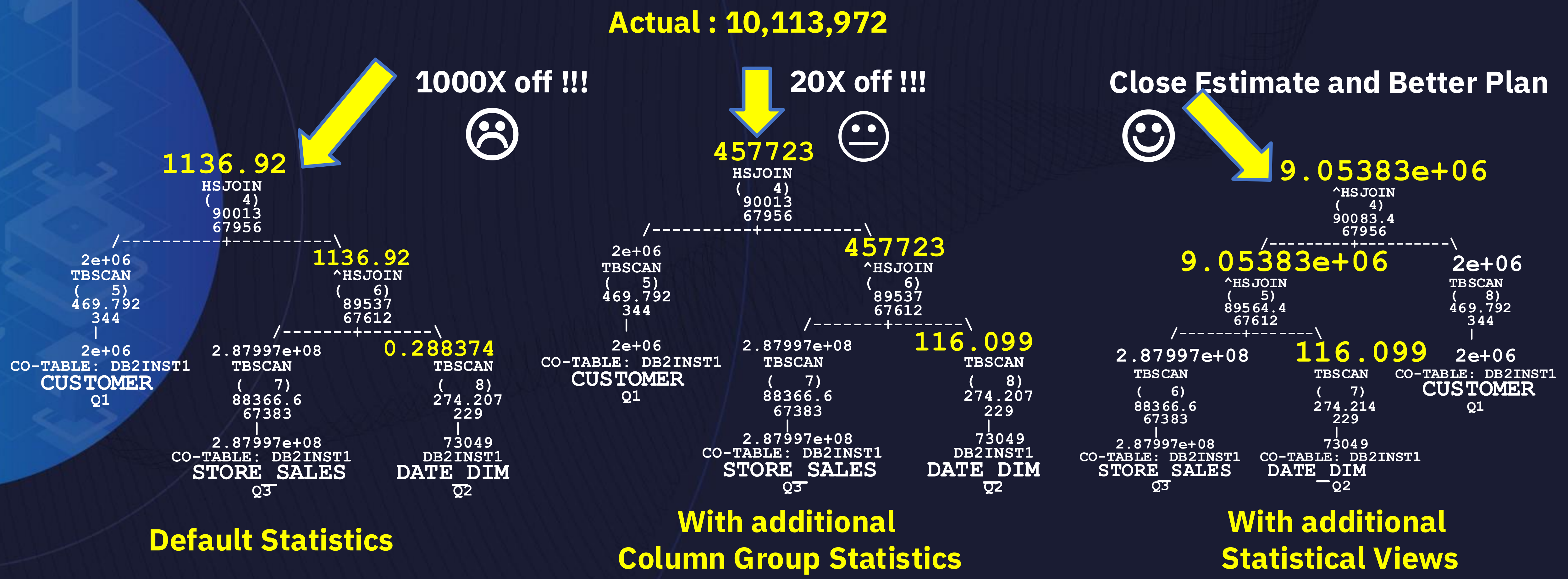
- What is cardinality estimation?

Cardinality Estimation

- Cardinality is the number of rows input to or output from an operator
- Generally reduced by predicates (increased with expanding joins)
- 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

1000X off !!! ☹

20X off !!! 😐

Close Estimate and Better Plan ☺

```
                  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

Can AI do Better?

Optimizer Challenges

- Performance Stability
- Development Effort
- Tuning Effort

AI Query Optimizer Goals

- Automate Everything
- Achieve Reliable Performance
- Simplify Optimizer Development

Benefits

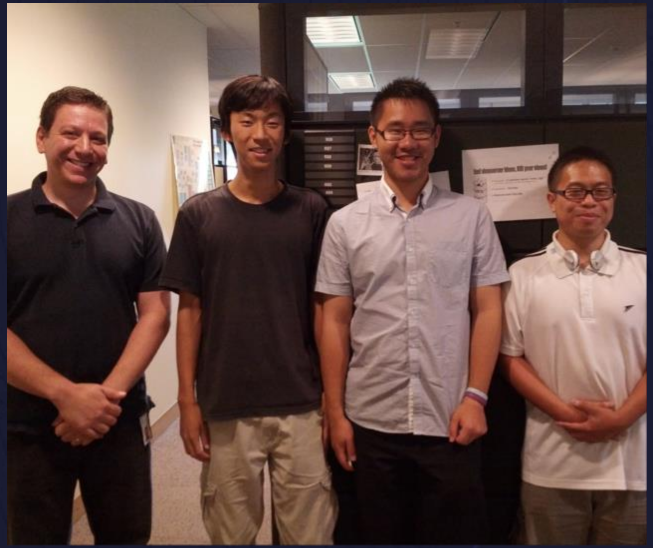
- Adapt to User Data
- Adapt to User Workloads
- Learn from Optimizer and Runtime feedback

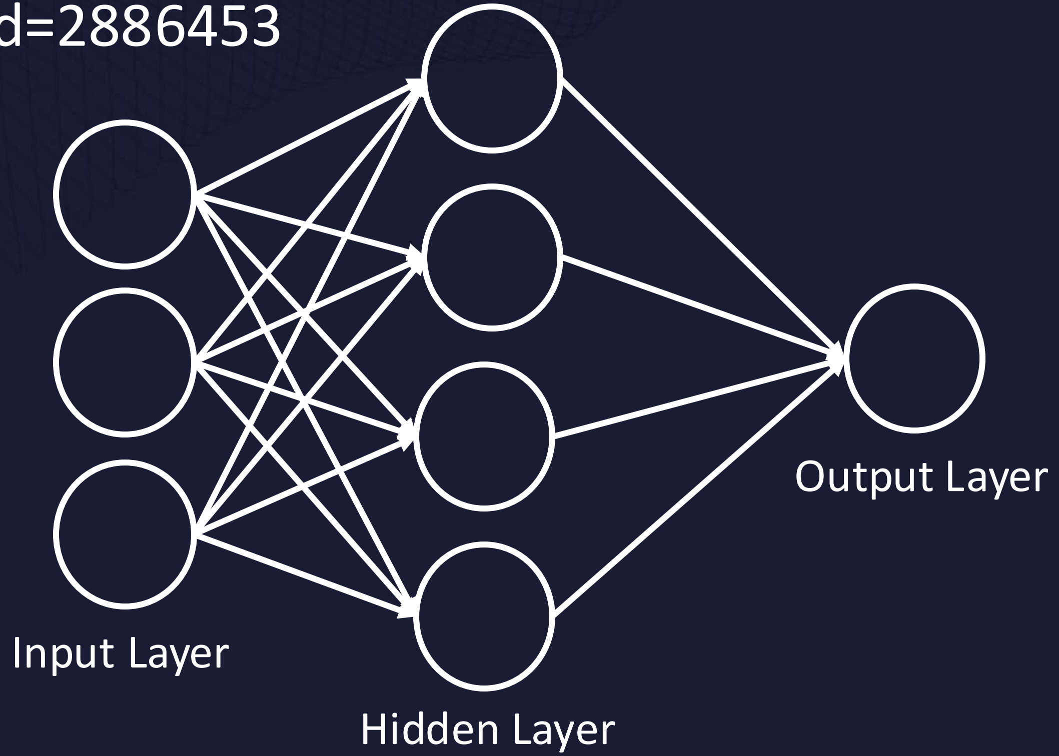
Infuse AI Gradually

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

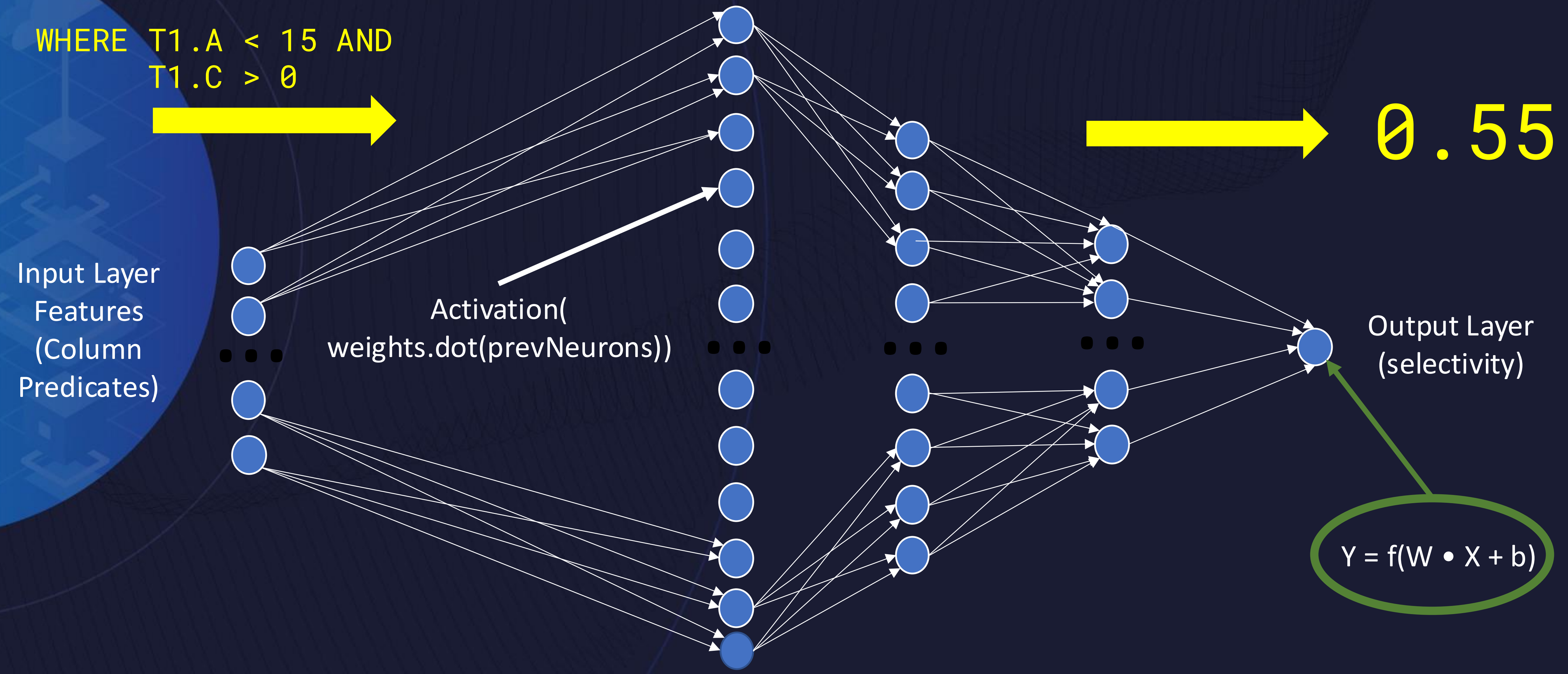
Our First Prototype in 2013
Research Paper Published in 2015

- "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

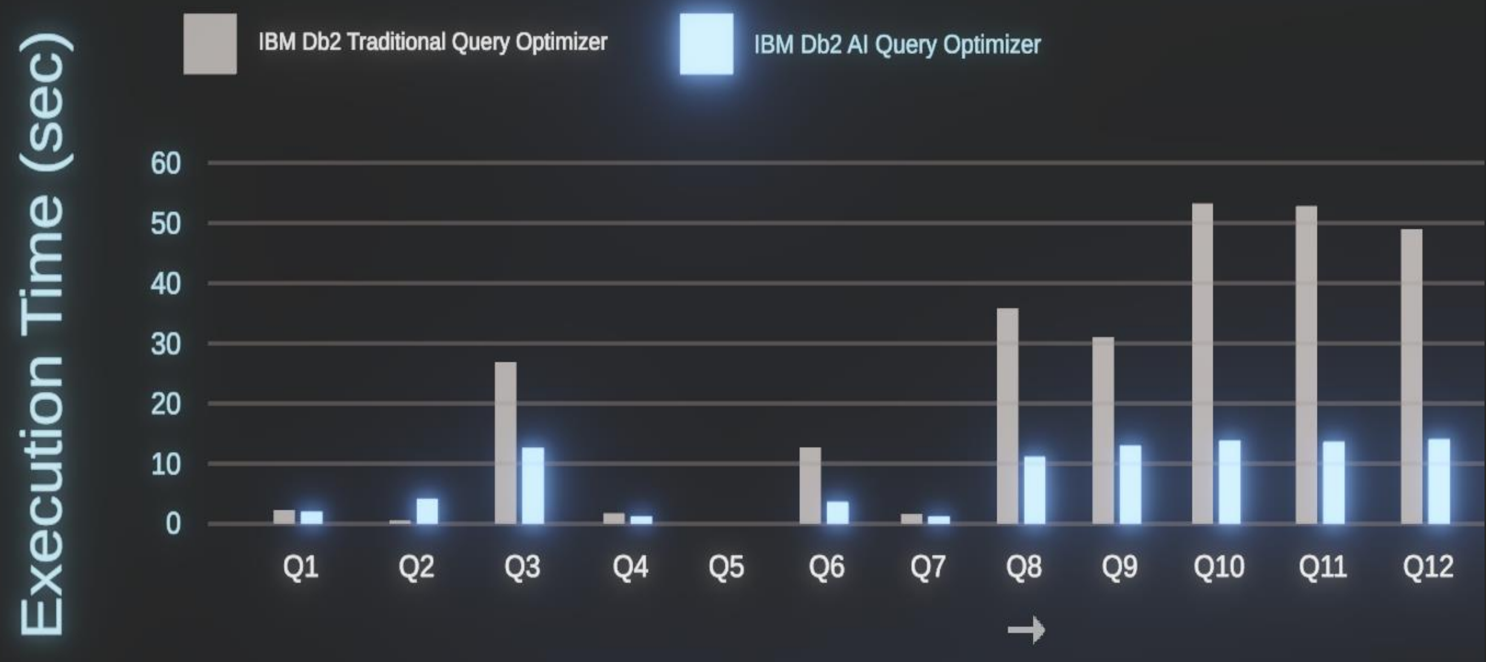


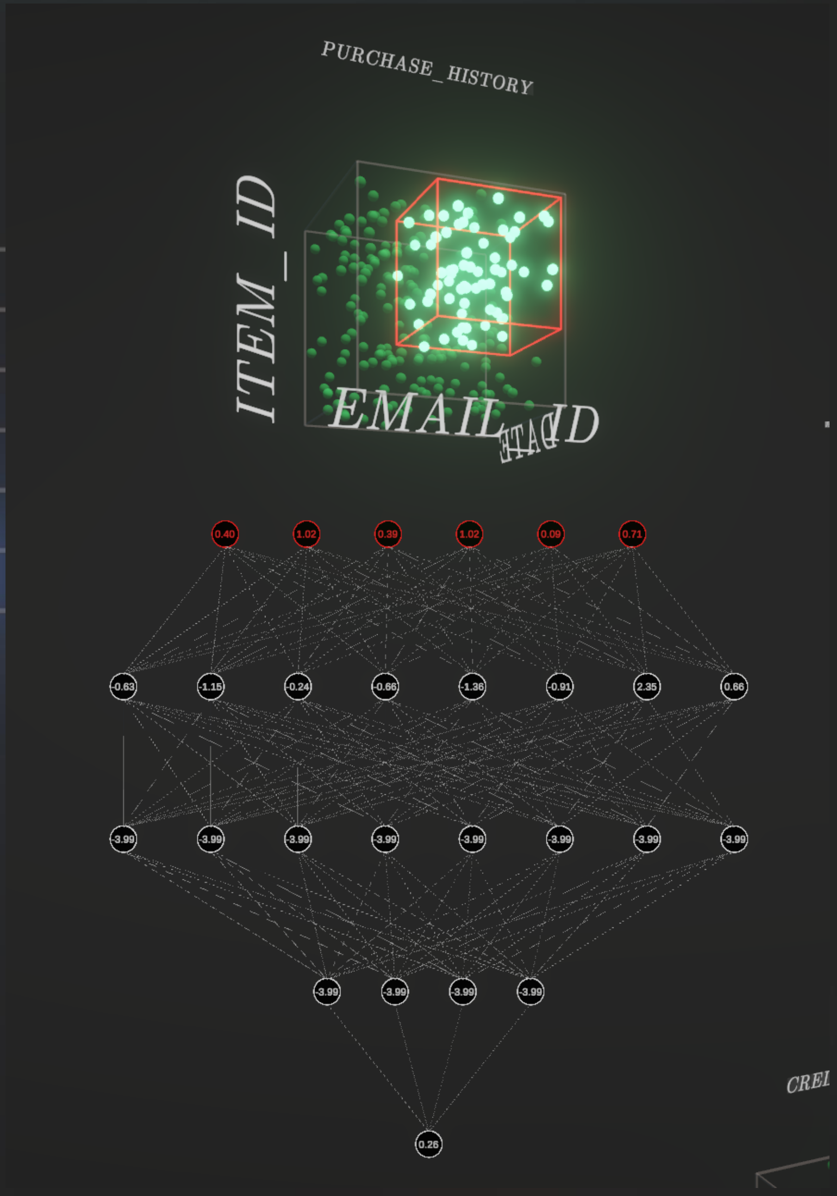


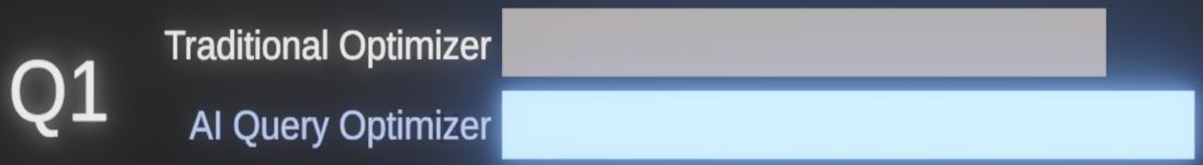
How Do Model Cardinality Estimates Work?



The Neural Networks Powering the Db2 AI Query Optimizer







AI Query Optimizer in V12

AI Optimizer Coverage in Phase 1 in V12

- Cardinality estimation for a subset of local predicates using an ML model
- Model training through auto-runstats
- Model prediction during query optimization
- Model policies
- Explain support
- DDL to disable/enable, revert, and drop models

Productization of the AI Query Optimizer

- SYSAIMODELS Catalog + SYSCAT Views
- Security / Access Control
- Audit
- Better Configuration Management
- Appropriate Error Messages
- Dependency Management

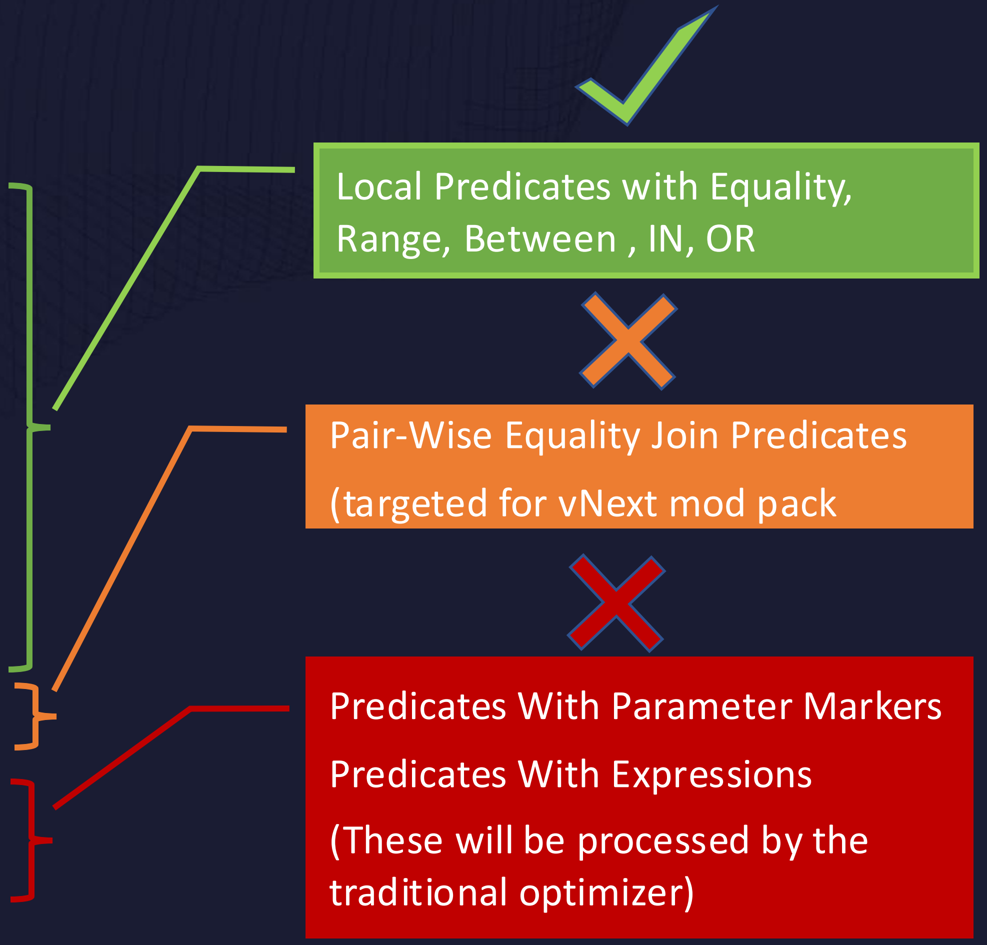
- Explain Support (Mod Pack)
- Activity Logging
- DDL / Model Policy for Better Model Management
- Increased Accuracy and Number of Columns in the Model
- Slightly Larger Model Size 20-30 Kb ➔ 20-60 Kb
- Improved Training Time Avg: 60s ➔ 40s

Predicate Support

- Supported
  - Local predicates with:
    - Equality
    - Range
    - BETWEEN
    - IN
    - OR
    - LIKE with supported patterns such as no wildcards (=) or trailing wildcard (BETWEEN
- 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

Examples of Predicates Supported or Not Yet Supported in vNext

```
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
```



✓

Local Predicates with Equality, Range, Between , IN, OR

✗

Pair-Wise Equality Join Predicates

(targeted for vNext mod pack

✗

Predicates With Parameter Markers

Predicates With Expressions

(These will be processed by the traditional optimizer)

Interesting Scenarios

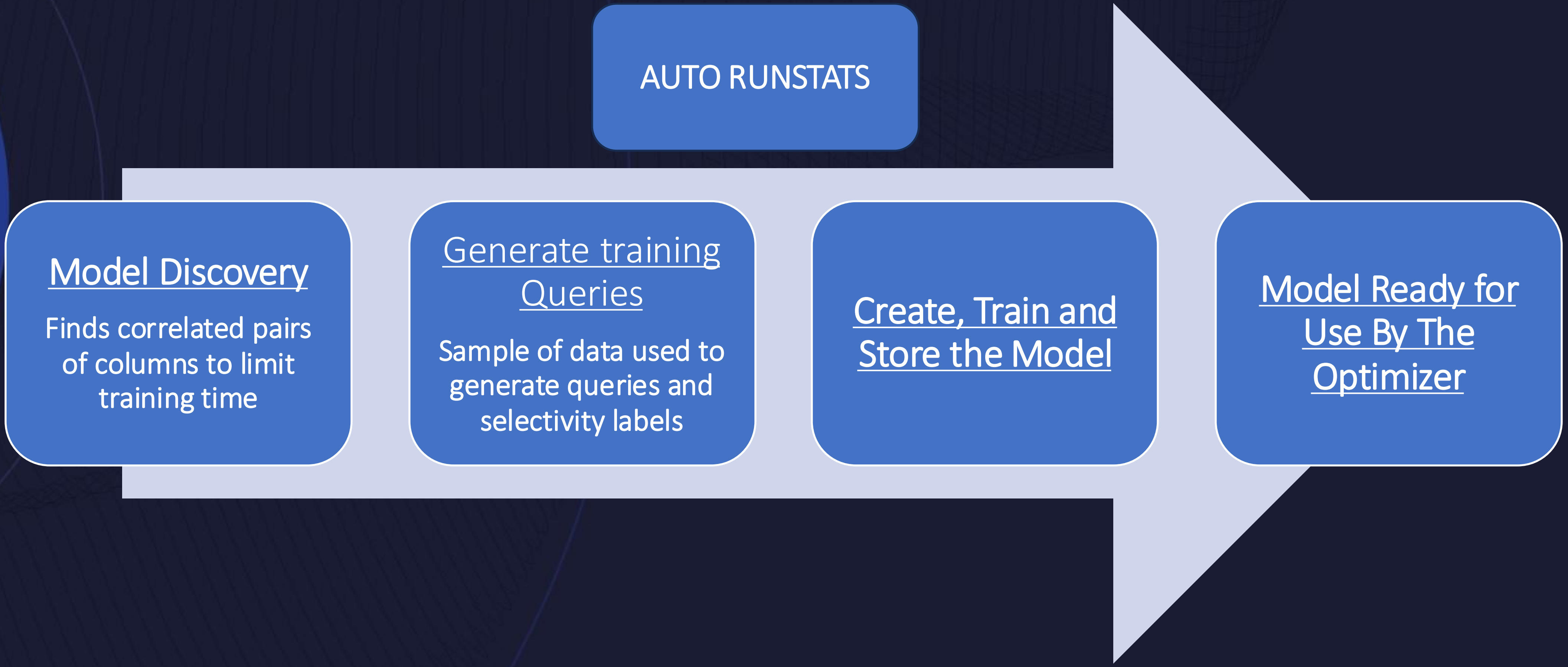
- Correlation between columns with multiple range predicates

  -
    ```
    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 equality predicates and range predicates

  -
    ```
    SELECT GUEST_LAST_NAME, ARRIVAL_DATE, DEPARTURE_DATE
    FROM
       HOTEL_DB
    WHERE
       DATE_COL BETWEEN  ‘2019-08-01’ and  ‘2019-08-31’) AND
       COMPANY = ‘IBM’
    ```

Training the Model (1/2)



AUTO RUNSTATS

Model Discovery

Finds correlated pairs of columns to limit training time

Generate training Queries

Sample of data used to generate queries and selectivity labels

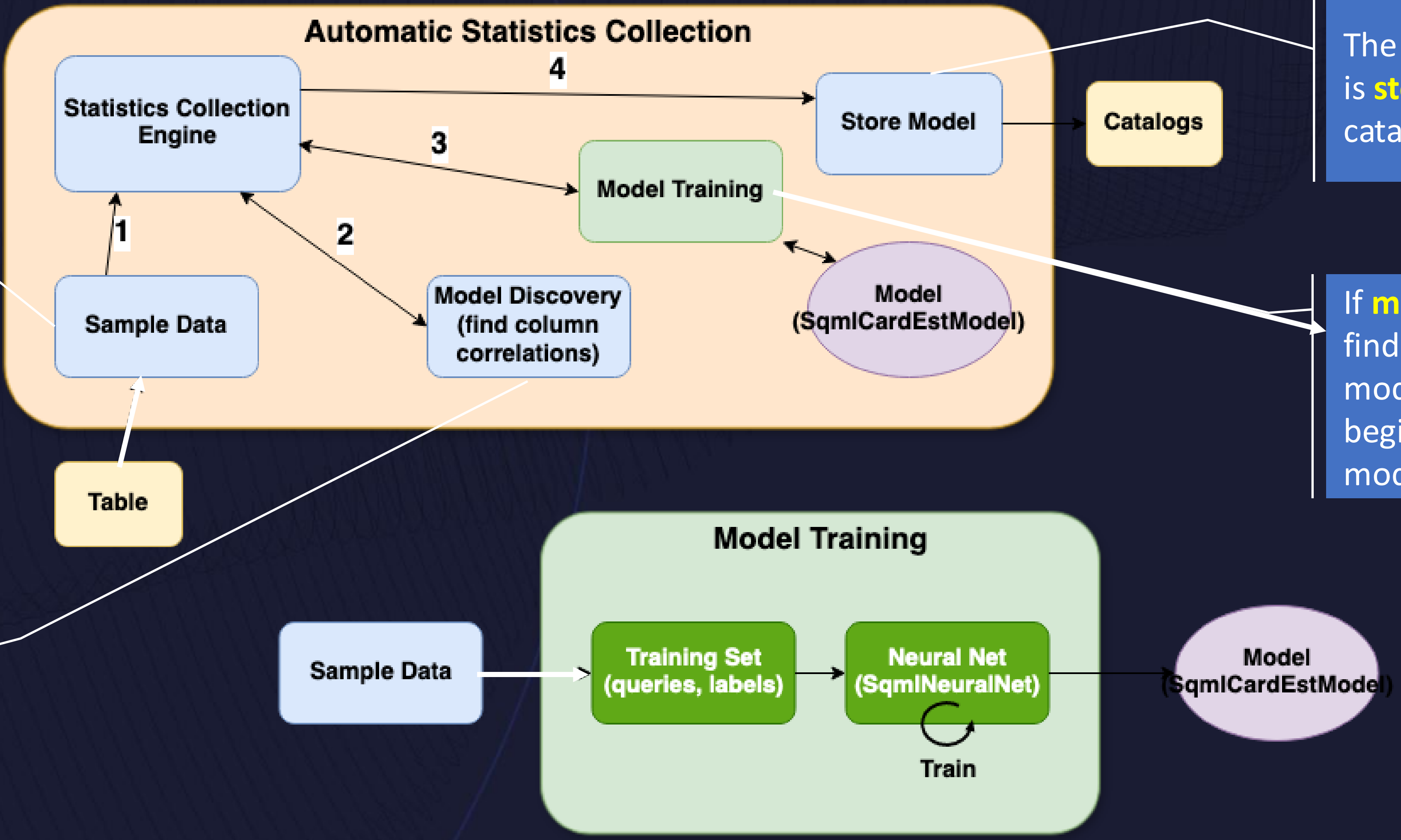
Create, Train and Store the Model

Model Ready for Use By The Optimizer

Training the Model (2/2)

During automatic statistics collection, **sample data** is retrieved from the table.

Model discovery finds correlated columns using the sample data

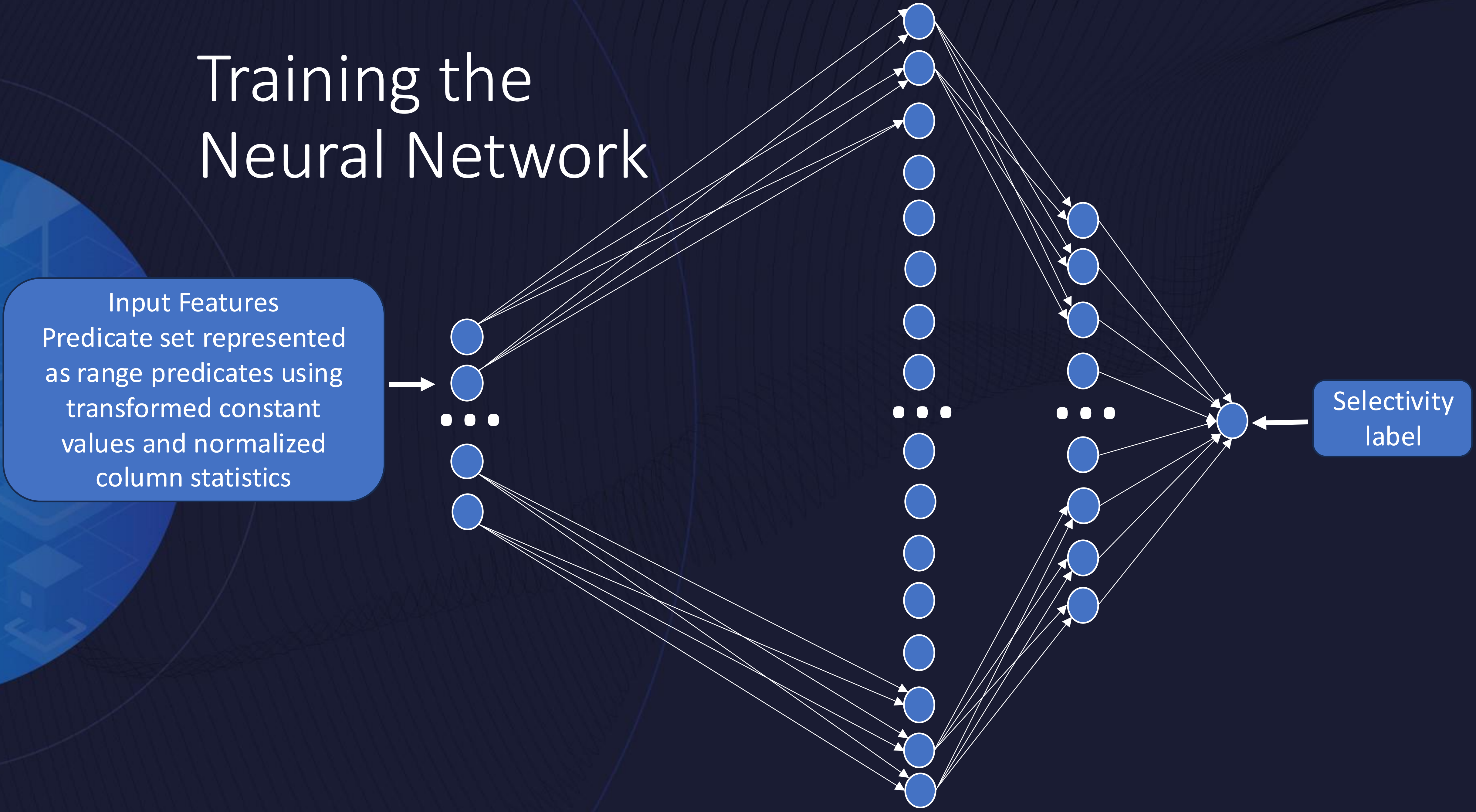


Automatic Statistics Collection
Statistics Collection Engine
4
Store Model
Catalogs
3
Model Training
1
2
Sample Data
Model Discovery (find column correlations)
Model (SqmlCardEstModel)
Table
Model Training
Sample Data
Training Set (queries, labels)
Neural Net (SqmlNeuralNet)
Train
Model (SqmlCardEstModel)

The trained model is **stored** in the catalogs

If **model discovery** finds correlations, model training begins to create a model

Training the Neural Network



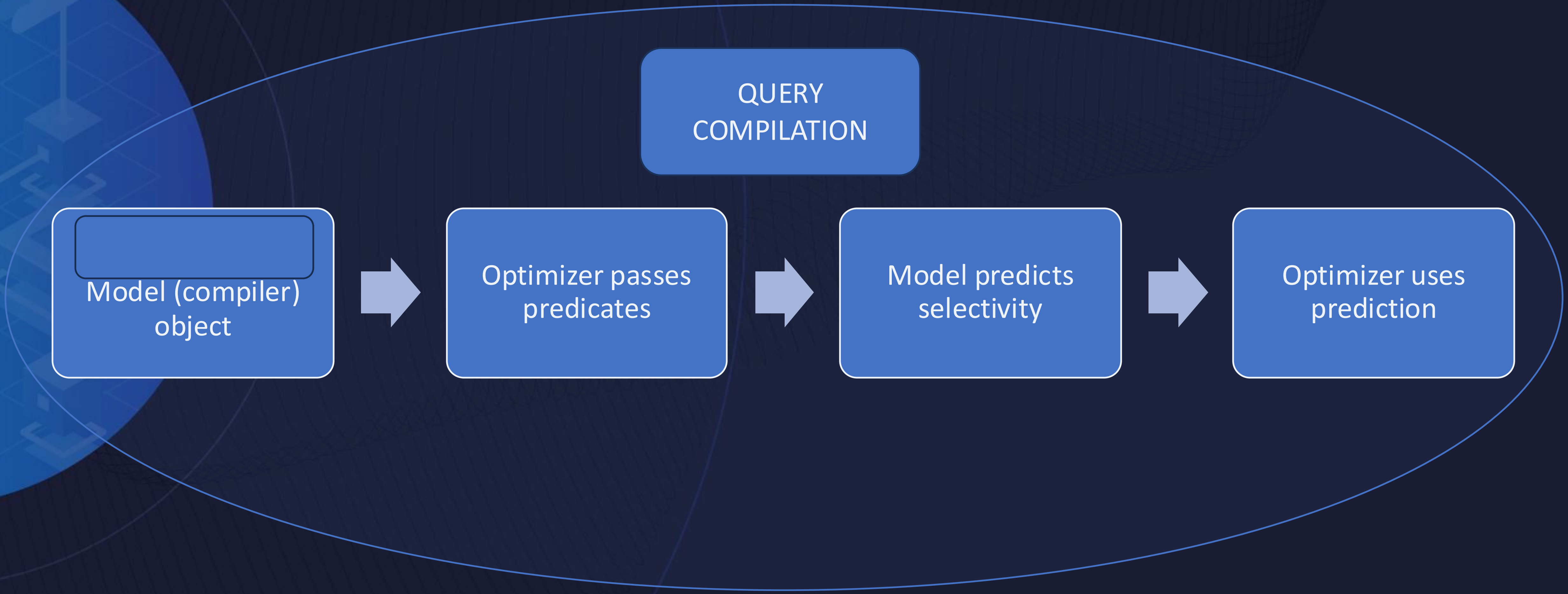
Input Features
Predicate set represented as range predicates using transformed constant values and normalized column statistics

Selectivity label

Retraining a Model

- When
  - Enough data change
  - Trigger stats collection
- 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 (1/1)



QUERY COMPILATION

Model (compiler) object

Optimizer passes predicates

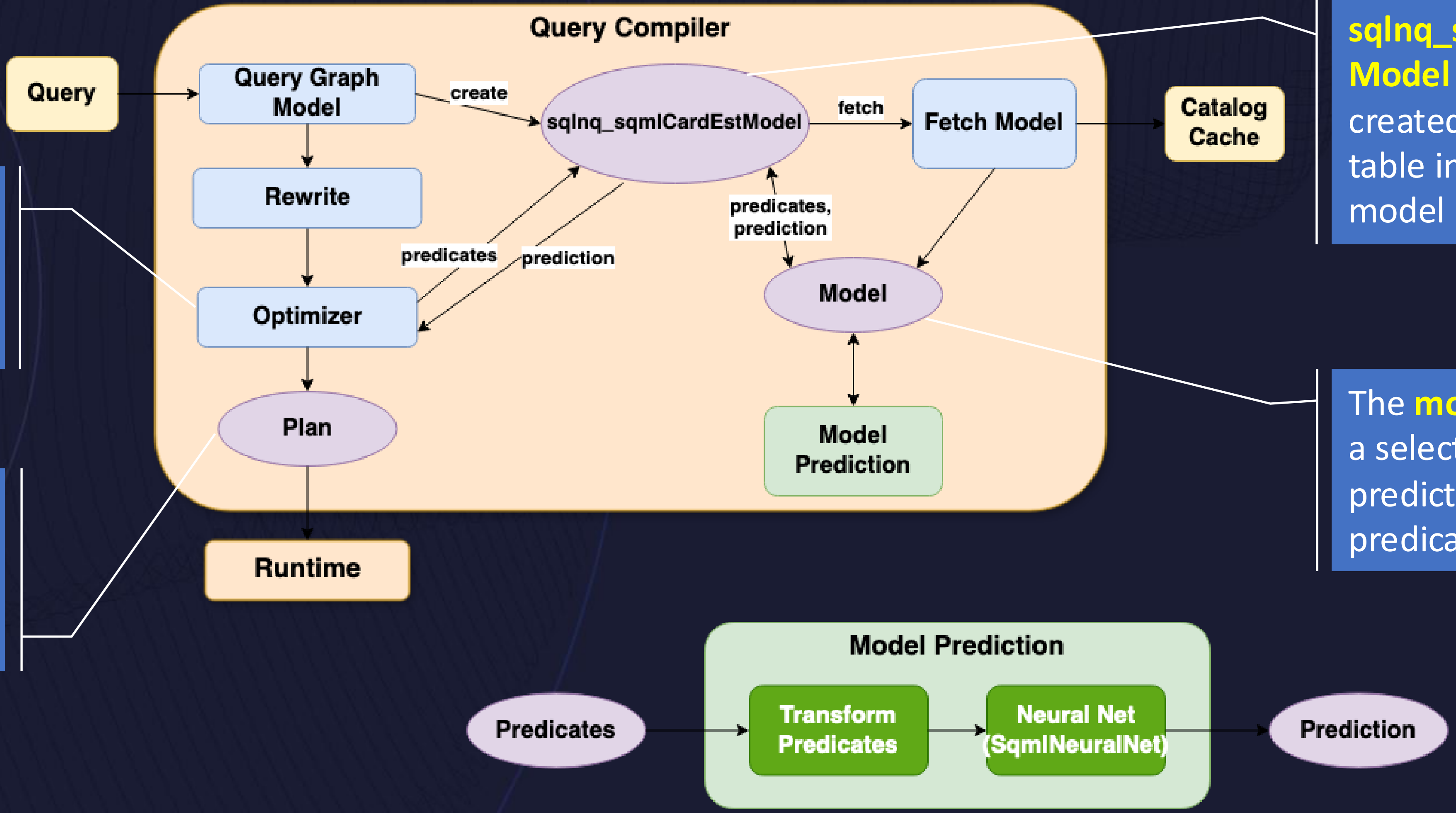
Model predicts selectivity

Optimizer uses prediction

Using the Model (2/2)

The **Optimizer** sends eligible predicates to the model

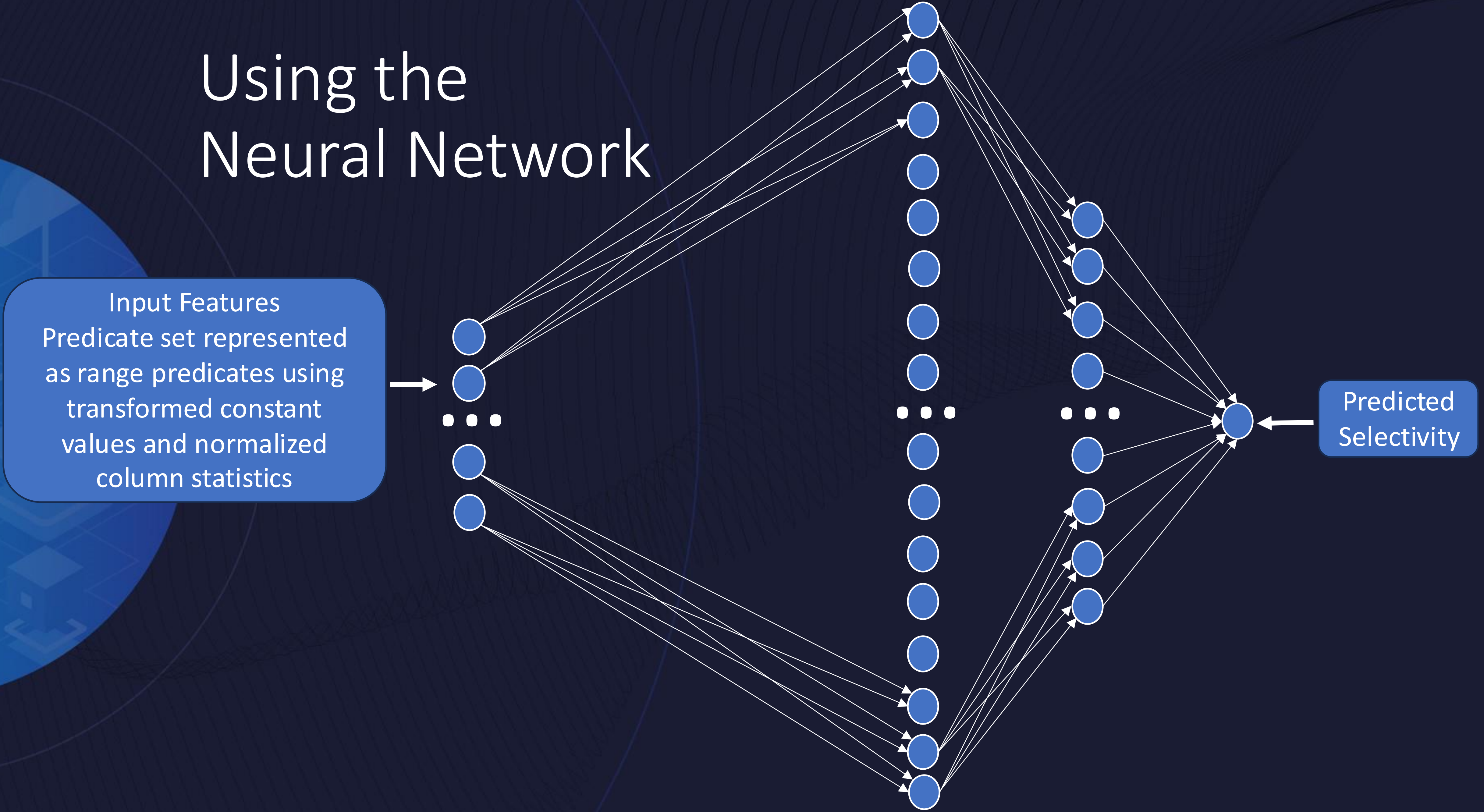
The **Optimizer** uses the prediction for cardinality estimation



sqlnq_sqmlCardEst Model object created for each table in query with model

The **model** obtains a selectivity prediction for the predicate set

Using the Neural Network

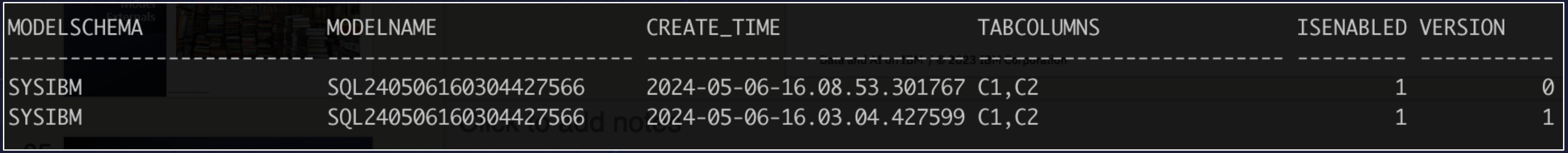


Input Features
Predicate set represented as range predicates using transformed constant values and normalized column statistics

Predicted Selectivity

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

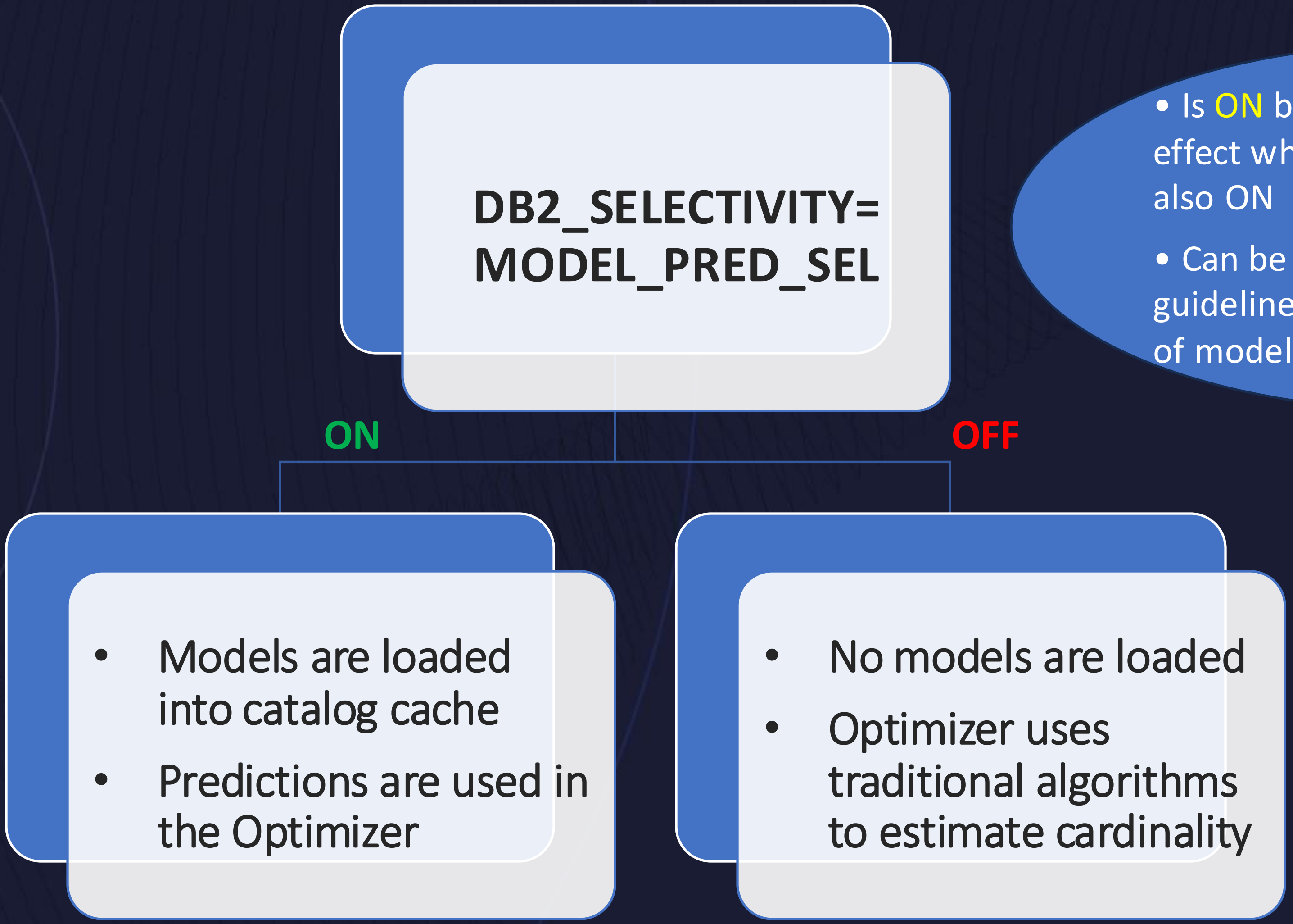
Automatic maintenance	(AUTO_MAINT) = ON
Automatic AI maintenance	(AUTO_AI_MAINT) = ON
AI Optimizer	(AUTO_AI_OPTIMIZER) = OFF
Automatic Model Discovery	(AUTO_MODEL_DISCOVER) = ON

  - Turning on the AI Optimizer
    - db2 update db cfg for <dbname> using AUTO_AI_OPTIMIZER ON

Controlling Model Use in the 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
- This is a good way of validating performance without dropping a model

Controlling Model Usage in Optimizer



DB2_SELECTIVITY=MODEL_PRED_SEL

ON

- Models are loaded into catalog cache
- Predictions are used in the Optimizer

OFF

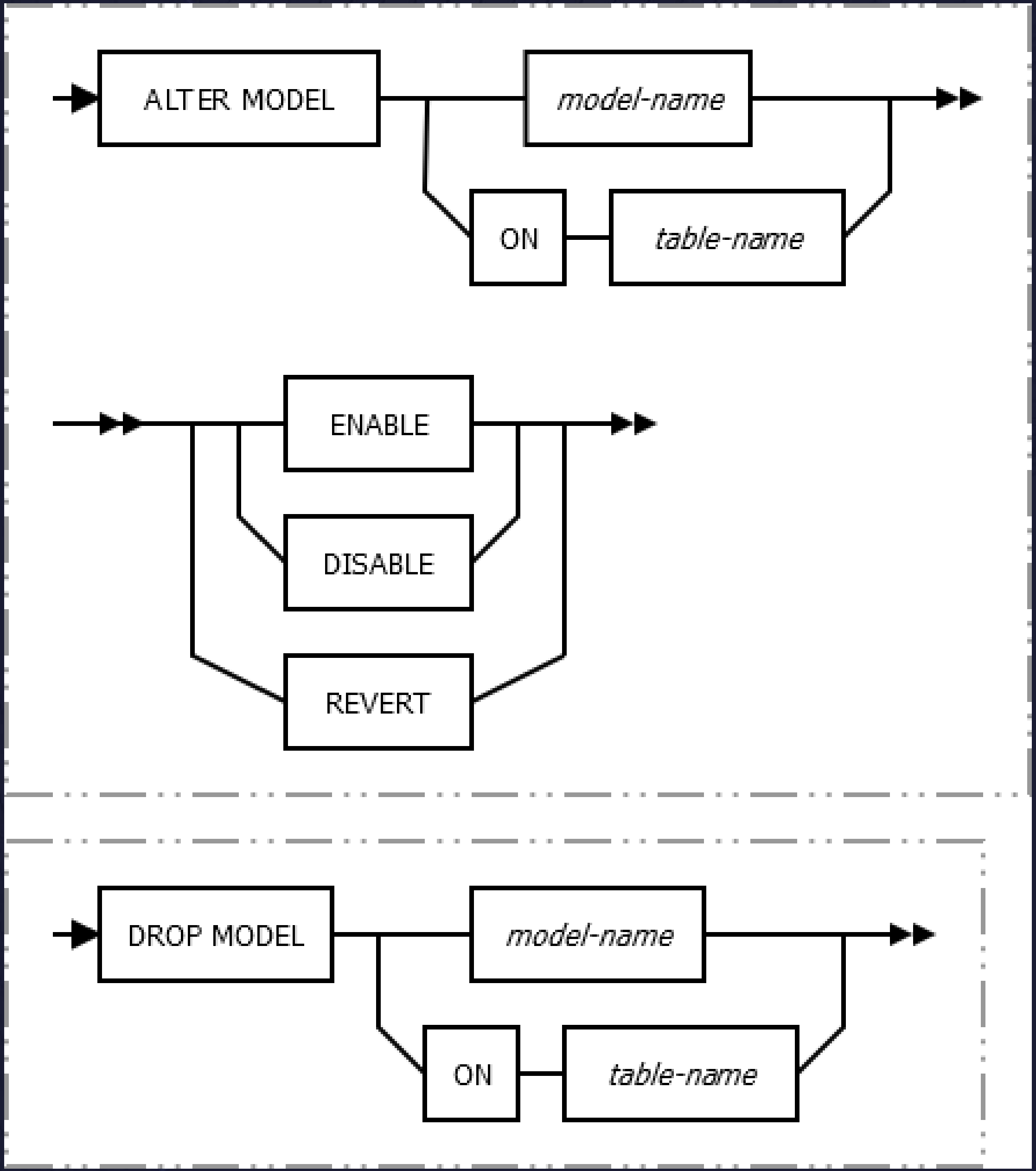
- No models are loaded
- Optimizer uses traditional algorithms to estimate cardinality

- Is ON by default but only takes effect when AUTO_AI_OPTIMIZER is also ON
- Can be embedded into a guideline/profile to control the use of models on a per query basis

DDL for Model Control

- Two new DDL commands: ALTER MODEL and DROP MODEL
  - Can specify a model by using a table name
- DROP MODEL will drop models
- ALTER MODEL will alter the model
  - ENABLE/DISABLE controls model discovery/training/usage
- REVERT swaps most recent model with older model

DDL


ALTER MODEL
model-name
ON
table-name
ENABLE
DISABLE
REVERT
DROP MODEL
model-name
ON
table-name

Model Discovery Logging

Entries added to 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")
```

Model Training Logging

Entries added to statistics log

```
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>
```

Identify Model Usage in EXPLAIN

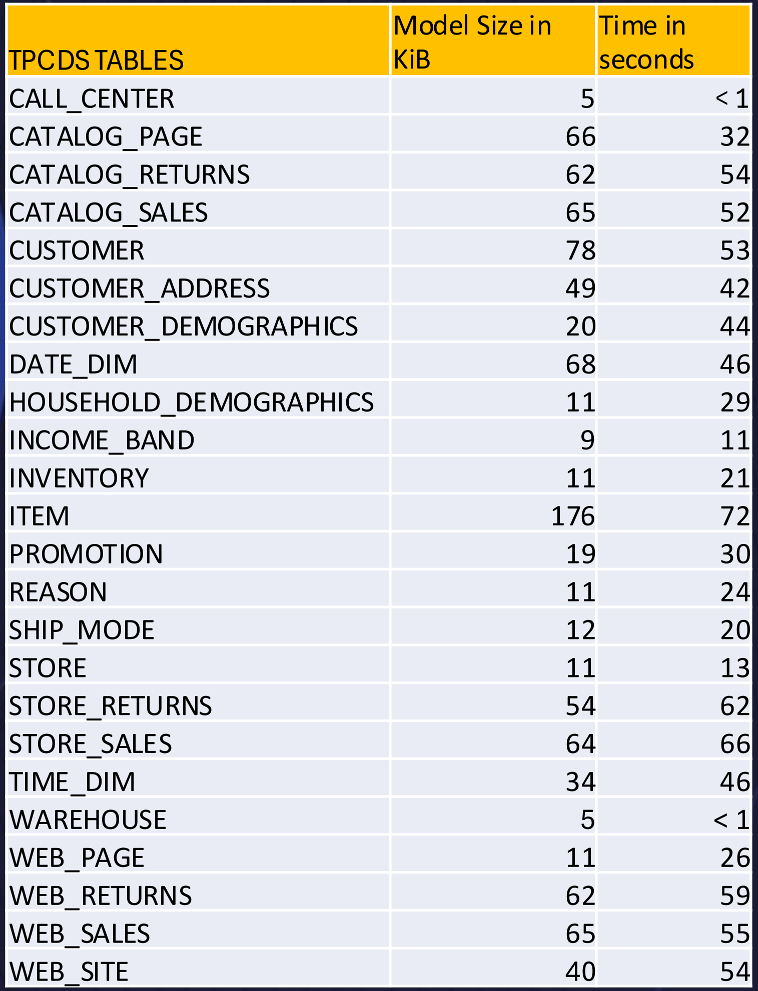
- Explain plan operators indicate if an individual predicate had its selectivity computed using a model
- Each relevant operator include the list of predicates the model computed the combined selectivity for
- Model information will also be listed in the “objects used” and includes the columns the model 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
                           …
```

Results

Model Size and Training Time



TPCDS TABLES	Model Size in KiB	Time in seconds
CALL_CENTER	5	< 1
CATALOG_PAGE	66	32
CATALOG_RETURNS	62	54
CATALOG_SALES	65	52
CUSTOMER	78	53
CUSTOMER_ADDRESS	49	42
CUSTOMER_DEMOGRAPHICS	20	44
DATE_DIM	68	46
HOUSEHOLD_DEMOGRAPHICS	11	29
INCOME_BAND	9	11
INVENTORY	11	21
ITEM	176	72
PROMOTION	19	30
REASON	11	24
SHIP_MODE	12	20
STORE	11	13
STORE_RETURNS	54	62
STORE_SALES	64	66
TIME_DIM	34	46
WAREHOUSE	5	< 1
WEB_PAGE	11	26
WEB_RETURNS	62	59
WEB_SALES	65	55
WEB_SITE	40	54

Average Model Size = ~42KB

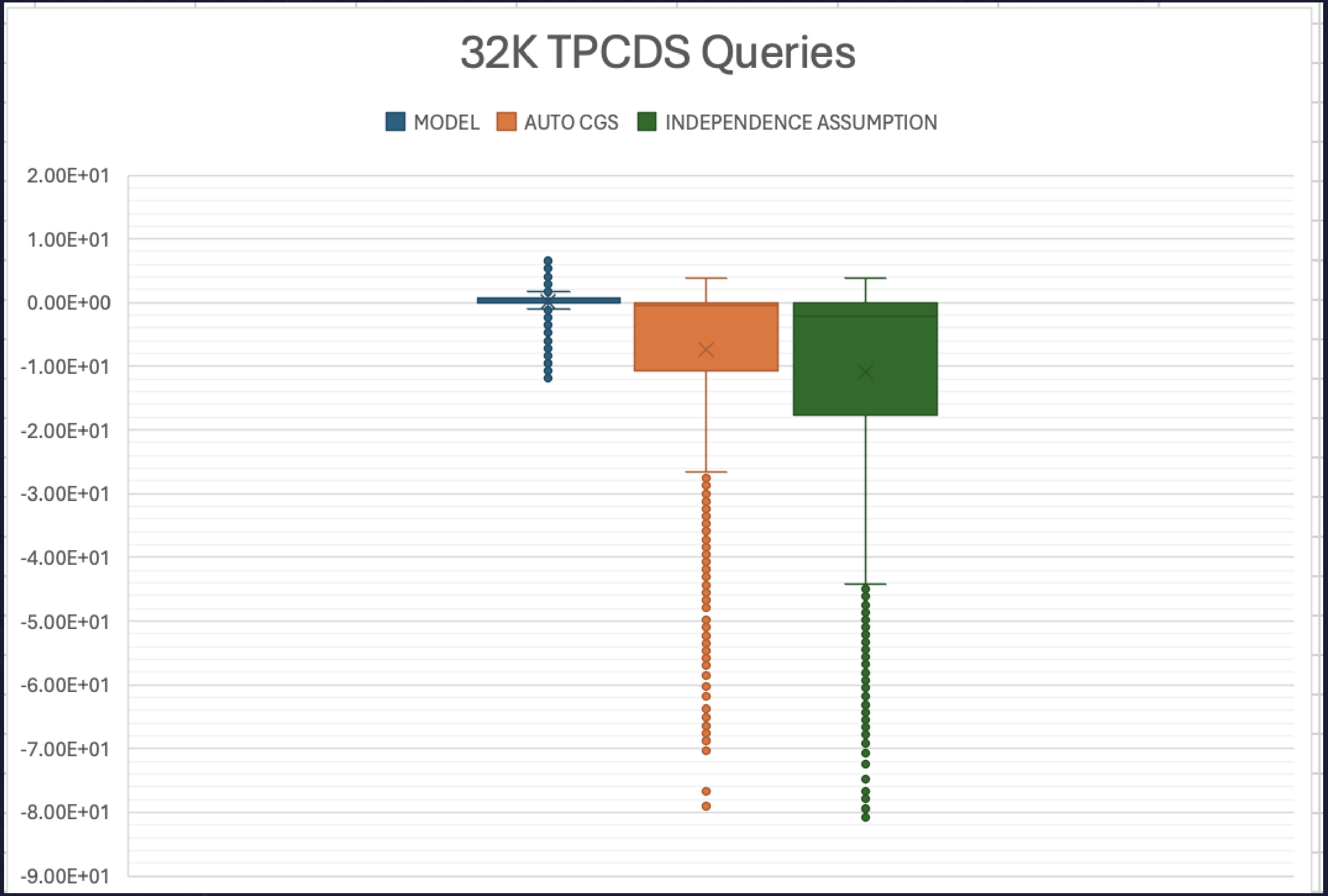
Average Training Time = ~38s

Cardinality Estimation Accuracy

32K Generated Queries against TPCDS schema

1-5 predicates consisting of equality, range, IN

Non-empty query results

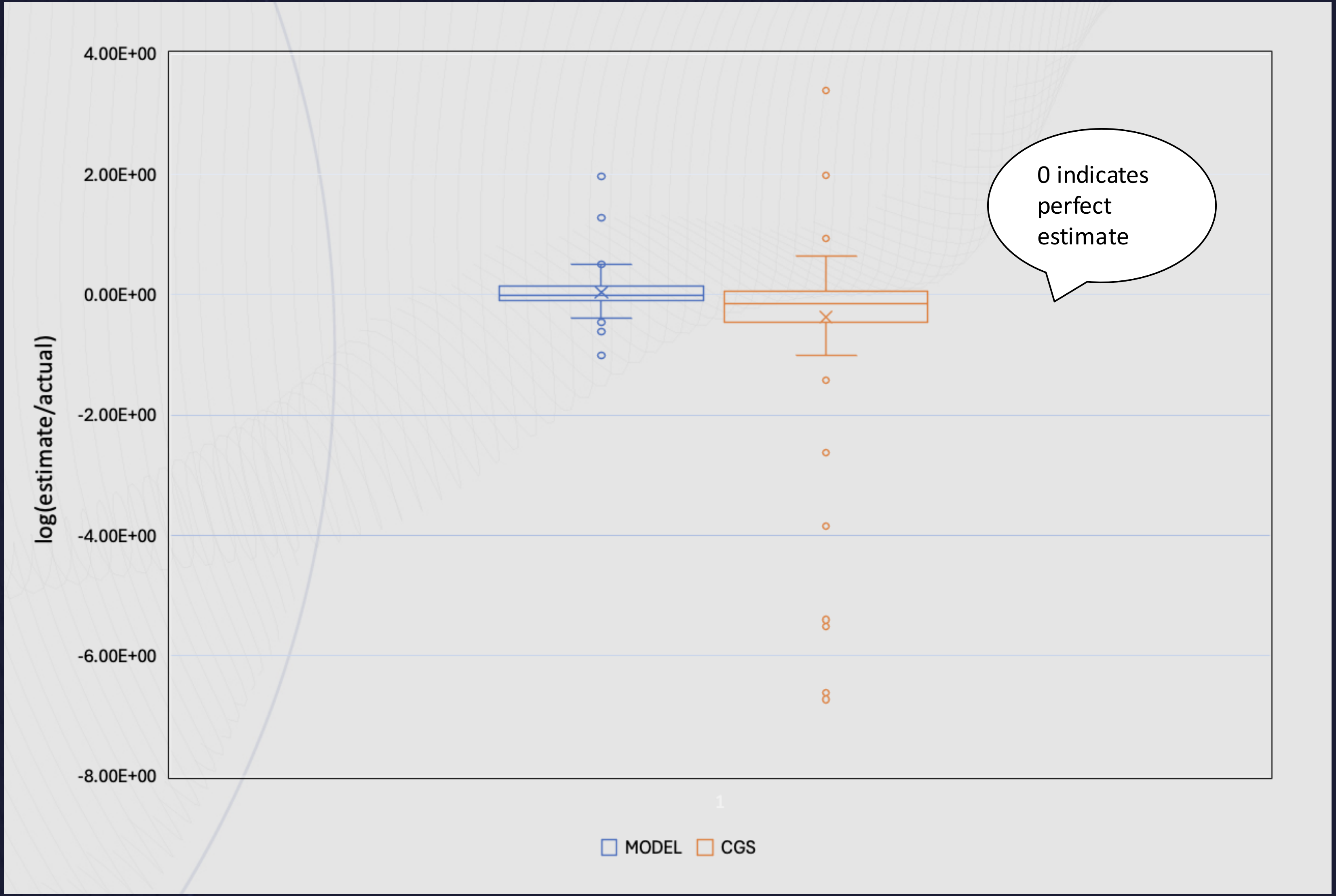


Cardinality Estimation Accuracy

Queries based on motor-vehicle schema

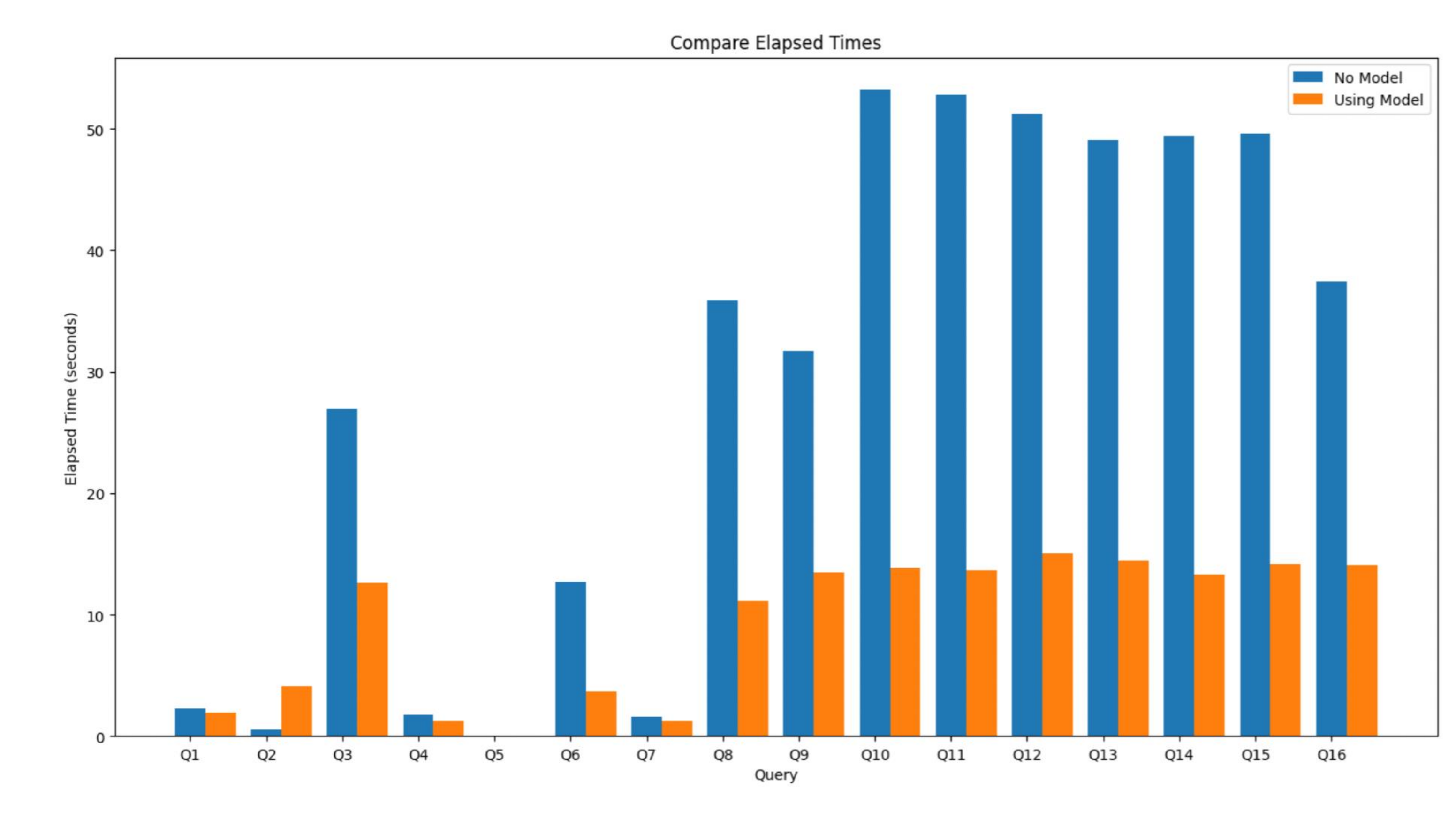
Equality and IN predicates

All required combinations of column group stats



0 indicates perfect estimate

Performance of Some Problematic Queries From a Couple of Db2 Users



AI Query Optimizer With a Problematic Query

```
CHD.PAY_0 IN (0,1,2) AND
CHD.PAY_2 BETWEEN 0 AND 2 AND
CHD.BILL_AMT1 BETWEEN 150 AND 746814
AND
…
```

Traditional Algorithms

```
                     4838.66 (4.5729e+06)
                     HSJOIN
                     (   4)
                     40713.1
                      6668
           /----------+-----------\
   100000                      4838.66 (4.5729e+06)
   TBSCAN                      HSJOIN
   (   5)                      (   6)
   1510.71                     38986.2
    19                          6649
    |                  /----------+-----------\
   100000          100000                    7.9354 (15032)
 CO-TABLE: DEMO    TBSCAN                    HSJOIN
 INSURANCE_HISTORY (   7)                    (   8)
    Q4             3565.28                   35295.6
                    37                        6612
                    |               /----------+-----------\
                  100000       10.102 (1503)           128.827 (127)
            CO-TABLE: DEMO     HSJOIN                  TBSCAN
             PURCHASE_HISTORY  (   9)                  (  12)
                  Q5           35180.2                 114.982
                                6596                    16
                      /-------+--------\                |
            (50245)7001.44          (106) 101.207       164
                 TBSCAN                TBSCAN      CO-TABLE: DEMO
                 (  10)                (  11)     SENTIMENT_SCORE_DATA
                 35047.6               114.982          Q1
                  6580                  16
                  |                     |
                 99995                  164
            CO-TABLE: DEMO        CO-TABLE: DEMO
            CREDIT_HISTORY_DATA     POLICE_DATA
```

AI Model

```
                              37787.2 (4.5729e+06)
                              HSJOIN
                              (   4)
                              44985.6
                               6668
                     /-----------+-----------\
      (4.5729e+06) 37787.2                    100000
            HSJOIN                            TBSCAN
            (   5)                            (  12)
            43151.7                           1510.71
             6649                              19
    /----------+----------\                    |
100000                (7516) 61.971          100000
TBSCAN                    HSJOIN         CO-TABLE: DEMO
(   6)                    (   7)          INSURANCE_HISTORY
3565.28                   39450.2             Q4
 37                        6612
 |                /---------+----------\
100000        54677.3 (50245)     79.5008 (82)
CO-TABLE: DEMO    TBSCAN               HSJOIN
 PURCHASE_HISTORY (   8)               (   9)
 Q5               39085.2              230.664
                   6580                 32
                   |            /-------+--------\
                  99995    128.827 (127)      101.207 (106)
          CO-TABLE: DEMO   TBSCAN             TBSCAN
          CREDIT_HISTORY_DATA (  10)          (  11)
                  Q3       114.982            114.982
                            16                 16
                            |                  |
                            164                164
                     CO-TABLE: DEMO      CO-TABLE: DEMO
                   SENTIMENT_SCORE_DATA   POLICE_DATA
                            Q1                 Q2
```

Learn More

- 11.5.6 Technology Preview
- The AI Query Optimizer in Db2
- AI Query Optimizer Features in Db2 Version 12.1
- CASCON 2015: Cardinality estimation using neural networks
- CASCON 2021 Best Industry Paper: Query predicate selectivity using machine learning in Db2:

Tridex

Tridex Db2 LUW
September 19th
2024

Machine Learning Optimization For Production Use in vNext

Calisto Zuzarte

Calisto@ca.ibm.com