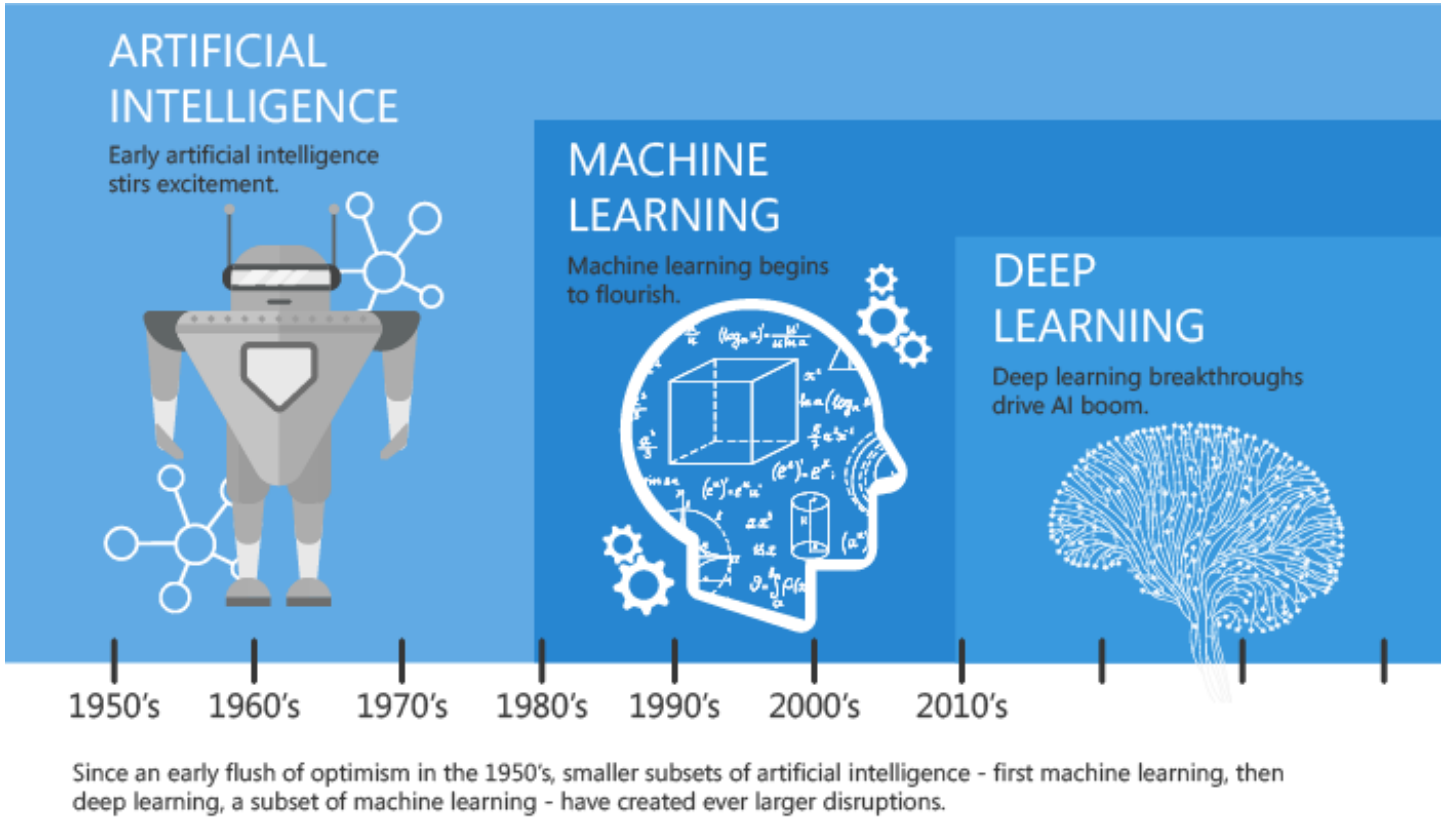
Big Data Analysis Enabler(TM) for Oracle Database®

Raymond,

2024. 06. 13

Why Big Data Analysis Enabler ?



Since an early flush of optimism in the 1950's, smaller subsets of artificial intelligence - first machine learning, then deep learning, a subset of machine learning - have created ever larger disruptions.

- **Enables to combine Python, R, JAVA with Oracle Database, ie, data sources, result repository and more.**
- **Enables Machine Learning, Deep Learning Inference continuously and in real time on the production line.**

BDAE™ stands for Big Data Analysis Enabler.

Big Data Related Platforms

.. RDBMS as Repositories

Manufacturing data is mostly **structured** or semi-structured, Especially High Tech. In most cases, data subject to analysis is automatically collected and the starting point is on Database rather than files.

Big Data Related Platforms

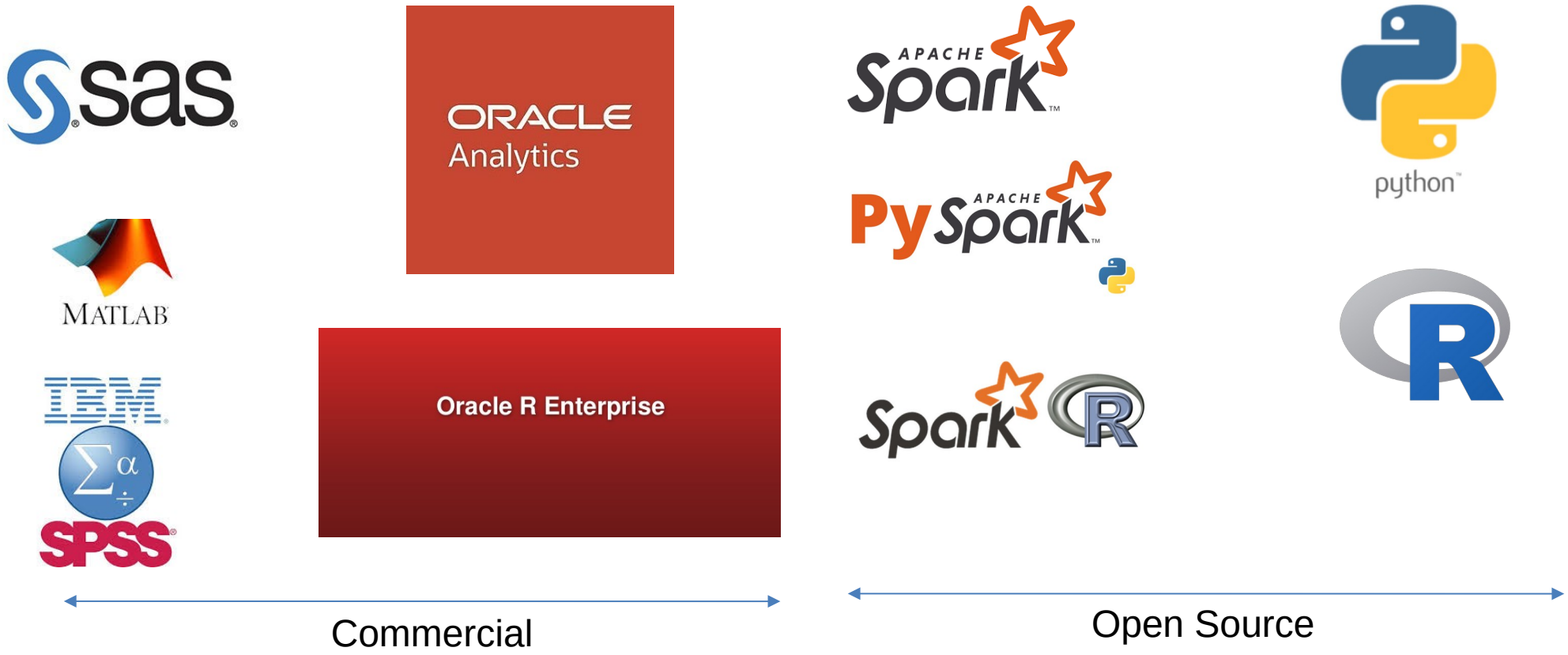
.. Hadoop Ecosystem as Repositories and more

The purpose of the existence of Phoenix and Hive is also to use SQL statements based on structured formats.

Analytic Engines and ..

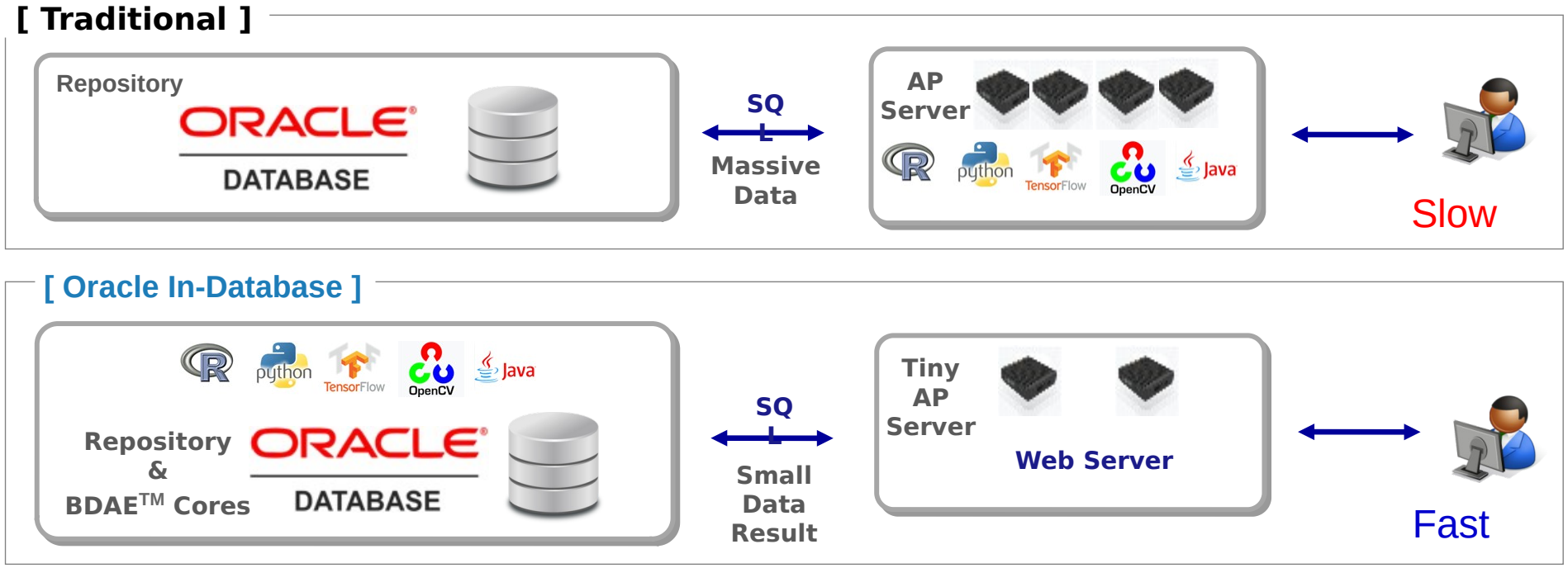
.. Analysis related

Commercial – SAS, MATLAB, SPSS, Oracle Analytics, Oracle R Enterprise, ..
R Engine related – Oracle R Enterprise, SparkR, ..
Python Engine related – PySpark, ...



Big Data Analysis Enabler™
is different from others ?

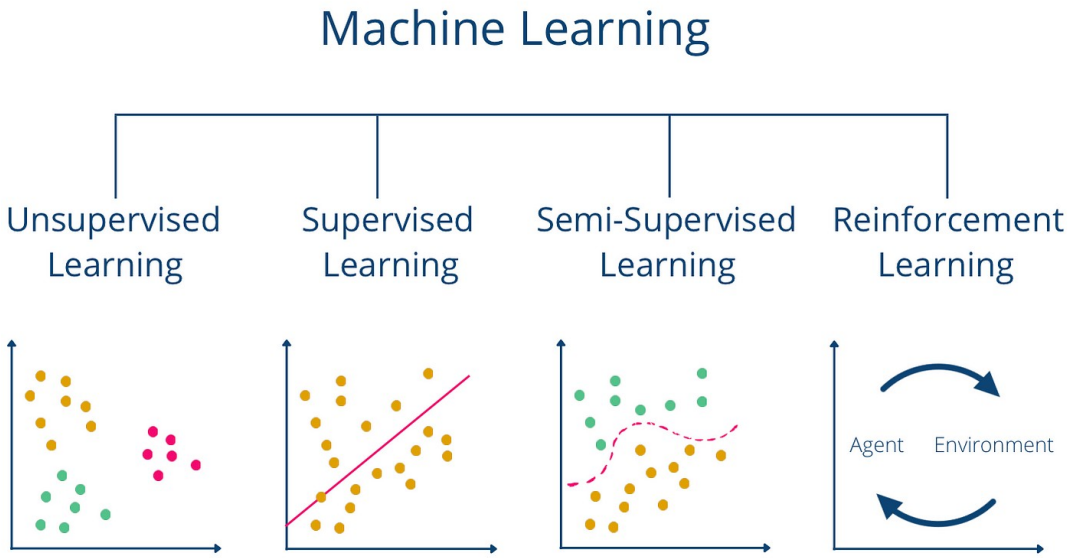
Assuming that Oracle Database ® is an OS(Operating System), BDAE is a Device Driver. It's not simple an application. Just as Device Drivers must be created in accordance with Operating System Interfaces, BDAE was written in accordance with Oracle In-Database Interfaces, and the call is also determined by Oracle DB.

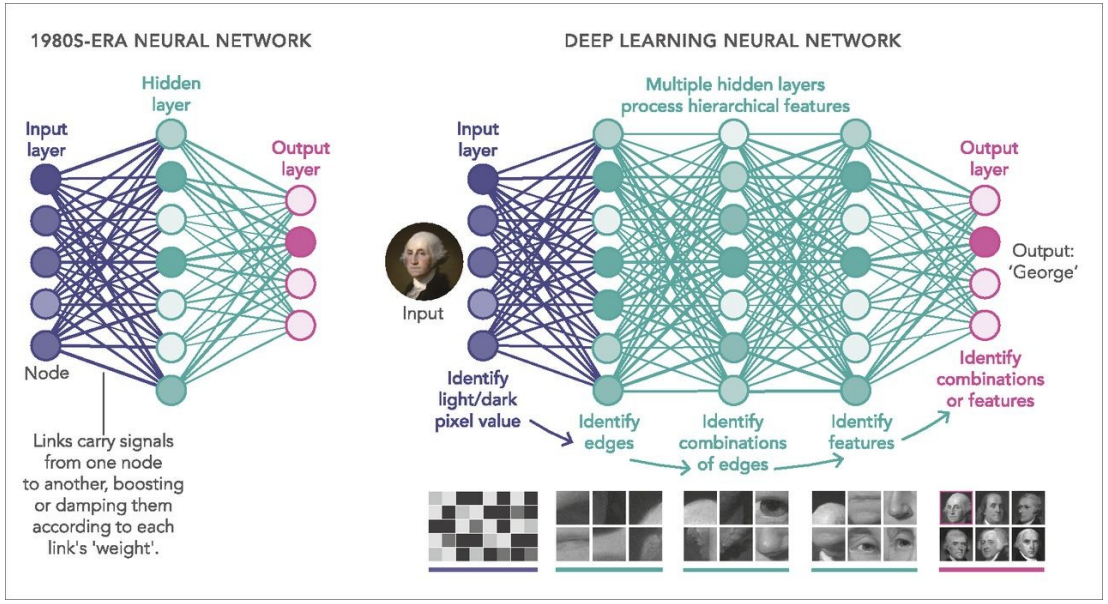


Big Data Analysis Enabler™ ?

supports Python/R/JAVA with Oracle In-Database®.

BDAE™ exists to load user algorithms in real time based on the performance and stability of the Oracle database, train algorithms only SQL statements without data movement, and make inference in production lines.



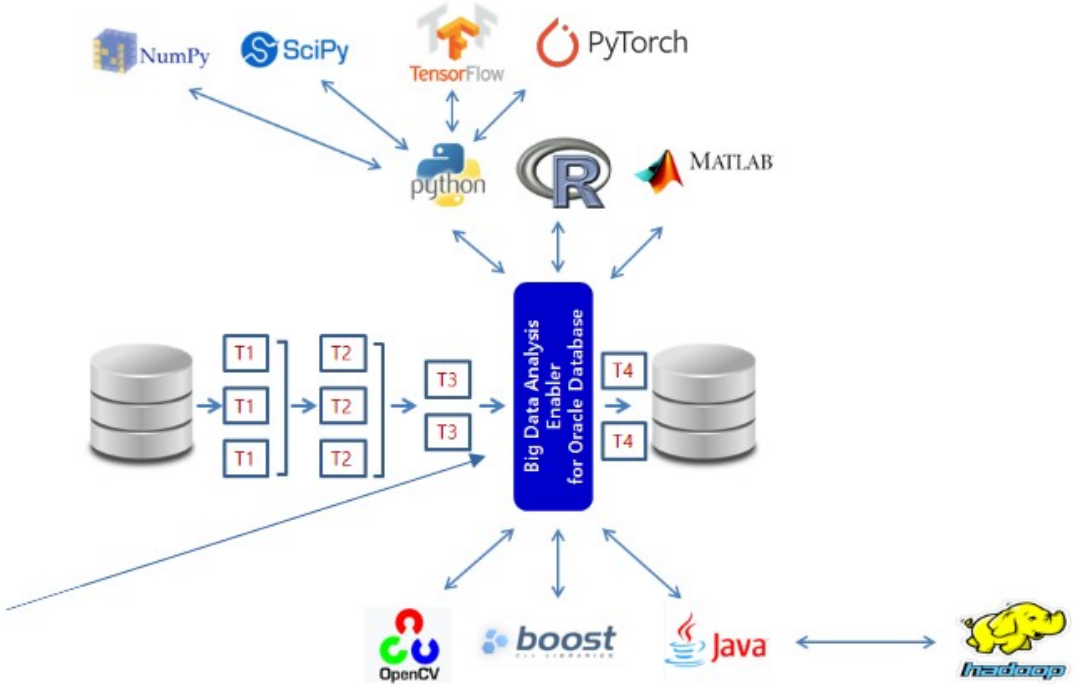


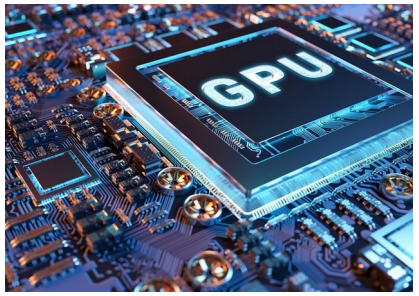
※ How many of the algorithms themselves are parallel distributed processing? Never be confused.

Big Data Analysis Enabler™

BDAE™ exists on the same server as Oracle Instance ..

The most important thing is the data loading location of analysis engines such as Python and R. BDAE is not an executable program. It is a shared library created according to Oracle Data Interface protocols, and the Oracle Database automatically loads it.





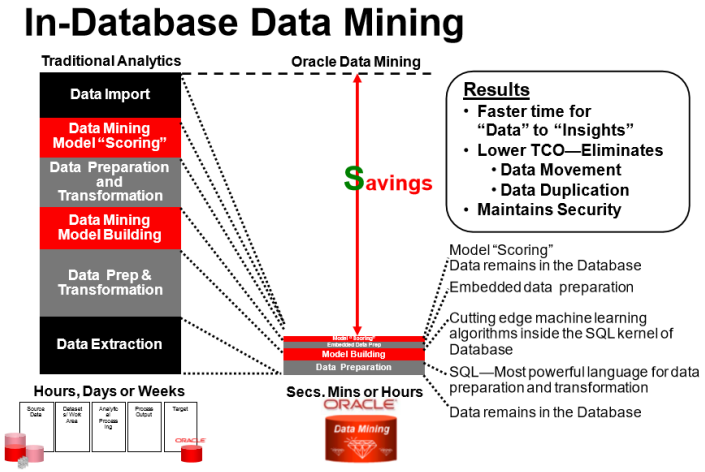
Oracle Database's
Parallel Processing
Intermediate Data Location

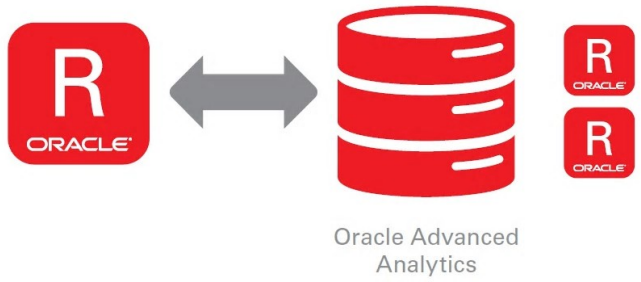
Big Data Analysis Enabler™
Uses Oracle In-Database, In-Memory (In case of Exadata).

Oracle In-Database :
No Massive Data Movement

Oracle In-Memory :
In case of Exadata, Not Hard-Disk, Data are in Flash Disk.

.. means “Fastest” although Python, R uses.





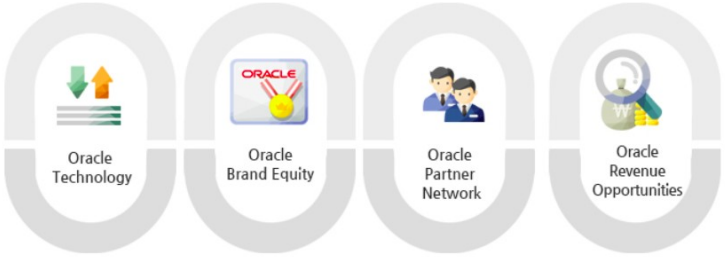
Summary of Big Data Analysis Enabler™
Advantages

1. Excellent development **Productivity**
   It can be immediately created, modified, and executed dynamically using Python and R languages.
2. **Performance**(Until you see the final result based on R and Python)
   No Massive Data Movement, Overall Performance is the fastest with Python/R related products.
3. **Excellent Connectivity** (Easy and simple to execute)
   Because the execution form is an SQL statement, connectivity is the best.
4. **Parallel Processing**
   R and Python modules do not need to consider parallel processing, Oracle Database will do.
5. **Wide range of use**, Real Time, Batch, Pre-processing ...
   It is suitable as an add-on to existing products and can be used immediately without changing customers' R and Python modules.
6. **No Application Servers needed**, Easy to support most of Solutions adds-on.
7. Based on the advantages of Oracle Database, **you can focus on your analysis work** without having to consider data integrity, parallelism, backup, and duplication.
8. Since the output of BDAE™ is done in memory, **no separate space is needed** for data analysis.

Big Data Analysis Enabler™
… Business Targeting.

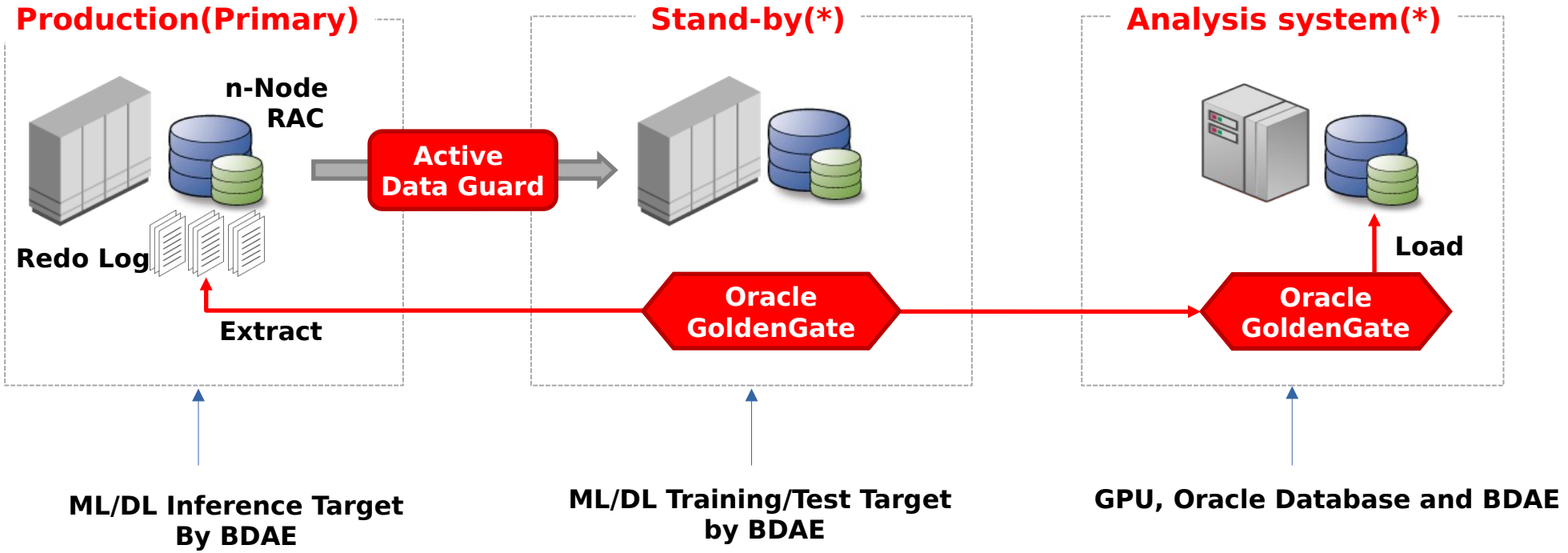
1. Adds-On
   BDAE™ integrates existing solutions or customer analysts' Python and R codes at the SQL statement level. Because it executes and receives results as SQL statements, it can be installed and executed immediately without changing the schema when installing an existing third-party solution or an algorithm written by the customer.

2. As a Independent, Analysis Platform
   Provide BDAE independently to customers for real-time or development use.
   Oracle ESL (Embedded Software License) for small size customers.



Ideal Architecture for BDAE

BDAE can show the best performance in Oracle RAC. This is because parallel distributed processing is possible. However, in the case of Deep Learning, performance is more often determined by the GPU. Ideally, all systems should automatically move and collect data from DB to DB in near real time, for example .. High Tech.



Restriction of Big Data Analysis Enabler™

- ✓ Since all 4 methods are fixed according to the Oracle In-Database® Interface Protocols, no further expansion is possible. 4 methods are enough to do database jobs.

- ✓ The session must be closed after the execution finished because it cannot respond to possible memory leaks of numerous un-proven Python and R packages and no wait for GC, eg. Python. When closed, all memory is cleaned and Database's stability is maintained. (We can set maximum memory usage.)

- ✓ Big Data Analysis Enabler is simple, powerful, and ready to deploy now. All efforts should be directed to developing the analysis itself.

- ✓ BDAE supports only Linux OS, not UNIX and Windows.

※ Most restrictions are due to installation of Oracle Database, Python, R, etc.
Ubuntu distribution is the best Linux for NVIDIA GPUs, etc., but recently Oracle does not support Ubuntu.

Big Data Analysis Enabler™
has no schema restrictions and also supports Dynamic SQL.

Preprocessing work at the manufacturing site for analysis accounts for more than 50 ~ 70% of the total analysis. Although Oracle Database is expensive, it supports a higher level of SQL than Open Source Based Database or Hive on Hadoop(R) using ANSI-SQL, Oracle Database(R) can effectively perform this preprocessing part.

BDAE™ supports this form of preprocessing SQL.

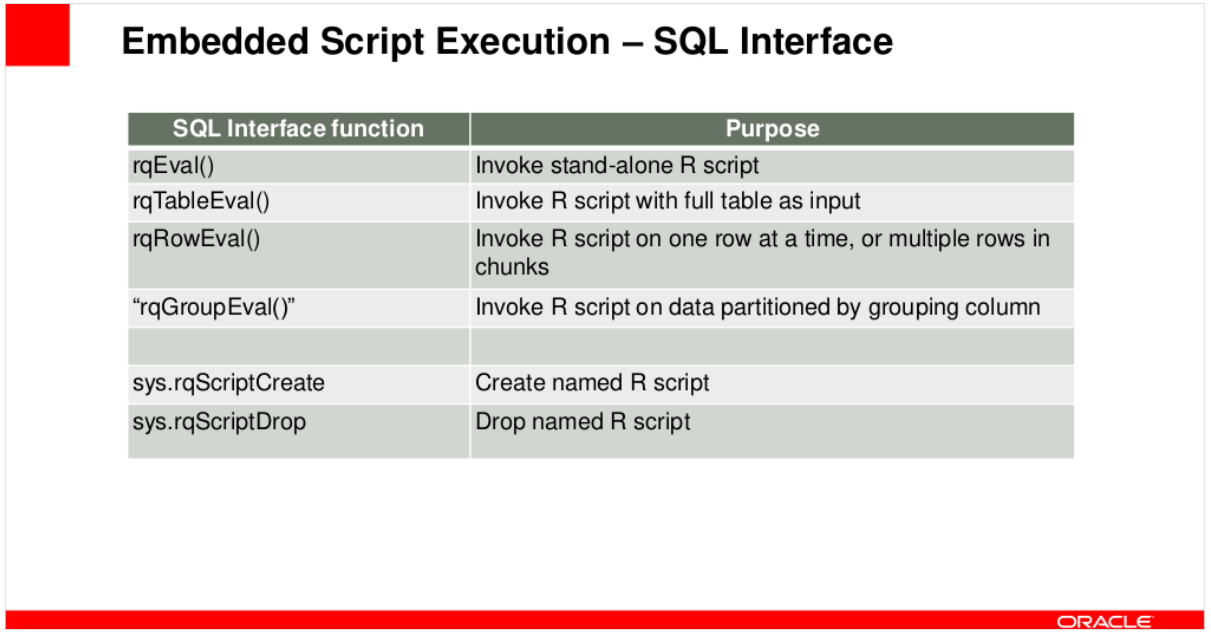
The biggest advantage of BDAE™ is that it uses the parallel processing of Oracle Database®. Therefore, there is no need to consider parallel processing internally in Python or R, and you can just focus on the algorithm itself.

Like PySpark, where a single line of code causes parallel processing, you can do the same thing using BDAE. Of course, it is much faster because shuffling does not occur like Spark or RHive.

BDAE™ - 4 Functions of Oracle In-Database® API

The left part is provided by Oracle R Enterprise, and the right part is a function provided by BDAE.

※ The function name can be determined by the customer, but the number of arguments cannot.



Embedded Script Execution – SQL Interface

SQL Interface function	Purpose
rqEval()	Invoke stand-alone R script
rqTableEval()	Invoke R script with full table as input
rqRowEval()	Invoke R script on one row at a time, or multiple rows in chunks
"rqGroupEval()"	Invoke R script on data partitioned by grouping column
sys.rqScriptCreate	Create named R script
sys.rqScriptDrop	Drop named R script

ORACLE

SQL Interface function	
asEval()	Same rqEval()
asTableEval()	Same rqTableEval()
asRowEval()	Same rqRowEval()
asGroupEval()	Same rqGroupEval()
apEval()	Invoke stand-alone Python module
apTableEval()	Invoke Python module with full table as input
apRowEval()	Invoke Python module on one row at a time, or multiple rows in chunks
apGroupEval()	Invoke Python module on data partitioned by grouping column

BDAE™ - 4 Functions' Arguments

The Number of Arguments, and Types can not be modified.

※ The function name can be determined by the customer, but the number of arguments, types of them cannot.

rq*Eval() Table Functions

rqEval, rqTableEval, "rqGroupEval", rqRowEval

```
rq*Eval(
  cursor(select * from <table-1>),
  cursor(select * from <table-2>),
  'select <column list> from <table-3> t',
  <grouping col-name from table-1
   or num rows>,
  '<R-script-name>')
```

- Input cursor – Depending on the function, input passed as a whole table, group, or one row at a time to the R closure (not for rqEval)
- Parameters cursor – Parameters are specified through a select statement, scalars only – single row
- Output table definition – a query specifying the format of the result
  If NULL, output is a serialized BLOB
  If 'PNG', images only as BLOB column
  If 'XML', XML string of images and return values
- Group name (optional) – Name of the grouping column
- Number of rows (optional) – number of rows to provide to function at one time
- Name of R function in repository to execute

as*Eval(), ap*Eval()

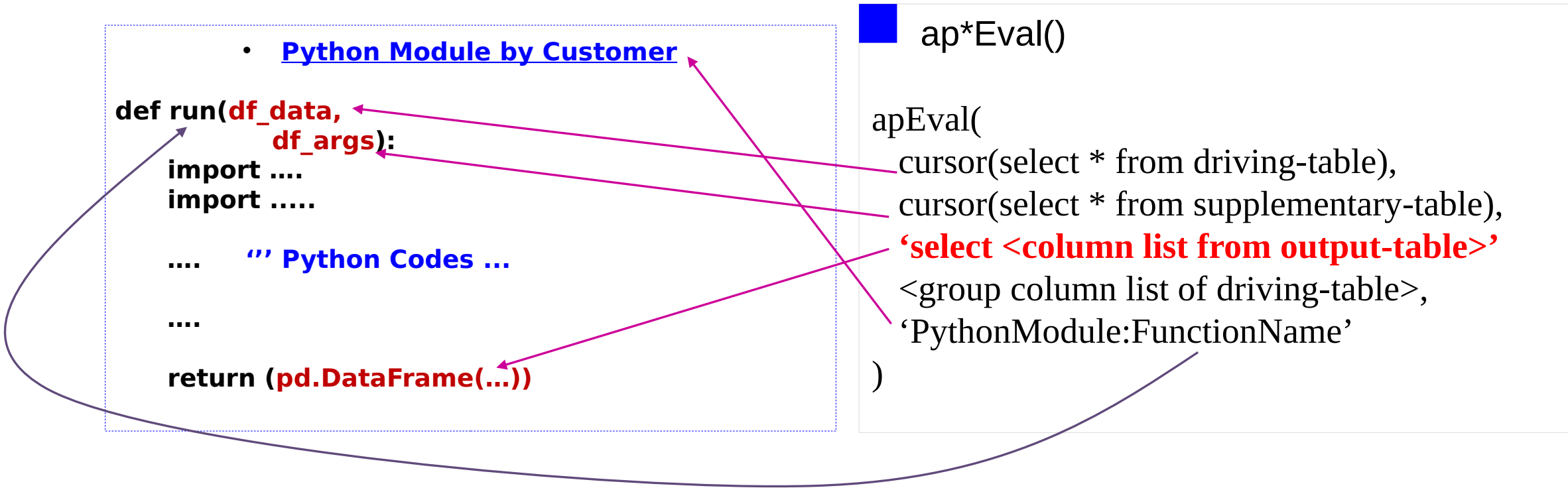
apEval(
  cursor(select * from driving-table),
  cursor(select * from supplementary-table),
  'select <column list from output-table>'
  <group column list of driving-table>,
  'PythonModuleName:start_function_name'
)

※ output-table can be a view or table, or
select .... from dual (dynamic)

Relations between 4 Functions' Arguments and Python Code

Python requires both a module name and a function name.

※ All data is delivered in Pandas DataFrame format.



Relations between 4 Functions' Arguments and R Code

Since R is called by BDAE, the function name can be fixed to function as shown below.
No need to change.

※ All data is delivered in R's data.frame format.

- **R fuction by Customer**

```
function(df_data,
         df_args)
{
  library(...) #
               #
  library(...)

  ....          # R Codes ...
  ...
  return (data.frame(...))
}
```

as*Eval()

```
as*Eval(
  cursor(select * from driving-table),
  cursor(select * from supplementary-table),
  'select <column list from output-table>'
  <group column list of driving-table>,
  'RModuleName'
)
```

Big Data Analysis Enabler™
does not write any data into Any Tables(Tablespaces).

Because all data is created dynamically in memory, it does not use storage space. BDAE™ cannot change the source data, and this is due to Oracle In-Database® API policies.

If you insert some tables, you can use as followings ..

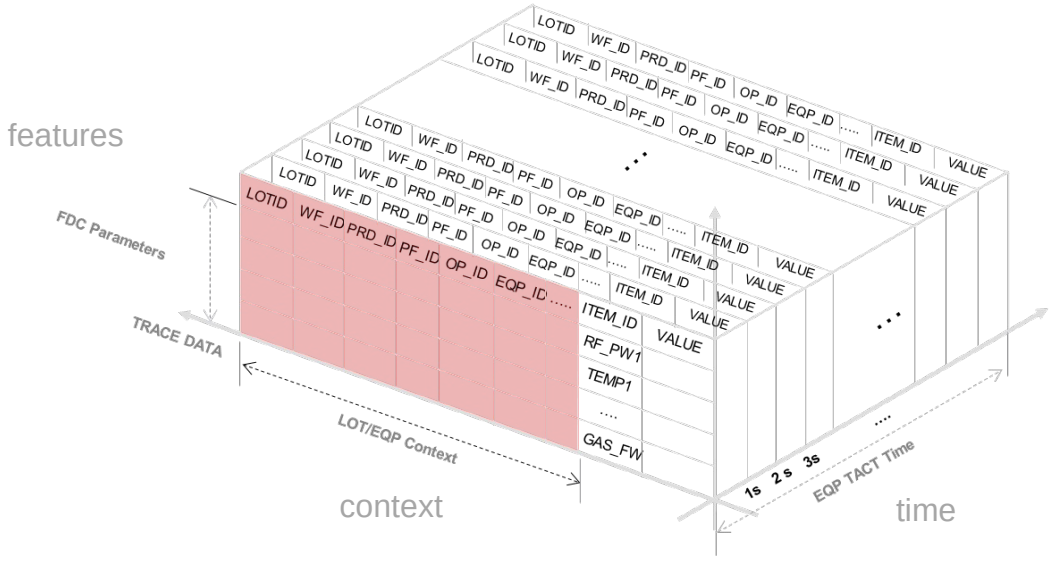
"INSERT INTO [TABLE] SELECT … table(apTableEval(….) ..)"
OR
"CREATE TABLE [TABLE] AS SELECT … table(apTableEvala(….)..)"

※ Why table function ?
In Oracle Database, you can only return table-like data with table functions, not Package, not general functions.

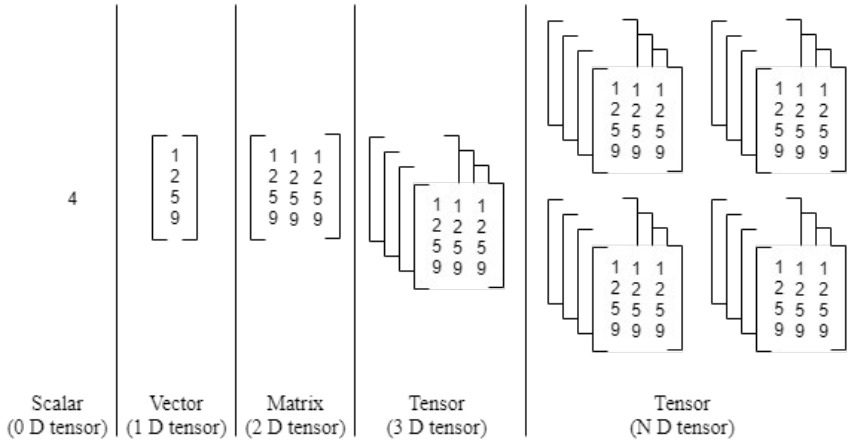
High-Tech. FDC Trace Data,

BDAE™ provides parallel processing for massive data ..

FDC Trace Data is a very large amount of data, and is also data for learning various algorithms. In semiconductors, this amount of data in an uncompressed state is approximately 4 to 6 PB over a 3-month storage cycle.



Trace Data per 1 LOT/1 EQP



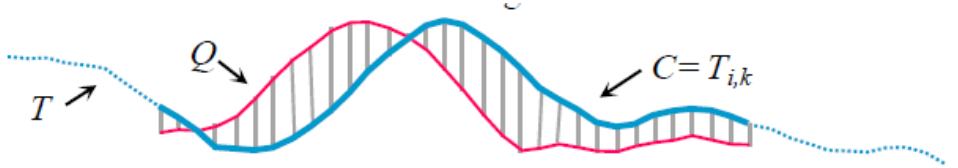
Tensors

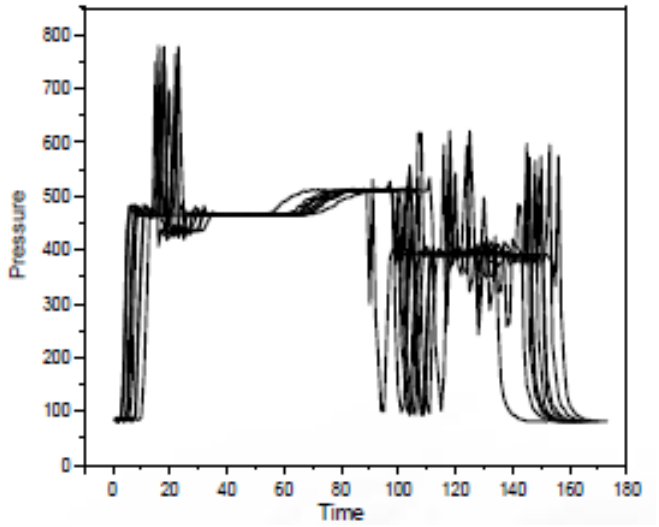
High-Tech. FDC Trace Data

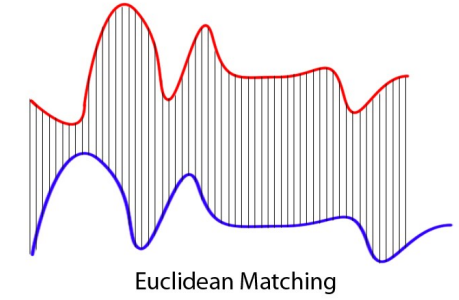
Applying DTW algorithm using BDAE.

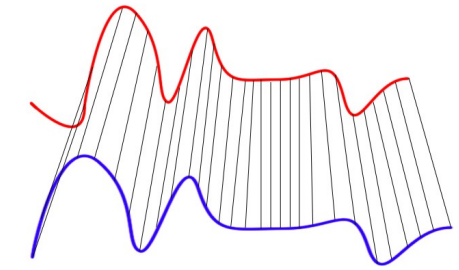
```
SQL>  SELECT  *
   2    FROM TABLE (apGroupEvalParallel (
   3                    cursor (
   4                            SELECT *
   5                            FROM TRACE_DATA
   6                            WHERE EQP_ID = 'EPS001'
   7                               AND LOT_ID = 'LOT001'
   8                               AND ETC = '.....'
   9              ),
  10                    cursor(SELECT * FROM GOLDEN_EQUIPMENT ...),
  11                    'SELECT  CAST(''A'' AS VARCHAR2(40)) ITEM_ID,
                              1.0 SIMILARITY FROM DUAL',
  11                    'EQP_ID, LOT_ID, ....',
  12                    'DefectUtil:FastDTW');
```

PARAMETER_ID	SIMILARITY
RF_POWER_1	2.23
O2_PUMP_1	0.5
Ch1_TEMP_1	2.1
.....	.....
.....	.....






Euclidean Matching


Dynamic Time Warping Matching

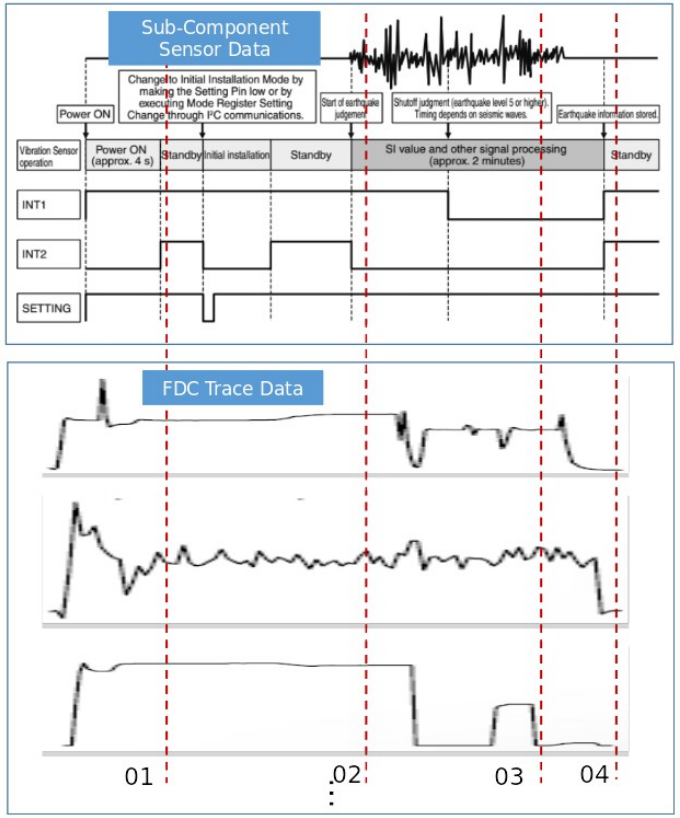
High-Tech. FDC Trace Data

From the perspective of FAB as a whole ..

The number of parameters involved in processing LOT from FDC data is now approaching or exceeding 1,000, and each time series data has length variability.
In a situation where it is difficult to predict defects
using descriptive statistics depending on the micro-process in FAB,
the development of Deep Learning is remarkable.

For example, you can see papers being published that utilize
not only RNN, LSTM, and GRU, but also CNN algorithms
using chart images. (Because it is mostly a security matter,
it is not known exactly what algorithm is used.)

With the advent of the Big Data era, Pattern Recognition or
Pattern Matching became popular, but the amount of pre-processing
was very high due to the variability in the length of parameters
(sensor data).



※ The maximum number of columns in an Oracle table, including virtual ones,
is 1,000. Due to the Oracle structure, having too many columns is not good for performance.

High-Tech. FDC Trace Data

BDAE™ will be powered by Oracle advanced SQLs

If you blindly load DB data in Python, a lot of memory and CPU will be used even before analysis begins. Numpy, Pandas, etc. are good packages, but it is important to remember that when there is a lot of data, Oracle Database can perform the preprocessing part much faster. Below shows how SMA, EMA, etc. are performed using SQL statements. (eg.)

```
SELECT  eqp_id, recipe_id, …, time, parameter_name, sma, ema
FROM    (
          SELECT eqp_id, recipe_id, time, …, parameter_name, parameter_value
          FROM  trace_data
          WHERE 1=1
              AND time between … and …
              AND step_seq = ‘ …’
              …
          ) a
MODEL
   PARTITION by (a.parameter_name, 2 / (1 + count(*) over (partition by a.parameter_name))
              smoothing_constant )
   DIMENSION by (row_number() over (partition by a.parameter_name order by a.time) rn)
   MEASURES (a.time, a.parameter_value, sma, 0 ema)
   (
      ema[1] = a.parameter_value[1],
      ema[rn > 1] order by rn = ( cv(smoothing_constant) * (parameter_value[cv()] - ema[cv() -1]) ) + ema[cv() -1]
   )
ORDER by eqp_id, a.recipe_id, …, a.time, a.parameter_name;
```

High-Tech. FDC Trace Data

BDAE™ enables to use Oracle Parallelism (#1)

Most Machine Learning and Deep Learning algorithms require data preprocessing. Of course, it is supported by Python packages. This too should be left to Oracle Database if its performance is better.
(eg, Min-Max Scaler, Standardization Scaler (z-Score) …)

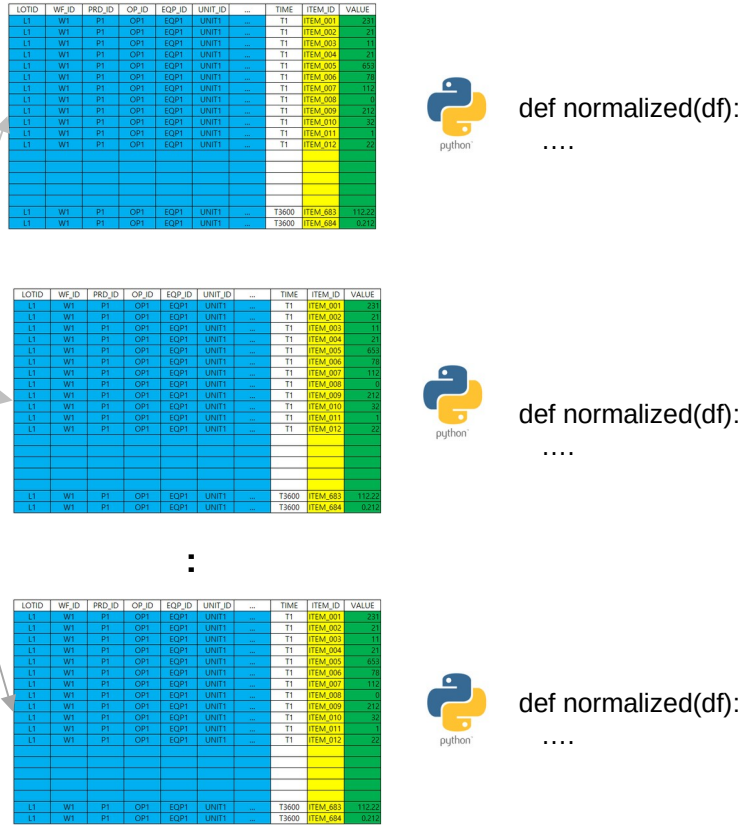
```
WITH TARGET_TBLE AS
(
   SELECT /*+ parallel(5) */ * from FDC_TRACE
   WHERE 1=1
       AND EQP_ID='EQP-200’
    --  AND UNIT_ID='UNIT-02’
    -- AND LOT_ID='LOTB-101’
    -- AND WAFER_ID='LOTB-101-01’
    -- AND RECIPE='RECIPE-200'
)
SELECT EQP_ID, UNIT_ID, LOT_ID, WAFER_ID, RECIPE, PARAM_ID,
  (VALUE - (AVG(VALUE) OVER (PARTITION BY PARAM_ID)))
          / (STDDEV(VALUE) OVER (PARTITION BY PARAM_ID)) AS
          NORMALIZED_VALUE
FROM TARGET_TBLE;
```

High-Tech. FDC Trace Data

BDAE™ enables to use Oracle Parallelism(#2)

BDAE uses Oracle Parallelism according your parallel hint(/*+ parallel(5) */), The Python module is not involved in parallel processing, and BDAE collects data using the parallel processing key and puts it into the corresponding function.

```
SELECT /*+ parallel(5) */*
    FROM table(apGroupEvalParallel(
        CURSOR(
            WITH TARGET_TBLE AS
            (
         SELECT /*+ parallel(5) */ * from FDC_TRACE
         WHERE 1=1
           AND EQP_ID='EQP-200'
           AND UNIT_ID='UNIT-02'
            )
           SELECT EQP_ID, UNIT_ID, LOT_ID, WAFER_ID, RECIPE, PARAM_ID,
              (VALUE - (AVG(VALUE) OVER (PARTITION BY PARAM_ID)))
             / (STDDEV(VALUE) OVER (PARTITION BY PARAM_ID)) AS  NORMALIZED_VALUE
                  FROM TARGET_TBLE
        ) ,
        NULL,
        'SELECT CAST("A" AS VARCHAR2(40)) EQP_ID,
              CAST("A" AS VARCHAR2(40)) UNIT_ID,
              CAST("A" AS VARCHAR2(40)) LOT_ID,
              CAST("A" AS VARCHAR2(40)) WAFER_ID,
              CAST("A" AS VARCHAR2(40)) RECIPE,
              CAST("A" AS VARCHAR2(40)) RESULT
              FROM DUAL',
        'EQP_ID,UNIT_ID,LOT_ID,WAFER_ID,RECIPE,PARAM_ID',
       'Standardization:normalize'));
```



High-Tech. FDC Trace Data

Wrapping BDAE™ SQLs .. Final Approach (#1)

Like PySpark or Spark RDD, One Single code will do parallel processing and fetch the result . On Right Figure, df = FDCDescribeParallel.get_fdc_descriptive_statistics() will cause Oracle parallel processing.
You can see Embedded SQLs are the base of Oracle In-Database, Parallel Processing.

FDCDescribeParallel.py

```
def get_fdc_descriptive_statistics_eqp(df_arg):

    ....

    SQL = "SELECT /*+ parallel(5) */* \
          FROM table(APGROUPEVALPARALLEL(\
                CURSOR(SELECT * FROM FDC_TRACE \
                      WHERE EQP_ID='{}' \
                ),\
                CURSOR(SELECT EQP_ID, UNIT_ID FROM FDC_TRACE WHERE ROWNUM < 1000001),\
                'SELECT CAST(''A'' AS VARCHAR2(40)) EQP_ID, \
                        CAST(''A'' AS VARCHAR2(40)) UNIT_ID,\
                        CAST(''A'' AS VARCHAR2(40)) LOT_ID,\
                        CAST(''A'' AS VARCHAR2(40)) WAFER_ID,\
                        CAST(''A'' AS VARCHAR2(40)) RECIPE,\
                        CAST(''A'' AS VARCHAR2(40)) PARAM_ID,\
                        CAST(''A'' AS VARCHAR2(40)) KEY,\
                        1.0 VALUE FROM DUAL',\
                        'EQP_ID,UNIT_ID,LOT_ID,WAFER_ID,RECIPE,PARAM_ID',\
               'ParallelDesc:describe'))".format(df_arg['EQP_ID'][0])

    df = pd.read_sql_query(SQL, conn)
    conn.close()
    return df
```

Using Your Component

```
[2]: import sys

     sys.path.append('/home/oracle/python')
```

```
[3]: import FDCDescribeParallel
```

```
[4]: df = FDCDescribeParallel.get_fdc_descriptive_statistics()
```

check parallel processing

```
[5]: df
```

[5]:

	eqp_id	unit_id	lot_id	wafer_id	recipe	param_id	key	value
0	EQP-200	UNIT-02	LOTB-101	LOTB-101-01	RECIPE-200	param_b-10	count	15000.000000
1	EQP-200	UNIT-02	LOTB-101	LOTB-101-01	RECIPE-200	param_b-10	mean	74795.254639
2	EQP-200	UNIT-02	LOTB-101	LOTB-101-01	RECIPE-200	param_b-10	std	43302.278170
3	EQP-200	UNIT-02	LOTB-101	LOTB-101-01	RECIPE-200	param_b-10	min	0.002450
4	EQP-200	UNIT-02	LOTB-101	LOTB-101-01	RECIPE-200	param_b-10	25%	37297.669422
...	...	...	...	...	...	...	...	...
3195	EQP-300	UNIT-01	LOTC-101	LOTC-101-01	RECIPE-300	param_b-99	min	-0.094540
3196	EQP-300	UNIT-01	LOTC-101	LOTC-101-01	RECIPE-300	param_b-99	25%	37298.084477
3197	EQP-300	UNIT-01	LOTC-101	LOTC-101-01	RECIPE-300	param_b-99	50%	74794.962558
3198	EQP-300	UNIT-01	LOTC-101	LOTC-101-01	RECIPE-300	param_b-99	75%	112292.827815
3199	EQP-300	UNIT-01	LOTC-101	LOTC-101-01	RECIPE-300	param_b-99	max	149789.850186

3200 rows × 8 columns

High-Tech. FDC Trace Data
Wrapping BDAE™ SQLs .. Final Approach(#2)

Because it is up to you, after creating the Python Wrapper as shown below, simply pass the Contexts as Arguments and you will receive the final result through parallel processing.

```
SELECT * FROM table (
  apEval(
    cursor(SELECT 'EQP-01' EQP_ID,
                  'UNIT-100' UNIT_ID,
                  ...
           FROM dual),
    'OUTPUT_VIEW_OR_TABLE_NAME',
    'FDC_ALGORITHM_01:run_final')
  );
```


Final Run for Final Results by Client

FDC_ALGORITHM_01.py OR In DB

```
import FDCWrapperNomalize
import FDCWrapperGoldenData
...
def run_final(df_args):
      # parallel internally.
      df_target = FDCWrapperNomalize.get(df_args)
      df_reference = FDCWrapperGoldenData.get(df_args)
      ... algorithm ...

      return pd.DataFrame(..)
```

High-Tech. FDC Trace Data

BDAE™ enables to use Oracle Parallelism.

Parallel processing does not occur automatically, so you must create an Oracle package and function as shown below to suit the customer's schema.
The BDAE Core part does not change, and parts that fit the customer's schema can be created immediately with simple DDL commands. APGRPEVALIMPL is the core of BDAE.

```
CREATE OR REPLACE FUNCTION    apGroupEvalParallel (
                        inp_cur IN fdc_tracePkg.cur,
                        par_cur SYS_REFCURSOR,
                        out_qry VARCHAR2,
                        grp_col VARCHAR2,
                        exp_nam VARCHAR2)
RETURN ANYDATASET PIPELINED PARALLEL_ENABLE
  (
PARTITION inp_cur BY HASH(EQP_ID,UNIT_ID,LOT_ID,WAFER_ID,RECIPE,PARAM_ID))
CLUSTER inp_cur BY (EQP_ID,UNIT_ID,LOT_ID,WAFER_ID,RECIPE,PARAM_ID)
USING APGRPEVALIMPL;
```

High-Tech. FDC Trace Data

BDAE™ provides serialization for Python and R Models

ML (e.g. Support Vector Machine)

$$y(x) = w^T \phi(x) + b$$

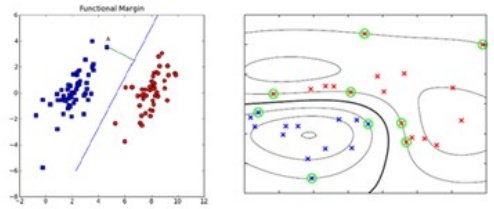
- According to Bishop's Machine Learning Book.

$$\arg\max_{w,b} \left\{ \min_n \left(label \cdot (w^T x + b) \right) \cdot \frac{1}{\|w\|} \right\}$$

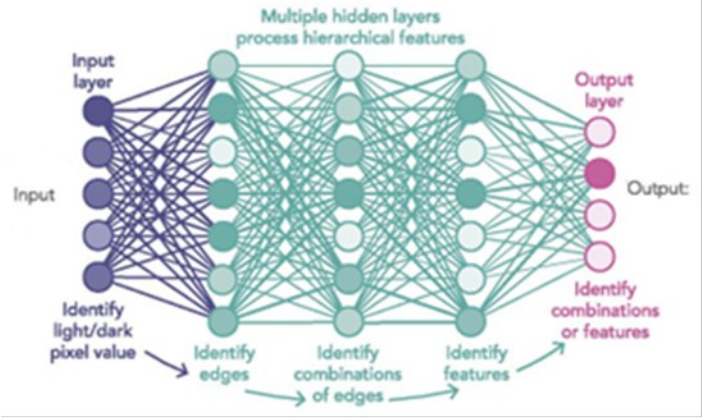
- Using Lagrange Multipliers

$$\max_{\alpha} \left[\sum_{i=1}^{m} \alpha - \frac{1}{2} \sum_{i,j=1}^{m} label^{(i)} \cdot label^{(j)} \cdot a_i \cdot a_j \langle x^{(i)}, x^{(j)} \rangle \right] \quad \text{Where} \quad \alpha \geq 0, and \sum_{i-1}^{m} \alpha_i \cdot label^{(i)} = 0$$

Applied by Slack variable $\quad c \geq \alpha \geq 0, and \sum_{i-1}^{m} \alpha_i \cdot label^{(i)} = 0$



Deep Learning Logistic Regression



$$y(m) = \sigma\left(\sum_{i=1}^{N} w_i x_i(m) + b \right)$$

$$w_j \leftarrow w_j - \frac{\eta}{B} \sum_{m=rB}^{(r+1)B} x_i(m)(y(m) - t(m)),\ 1 \leq j \leq N$$

$$b \leftarrow b - \frac{\eta}{B} \sum_{rB}^{(r+1)B} (y(m) - t(m))$$

$$L = -\frac{1}{V} \sum_{m=1}^{V} t(m) \ln y(m) + (1 - t(m) \ln(1 - y(m))$$

The second argument of the four SQLs provided by BDAE™ is responsible for reusing serialized ML/DL models. In other words, predictions, etc. can be performed using de-serialization, which stores the model itself (y=..) and uses it immediately for predictions !

BDAE, Best Practice

BDAE™ Best Practice for Massive Sensor Data

FDC Sensor Data are very big, Pandas DataFrame not fit handle this, because too many memories occupied and slow.

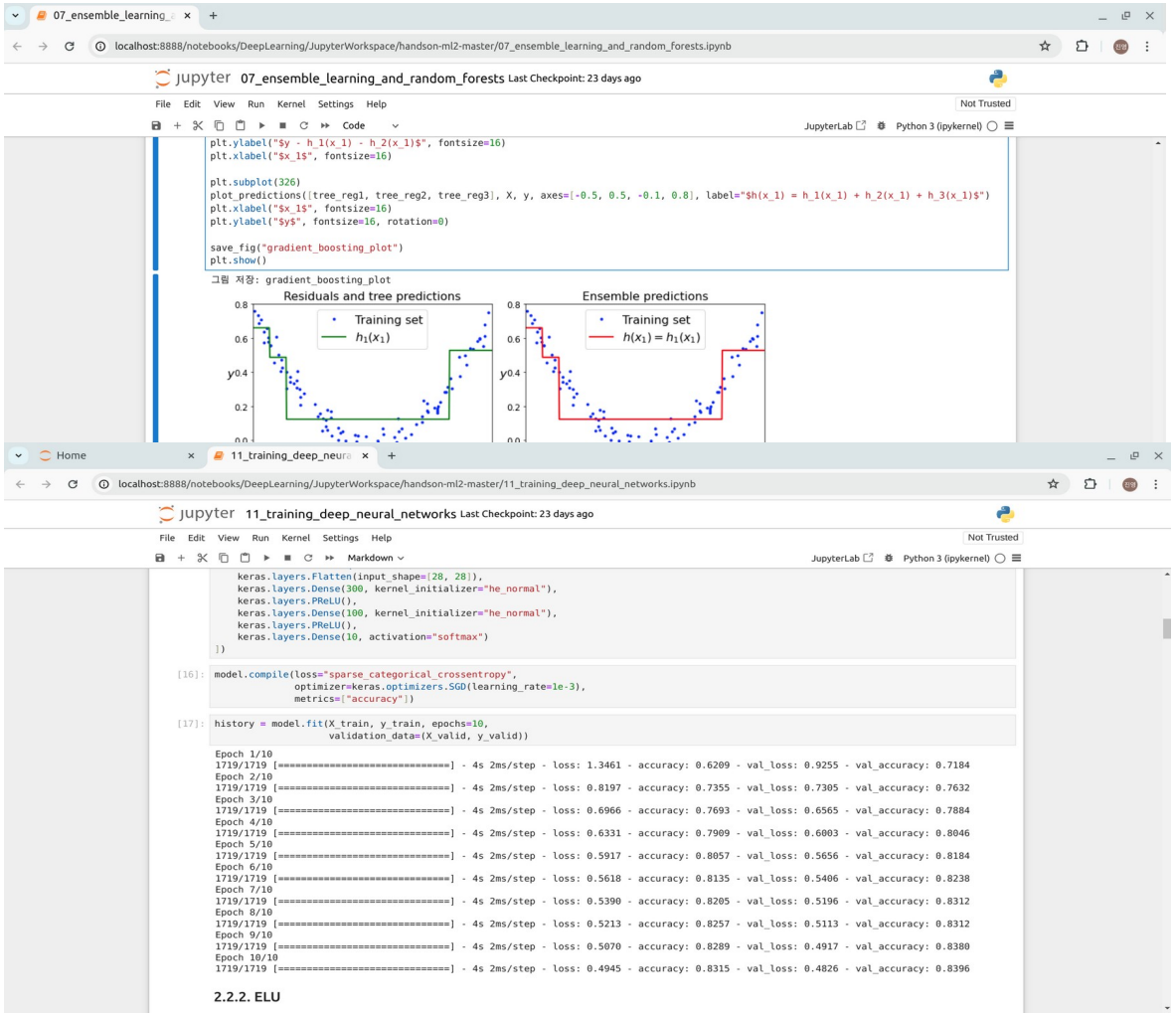
Oracle No. of Max Columns …. 1,000
Many Columns cause your system slow, ..
In case of FDC, Interface-A….

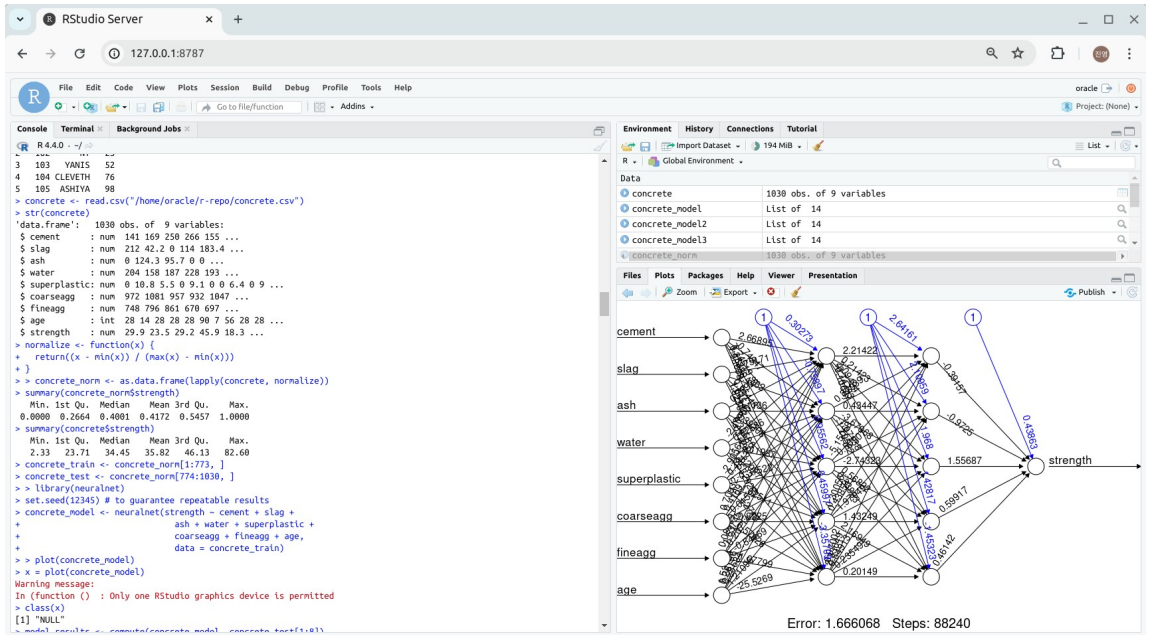
How ?

	EQP_ID	UNIT_ID	LOT_ID	WAFER_ID	RECIPE	PARAM_ID	TIMEKEY	VALUE
1	EQP-200	UNIT-02	LOTB-101	LOTB-101-01	RECIPE-200	param_b-36	time-765	142349.98956373942
2	EQP-200	UNIT-02	LOTB-101	LOTB-101-01	RECIPE-200	param_b-36	time-766	142340.37162073367
3	EQP-200	UNIT-02	LOTB-101	LOTB-101-01	RECIPE-200	param_b-36	time-767	142329.78790367243
4	EQP-200	UNIT-02	LOTB-101	LOTB-101-01	RECIPE-200	param_b-36	time-768	142320.36438901207
5	EQP-200	UNIT-02	LOTB-101	LOTB-101-01	RECIPE-200	param_b-36	time-769	142310.05912591403
6	EQP-200	UNIT-02	LOTB-101	LOTB-101-01	RECIPE-200	param_b-36	time-770	142298.66922700586
7	EQP-200	UNIT-02	LOTB-101	LOTB-101-01	RECIPE-200	param_b-36	time-771	142290.68556251022
8	EQP-200	UNIT-02	LOTB-101	LOTB-101-01	RECIPE-200	param_b-36	time-772	142280.16142178886
9	EQP-200	UNIT-02	LOTB-101	LOTB-101-01	RECIPE-200	param_b-36	time-773	142268.87071552715
10	EQP-200	UNIT-02	LOTB-101	LOTB-101-01	RECIPE-200	param_b-36	time-774	142260.75427998268
11	EQP-200	UNIT-02	LOTB-101	LOTB-101-01	RECIPE-200	param_b-36	time-775	142249.3583503445
12	EQP-200	UNIT-02	LOTB-101	LOTB-101-01	RECIPE-200	param_b-36	time-776	142240.21931166851
13	EQP-200	UNIT-02	LOTB-101	LOTB-101-01	RECIPE-200	param_b-36	time-777	142230.30554217254
14	EQP-200	UNIT-02	LOTB-101	LOTB-101-01	RECIPE-200	param_b-36	time-778	142220.21813966605
15	EQP-200	UNIT-02	LOTB-101	LOTB-101-01	RECIPE-200	param_b-36	time-779	142210.87260590505
16	EQP-200	UNIT-02	LOTB-101	LOTB-101-01	RECIPE-200	param_b-36	time-780	142200.64781496694
17	EQP-200	UNIT-02	LOTB-101	LOTB-101-01	RECIPE-200	param_b-36	time-781	142189.79580956438
18	EQP-200	UNIT-02	LOTB-101	LOTB-101-01	RECIPE-200	param_b-36	time-782	142179.84841075834
19	EQP-200	UNIT-02	LOTB-101	LOTB-101-01	RECIPE-200	param_b-36	time-783	142169.85659521911
20	EQP-200	UNIT-02	LOTB-101	LOTB-101-01	RECIPE-200	param_b-36	time-784	142159.58598922772
21	EQP-200	UNIT-02	LOTB-101	LOTB-101-01	RECIPE-200	param_b-36	time-785	142150.5230775126

Data Scientists' Development Environments..

Python : Jupyter Notebook or PyCharm, R : Rstudio





Data Scientists' Development Environments..

Python : Jupyter notebook or PyCharm, .. R : Rstudio

In the case of Python, the starting point of the module name (class, etc.) must also be provided. Therefore, it is best to do the following while making it.

```
def start_point_function(df_data, df_arg):
    …

    pdf = pd.DataFrame(…)
    return (pdf)
```

※ Indentation is very important when coding in Python. There is no major problem if you develop and paste it in Jupyter notebook. This is because Jupyter notebook automatically fills tab with space characters.

How to use BDAE ?

How many SQL Queries are needed for analysis?

In the case of apTableEval() and apGroupEval(), which are the most used, there are only two SQL Queries.

- Driving Table Query
- Argument Table Query

And in the Python analysis code, the results of these two queries are delivered as a pandas DataFrame.

If you create a **Python Wrapper module for these units of Query**, you can use it to **analyze something like multiple Queries**.

Installations (#1. OS)

Conditions for BDAE Installation

BDAE will also be built and installed on the OS where Oracle Database is installed, but the problem is the installation of Python and R.

1. **Microsoft Windows** is excluded because installing Oracle Database is implicitly **not recommended**.

2. The reason why **Oracle Linux is best** for Linux is because Oracle Exadata is based on this OS.

3. **UNIX (HP-UX, AIX) is a good OS**, but installation of **Python and R does not seem to be easy**, and above all, installation of NVIDIA GPU SDK, CUDA, etc. is not possible.

4. **BDAE is initially built** and distributed **according to** the Customer's **Linux version**, **Oracle Database version**, **Python version**, and **R version** to be used.

Installations

Conditions for BDAE Installation

5. The **Python version** is determined depending on the Python packages that **Customer want to use**, and using Anaconda, it is tested in the development environment and then copied into operation.

6. In rare cases, **additional Linux OS packages may be required** depending on the Python and R packages, and this setting is also required.

7. When installing a **GPU**, you must **proceed with the installation in the following order** and practice the installation several times before performing the final installation.

1) NVIDIA GPU CUDA, Toolkit Installation – Check
2) Customer’s Algorithm base on GPU Test and Check
3) Oracle Database Installation
4) BDAE Installation

Installations

Conditions for BDAE Installation

8. BDAE conducted installation tests in the following environment.

Oracle Enterprise Linux 6, 7, 8
CentOS 6, 7, 8 (not support now)
Oracle Database 10g, 11i, 12c, 19c, 23c
NVIDIA, lastest 12.x, (**NVIDA RTX3060 12GB**, GTX 1050 4GB, GTX 960 2GB)
Python 3.6, 3.8, 3.9
R 3.4.x, 4.4.0 (lastest 2024.6)
JAVA 1.8.x (OpenJDK)

※ Since the build occurs automatically when installing Python and R packages, installation errors may occur frequently. In particular, R may be more difficult to install because there are packages in several languages (C, C++, Fortran, ..).

In the case of Python, installation has become relatively easier thanks to Anaconda.

Summary

Additional comments about BDAE™

- BDAE™ is not an analysis solution, but the special tool that supports various solutions at a lower level.
- BDAE™ is designed to do little change to the analyst's algorithmic code.
- BDAE can be used various fields (e.g. MES, SPC, and QMS in general manufacturing (Smart Factory) to High-Tech. ..) It can be used as an Adds-On on most systems without system changes.
- BDAE is not only used for ML/DL, but can be used in various areas such as ETL, visualization, and more..
- The biggest advantage of BDAE is that it can be applied immediately and patches can be made during operation.

DEMO

Demo Web for Big Data Analysis Enabler™
but, DEMO Web not belong to BDAE

Simplest 3 Steps to use Big Data Analysis Enabler™

1) Data Scientists register Python or R Module.
   Data Scientist must test itself by Jupyter Notebook or any tools.
   All Out format must be Pandas DataFrame in case of Python
   In case of R, All Out format must be data.frame.

2) Register SQL using above module. This SQL output will be provided Input of Python or R.

3) Run (using Any SQL client program)