Session 7: Oracle R Enterprise 1.5.1 OAAgraph Package

Oracle Spatial and Graph PGX Graph Algorithms
Oracle R Technologies

Mark Hornick
Director, Advanced Analytics and Machine Learning
mark.hornick@oracle.com

October 2018

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

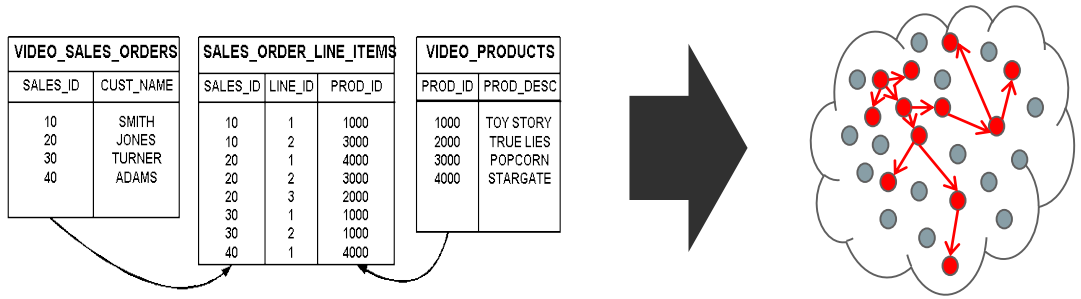
Topics

- Graph Analysis and Machine Learning
- ORE integration with Oracle Spatial and Graph option for PGX
- OAAgraph interface
- OAAgraph examples

Graph Analysis And Machine Learning

Graph Analysis

- A methodology in data analysis
- Represent your data as a graph
  - Data entities become nodes
  - Relationships become edges



VIDEO_SALES_ORDERS

SALES_ID	CUST_NAME
10	SMITH
20	JONES
30	TURNER
40	ADAMS

SALES_ORDER_LINE_ITEMS

SALES_ID	LINE_ID	PROD_ID
10	1	1000
10	2	3000
20	1	4000
20	2	3000
20	3	2000
30	1	1000
30	2	1000
40	1	4000

VIDEO_PRODUCTS

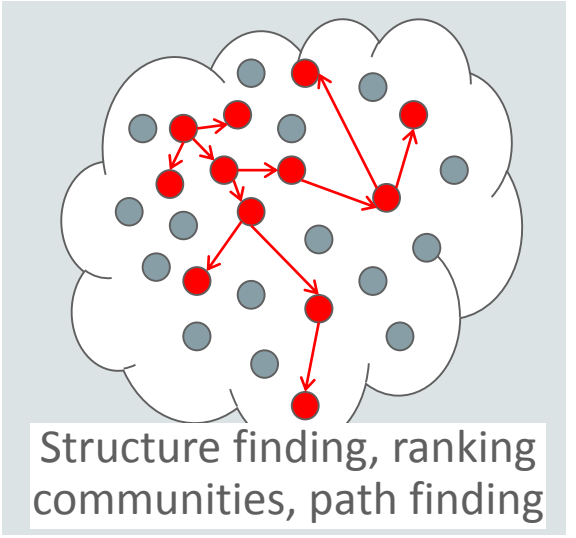
PROD_ID	PROD_DESC
1000	TOY STORY
2000	TRUE LIES
3000	POPCORN
4000	STARGATE

- Analyze fine-grained relationships through the graph
  - Navigate multi-hop relationships quickly
  - Without computing expensive joins repeatedly

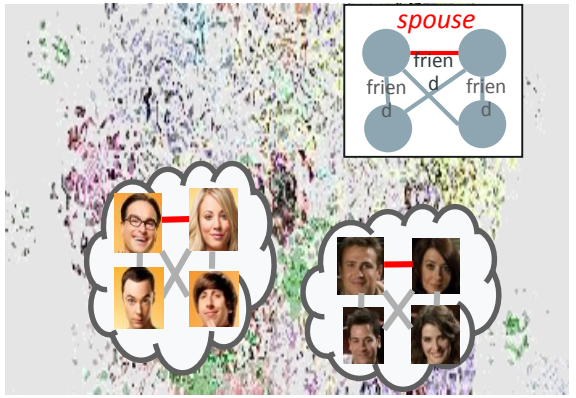
Graph Analysis

Inter-relationships between data and networks are growing in importance

- Graphs are everywhere
  - Facebook (friends of friends), Twitter, LinkedIn, etc.
    - Most data has inter-relationships that contain insights
- Two major types of graph algorithms
  - Computational Graph Analytics: Analysis of entire Graph
    - Influencer ID, community detect, pattern machine, recommendations
  - Graph Pattern Matching
    - Queries that find sub-graphs fitting relationship patterns



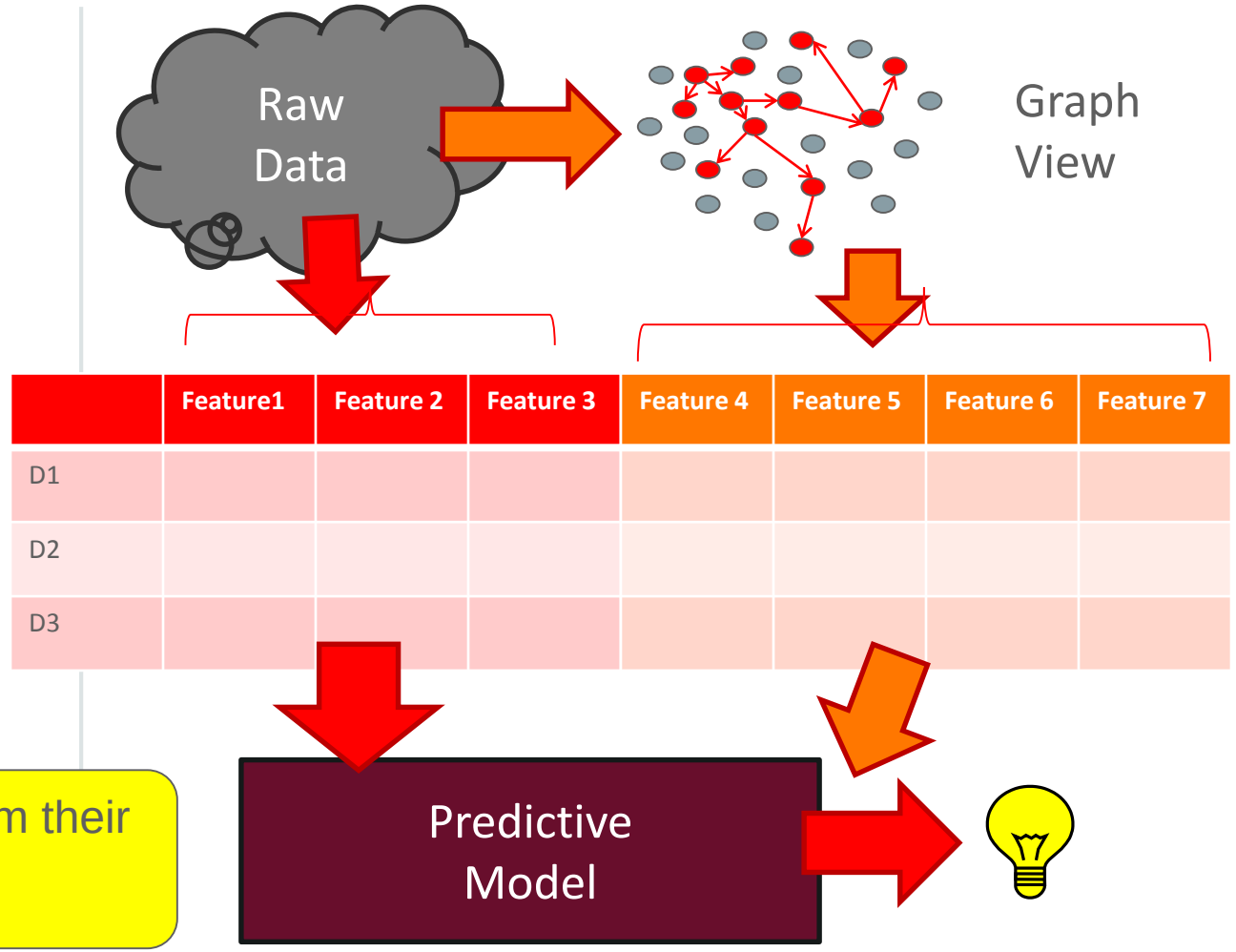
Structure finding, ranking communities, path finding



Graph Analysis and Machine Learning

- Graph analysis can augment Machine Learning
  - Typical machine learning techniques create/train models based on observed features
  - Graph analysis can provide additional *strong* signals
  - That make predictions more accurate

e.g. Can you identify groups of close customers from their call graph in order to predict customer churn?

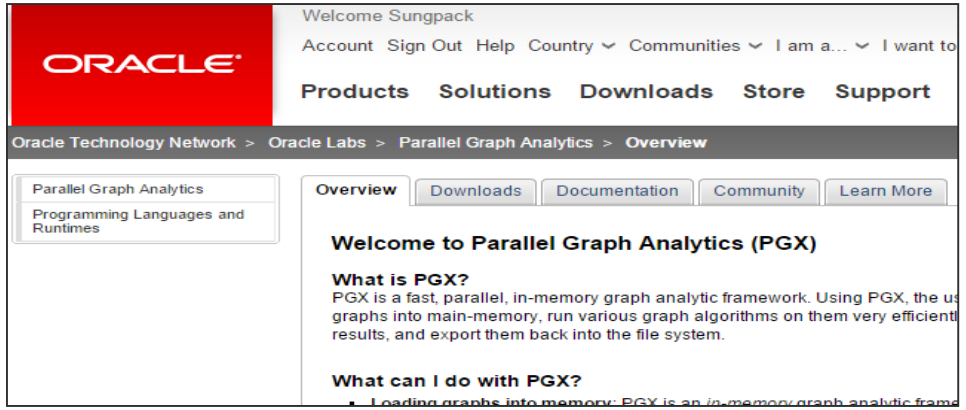


ORE integration with Oracle Spatial and Graph option for PGX

What is PGX?

- PGX (Parallel Graph Analytics)
  - An in-memory graph analysis engine
  - Originated from Oracle Labs
  - Provides fast, parallel graph analysis
    - Built-in Algorithm Packages
    - Graph Query (Pattern-Matching)
    - Custom Algorithm Compilation (Advanced Use case)
  - Integrated with Oracle Product(s)
    - Oracle Big Data Spatial and Graph (with BDA)
    - Property Graph Support at RDBMS 12.2c (Planned)
  - 35+ graph algorithms
  - Exceeds open source tool capabilities

OTN Technology Preview



Oracle Big Data and Spatial and Graph

PGX Graph Algorithms

- Ranking
  - Pagerank (+ variants)
  - Vertex Betweenness Centrality (including approximations)
  - Closeness Centrality
  - Eigenvector Centrality
  - Degree Centrality
  - Hyperlink-Induced Topic Search (HITS)
- Path Finding
  - Dijkstra (+ variants)
  - Bellman Ford (+ variants)
  - Hop Distance (+ variants)
  - Fattest path
- Partitioning
  - Weakly and Strongly Connected Components
  - Conductance and Modularity
  - Community Detection
- Recommendation
  - Twitter's whom-to-follow
  - Matrix Factorization
- Other
  - Breadth First Search with filter
  - Triangle Counting
  - Degree Distribution
  - K-core
  - Adamic Adar

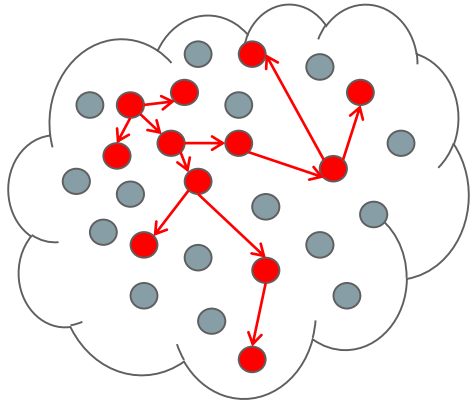
OAAgraph

An R interface integrating PGX and ORE/ORAAH for Machine Learning

Why an R interface to Graph?

- Single, unified interface across complementary technologies
  - Work with R data.frames and convenient functions across ML and graph
  - Results returned as R data.frames allows further processing in R env
- R users take advantage of multiple, powerful technologies
  - Highly scalable PGX engine on both Oracle Database and Hadoop
  - Integrated with **Oracle R Enterprise**, part of Oracle Database Advanced Analytics option
  - Integrated with **Oracle R Advanced Analytics for Hadoop**, part of Oracle Big Data Connectors

Graph Analytics



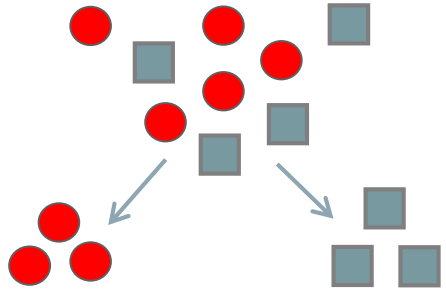
Compute graph metric(s)

Add to structured data

Explore graph or compute new metrics using ML result

Add to graph

Machine Learning



Build predictive model using graph metric

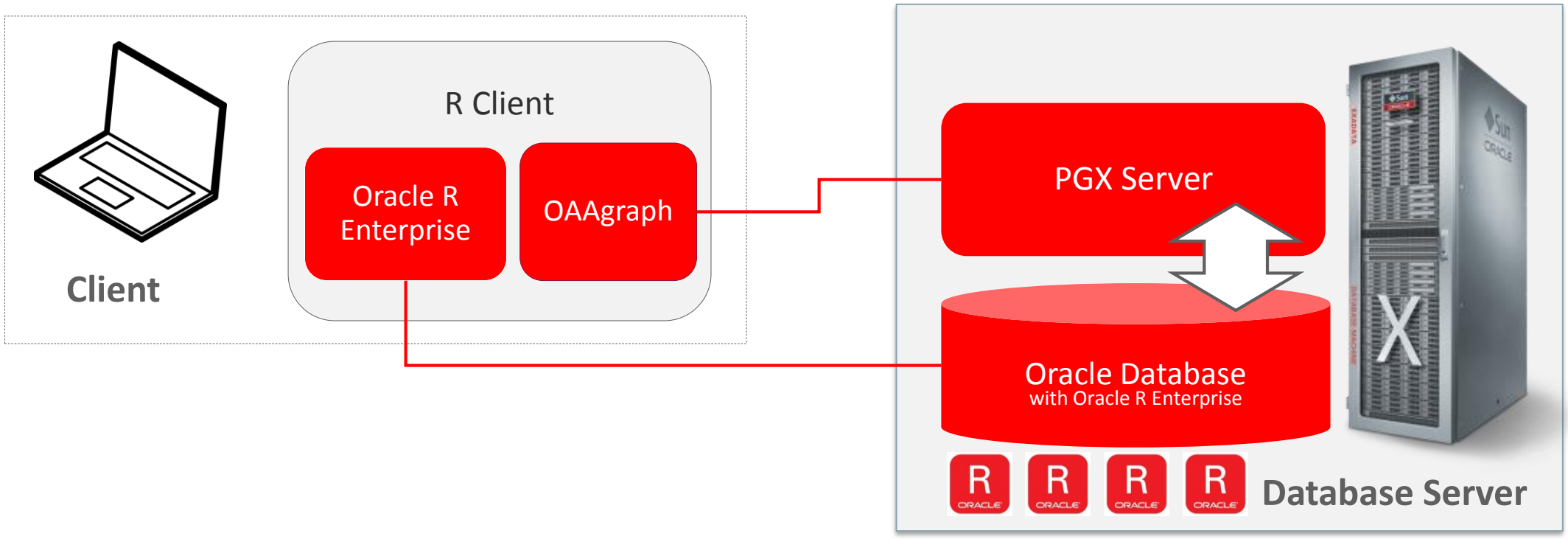
Build model(s) and score or classify data

Approach problem from two perspectives

*Requires Oracle Spatial and Graph option for PGX

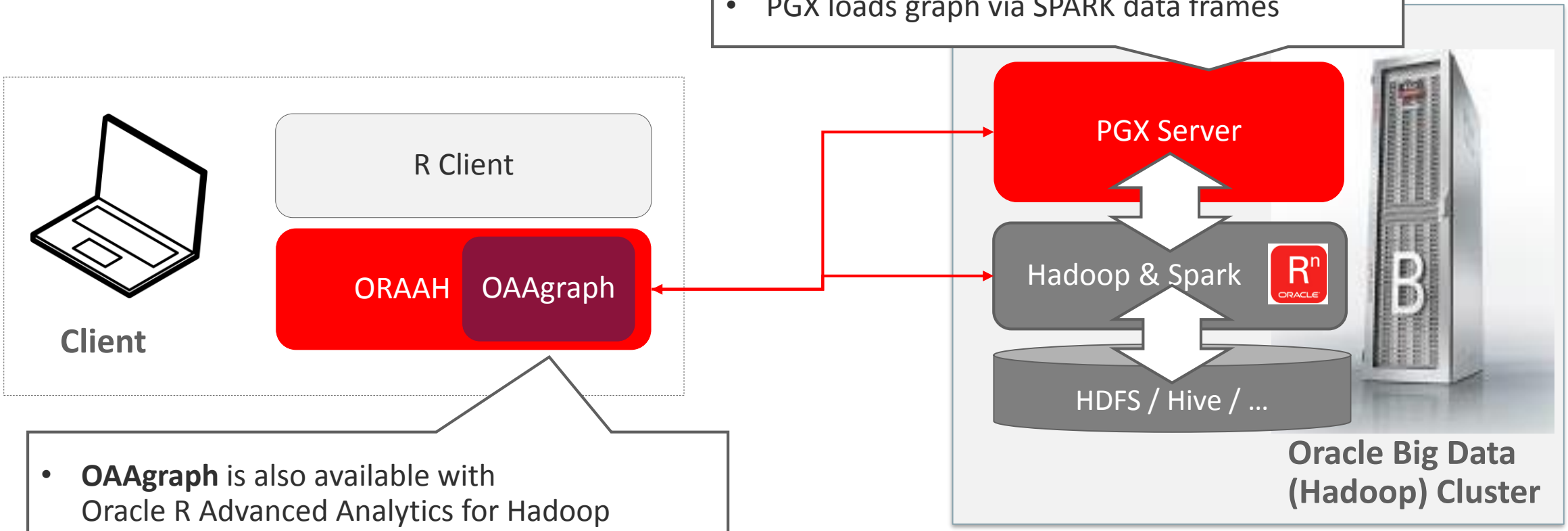
ORACLE

OAAgraph Architecture with Oracle Database



OAAgraph Architecture with Spark/Hadoop

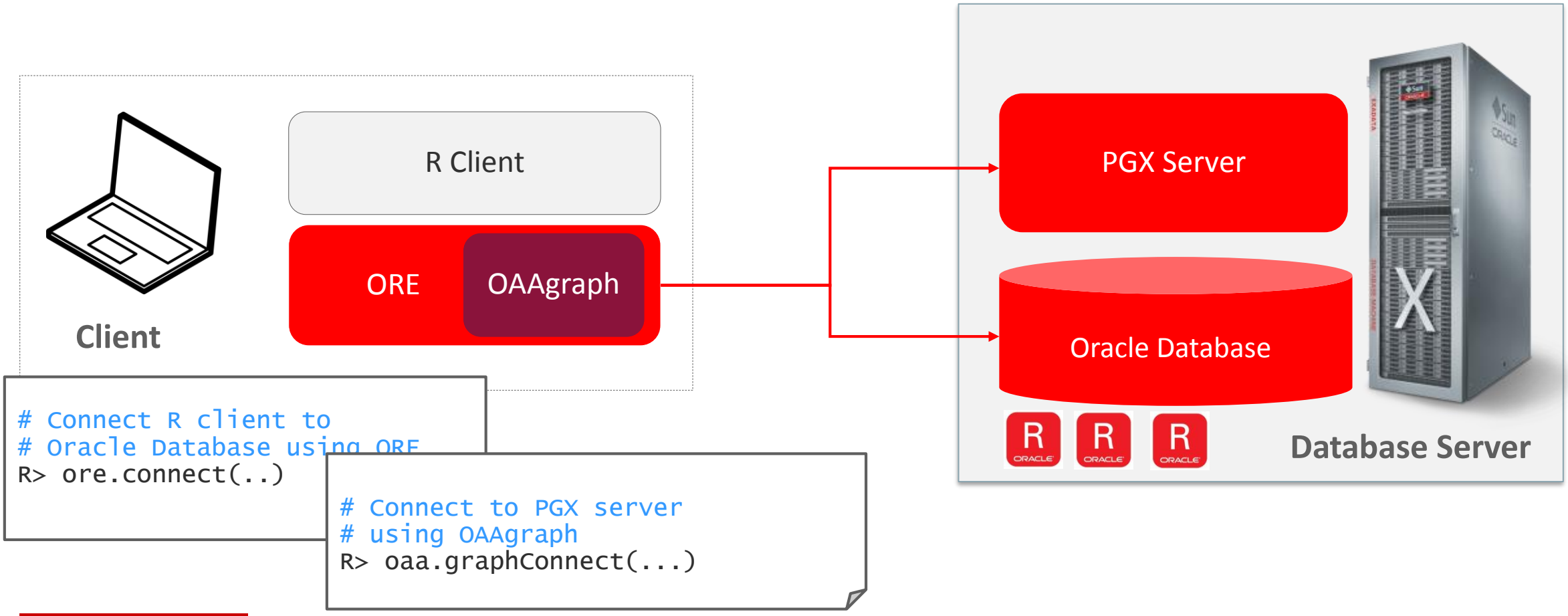
- **OAAgraph** gives remote control of PGX server
- PGX loads graph via SPARK data frames



- **OAAgraph** is also available with Oracle R Advanced Analytics for Hadoop

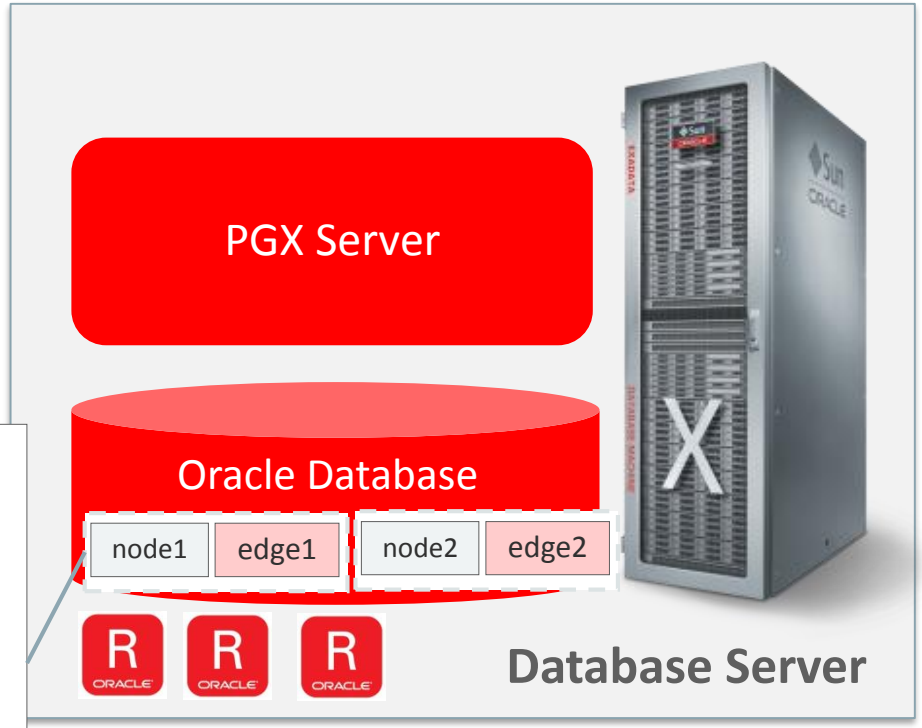
Execution Overview (ORE)

- Initialization and Connection



Execution Overview (ORE)

- Data Source
  - Graph data represented as two tables
    - Nodes and Edges
  - Multiple graphs stored in database
    - Using user-specified, unique table names



Node Table

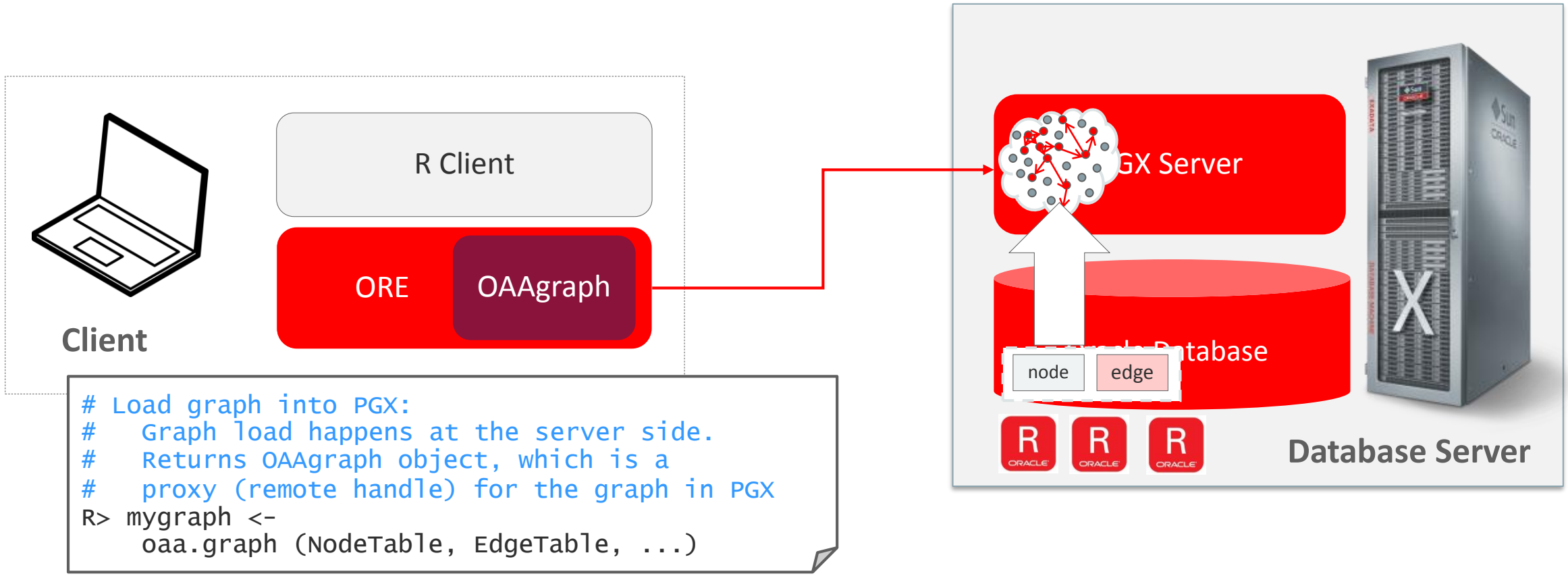
Node ID	Node Prop 1 (name)	Node Prop 2 (age)	…
1238	John	39	…
1299	Paul	41	…
4818	…	…	…

Edge Table

From Node	To Node	Edge Prop 1 (relation)	…
1238	1299	Likes	…
1299	4818	FriendOf	…
1299	6637	FriendOf	…

Execution Overview (ORE)

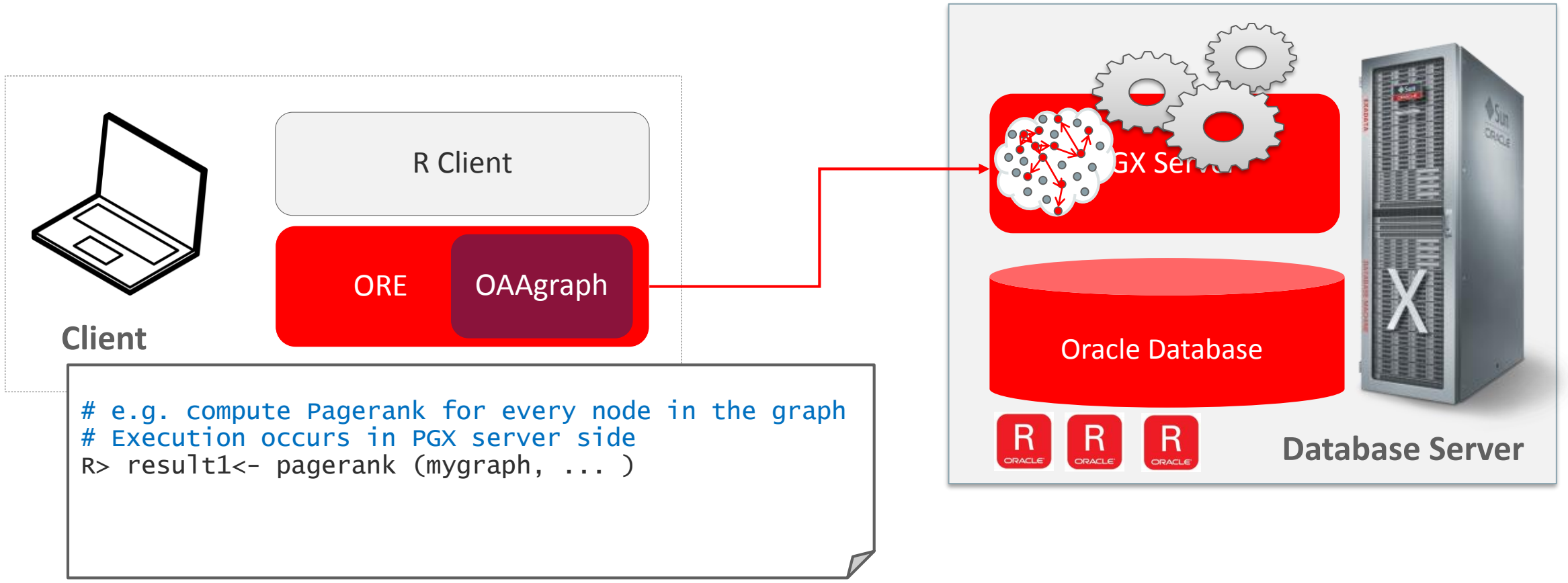
- Loading Graph




```
# Load graph into PGX:
#   Graph load happens at the server side.
#   Returns OAAgraph object, which is a
#   proxy (remote handle) for the graph in PGX
R> mygraph <-
    oaa.graph (NodeTable, EdgeTable, ...)
```

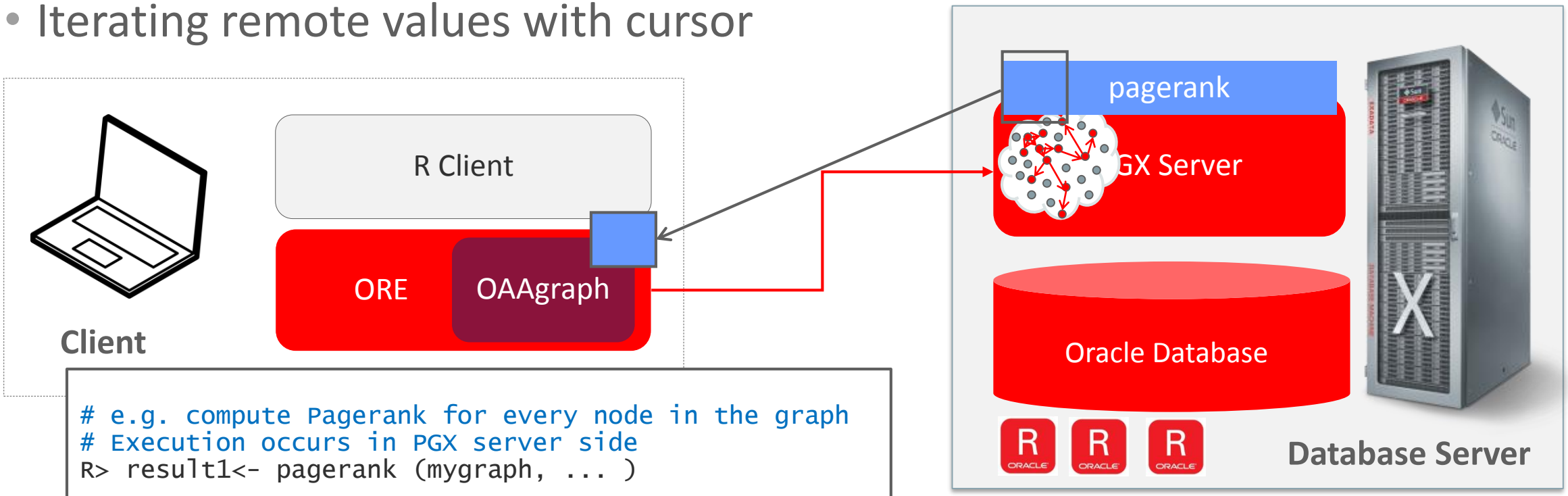
Execution Overview (ORE)

- Running Graph Algorithm



Execution Overview (ORE)

- Iterating remote values with cursor

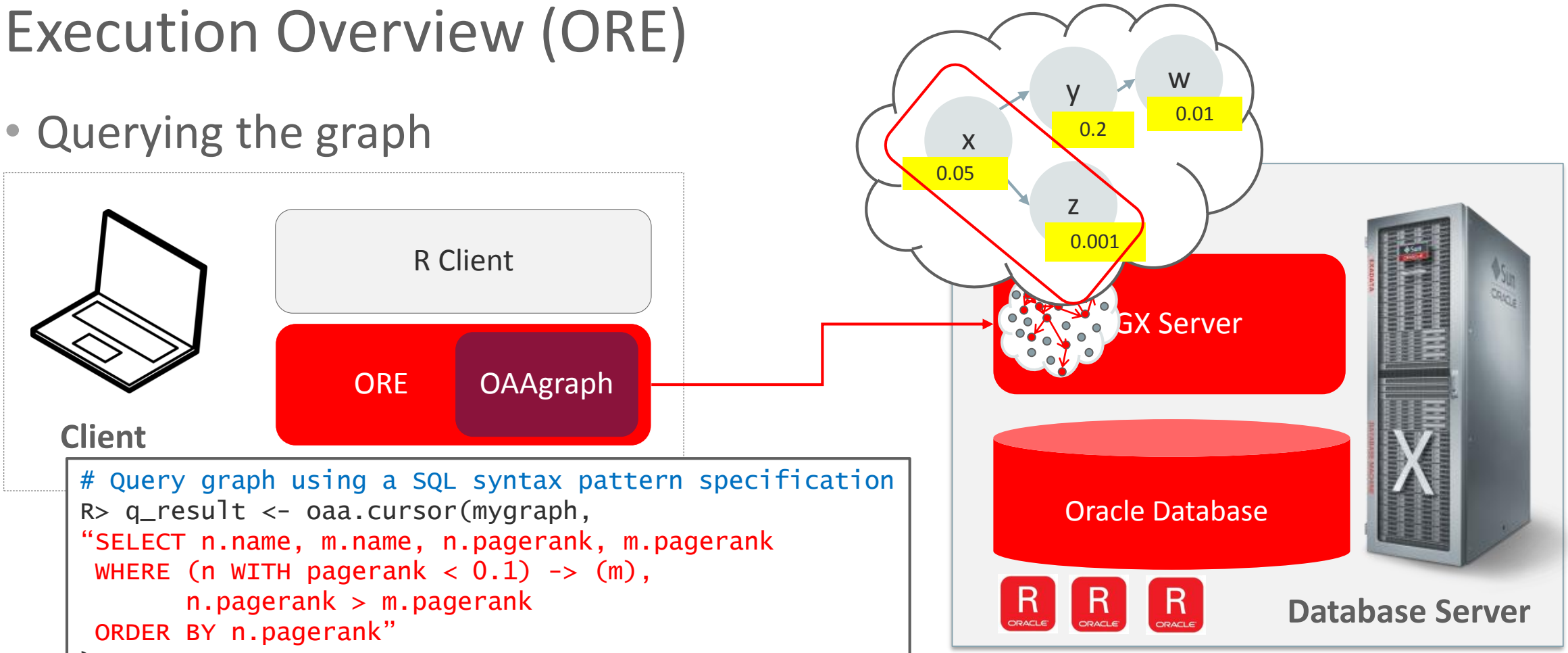



```
# e.g. compute Pagerank for every node in the graph
# Execution occurs in PGX server side
R> result1<- pagerank (mygraph, ... )

# Return value is a "cursor" object
# for the computed result:
#   client can get local data frames by oaa.next()
R> df <- oaa.next(result1, 10)
```

Execution Overview (ORE)

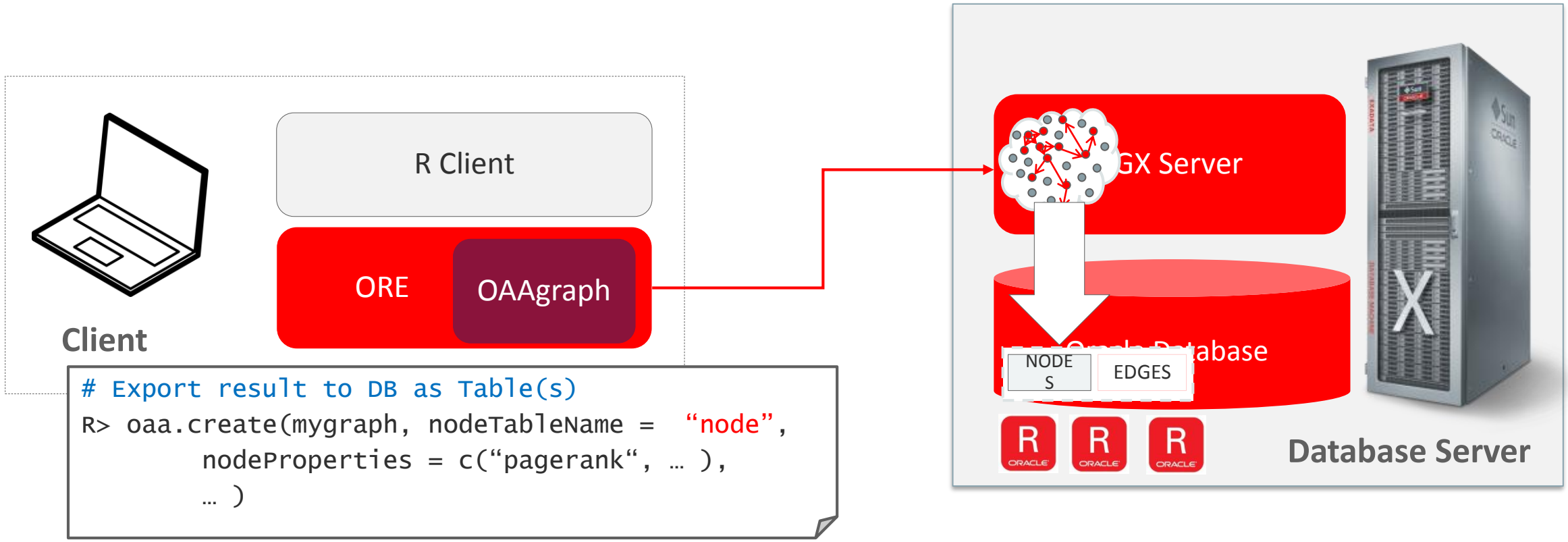
- Querying the graph




```
# Query graph using a SQL syntax pattern specification
R> q_result <- oaa.cursor(mygraph,
"SELECT n.name, m.name, n.pagerank, m.pagerank
 WHERE (n WITH pagerank < 0.1) -> (m),
       n.pagerank > m.pagerank
 ORDER BY n.pagerank"
)
# Returns a cursor to examine results
R> df <- oaa.next(q_result, 10)
```

Execution Overview (ORE)

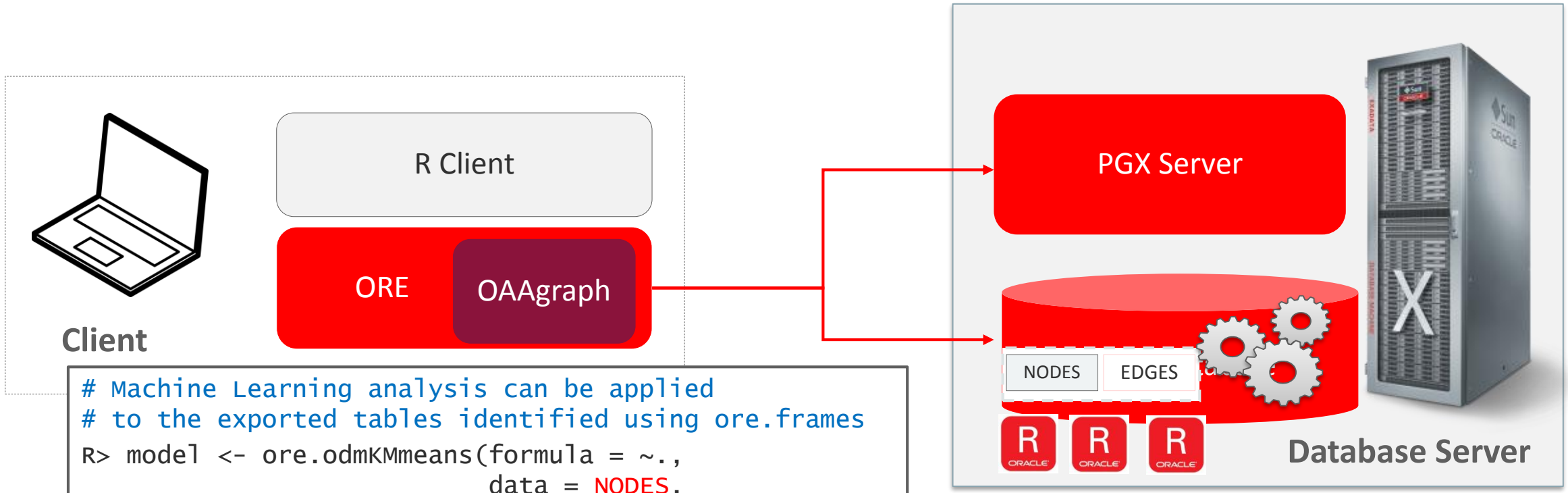
- Exporting the result to DB




```
# Export result to DB as Table(s)
R> oaa.create(mygraph, nodeTableName =  "node",
        nodeProperties = c("pagerank", … ),
        … )
```

Execution Overview (ORE)

- Continuing analysis with ORE Machine Learning




```
# Machine Learning analysis can be applied
# to the exported tables identified using ore.frames
R> model <- ore.odmKMmeans(formula = ~.,
                           data = NODES,
                           num.centers = 5,…)
R> scores <- predict(model, NODES, …)
…
```

PGX Performance

X86 Server Xeon E5-2660 2.2Ghz 2 socket x 8 cores x 2HT 256GB DRAM	#Nodes	#Edges	Closeness Centrality (PGX)	Closeness Centrality (igraph)	Betweenness Centrality (PGX)	Betweenness Centrality (igraph)
Epinion	75,879	811,480	17 sec	6.27 min	45 sec	11.5 min
Google	107,614	2,4476,570	15 sec	132 min	30 min	290 min
Churner	3,000,000	16,519,402	26 min	5+ days	20 hrs	5+ days
Patent	3,774,768	33,037,894	6.6 hrs	5+ days	48 hrs	5+ days
LiveJ	4,846,609	85,702,474	5.8 hrs	5+ days	60 hrs	5+ days

Effectively does not complete!

OAAgraph Examples

Example: Connecting to PGX

```
library(ORE)
library(OAAgraph)

dbHost     <- "<DATABASE_HOST>"
dbUser     <- "<DATABASE_USERNAME>"
dbPassword <- "<DATABASE_PASSWORD>"
dbSid      <- "<DATABASE_SID>"
pgxBaseUrl <- "<PGX_BASE_URL>"
ore.connect(host = dbHost, user = dbUser, password = dbPassword, sid = dbSid)
oaa.graphConnect(pgxBaseUrl = pgxBaseUrl, dbHost = dbHost,
                 dbSid = dbSid, dbUser = dbUser, dbPassword = dbPassword)
```

Example: Create some data – node and edge tables

```
#-- Create the node table in Oracle Database
VID <- c(1, 2, 3, 4, 5)
NP1 <- c("node1", "node2", "node3", "node4", "node5")
NP2 <- c(111.11, 222.22, 333.33, 444.44, 555.55)
NP3 <- c(1, 2, 3, 4, 5)
nodes <- data.frame(VID, NP1, NP2, NP3)
ore.drop(table="MY_NODES")
ore.create(nodes, table = "MY_NODES")

#-- Create the edge table in Oracle Database
EID <- c(1, 2, 3, 4, 5)
SVID <- c(1, 3, 3, 2, 4)
DVID <- c(2, 1, 4, 3, 2)
EP1 <- c("edge1", "edge2", "edge3", "edge4", "edge5")
EL <- c("label1", "label2", "label3", "label4", "label5")
edges <- data.frame(EID, SVID, DVID, EP1, EL)
ore.drop(table="MY_EDGES")
ore.create(edges, table = "MY_EDGES")
```

Example: Create some data – node and edge tables

```
#-- Create a graph in PGX from the node and edge tables in the database
graph <- oaa.graph(MY_EDGES, MY_NODES, "myPgxGraph")
names(graph, "nodes")
names(graph, "edges")

#-- See result of countTriangles function, which gives an overview of the
#--   number of connections between nodes in neighborhoods
countTriangles(graph, sortVerticesByDegree=FALSE)

#-- See results from degree algorithm variants, note the graph nodes
#--   are augmented with new properties as indicated by the 'name' argument
degree(graph, name = "OutDegree")
degree(graph, name = "InDegree", variant = "in")
degree(graph, name = "InOutDegree", variant = "all")
```

Example: Load graph from database tables

```
#-- Create a graph in PGX from the node and edge tables in the database
graph <- oaa.graph(MY_EDGES, MY_NODES, "myPgxGraph")
names(graph, "nodes")
names(graph, "edges")

#-- See result of countTriangles function, which gives an overview of the
#--   number of connections between nodes in neighborhoods
countTriangles(graph, sortVerticesByDegree=FALSE)

#-- See results from degree algorithm variants, note the graph nodes
#--   are augmented with new properties as indicated by the 'name' argument
degree(graph, name = "OutDegree")
degree(graph, name = "InDegree", variant = "in")
degree(graph, name = "InOutDegree", variant = "all")
```

Example: Create cursors to access the results

```
#-- Create a cursor including the degree properties
cursor <- oaa.cursor(graph, c("OutDegree", "InOutDegree", "InDegree"), "nodes")
oaa.next(cursor, 5)

#-- Create a cursor over the degree properties using
#--    the PGX SQL-like query language PGQL
cursor <- oaa.cursor(graph,
                     query = "select n.OutDegree, n.InOutDegree, n.InDegree where (n)
  order by n.OutDegree desc")

#-- View the first 5 entries from the cursor
oaa.next(cursor, 5)
```

Example: Create cursors to access the results

```
#-- See results from the pagerank algorithm
pagerankCursor <- pagerank(graph, 0.085, 0.1, 100)
oaa.next(pagerankCursor, 5)

#-- Create a cursor over the pagerank property using PGQL
cursor <- oaa.cursor(graph,
                     query = "select n.pagerank where (n) order by n.pagerank desc")
oaa.next(cursor, 5)

#-- This could be done using the R interface as well...
cursor <- oaa.cursor(graph, "pagerank", ordering="desc")
oaa.next(cursor, 5)
```

Example: More graph analytics and create snapshots

```
#-- Compute the adamic adar index for edges
topEdges <- adamicAdarCounting(graph)
oaa.next(topEdges)

#-- List any graph snapshots available
oaa.graphSnapshotList()

#-- Export a binary snapshot of the whole graph into Oracle Database
#--   and view the listing again
oaa.graphSnapshotPersist(graph, nodeProperties = TRUE, edgeProperties = TRUE)
oaa.graphSnapshotList()
```

Example: Read snapshots and create tables from graphs

```
#-- Read the snapshot back into memory
graph2 <- oaa.graphSnapshot("myPgxGraph")

#-- Export the graph nodes and specific node properties from memory into a database table
oaa.create(graph2, nodeTableName = "RANKED_NODES", nodeProperties = TRUE)

#-- Export both nodes and edges as tables from memory into the database,
#--    but only export the pagerank node property
oaa.create(graph2, nodeTableName = "RANKED_GRAPH_N",
           nodeProperties = c("NP1", "pagerank"),
           edgeTableName = "RANKED_GRAPH_E")

#-- Export graph edges and their properties from memory into a database table
oaa.create(graph2, edgeTableName = "RANKED_EDGES", edgeProperties = TRUE)

#-- Free the graphs at the PGX server
oaa.rm(graph)
oaa.rm(graph2)
```

Two options to get more debugging output…

1. Before starting R

```
export DEBUG=true
```

2. In OAAgraph client:

```
oaa.log("debug")
```

Learn More about Oracle's R Technologies…

http://oracle.com/goto/R

R Technologies from Oracle

Bringing the Power of R to the Enterprise


ORACLE®