Combining Graph and Machine Learning Technology using R

Hassan Chafi
Oracle Labs

Mark Hornick
Oracle Advanced Analytics

February 2, 2017
Safe Harbor Statement

The following is intended to outline our research activities and 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.
Contents

• Graph Analysis and Machine Learning
  – Graph Analysis and Applications
  – Combining Graph Analysis with Machine Learning

• OAA.Graph
  – ORE/ORAAH and PGX
  – Integration

• Demo
Graph Analysis And Machine Learning
Big Data and Data Analysis

• The Big Data era is here
  – Volume
  – Velocity
  – Variety

• However, just storing and managing this data is not sufficient
  – Typically Big Data is low value per byte

➤ We want to get useful information out of the huge data sets

Methodologies:
• Classic OLAP
• Statistical analysis
• Machine learning
• Graph analysis
Graph Analysis

• A methodology in data analysis
• Represent your data as a graph
  – Data entities become nodes
  – Relationships become edges
• 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
Graph Analysis Examples

**Example Application**

- Money laundering pattern detection in bank applications
- Identify a chain of wire transfers, including an external entity, between two accounts of a single owner

➤ Graph pattern matching with cycle detection

Reachability Analysis

Find out how data entities are connected with each other via multiple hops
Graph Analysis Examples

**Centrality Analysis**
Identifying important entities from connections between data entities

**Example Application**
- Computer network vulnerability analysis
- Identify network components whose failure would cause the largest damage

→ *Betweenness Centrality* Computation
Graph Analysis Examples

• Example Application
  – Product recommendation for retail
  – Given an item, identify close items from user-item or item-feature graph and recommend those items
  – Given a user, identify close users who purchased similar items and recommend items popular among those

  ➔ Matrix (Graph) Factorization, Personalized Pagerank, ...
Graph Analysis Examples

**Example Application**
- Classification of data entities based on their relationship
- E.g. classify students from the same department by the courses that they take

➤ Label propagation, Relax Map, ...

**Community Analysis**
Identify grouping of data entities from their interconnection structure

---

<table>
<thead>
<tr>
<th>course_department</th>
<th>course_title</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dept of Math/CS</td>
<td>Data Structures</td>
<td>114</td>
</tr>
<tr>
<td>Dept of Math/CS</td>
<td>Computer Organization</td>
<td>104</td>
</tr>
<tr>
<td>Dept of Math/CS</td>
<td>Principles of Programming II</td>
<td>102</td>
</tr>
<tr>
<td>Dept of Math/CS</td>
<td>Database Design I</td>
<td>91</td>
</tr>
<tr>
<td>Dept of Math/CS</td>
<td>Operating Systems</td>
<td>89</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>course_department</th>
<th>course_title</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dept of Math/CS</td>
<td>Ordinary Differential Equation</td>
<td>72</td>
</tr>
<tr>
<td>Dept of Math/CS</td>
<td>Set Theory</td>
<td>71</td>
</tr>
<tr>
<td>Dept of Math/CS</td>
<td>Probability and Statistics</td>
<td>65</td>
</tr>
<tr>
<td>Dept of Math/CS</td>
<td>Linear Algebra</td>
<td>55</td>
</tr>
<tr>
<td>Dept of Math/CS</td>
<td>Modern Algebra I</td>
<td>53</td>
</tr>
</tbody>
</table>
Graph Analysis and Other Data Analyses

• Naturally, graph analysis pairs well with other data analyses
  – Traditional analysis steps favors tabular data representation
  – Graph analysis can occur as a separate data processing step
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?
Example – SNS Stream analysis

• Twitter streams can often be manipulated to achieve some goal
  – Social and viral marketing (or alternative fact based news)
  – True view on trends can be polluted by these streams
  – How can we eliminate such noise?

Different accounts re-tweet messages (mostly) from certain accounts

These are bots to make other accounts look more important
Data Processing Steps: Creating Graph

- Create graph representation from tables data set

<table>
<thead>
<tr>
<th>Authors</th>
<th>Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>name</td>
</tr>
<tr>
<td>0x01</td>
<td>John</td>
</tr>
<tr>
<td>0x02</td>
<td>Jane</td>
</tr>
<tr>
<td>0x03</td>
<td>Bob</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Message body</th>
<th>message_id</th>
<th>analysis_group_id</th>
<th>snippet</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x01</td>
<td>0x01</td>
<td>I like #Mercedes</td>
</tr>
<tr>
<td>0x02</td>
<td>0x02</td>
<td>0x01</td>
<td>RT @john &quot;I like #...</td>
</tr>
<tr>
<td>0x03</td>
<td>0x03</td>
<td>0x02</td>
<td>Regarding RT, I still ...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Topics</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td></td>
</tr>
<tr>
<td>0x01</td>
<td>Mercedes</td>
</tr>
<tr>
<td>0x02</td>
<td>Cadillac</td>
</tr>
</tbody>
</table>

Authors

- John
- Jane

Messages

- I like #Mercedes
- @RT John "I like #Mercedes"
- Regarding RT, I still think the new #Cadillac is way cooler

Topic

- Mercedes
- Cadillac

Post

- about
- retweet

Copyright © 2017 Oracle and/or its affiliates. All rights reserved.
Data Processing Steps: Analyzing Graph

• Analyze Graph
  – Extract Retweet (RT) between accounts
  – Focus only on Total RT counts between accounts
Data Processing Steps: Statistical Analysis

- Compute objective function for top-100 RT’ed accounts
- Identify anomalies from simple statistical methods

Certain accounts have obviously unnatural deviation
Data Processing Steps: Analyzing Graph

- Resume from previous step’s results
- Already identified targets (among top RT’ed accounts) and bots
- Analyze graph even further to identify more target accounts
Analysis Result

Data Set

• Data acquisition period: 2016-Feb (1 week)
• Number of topics: 675
• Number of messages: 2.6 million
• Number of accounts: 788,360

• # RT Bots: 3,092
• # RT Beneficiaries: 5
• # Removed messages: 551,177

⇒ Significant changes in important accounts and trends
What is PGX (part of BDSG)?

• 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
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
PGX Performance (Algorithm Computation)

- Comparisons against existing graph engines
  - GraphX (Spark)
  - GraphLab (Dato)

- With seven popular algorithms
  - Pagerank (exact and approx), Weakly Connected Components, Single-Source Shortest Path, Hop-Distance (BFS), Eigen Vector, K-Core

- On Two Graph instances
  - Twitter Graph (TWT): 41 million nodes, 1.4 Billion edges
  - Web Graph (WEB): 77 millions nodes, 2.9 Billion edges

Hardware: Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz - 256 RAM
Network: Melanox Infiniband (56Gbps)
PGX Performance (Algorithm Computation)

Comparisons against existing graph engines:
- GraphX (Spark)
- GraphLab (Dato)

With seven popular algorithms:
- Pagerank (exact and approx)
- Weakly Connected Components
- Single-Source Shortest Path
- Hop-Distance (BFS)
- Eigen Vector
- K-Core

On Two Graph instances:
- Twitter Graph (TWT): 41 million nodes, 1.4 Billion edges
- Web Graph (WEB): 77 millions nodes, 2.9 Billion edges

Hardware: Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz - 256 RAM
Network: Melanox Infiniband (56Gbps)
PGX Performance (Query) vs. Neo4j

Hardware: 88 x Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz - 256 RAM
Warmup: ignore the first two runs, measure the third run
Index usage: no special indexes (use defaults)

Numbers shown are “hot” numbers: Neo4j is no longer accessing disk.
PGX (Single Node) Performance on SPARC

SPARC M7 up to 1.5x faster per core than x86

<table>
<thead>
<tr>
<th>Graph Algorithm</th>
<th>Workload Size</th>
<th>4-chip X86 E5 v3</th>
<th>4-chip SPARC T7-4</th>
<th>SPARC per chip Advantage</th>
<th>SPARC per core Advantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSSP Bellman-Ford</td>
<td>448M vertices, 17.2B edges</td>
<td>39.2s</td>
<td>14.7s</td>
<td>2.7x</td>
<td>1.5x</td>
</tr>
<tr>
<td></td>
<td>233M vertices, 8.6B edges</td>
<td>21.3s</td>
<td>8.5s</td>
<td>2.5x</td>
<td>1.4x</td>
</tr>
<tr>
<td>PageRank</td>
<td>448M vertices, 17.2B edges</td>
<td>136.7s</td>
<td>62.6s</td>
<td>2.2x</td>
<td>1.2x</td>
</tr>
<tr>
<td></td>
<td>233M vertices, 8.6B edges</td>
<td>72.1s</td>
<td>27.6s</td>
<td>2.6x</td>
<td>1.5x</td>
</tr>
</tbody>
</table>

• Graph computations accelerated by SPARC’s memory bandwidth
  – Bellman-Ford/SSSP (single-source shortest path) – optimal route or connection
  – PageRank - measuring website importance
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)

Explore graph or compute new metrics using ML result

Add to graph

Machine Learning

Add to structured data

Build predictive model using graph metric

Build model(s) and score or classify data

Add to graph
Oracle R Enterprise

• Use Oracle Database as a high performance compute environment
• Transparency layer
  – Leverage proxy objects (ore.frames) - data remains in the database
  – Overload R functions that translate functionality to SQL
  – Use standard R syntax to manipulate database data
• Parallel, distributed algorithms
  – Scalability and performance
  – Exposes in-database machine learning algorithms from ODM
  – Additional R-based algorithms executing and database server
• Embedded R execution
  – Store and invoke R scripts in Oracle Database
  – Data-parallel, task-parallel, and non-parallel execution
  – Use open source CRAN packages
### Machine Learning algorithms in-Database

**Classification**
- Decision Tree
- Logistic Regression
- Naïve Bayes
- Support Vector Machine
- RandomForest

**Regression**
- Linear Model
- Generalized Linear Model
- Multi-Layer Neural Networks
- Stepwise Linear Regression
- Support Vector Machine

**Clustering**
- Hierarchical k-Means
- Orthogonal Partitioning Clustering

**Market Basket Analysis**
- Apriori – Association Rules

**Attribute Importance**
- Minimum Description Length

**Anomaly Detection**
- 1 Class Support Vector Machine

**Feature Extraction**
- Nonnegative Matrix Factorization
- Principal Component Analysis
- Singular Value Decomposition

**Time Series**
- Single Exponential Smoothing
- Double Exponential Smoothing

---

...plus open source R packages for algorithms in combination with embedded R data- and task-parallel execution
Oracle R Advanced Analytics for Hadoop

Using Hadoop/Hive/Spark Integration, plus R Engine and Open-Source R Packages

Oracle R Advanced Analytics for Hadoop (ORAAH)
on Hadoop Cluster

R interface to HQL Basic Statistics, Data Prep, Joins and View creation

Parallel, distributed algorithms:
- ORAAH Spark algorithms: Deep Neural Network, GLM, LM
- Spark MLlib algorithms: LM, GLM, LASSO, Ridge Regression, Decision Trees, Random Forests, SVM, k-Means, PCA

Use of Open-source R packages via custom R Mappers / Reducers

R Client

R Analytics
Oracle R Advanced Analytics for Hadoop

SQL Client

SQL Developer
Other SQL Apps

Copyright © 2017 Oracle and/or its affiliates. All rights reserved.
## Oracle R Advanced Analytics for Hadoop 2.7.0

### Machine Learning algorithms

<table>
<thead>
<tr>
<th>Classification</th>
<th>Regression</th>
<th>Feature Extraction</th>
<th>Attribute Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLM ORAAH</td>
<td>MLP Neural Networks ORAAH</td>
<td>Non-negative Matrix Factorization</td>
<td>Principal Components Analysis</td>
</tr>
<tr>
<td>Logistic Regression ORAAH</td>
<td>LASSO</td>
<td>Collaborative Filtering (LMF)</td>
<td></td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>Ridge Regression</td>
<td>Singular Value Decomposition</td>
<td></td>
</tr>
<tr>
<td>Random Forests</td>
<td>Support Vector Machines</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision Trees</td>
<td>Random Forest</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support Vector Machines</td>
<td>Linear Regression</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Clustering</strong></td>
<td><strong>Basic Statistics</strong></td>
<td><strong>Feature Extraction</strong></td>
<td><strong>Attribute Importance</strong></td>
</tr>
<tr>
<td>Hierarchical k-Means</td>
<td>Correlation/Covariance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hierarchical k-Means</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gaussian Mixture Models</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**OAAgraph Architecture with Oracle Database**

- **OAAgraph** gives remote control of PGX server
- PGX loads graph from database (ore.frames)

- **OAAgraph** is an additional R package that comes with Oracle R Enterprise
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

# Connect R client to Oracle Database using ORE
R> ore.connect(..)

# Connect to PGX server using OAAgraph
R> oaa.init(..)
R> oaa.graphConnect(...)

Client

R Client

ORE

OAAngraph

Database Server

PGX Server

Oracle Database
**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

<table>
<thead>
<tr>
<th>Node Table</th>
<th>Edge Table</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Node</strong></td>
<td><strong>From</strong></td>
</tr>
<tr>
<td><strong>Prop 1</strong></td>
<td><strong>Node</strong></td>
</tr>
<tr>
<td><strong>ID</strong></td>
<td><strong>To Node</strong></td>
</tr>
<tr>
<td>1238</td>
<td>John</td>
</tr>
<tr>
<td>1299</td>
<td>Paul</td>
</tr>
<tr>
<td>4818</td>
<td>...</td>
</tr>
</tbody>
</table>

**Database Server**

- Oracle Database
- PGX Server
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, ...)

Copyright © 2017 Oracle and/or its affiliates. All rights reserved.
Execution Overview (ORE)

• Running Graph Algorithm

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

```r
# 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) \rightarrow (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

```r
# 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.odmKMeans(formula = ~.,
   data = NODES,
   num.centers = 5,...)
R> scores <- predict(model, NODES, ...)
...
Demo

• Environment
  – OAAGraph with ORAAH
  – PGX + SPARK + HDFS

• Dataset
  – Persons: name, age, zip, ...
  – Calls: phone calls person-to-person
Demo Scenario

- Load *persons* data into ORAAH
- Check the data set
- Cluster *persons* by their age with K-means

- Load *calls* data into ORAAH
- Create an OAAgraph object with *persons* and *calls*
- Compute Pagerank and check results

- Export results back to ORAAH
- Cluster *persons* by their age AND pagerank values (with K-means)
Summary

• Powerful, scalable graph analytics enabled from R
• Cross-pollinate graph analytics and machine learning
Hardware and Software
Engineered to Work Together
RDBMS queries (1)

Q1
\[
\text{SELECT } n \text{ WHERE } (n:\text{macro WITH name = 'ksqget'}) \text{ ORDER BY } n
\]

Q2
\[
\text{SELECT } n \text{ WHERE } (n:\text{macro WITH name =~ 'EVTDV\$'}) \text{ ORDER BY } n
\]

Q3
\[
\text{SELECT } f.\text{name WHERE} \ (n:\text{macro WITH name = 'ksqget'}) \leftarrow[c:\text{expands}] \Rightarrow (f:\text{source_file}) \text{ ORDER BY } f.\text{name LIMIT 10}
\]

Q4
\[
\text{SELECT } c.\text{name_file_id, c}\text{.use_end_line, c}\text{.name_file_id, c}\text{.use_start_line, c}\text{.name_start_line, c}\text{.name_start_column WHERE} \ (n:\text{macro WITH name = 'ksqget'}) \leftarrow[c:\text{expands}] \Rightarrow (f:\text{source_file WITH name = 'rdbms/src/server/vos/ksfd.c'}) \text{ ORDER BY } c.\text{use_file_id, c}\text{.use_end_line, c}\text{.name_file_id, c}\text{.use_start_line, c}\text{.name_start_line, c}\text{.name_start_column LIMIT 10}
\]

Q5
\[
\text{SELECT } n1.\text{id(), n2.id(), n3.id()} \text{ WHERE} \ (n1:\text{field}) \leftarrow[c:\text{isa_type}] \rightarrow (n2:\text{function_type}) \leftarrow[c:\text{has_param_type}] \rightarrow (n3:\text{struct}) \leftarrow[c:\text{contains}] \rightarrow (n1) \text{ ORDER BY n1, n2, n3 LIMIT 20}
\]

Q6
\[
\text{SELECT } f.\text{name, c}\text{.use_start_line WHERE} \ (n:\text{macro WITH name = 'ksqget'}) \leftarrow[c:\text{expands}] \Rightarrow (f:\text{source_file}) \leftarrow[e:\text{expands}] \Rightarrow (m \text{ WITH name = 'KSQO_GLOBAL'}, c\text{.use_start_line} \leftarrow[e\text{.use_start_line}] \text{ AND } e\text{.use_start_line} \leftarrow[c\text{.use_start_line}] + 2 \text{ ORDER BY } f.\text{name, c}\text{.use_start_line LIMIT 20}
\]

Q7
\[
\text{SELECT } f.\text{name WHERE} \ (n:\text{macro WITH name = 'KSQO_GLOBAL'}) \leftarrow[c:\text{expands}] \Rightarrow (f:\text{source_file}) \leftarrow[e:\text{expands}] \Rightarrow (n), c\text{.use_start_line} \leftarrow[e\text{.use_start_line}] \text{ ORDER BY } f.\text{name LIMIT 20}
\]

Q8
\[
\text{SELECT } n1 \text{ WHERE} \ (n1:\text{function}) \leftarrow[e\text{.calls}] \Rightarrow (n2:\text{function}) \text{ ORDER BY } e\text{.use_start_line DESC, n1\text{.name ASC, n2\text{.name ASC LIMIT 10}}}
\]
RDBMS queries (2)

Q9
```sql
SELECT COUNT(*), MIN(e.use_end_line), MAX(e.use_start_line), AVG(e.name_start_line), SUM(e.name_start_column)
WHERE (:function) -[e:calls]-> (:function), e.name_start_line != -1
```

Q10
```sql
SELECT f.name, o.name, r.label(), COUNT(*)
WHERE
(f:source_file WITH name = 'rdbms/src/generic/psm/kgfk.c') -[:file_contains|contains]-> () -
[r:calls|reads|writes]-> () <-[:file_contains|contains]- (o:source_file)
GROUP BY
f, o, r.label()
ORDER BY f.name, o.name, r.label()
LIMIT 20
```

Q11
```sql
SELECT f.name, o.name, r.label(), COUNT(*)
WHERE
(f:source_file) -[:file_contains|contains]-> () -[r:calls|reads|writes]-> () <-[:file_contains|contains]-
(o:source_file)
GROUP BY
f, o, r.label()
ORDER BY f.name, o.name, r.label()
LIMIT 20
```

Q12
```sql
SELECT n.id(), n.outDegree()
WHERE (n:function)
ORDER BY n.outDegree() DESC
LIMIT 10
```
### RDBMS queries (3)

**Q13**

```
SELECT n.id() WHERE (n WITH name = 'ksqget' OR name =~ 'EVTDV\$') ORDER BY n
```

**Q14**

```java
PATH contains := () -[:file_contains|dir_contains]-> ()
SELECT m
WHERE
    (d:directory WITH name = 'rdbms/src/client') -[:contains*]-> (f),
    (f) -[:file_contains]-> (m:macro WITH name =~ 'EVTDV\$')
```

**Q15**

```java
PATH includes := () -[:includes]-> ()
SELECT f.name
WHERE
    (f) -[:includes*]-> (h:source_file WITH name = 'rdbms/include/kge.h')
ORDER BY f.name
LIMIT 20
```

**Q16**

```
PATH contains := () -[:file_contains|contains]-> ()
SELECT f.name, o.name, r.label(), COUNT(*)
WHERE
    (f:source_file WITH name = 'rdbms/src/generic/psm/kgfk.c') -[:contains*]-> (x),
    (o:source_file) -[:contains*]-> (y),
    (x) -[:calls|reads|writes]-> (y)
GROUP BY
    f, o, r.label()
ORDER BY o.name
LIMIT 20
```
RDBMS queries (4)

Q17

```
PATH contains := () -[:file_contains|contains]--> ()
SELECT f.name, o.name, r.label(), COUNT(*)
WHERE
  (f:source_file) -[:contains*]--> (x),
  (o:source_file) -[:contains*]--> (y),
  (x) -[r:calls|reads|writes]--> (y)
GROUP BY
  f, o, r.label()
ORDER BY o.name
LIMIT 20
```