ORACLE

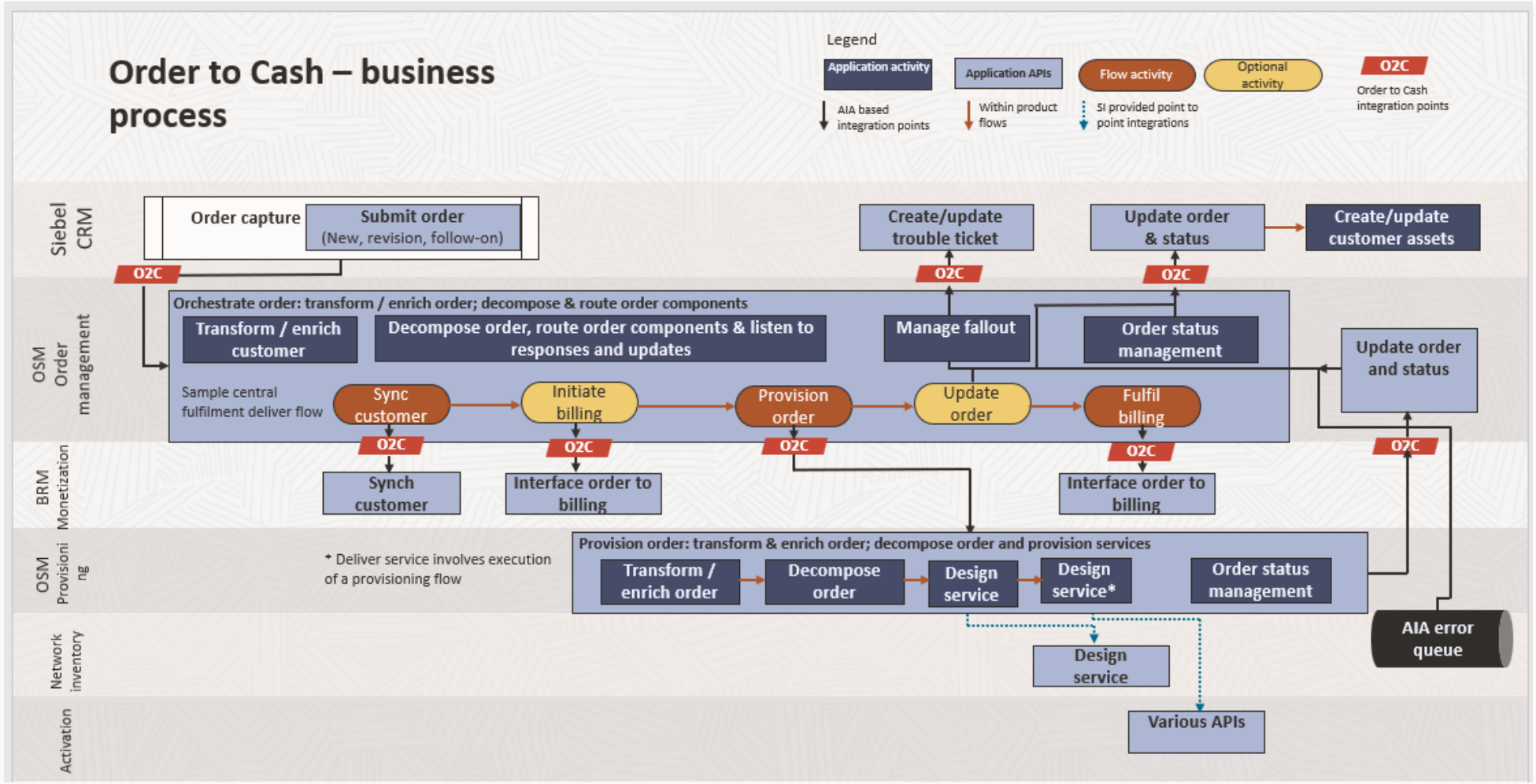
From SQL Patterns in Event Logs to Graph Insights

An introduction to Process Mining

Hans Viehmann
Director Product Management – Spatial and Graph
nlOUG DB World 2026
November 13, 2026

Process model today

Are you sure?



Copyright © 2026, Oracle and/or its affiliates

What is Process Mining and why is it important?



Discovery

Conformance



Optimization

Copyright © 2026, Oracle and/or its affiliates

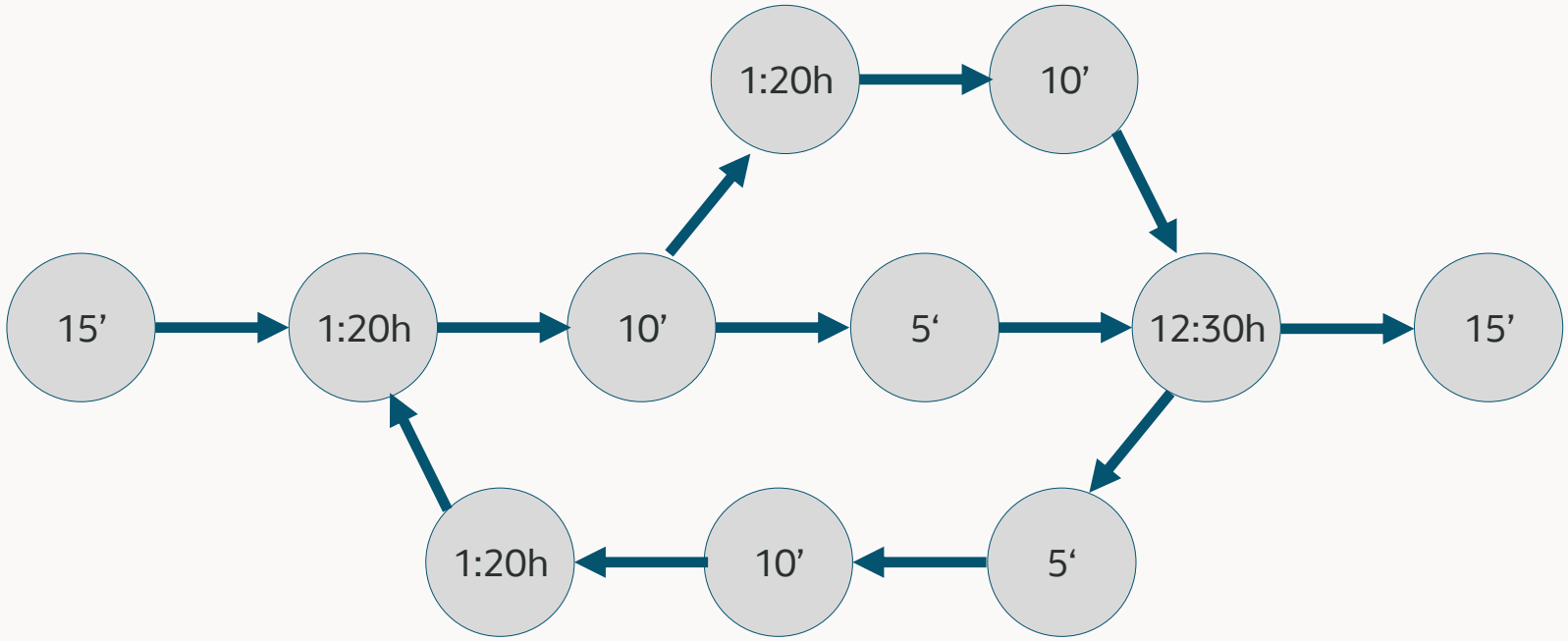
Using event logs as a data source

Demo dataset

CASEID	ACTIVITYNAME	ACTOR	STARTTIMESTAMP	ENDTIMESTAMP	VENDOR	INVOICEAMOUNT	ROLE
1	Invoice Entry	Adams	11.02.26 09:43:01.000	11.02.26 12:05:24.000	Stellar Traders	2873.96	Invoice Processor
1	Customer Payment Check	Adams	11.02.26 12:05:24.000	11.02.26 13:54:26.000	Stellar Traders	2873.96	Invoice Processor
1	Credit Memo Entry	Adams	11.02.26 13:54:26.000	11.02.26 15:44:38.000	Stellar Traders	2873.96	Invoice Processor
1	Customer Refund	Adams	11.02.26 15:44:38.000	11.02.26 17:51:44.000	Stellar Traders	2873.96	Invoice Processor
1	Invoice Re-issue	Adams	11.02.26 17:51:44.000	11.02.26 19:05:53.000	Stellar Traders	2873.96	Invoice Processor
2	Invoice Entry	Ford	09.02.26 10:03:03.000	09.02.26 19:12:49.000	Stellar Traders	717.73	Invoice Processor
2	Payment Confirmation	Ford	10.02.26 09:12:49.000	10.02.26 14:47:47.000	Stellar Traders	717.73	Invoice Processor
2	Refund With Special Voucher	Ford	10.02.26 14:47:47.000	11.02.26 15:15:56.000	Stellar Traders	717.73	Invoice Processor
2	Customer Memo Completion	Clark	11.02.26 15:15:56.000	11.02.26 23:35:21.000	Stellar Traders	717.73	Team Lead Invoicing
2	Invoice Approval	Clark	12.02.26 09:35:21.000	12.02.26 17:05:44.000	Stellar Traders	717.73	Team Lead Invoicing
2	Credit Memo Creation	Ford	12.02.26 17:05:44.000	12.02.26 23:44:07.000	Stellar Traders	717.73	Invoice Processor
2	Credit Memo Completion	Ford	15.02.26 09:44:07.000	15.02.26 15:59:49.000	Stellar Traders	717.73	Invoice Processor
2	Invoice Re-issue	Ford	15.02.26 15:59:49.000	15.02.26 22:57:42.000	Stellar Traders	717.73	Invoice Processor
3	Invoice Entry	Adams	15.03.26 09:12:15.000	15.03.26 10:42:50.000	Blue Ridge Imp…	4554.8	Invoice Processor
3	Payment Confirmation	Adams	15.03.26 10:42:50.000	15.03.26 12:11:44.000	Blue Ridge Imp…	4554.8	Invoice Processor
3	Refund With Standard Vouch…	Adams	15.03.26 12:11:44.000	15.03.26 13:36:21.000	Blue Ridge Imp…	4554.8	Invoice Processor
3	Customer Memo Completion	Jones	15.03.26 13:36:21.000	15.03.26 18:17:14.000	Blue Ridge Imp…	4554.8	Manager Invoicing
3	Invoice Rejection	Jones	16.03.26 09:17:14.000	16.03.26 12:35:54.000	Blue Ridge Imp…	4554.8	Manager Invoicing
3	Refund With Standard Vouch…	Adams	16.03.26 12:35:54.000	16.03.26 13:48:59.000	Blue Ridge Imp…	4554.8	Invoice Processor
3	Customer Memo Completion	Jones	16.03.26 13:48:59.000	16.03.26 17:03:09.000	Blue Ridge Imp…	4554.8	Manager Invoicing

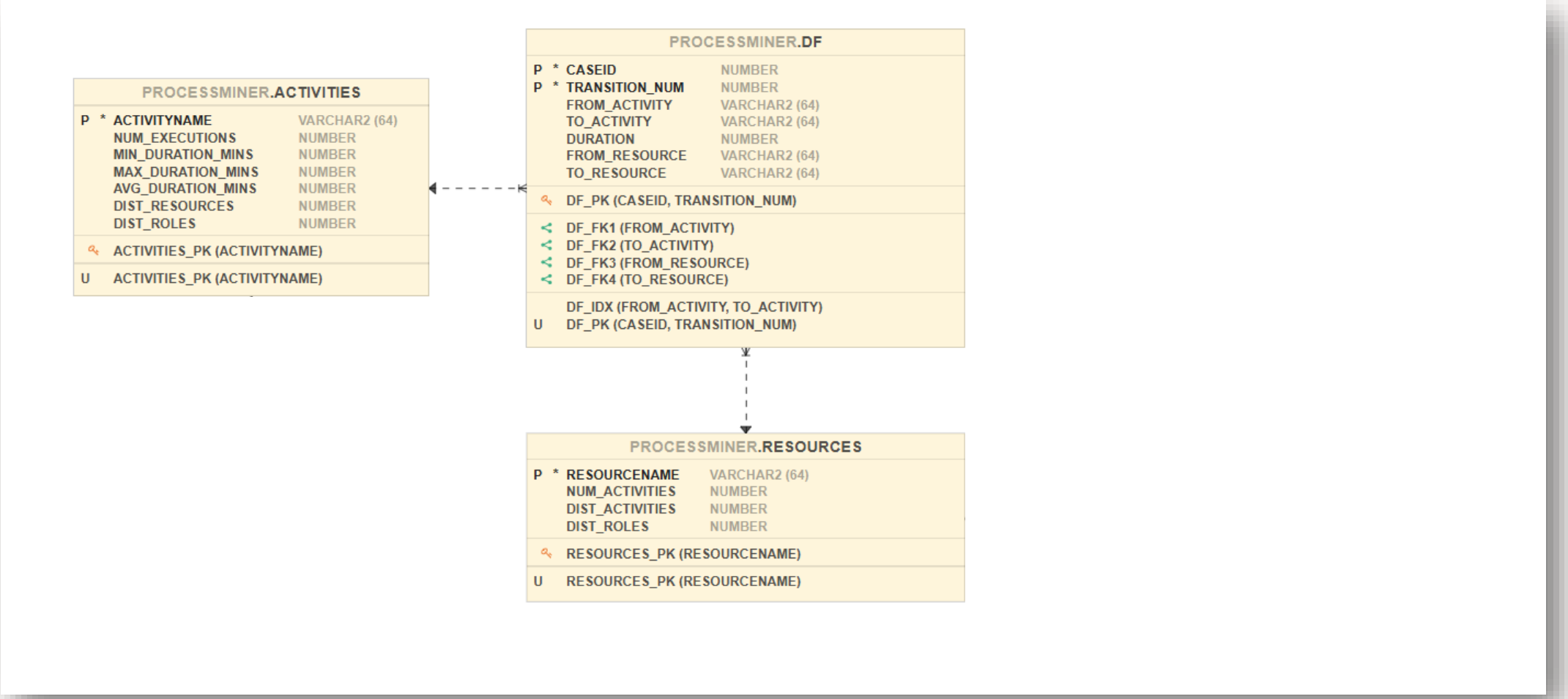
Copyright © 2026, Oracle and/or its affiliates

Process flows derived from event logs



Copyright © 2026, Oracle and/or its affiliates

Relational data model



Copyright © 2026, Oracle and/or its affiliates

General statistics per process instance

Using MATCH_RECOGNIZE to partition, sort, and evaluate

```
select * from eventlog match_recognize (
   partition by caseid
   order by starttimestamp
 measures
   count(activityname) as num_steps,
   a.starttimestamp as startcase,
   LAST(b.endtimestamp) as endcase,
   LAST(b.endtimestamp) - a.starttimestamp as
     duration
 pattern (a b+)
 define
   b as b.caseid = a.caseid
 );
```

PATTERN (a b+) returns first row (a) and one or more following rows (b) with same caseid

CASEID	ACTIVITYNAME	STARTTIMESTAMP
9	Invoice Entry	19.02.26 13:36:36.000
9	Customer Payment Check	19.02.26 14:43:19.000
9	Credit Memo Entry	19.02.26 15:48:18.000
9	Customer Refund	19.02.26 16:30:46.000
9	Invoice Re-issue	19.02.26 17:48:24.000

Using LAST() to compute total duration per case

CASEID	NUM_STEPS	DURATION
9	5	+00 04:39:30.000000

Copyright © 2026, Oracle and/or its affiliates

Finding rework within a case

Using MATCH_RECOGNIZE to identify consecutive steps satisfying a condition

```
select * from eventlog match_recognize (
   partition by caseid
   order by starttimestamp
 measures
   match_number() as match_num,
   a.activityname as rework_activity
 all rows per match
   pattern (a b)
   define
     b as b.activityname = a.activityname
);
```

Eventlog

135	Customer Memo Completion	Clark
135	Invoice Approval	Clark
135	Invoice Approval	Jones
135	Credit Memo Creation	James
135	Credit Memo Completion	James
135	Invoice Re-issue	James
137	Invoice Entry	Ford
137	Customer Payment Check	Ford

ALL ROWS PER MATCH returns both rows

CASEID	MATCH_NUM	REWORK_ACTIVITY	ACTOR
135	1	Invoice Approval	Clark
135	1	Invoice Approval	Jones

Copyright © 2026, Oracle and/or its affiliates

Example patterns and options

Pattern definition – each variable corresponds to a Boolean condition

- POSIX extended regular expression syntax
- Concatenation (a b c)
- Quantifiers {n}, {n,m}, *, +, ?
- Alternation (a | b | c)
- Grouping
- Permutation
- …

Restart search after match

- AFTER MATCH SKIP TO NEXT ROW
- AFTER MATCH SKIP PAST LAST ROW
- AFTER MATCH SKIP TO FIRST <variable>
- …

CASEID	MATCH…	VAR…	ACTIVITYNAME
87	1	A	Refund With Standard Voucher
87	1	X	Customer Memo Completion
87	1	X	Invoice Rejection
87	1	B	Refund With Standard Voucher
87	2	A	Refund With Standard Voucher
87	2	X	Customer Memo Completion
87	2	X	Invoice Rejection
87	2	B	Refund With Standard Voucher

```
pattern (a x+ b)
define
   b as b.activityname = a.activityname,
   x as x.activityname <> a.activityname
```


Copyright © 2026, Oracle and/or its affiliates

Creating a Directly Follows Graph (DFG) with MATCH_RECOGNIZE

Identifying all transitions within the same case

CASEID	ACTIVITYNAME	STARTTIMESTAMP
9	Invoice Entry	19.02.26 13:36:36.000
9	Customer Payment Check	19.02.26 14:43:19.000
9	Credit Memo Entry	19.02.26 15:48:18.000
9	Customer Refund	19.02.26 16:30:46.000
9	Invoice Re-issue	19.02.26 17:48:24.000
14	Invoice Entry	11.02.26 16:38:04.000
14	Payment Confirmation	12.02.26 09:11:51.000
14	Refund With Standard V…	12.02.26 10:46:54.000
14	Customer Memo Completi…	12.02.26 11:36:02.000
14	Invoice Rejection	12.02.26 13:49:21.000
14	Refund With Standard V…	12.02.26 16:44:19.000



CASEID	STEP	FROM_ACTIVITY	TO_ACTIVITY
9	1	Invoice Entry	Customer Payment Ch…
9	2	Customer Payment …	Credit Memo Entry
9	3	Credit Memo Entry	Customer Refund
9	4	Customer Refund	Invoice Re-issue
14	1	Invoice Entry	Payment Confirmation
14	2	Payment Confirmat…	Refund With Standar…
14	3	Refund With Stand…	Customer Memo Compl…
14	4	Customer Memo Com…	Invoice Rejection
14	5	Invoice Rejection	Refund With Standar…
14	6	Refund With Stand…	Customer Memo Compl…
14	7	Customer Memo Com…	Invoice Approval

Eventlog

Transitions

Copyright © 2026, Oracle and/or its affiliates

Populating the edge table (1): Directly follows transitions

MATCH_RECOGNIZE turns ordered events into process graph edges

```
select * from eventlog match_recognize (
   partition by caseid
   order by starttimestamp
 measures
   match_number() as transition_num,
   a.activityname as from_activity,
   b.activityname as to_activity,
   b.starttimestamp - a.endtimestamp as
     duration,
   a.actor as from_resource,
   b.actor as to_resource
 after match skip to next row
   pattern (a b)
   define b as b.caseid = a.caseid
);
```

Default behavior: Continue after match

CASEID	ACTIVITYNAME	STARTTIMESTAMP
9	Invoice Entry	19.02.26 13:36:36.000
9	Customer Payment Check	19.02.26 14:43:19.000
9	Credit Memo Entry	19.02.26 15:48:18.000
9	Customer Refund	19.02.26 16:30:46.000
9	Invoice Re-issue	19.02.26 17:48:24.000

Using AFTER MATCH SKIP TO NEXT ROW

CASEID	ACTIVITYNAME	STARTTIMESTAMP
9	Invoice Entry	19.02.26 13:36:36.000
9	Customer Payment Check	19.02.26 14:43:19.000
9	Credit Memo Entry	19.02.26 15:48:18.000
9	Customer Refund	19.02.26 16:30:46.000
9	Invoice Re-issue	19.02.26 17:48:24.000

Copyright © 2026, Oracle and/or its affiliates

Populating the edge table (2): Handovers between resources

MATCH_RECOGNIZE creates the basis for a social graph

```
select * from eventlog match_recognize (
   partition by caseid
   order by starttimestamp
 measures
   match_number() as transition_num,
   a.activityname as from_activity,
   b.activityname as to_activity,
   b.starttimestamp - a.endtimestamp as
      duration,
   a.actor as from_resource,
   b.actor as to_resource
 after match skip to next row
   pattern (a b)
   define
     b as b.actor != a.actor          );
```

Eventlog

CASEID	ACTIVITYNAME	ACTOR	ROLE
46	Refund With Special Voucher	Smith	Invoice Processor
46	Customer Memo Completion	Clark	Team Lead Invoicing
46	Invoice Approval	Clark	Team Lead Invoicing
46	Credit Memo Creation	Smith	Invoice Processor
46	Credit Memo Completion	Smith	Invoice Processor
46	Invoice Re-issue	Smith	Invoice Processor
52	Invoice Entry	Adams	Invoice Processor
52	Payment Confirmation	Adams	Invoice Processor
52	Refund With Standard Voucher	Adams	Invoice Processor
52	Customer Memo Completion	Jones	Manager Invoicing

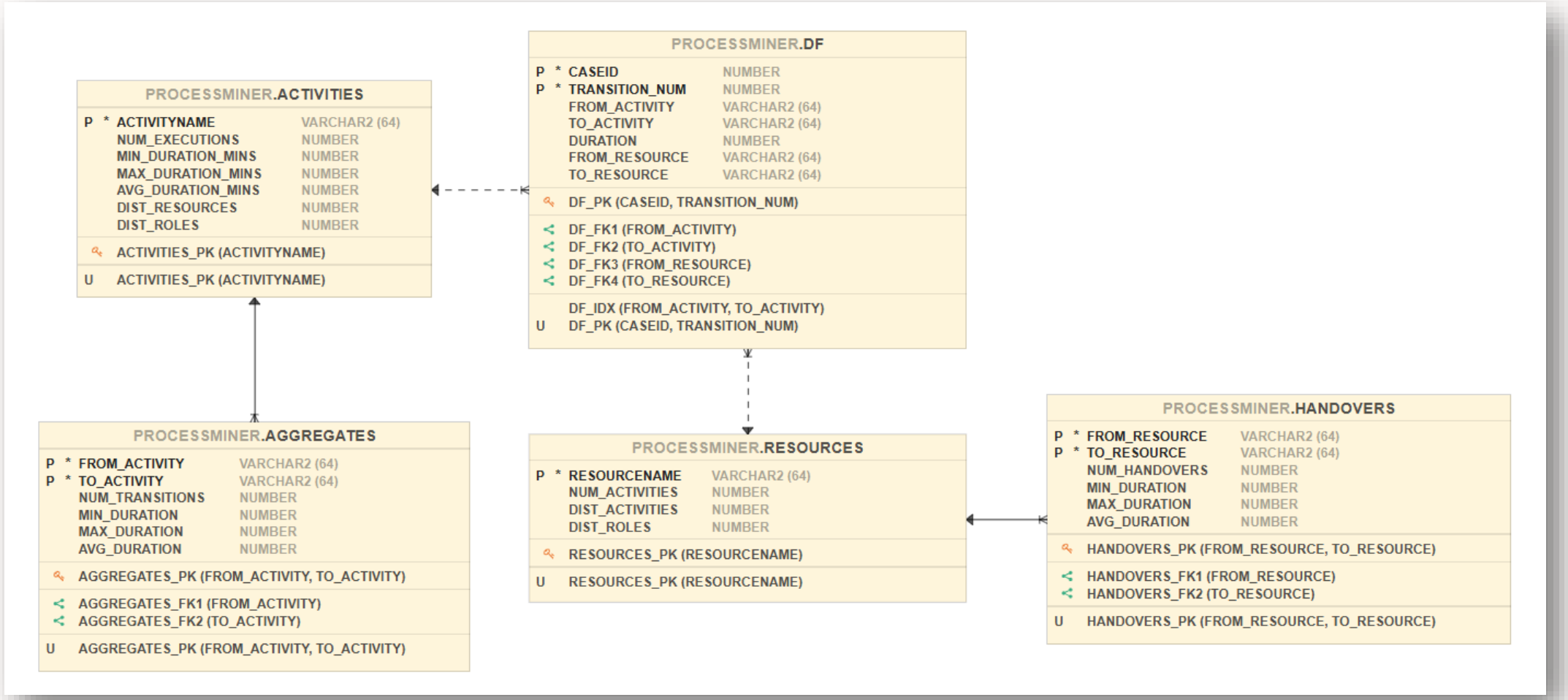
Note: Patterns are matched only within partitions

CA…	TR…	FROM_ACTIVITY	TO_ACTIVITY	FROM…	TO_R…
46	1	Refund With Special Voucher	Customer Memo Completion	Smith	Clark
46	2	Invoice Approval	Credit Memo Creation	Clark	Smith
52	1	Refund With Standard Voucher	Customer Memo Completion	Adams	Jones
52	2	Invoice Rejection	Refund With Standard Voucher	Jones	Adams
52	3	Refund With Standard Voucher	Customer Memo Completion	Adams	Jones
52	4	Invoice Approval	Credit Memo Creation	Jones	Adams

Copyright © 2026, Oracle and/or its affiliates

Reminder: Relational data model

Tables DF and HANDOVERS populated with MATCH_RECOGNIZE



Copyright © 2026, Oracle and/or its affiliates

Mapping relational tables to a graph

More than one mapping possible

- Model depends on planned queries and analyses

Different from relational model

- N:m relationships are no issue

Definition of graph in PGQL DDL

- VERTEX TABLES clause and EDGE TABLES clause list the source tables
- Defaults are taken from Oracle Data Dictionary

CREATE PROPERTY GRAPH creates metadata only

- No physical data copy

```
CREATE PROPERTY GRAPH socialgraph
  VERTEX TABLES (
    resources
      LABEL actor
      PROPERTIES ARE ALL COLUMNS
  )
  EDGE TABLES (
    handovers
      SOURCE KEY (from_resource) REFERENCES
                              resources (resourcename)
      DESTINATION KEY (to_resource) REFERENCES
                              resources (resourcename)
      LABEL handover
      PROPERTIES ARE ALL COLUMNS
  )
```


Copyright © 2024, Oracle and/or its affiliates.

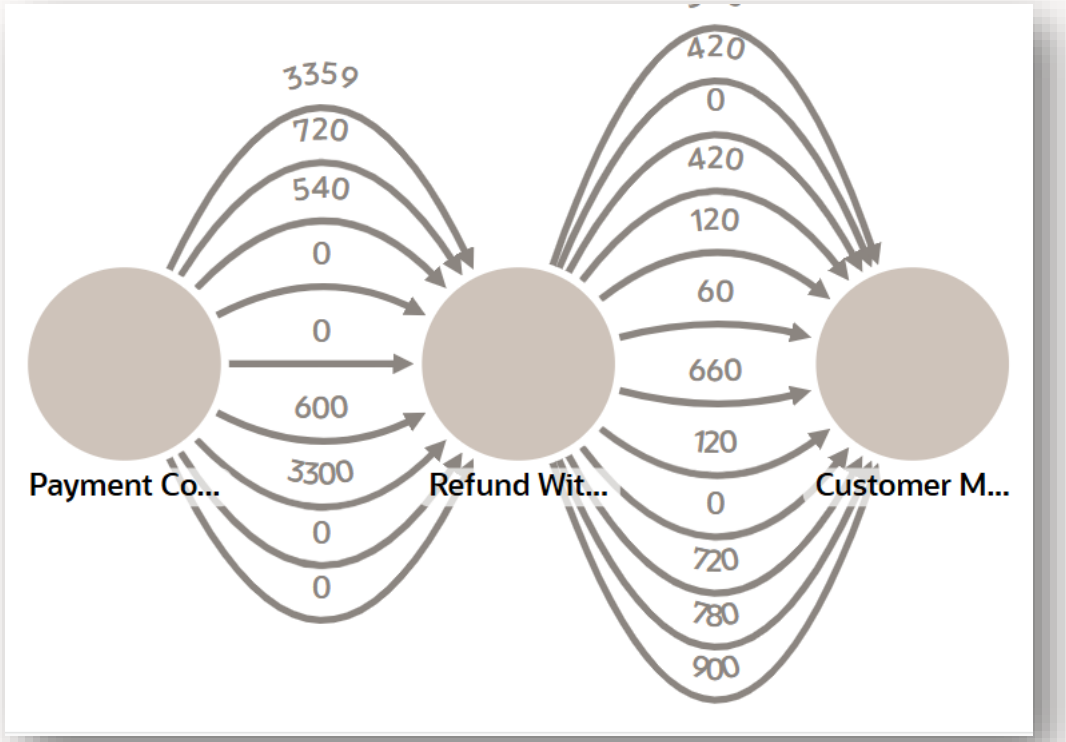
Demo

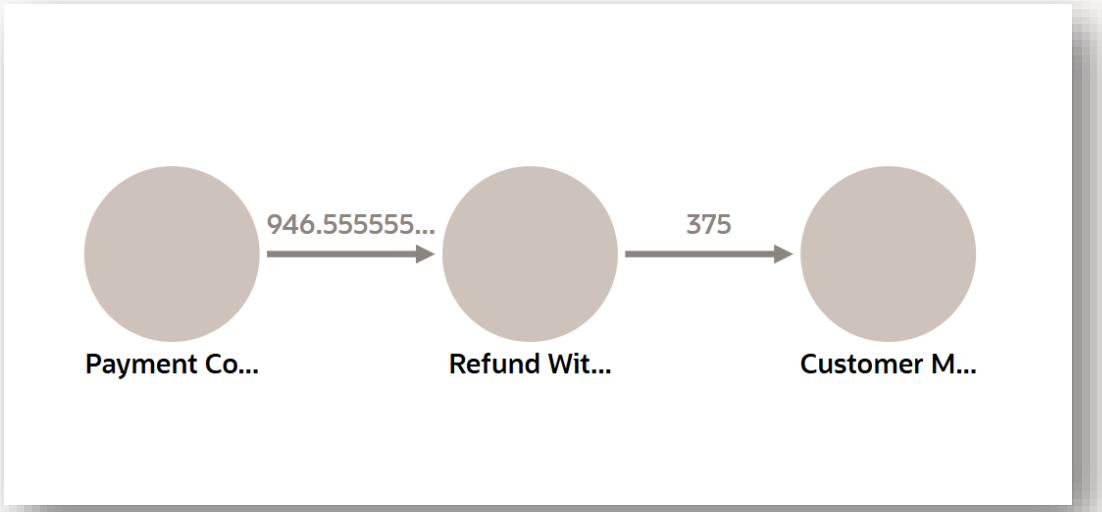
Property graphs in SQL Developer in VS Code



Representing the Directly Follows (DF) and Aggregates table as a graph

Edge label DF and AGG_TRANS respectively, both in the same graph





Copyright © 2026, Oracle and/or its affiliates

Anatomy of a GRAPH_TABLE query

```
SELECT * FROM GRAPH_TABLE( PROCESSGRAPH
MATCH (origin) -[step is transition]->{4} (dest)
KEEP ANY SHORTEST PATH
WHERE origin.activityname='Invoice Entry'
ONE ROW PER STEP (o,s,d)
COLUMNS (o.activityname as origin, s.num_transitions as transition, d.activityname as
dest))
```

- MATCH clause to specify patterns
- KEEP clause for path modes
- WHERE clause (optionally inside vertex or edge pattern)
- ONE ROW PER STEP (optional) splitting paths in vertex-edge-vertex triples mostly for visualization purposes
- COLUMNS clause to specify the output of the GRAPH_TABLE function

Copyright © 2026, Oracle and/or its affiliates

Example patterns and options

Pattern definition in MATCH clause

- Vertex definition ()
- Edge definition ->, -[]->, <-[]-, -[]-, <-[]->
- Label (a IS activity), -[t IS transition]->
- Quantifiers {n}, {n,m}, *, +
- Alternation (a | b | c)
- …

Number of output rows

- ONE ROW PER MATCH
- ONE ROW PER STEP
  - retrieve information from edges on paths
- ONE ROW PER VERTEX
  - retrieve information from vertices along paths

Path finding goals

- ANY
- ANY SHORTEST
- ALL SHORTEST
- SHORTEST k
- ALL

Path modes

- WALK: no filtering of paths
- TRAIL: paths with repeated edges are not returned
- ACYCLIC: paths with repeated vertices are not returned
- SIMPLE: paths with repeated vertices are not returned unless the repeated vertex is first and last

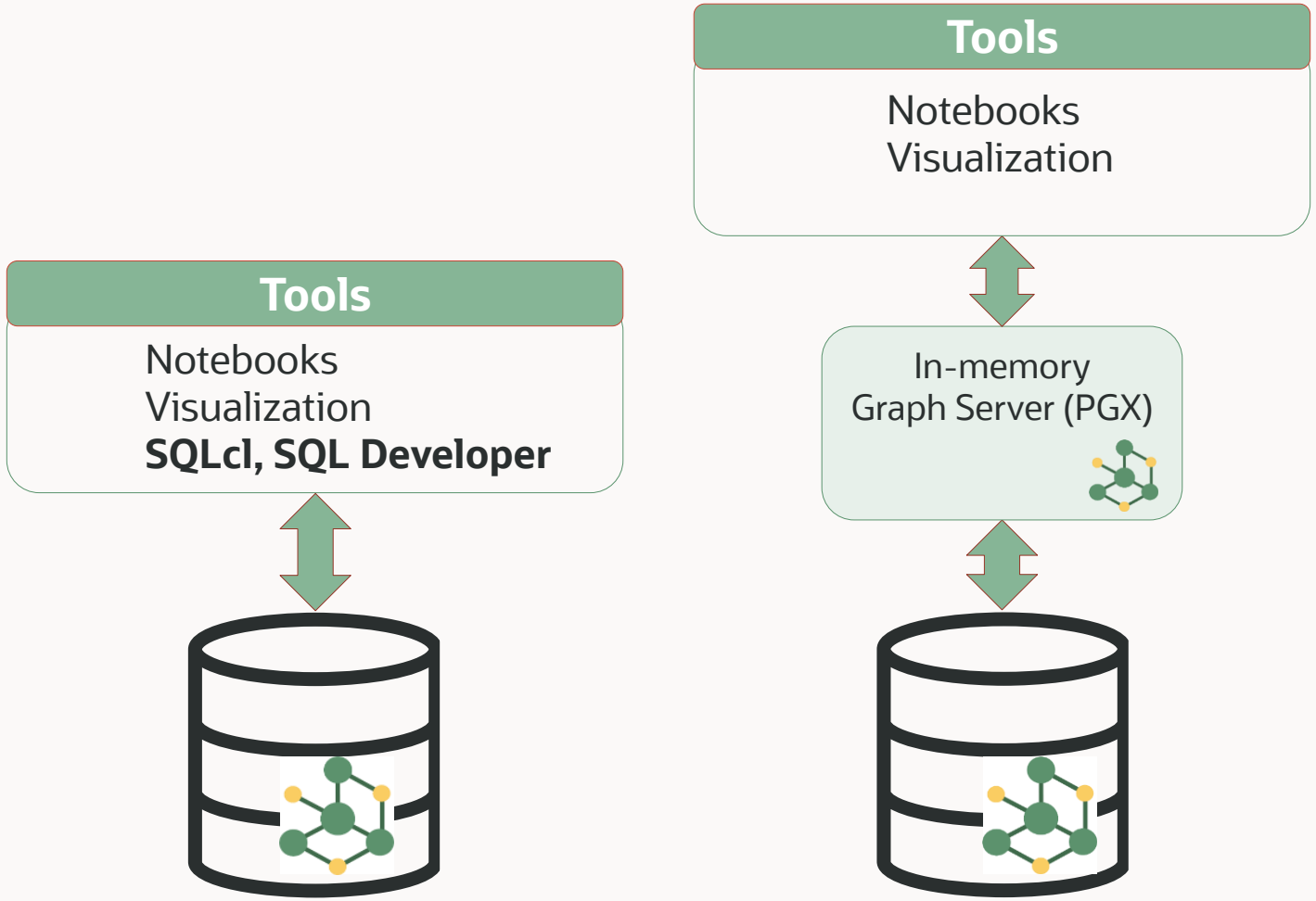
See pgql-lang.org for details and examples

Copyright © 2026, Oracle and/or its affiliates

Demo flow

Getting to know Graph Studio in ADB

- Modeling a graph
  - based on primary key/foreign key relationships
- Creating the graph
- Loading the graph into memory
- Working with the Notebook UI



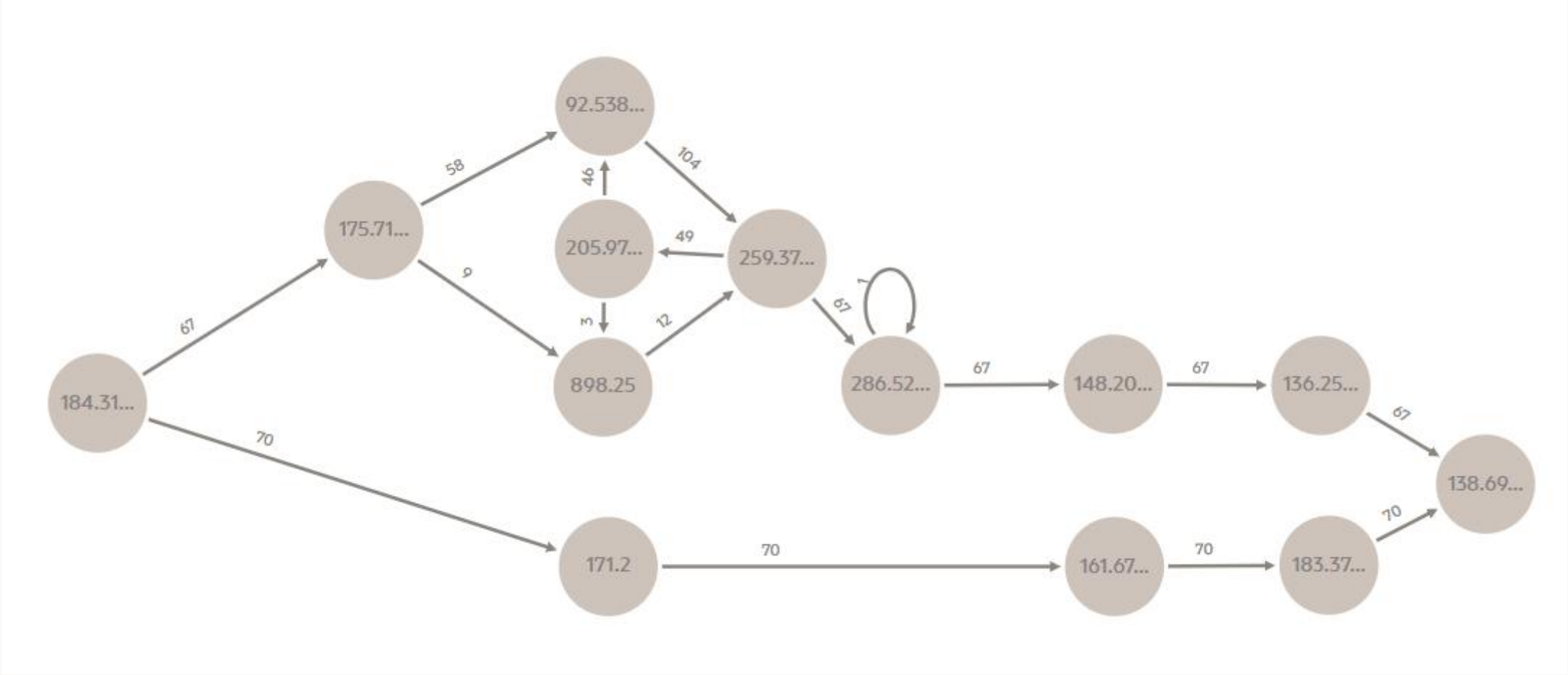
Copyright © 2026, Oracle and/or its affiliates

Demo

Property graphs in Graph Studio in ADB

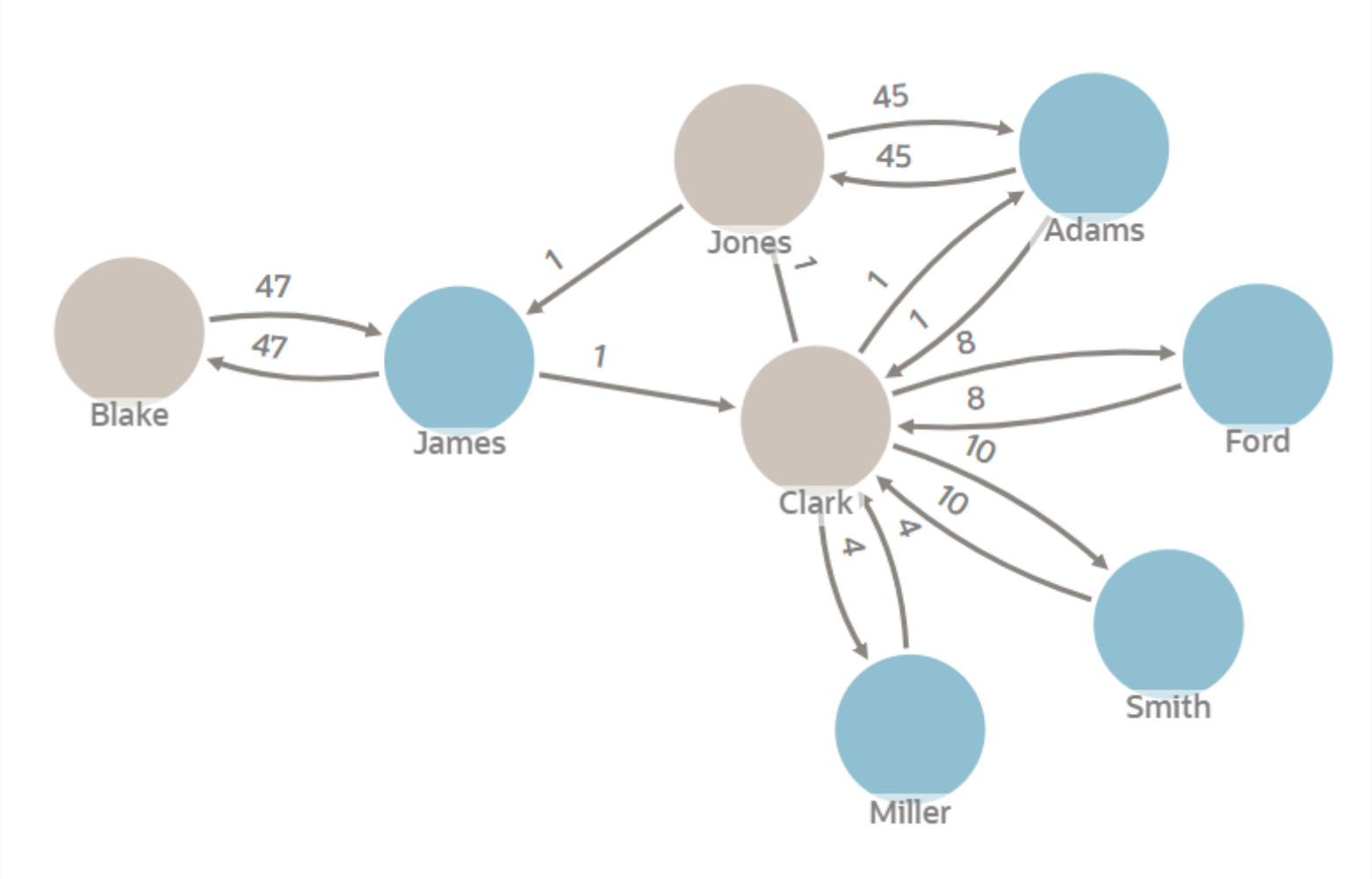


Process graph



Copyright © 2026, Oracle and/or its affiliates

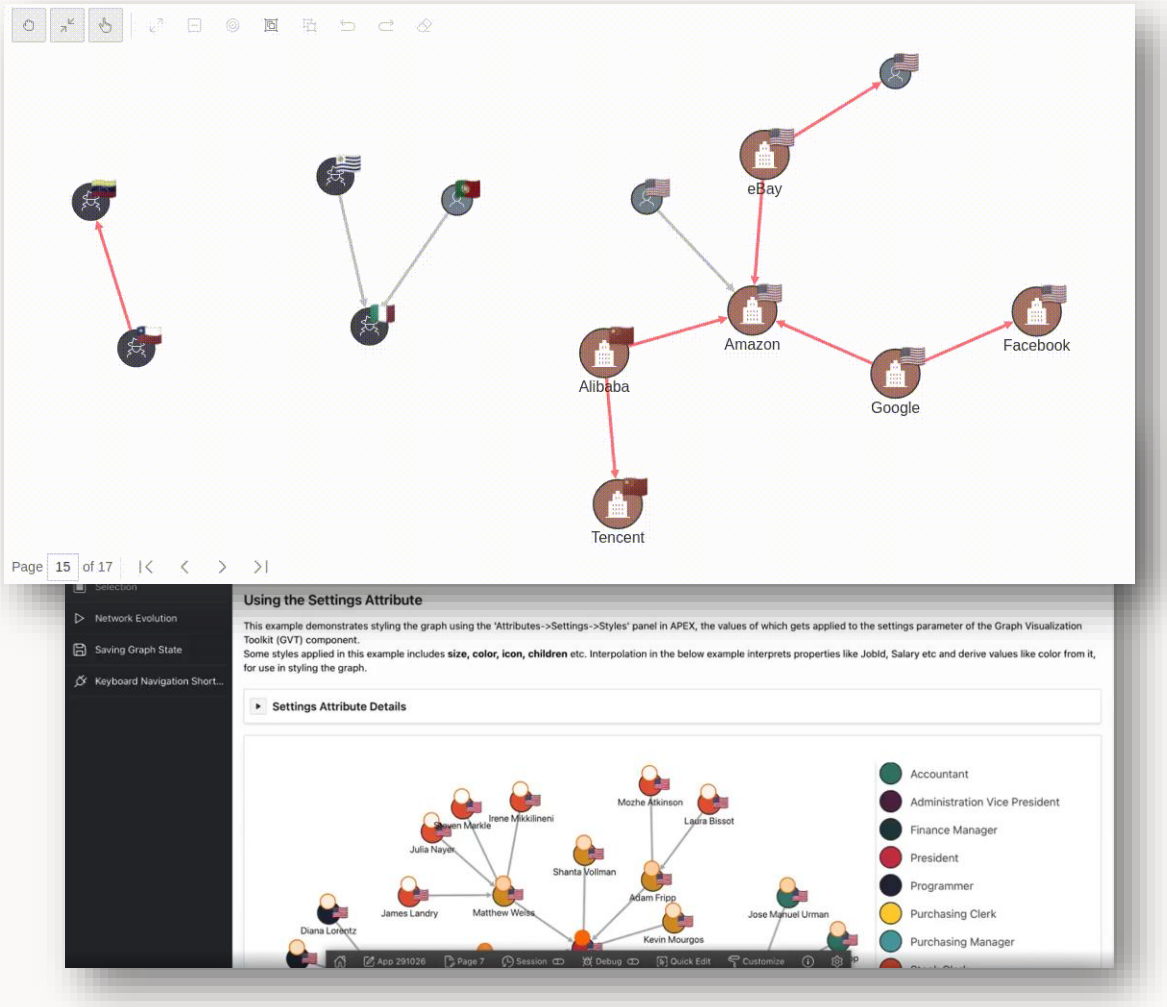
Social graph



Copyright © 2026, Oracle and/or its affiliates

Build a Process Mining dashboard in APEX

- Graph visualization plug-in for Apex
  - Uses Graph Visualization Toolkit
  - Based on Javascript
- Explore the graph vertices and edges
- Interact with the graph visualization
  - changing graph layouts, grouping or ungrouping, removing selected vertices or edges, …
- Style vertices and edges (size, color, icon, label values, …)
- Visualize and study the evolution of the graph over time
- Sample application on oracle.github.io/apex/



Copyright © 2026, Oracle and/or its affiliates

When working on relationships,
think graphs

When analyzing timeseries,
think MATCH_RECOGNIZE

When combining graphs and relational,
think converged database

Copyright © 2026, Oracle and/or its affiliates

Further information on Oracle Graph

Oracle Graph technologies landing page on oracle.com

- www.oracle.com/database/graph

Videos on Youtube

- www.youtube.com/c/OracleSpatialandGraph

Oracle LiveLabs: Search for „graph“ on

- livelabs.oracle.com

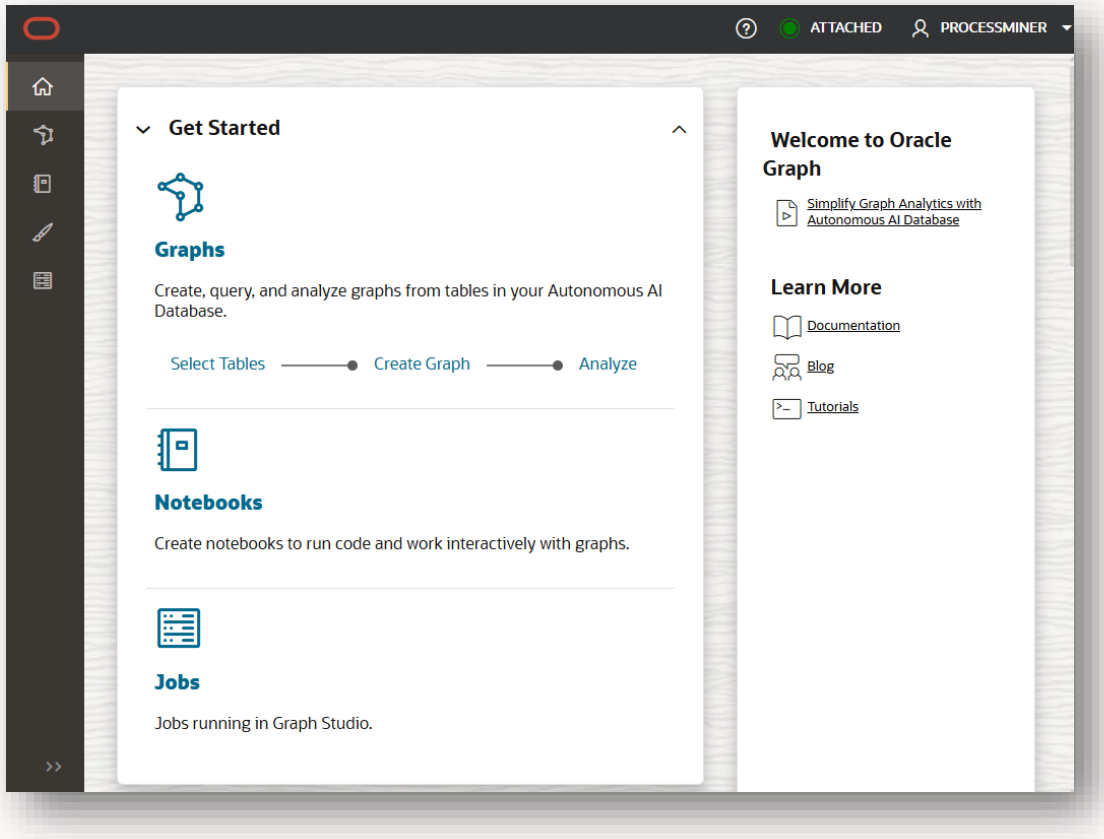
Blogs

- Database Insider (category: Graph)
- medium.com (tag: Oracle Graph)

Documentation

- Oracle Property Graph documentation
- Graph Studio in ADB documentation
- PG Viz Developers Guide

Skills repository on Github



Copyright © 2026, Oracle and/or its affiliates

Thank you

Hans Viehmann

hans.viehmann@oracle.com

Copyright © 2026, Oracle and/or its affiliates


ORACLE