ORACLE

Multi-Edge support and Future-Proofing

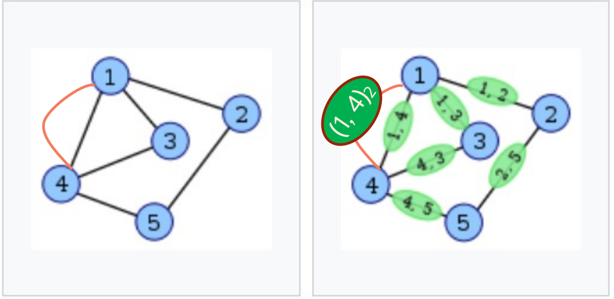
Souripriya Das, Ph.D.

Architect, Spatial and Graph,
Oracle America Inc.

January 19, 2023

Background: Edges as Vertices, Multigraph, Multi-Edge (or Parallel Edges)

- A Line Graph (LG) converts edges to vertices and then eliminates the original vertices and …



Graph G

Vertices in L(G) constructed from edges in G

What if G is a multigraph?

Two parallel edges between vertices 1 and 4.

Both edges cannot be named (1, 4). A custom name, e. g., $(1, 4)_2$, may be used.

- RDF-star, keeps, and allows unrestricted use of, both edge-vertices and the original vertices, as vertices.
- Property Graph (PG) supports it too, but limits edge-vertices to only connect *to* scalar values.

Labeled Multidigraphs are Not Uncommon in Practice

Examples:

- :servedAs :POTUS" (Cleveland Grover did two non-consecutive terms)
- :deposit :myBankAccount (multiple transactions by same person to same account)
- :called :mySister (call data records: multiple calls by same person to his/her sister)
- :hasManager :myManager (multiple stints)
- :won :Wimbledon (same person wins multiple times)
- :won :SoccerWorldCup (same country wins multiple times)
- etc.

Modeling Multi-Edge and Handling Transition of a Property to Multi-Edge

Data

x	y	color	type
A	B	red	--
B	C	blue	__
B	D	blue	__
C	D	green	__
C	**D**	blue	--

Column Names

1. name → implicit or explicit
2. iid → implicit id
3. oid → occurrence id
4. x → subject of triple
5. y → object of triple

RDFn Property Tables: **no** :occurrenceOf

:knows

name	x	y
akb	A	B
bkc	B	C
bkd	B	D
ckd	C	D
cd2	**C**	**D**

:color

name	color
akb	red
bkc	blue
bkd	blue
ckd	green
cd2	blue

:dtype

name	dtype
akb	--
bkc	__
bkd	__
ckd	__
cd2	--

RDF-star Property Tables: :occurrenceOf as needed

:knows

iid	x	y
akb	A	B
bkc	B	C
bkd	B	D
ckd	C	D

:occurrenceOf

oid	iid
cd1	ckd
cd2	ckd

:color

oid	color
akb	red
bkc	blue
bkd	blue
cd1	green
cd2	blue

:dtype

oid	dtype
akb	--
bkc	__
bkd	__
cd1	__
cd2	--

RDF-star Property Tables: :occurrenceOf ALWAYS

:knows

iid	x	y
akb	A	B
bkc	B	C
bkd	B	D
ckd	C	D

:occurrenceOf

oid	iid
ab1	akb
bc1	bkc
bd1	bkd
cd1	ckd
cd2	ckd

:color

oid	color
ab1	red
bc1	blue
bd1	blue
cd1	green
cd2	blue

:dtype

oid	dtype
ab1	--
bc1	__
bd1	__
cd1	__
cd2	--

RDFn Named Triples:
Data Size, Burden on Creator, Query Complexity/Efficiency, Future-Proof?

Data

x	y	color	type
A	B	red	--
B	C	blue	__
B	D	blue	__
C	D	green	__
C	**D**	blue	--

Key Measures

1. # of core triples: 4 + 1 = 5
2. # of custom IRIs: 1
3. Qry Complexity: Simple.
4. # of triple-patterns: 1
5. Is future-proof? YES

:A :knows :B .
:B :knows :C .
:B :knows :D .
:C :knows :D .
:C :knows :D | :cd2 .

Data: Core triples only

RDFn Property Tables: **no** :occurrenceOf

:knows

name	x	y
akb	A	B
bkc	B	C
bkd	B	D
ckd	C	D
cd2	**C**	**D**

:color

name	color
akb	red
bkc	blue
bkd	blue
ckd	green
cd2	blue

:dtype

name	dtype
akb	--
bkc	__
bkd	__
ckd	__
cd2	--

Query: Count occurrences of (asserted) :knows edges.

```
SELECT (count(*) as ?cnt) {
  ?x :knows :?y
}
```

BEFORE multi-edge

AFTER multi-edge

… same as BEFORE …

RDF-star with :occurrenceOf for extra edges in multi-edges: Data Size, Burden on Creator, Query Complexity/Efficiency, Future-Proof?

Data

x	y	color	type
A	B	red	--
B	C	blue	__
B	D	blue	__
C	D	green	__
C	**D**	blue	--

Key Measures

1. # of core triples: 4 + 1 = 5
2. # of custom IRIs: 1
3. Qry Complexity: UNION.
4. # of triple-patterns: 2.
5. Is future-proof? NO

RDF-star Property Tables: :occurrenceOf as needed

:knows

iid	x	y
akb	A	B
bkc	B	C
bkd	B	D
ckd	C	D

:occurrenceOf

oid	iid
cd2	ckd

:color

oid	color
akb	red
bkc	blue
bkd	blue
ckd	green
cd2	blue

:dtype

oid	dtype
akb	--
bkc	__
bkd	__
ckd	__
cd2	--

Data: Core triples only

```
:A :knows :B .
:B :knows :C .
:B :knows :D .
:C :knows :D .

:cd2 :occurrenceOf << :C :knows :D >> .
```

Query: Count occurrences of (asserted) :knows edges.

BEFORE multi-edge

```
SELECT (count(*) as ?cnt) {
  ?x :knows :?y
}
```

AFTER multi-edge

```
SELECT (count(*) as ?cnt) {
  { ?x :knows :?y }
  UNION { ?occ :occurrenceOf << ?x :knows ?y >>}
}
```

RDF-star with :occurrenceOf for all edges in multi-edges: Data Size, Burden on Creator, Query Complexity/Efficiency, Future-Proof?

Data

x	y	color	type
A	B	red	--
B	C	blue	__
B	D	blue	__
C	D	green	__
C	**D**	blue	--

Key Measures

1. # of core triples: 4 + 2*1 = 6
2. # of custom IRIs: 2*1 = 2
3. Qry Complexity: OPTIONAL.
4. # of triple-patterns: 2.
5. Is future-proof? NO

RDF-star Property Tables: :occurrenceOf as needed

:knows

iid	x	y
akb	A	B
bkc	B	C
bkd	B	D
ckd	C	D

:occurrenceOf

oid	iid
cd1	ckd
cd2	ckd

:color

oid	color
akb	red
bkc	blue
bkd	blue
cd1	green
cd2	blue

:dtype

oid	dtype
akb	--
bkc	__
bkd	__
cd1	__
cd2	--

Data: Core triples only

```
:A :knows :B .
:B :knows :C .
:B :knows :D .
:C :knows :D .
```

```
:cd1 :occurrenceOf << :C :knows :D >> .
:cd2 :occurrenceOf << :C :knows :D >> .
```

Query: Count occurrences of (asserted) :knows edges.

BEFORE multi-edge

```
SELECT (count(*) as ?cnt) {
  ?x :knows :?y
}
```

AFTER multi-edge

```
SELECT (count(*) as ?cnt) {
  ?x :knows :?y
  OPTIONAL { ?occ :occurrenceOf << ?x :knows ?y >>}
}
```

RDF-star with :occurrenceOf ALWAYS (for all edges): Data Size, Burden on Creator, Query Complexity/Efficiency, Future-Proof?

Data

x	y	color	type
A	B	red	--
B	C	blue	__
B	D	blue	__
C	D	green	__
C	**D**	blue	--

Key Measures

1. # of core triples: 4*2 + 1 = 9
2. # of custom IRIs: 4 + 1 = 5
3. Qry Complexity: Simple.
4. # of triple-patterns: 2.
5. Is future-proof? YES

Query: Count occurrences of (asserted) :knows edges.

```
SELECT (count(*) as ?cnt) {
  ?x :knows :?y . ?occ :occurrenceOf << ?x :knows ?y >>
}
```

BEFORE multi-edge

AFTER multi-edge

… same as BEFORE …

Data: Core triples only

:A :knows :B . **:ab1** :occurrenceOf << :A :knows :B >> .
:B :knows :C . **:bc1** :occurrenceOf << :B :knows :C >> .
:B :knows :D . **:bd1** :occurrenceOf << :B :knows :D >> .
:C :knows :D . **:cd1** :occurrenceOf << :C :knows :D >> .

:cd2 :occurrenceOf << :C :knows :D >> .

RDF-star Property Tables: :occurrenceOf ALWAYS

:knows

iid	x	y
akb	A	B
bkc	B	C
bkd	B	D
ckd	C	D

:occurrenceOf

oid	iid
ab1	akb
bc1	bkc
bd1	bkd
cd1	ckd
cd2	ckd

:color

oid	color
ab1	red
bc1	blue
bd1	blue
cd1	green
cd2	blue

:dtype

oid	dtype
ab1	--
bc1	__
bd1	__
cd1	__
cd2	--

Comparison of Key Measures: RDFn Named Triples vs. RDF-star Alternatives
Data Size, Burden on Creator, Query Complexity/Efficiency, Future-Proof?

Key Measures	RDFn Named Triples	:occurenceOf, for extra edges in multi-edges	:occurenceOf, for all edges in multi-edges	:occurenceOf, for all edges
1. # of core triples	5 = 4+1	5 = 4+1	6 = 4 + 2*1	9 = 2*4 + 1
2. # of custom IRIs	1	1	2	5
3. Query Complexity	Simple	UNION	OPTIONAL	Simple
4. # of triple-patterns	1	2	2	2
5. Is future-proof?	YES	NO	NO	YES

RDF-star Property Tables: :occurrenceOf as needed

:knows

iid	x	y
akb	A	B
bkc	B	C
bkd	B	D
ckd	C	D

:occurrenceOf

oid	iid
cd1	ckd
cd2	ckd

:color

oid	color
akb	red
bkc	blue
bkd	blue
cd1	green
cd2	blue

:dtype

oid	dtype
akb	--
bkc	__
bkd	__
cd1	__
cd2	--

RDFn Property Tables: no :occurrenceOf

:knows

name	x	y
akb	A	B
bkc	B	C
bkd	B	D
ckd	C	D
cd2	C	D

:color

name	color
akb	red
bkc	blue
bkd	blue
ckd	green
cd2	blue

:dtype

name	dtype
akb	--
bkc	__
bkd	__
ckd	__
cd2	--

RDF-star Property Tables: :occurrenceOf ALWAYS

:knows

iid	x	y
akb	A	B
bkc	B	C
bkd	B	D
ckd	C	D

:occurrenceOf

oid	iid
ab1	akb
bc1	bkc
bd1	bkd
cd1	ckd
cd2	ckd

:color

oid	color
ab1	red
bc1	blue
bd1	blue
cd1	green
cd2	blue

:dtype

oid	dtype
ab1	--
bc1	__
bd1	__
cd1	__
cd2	--

Multi-Edge Handling Costs: RDFn Named Triples vs. RDF-star Alternatives Data Size, Burden on Creator, Query Complexity/Efficiency, Future-Proof?

When a single property occurring in N triples transitions to multi-edge(s) adding m extra triples …

N ← #distinct triples using given property m ← #extra triples for the multi-edge(s) • N can be high, m usually small	RDFn Named Triples	:occurrenceOf, for extra edges in multi-edges	:occurrenceOf, for all edges in multi-edges	:occurrenceOf, for all edges
Data Size: # core triples (not counting "statement about statement" triples)	N + m	N + m	(N + m + 1) to (N + 2*m)	2*N + m
Burden (on data creator): # custom IRIs that the data creator has to provide)	(Data Size – N)			
COUNT Query Complexity/Efficiency: #patterns / Simple, UNION, OPTIONAL?	1/Simple	2/UNION	2/OPTIONAL	2/Simple
Future-Proof?: Pre-transition COUNT query still works?	YES	NO	NO	YES

… and the overhead increases (additively) when multiple properties transition to multi-edges!

ORACLE