On N-ary EdgeTypes

Consider the insert statement (for an empty database)

1 2 3 4 5 6

12345678901234567890123456789012345678901234567890123456789012

Insert (:Person{id:'Joe'})-[:WorksFor]->(:Company{name:'ABC'})

After parsing we have

{(!0=!0 PERSON NODETYPE (!1)[!1, CHAR] rows 0 IdCol=!1,

!1=TableColumn !1 Definer=-502 LastChange=!1 CHAR Table=!0 IdCol,

!2=!2 COMPANY NODETYPE (!3)[!3, CHAR] rows 0,

!3=TableColumn !3 Definer=-502 LastChange=!3 CHAR Table=!2,

!4=!4 WORKSFOR EDGETYPE (!7,!5)[!7, POSITION],[!5, POSITION] rows 0 Indexes:((!5)!6;(!7)!8) KeyCols: (!5=True,!7=True) Leaving !0[!6] LeaveCol=!5 Arriving !2[!8] ArriveCol=!7,

!5=TableColumn !5 Definer=-502 LastChange=!5 POSITION Table=!4 LeaveCol[!0],

!6=Index !6 Definer=-502 LastChange=!6 for !4 count 0 Key:Domain ROW (!5) Display=1[!5, POSITION] Kind=ForeignKey, CascadeUpdate, CascadeDelete Rows:MTree 0@0Domain ROW (!5) Display=1[!5, POSITION],

!7=TableColumn !7 Definer=-502 LastChange=!7 POSITION Table=!4 ArriveCol[!2],

!8=Index !8 Definer=-502 LastChange=!8 for !4 count 0 Key:Domain ROW (!7) Display=1[!7, POSITION] Kind=ForeignKey, CascadeUpdate, CascadeDelete Rows:MTree 0@0Domain ROW (!7) Display=1[!7, POSITION],

#9=GqlNode #9 #9 !0 PERSON NODETYPE (!1)[!1, CHAR] rows 0 IdCol=!1:GqlLabel PERSON NODETYPE PERSON NODETYPE {ID=Joe} !0 PERSON,#10 PERSON,

#10=GqlLabel PERSON NODETYPE PERSON NODETYPE,

#20=Joe,

#29=GqlEdge #29 #29 !4 WORKSFOR EDGETYPE (!7,!5)[!7, POSITION],[!5, POSITION] rows 0 Indexes:((!5)!6;(!7)!8) KeyCols: (!5=True,!7=True) Leaving !0[!6] LeaveCol=!5 Arriving !2[!8] ArriveCol=!7 ARROWBASE:GqlLabel WORKSFOR EDGETYPE WORKSFOR EDGETYPE !4 WORKSFOR,#30 WORKSFOR leaving #9 arriving #42 ARROWBASE,

#30=GqlLabel WORKSFOR EDGETYPE WORKSFOR EDGETYPE,

#42=GqlNode #42 #42 !2 COMPANY NODETYPE (!3)[!3, CHAR] rows 0:GqlLabel COMPANY NODETYPE COMPANY NODETYPE {NAME=ABC} !2 COMPANY,#43 COMPANY,

#43=GqlLabel COMPANY NODETYPE COMPANY NODETYPE,

#56=ABC,

%0=GraphInsertStatement %0[

(0=(0=GqlNode #9 #9 !0 PERSON NODETYPE (!1)[!1, CHAR] rows 0 IdCol=!1:GqlLabel PERSON NODETYPE PERSON NODETYPE {ID=Joe} !0 PERSON,#10 PERSON,

1=GqlEdge #29 #29 !4 WORKSFOR EDGETYPE (!7,!5)[!7, POSITION],[!5, POSITION] rows 0 Indexes:((!5)!6;(!7)!8) KeyCols: (!5=True,!7=True) Leaving !0[!6] LeaveCol=!5 Arriving !2[!8] ArriveCol=!7 ARROWBASE:GqlLabel WORKSFOR EDGETYPE WORKSFOR EDGETYPE !4 WORKSFOR,#30 WORKSFOR leaving #9 arriving #42 ARROWBASE,

2=GqlNode #42 #42 !2 COMPANY NODETYPE (!3)[!3, CHAR] rows 0:GqlLabel COMPANY NODETYPE COMPANY NODETYPE {NAME=ABC} !2 COMPANY,#43 COMPANY))])}

After Obey(), the database has currently

{[23, {23 PERSON NODETYPE (48) Display=0[48,Domain CHAR] rows 1 IdCol=48}]}

{[257, {vals = {(48=Joe)}}]} {(257=(150=(287=True)))}

{[48, {TableColumn 48 Definer=-502 LastChange=48 Domain CHAR Table=23 IdCol}]} ID

{[70, {70 COMPANY NODETYPE (96) Display=0[96,Domain CHAR] rows 1}]}

{[272, {vals = {(96=ABC)}}]} {(272=(202=(287=True)))}

{[96, {TableColumn 96 Definer=-502 LastChange=96 Domain CHAR Table=70}] NAME

{[119, {119 WORKSFOR EDGETYPE (150,202) Display=0[150,Domain POSITION],[202,Domain POSITION] rows 1 Indexes:((150)178;(202)232) KeyCols: (150=True,202=True) Leaving 23[178] LeaveCol=150 Arriving 70[232] ArriveCol=202}]}

{[287, {vals = {(150=257,202=272)}}]}

{[150, {TableColumn 150 Definer=-502 LastChange=150 Domain POSITION Table=119 LeaveCol[23]}]} LEAVING

{[178, {Index 178 Definer=-502 LastChange=178 for 119 count 1 Key:Domain ROW (150) Display=1[150,Domain POSITION] Kind=ForeignKey, CascadeUpdate, CascadeDelete Rows:MTree 1@0Domain ROW (150) Display=1[150,Domain POSITION]}]}

{257=(287)}

{[202, {TableColumn 202 Definer=-502 LastChange=202 Domain POSITION Table=119 ArriveCol[70]}]} ARRIVING

{[232, {Index 232 Definer=-502 LastChange=232 for 119 count 1 Key:Domain ROW (202) Display=1[202,Domain POSITION] Kind=ForeignKey, CascadeUpdate, CascadeDelete Rows:MTree 1@0Domain ROW (202) Display=1[202,Domain POSITION]}]}

{272=(287)}

We now want to change this to remove the highlighted features: what goes in their place?

LeaveCol and ArriveCol are saying that the column is a simple reference with orientations FROM and TO respectively

Leaving and Arriving are giving us the referencing indexes

IdCol says nothing at all

Consider the following, more general, representation:

1 2 3 4 5 6

12345678901234567890123456789012345678901234567890123456789012

Insert (:Person{id:'Joe'})-[:WorksFor]->(:Company{name:'ABC'})

{(!0=!0 PERSON NODETYPE (!1)[!1, CHAR] rows 0 IdCol=!1,

!1=TableColumn !1 Definer=-502 LastChange=!1 CHAR Table=!0 IdCol,

!2=!2 COMPANY NODETYPE (!3)[!3, CHAR] rows 0,

!3=TableColumn !3 Definer=-502 LastChange=!3 CHAR Table=!2,

!4=!4 WORKSFOR EDGETYPE (!7,!5)[!7, POSITION],[!5, POSITION] rows 0 Indexes:((!5)!6;(!7)!8) KeyCols: (!5=True,!7=True) Connects [!6=!0 FROM !5 LEAVING,!8=!2 TO !7 ARRIVING]}

!5=TableColumn !5 Definer=-502 LastChange=!5 POSITION Table=!4 Refs[!0 FROM LEAVING]}]}

!6=Index !6 Definer=-502 LastChange=!6 for !4 count 0 Key:Domain ROW (!5) Display=1[!5, POSITION] Kind=ForeignKey, CascadeUpdate, CascadeDelete Rows:MTree 0@0Domain ROW (!5) Display=1[!5, POSITION],

!7=TableColumn !7 Definer=-502 LastChange=!7 POSITION Table=!4 Refs[!2 TO ARRIVING]}]}

!8=Index !8 Definer=-502 LastChange=!8 for !4 count 0 Key:Domain ROW (!7) Display=1[!7, POSITION] Kind=ForeignKey, CascadeUpdate, CascadeDelete Rows:MTree 0@0Domain ROW (!7) Display=1[!7, POSITION],

#9=GqlNode #9 #9 !0 PERSON NODETYPE (!1)[!1, CHAR] rows 0 IdCol=!1:GqlLabel PERSON NODETYPE PERSON NODETYPE {ID=Joe} !0 PERSON,#10 PERSON,

#10=GqlLabel PERSON NODETYPE PERSON NODETYPE,

#20=Joe,

#29=GqlEdge #29 #29 !4 WORKSFOR EDGETYPE (!7,!5)[!7, POSITION],[!5, POSITION] rows 0 Indexes:((!5)!6;(!7)!8) KeyCols: (!5=True,!7=True) Connects [!6=!0 FROM !5 LEAVING,!8=!2 TO !7 ARRIVING]} FROM:GqlLabel WORKSFOR EDGETYPE WORKSFOR EDGETYPE !4 WORKSFOR,#30 WORKSFOR Refs #9 FROM #42 TO,

#30=GqlLabel WORKSFOR EDGETYPE WORKSFOR EDGETYPE,

#42=GqlNode #42 #42 !2 COMPANY NODETYPE (!3)[!3, CHAR] rows 0:GqlLabel COMPANY NODETYPE COMPANY NODETYPE {NAME=ABC} !2 COMPANY,#43 COMPANY,

#43=GqlLabel COMPANY NODETYPE COMPANY NODETYPE,

#56=ABC,

%0=GraphInsertStatement %0[

(0=(0=GqlNode #9 #9 !0 PERSON NODETYPE (!1)[!1, CHAR] rows 0 IdCol=!1:GqlLabel PERSON NODETYPE PERSON NODETYPE {ID=Joe} !0 PERSON,#10 PERSON,

1=GqlEdge #29 #29 !4 WORKSFOR EDGETYPE (!7,!5)[!7, POSITION],[!5, POSITION] rows 0 Indexes:((!5)!6;(!7)!8) KeyCols: (!5=True,!7=True) Connects [!6=!0 FROM !5 LEAVING,!8=!2 TO !7 ARRIVING]} FROM:GqlLabel WORKSFOR EDGETYPE WORKSFOR EDGETYPE !4 WORKSFOR,#30 WORKSFOR Refs #9 FROM #42 TO,

2=GqlNode #42 #42 !2 COMPANY NODETYPE (!3)[!3, CHAR] rows 0:GqlLabel COMPANY NODETYPE COMPANY NODETYPE {NAME=ABC} !2 COMPANY,#43 COMPANY))])}

Thus EdgeType has Connects CTree<long,(Qlx,long,string)>, and TableColumn has Refs CTree<long,(Qlx,long,string)> where the long might be undefined (POSITION) and the string might be a default. Similarly GqlEdge has Connects and Refs.

{[23, {23 PERSON NODETYPE (48) Display=0[48,Domain CHAR] rows 1}]}

{[257, {vals = {(48=Joe)}}]} {(257=(150=(287=True)))}

{[48, {TableColumn 48 Definer=-502 LastChange=48 Domain CHAR Table=23}]} ID

{[70, {70 COMPANY NODETYPE (96) Display=0[96,Domain CHAR] rows 1}]}

{[272, {vals = {(96=ABC)}}]} {(272=(202=(287=True)))}

{[96, {TableColumn 96 Definer=-502 LastChange=96 Domain CHAR Table=70}] NAME

{[119, {119 WORKSFOR EDGETYPE (150,202) Display=0[150,Domain POSITION],[202,Domain POSITION] rows 1 Indexes:((150)178;(202)232) KeyCols: (150=True,202=True) Connects [178=23 FROM 150 LEAVING,232=70 TO 202 ARRIVING]}

{[287, {vals = {(150=257,202=272)}}]}

{[150, {TableColumn 150 Definer=-502 LastChange=150 Domain POSITION Table=119 Refs[23 FROM LEAVING]}]}

{[178, {Index 178 Definer=-502 LastChange=178 for 119 count 1 Key:Domain ROW (150) Display=1[150,Domain POSITION] Kind=ForeignKey, CascadeUpdate, CascadeDelete Rows:MTree 1@0Domain ROW

(150) Display=1[150,Domain POSITION]}]}

{257=(287)}

{[202, {TableColumn 202 Definer=-502 LastChange=202 Domain POSITION Table=119 Refs[70 TO ARRIVING]}]}

{[232, {Index 232 Definer=-502 LastChange=232 for 119 count 1 Key:Domain ROW (202) Display=1[202,Domain POSITION] Kind=ForeignKey, CascadeUpdate, CascadeDelete Rows:MTree 1@0Domain ROW (202) Display=1[202,Domain POSITION]}]}

{272=(287)}

For element type definition: the metadata gives us a TList(…,CList<NodeType>,…) which we transform into a TList(…,NodeType,..) using a recursive algorithm (like Index building).

Var given = TList<Alts>..;

Var expanded = Expand(given,0,TList<NoAlts>.Empty);

Then copy expanded to a TList<NoAlts>.

TList<Alts> Expand (TList<Alts> g,int off,TList<Alts> x) // g.Length>off, x has alts only up to off-1

{..}