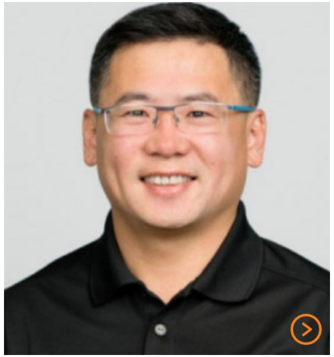
TigerGraph

GSQL Road To GQL (ISO) Standard

Mingxi Wu, VP Engineering

WHO AM I?

- Ph.D. in Database & Data Mining, University of Florida 2008
- SDE SQL server group, Microsoft 2007
- SDE relational database optimizer group, Oracle 2008-2011
- Lead SDE big data management group, Turn Inc. 2011-2014
- VP Engineering, TigerGraph 2014- now



“Graph analysis is possibly the single most effective competitive differentiator for organizations pursuing data-driven operations and decisions after the design of data capture.”

Gartner

"New and valuable insights come from finding links between well understood, integrated data."

IDC
Analyze the Future

“**Graph is the fastest way to connect data**, *especially when dealing with complex or large volumes of disparate data. Without graph, organizations have to rely on developers to write complex code that can take considerable time and effort. In some cases, it becomes impractical due to the complexity of data.”*

“Graph data platform is a new and emerging market that allows organizations to ***think differently and create new, intelligence-based business opportunities*** *that would otherwise be difficult to develop and support.”*

Forrester Research

Why Graph, Why not RDBMS
Graph handles relationship analytics better

RDBMS cannot easily model or store relationship data without complexity

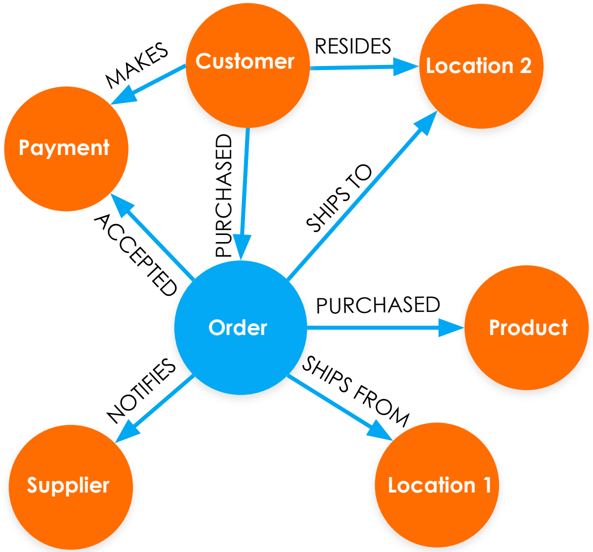
Performance degrades with number of relationships and database size

Query complexity grows with need for more JOINS

Adding new relationships or data types requires schema and modeling effort

Gartner

Graph Analytics Need Graph Query Language

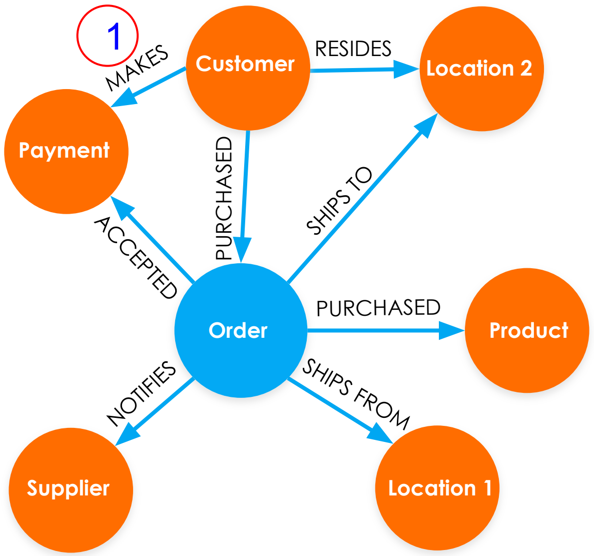


- **Definition** - Graph analytics is a set of analytic techniques that allows for the **exploration of relationships** between entities of interest such as organizations, people and transactions.
- **Forecasted growth** - 100% annually through 2022
- **What's driving the growth**
  - Need to ask **complex questions across complex data**, which is **not always practical or even possible at scale using SQL queries.** (RDBMS requires time-consuming & expensive table joins!)
- **What's needed for broad adoption of graph data stores**
  - Graph store can efficiently model, explore and query data with complex interrelationships across data silos, but the **need for specialized skills has limited their adoption to date.**

A Standard Graph Query Language (GQL) Comes To Rescue!

Source: Gartner - Top 10 data and analytics Trends for 2019

Graph Analytics Need Graph Query Language

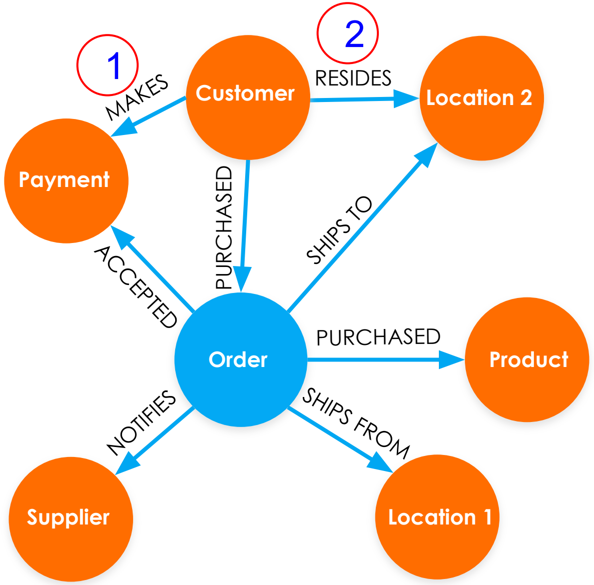


- **Definition** - Graph analytics is a set of analytic techniques that allows for the **exploration of relationships** between entities of interest such as organizations, people and transactions.
- **Forecasted growth** - 100% annually through 2022
- **What's driving the growth**
  - Need to ask **complex questions across complex data**, which is **not always practical or even possible at scale using SQL queries.** (RDBMS requires time-consuming & expensive table joins!)
- **What's needed for broad adoption of graph data stores**
  - Graph store can efficiently model, explore and query data with complex interrelationships across data silos, but the **need for specialized skills has limited their adoption to date.**

A Standard Graph Query Language (GQL) Comes To Rescue!

Source: Gartner - Top 10 data and analytics Trends for 2019

Graph Analytics Need Graph Query Language

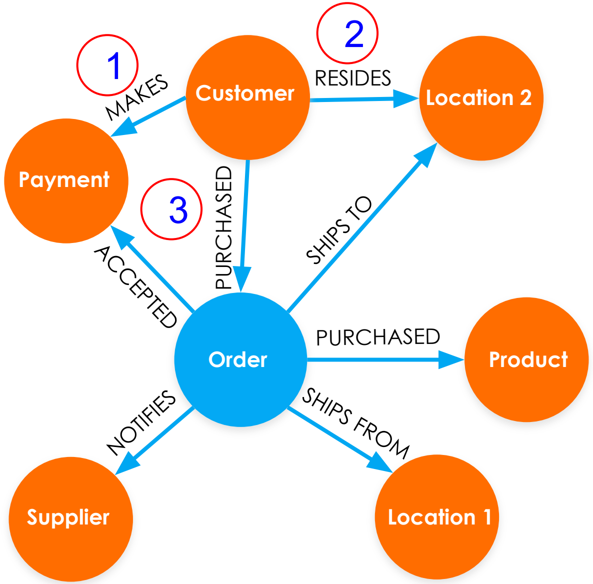


- **Definition** - Graph analytics is a set of analytic techniques that allows for the **exploration of relationships** between entities of interest such as organizations, people and transactions.
- **Forecasted growth** - 100% annually through 2022
- **What's driving the growth**
  - Need to ask **complex questions across complex data**, which is **not always practical or even possible at scale using SQL queries.** (RDBMS requires time-consuming & expensive table joins!)
- **What's needed for broad adoption of graph data stores**
  - Graph store can efficiently model, explore and query data with complex interrelationships across data silos, but the **need for specialized skills has limited their adoption to date.**

A Standard Graph Query Language (GQL) Comes To Rescue!

  Source: Gartner - Top 10 data and analytics Trends for 2019

Graph Analytics Need Graph Query Language

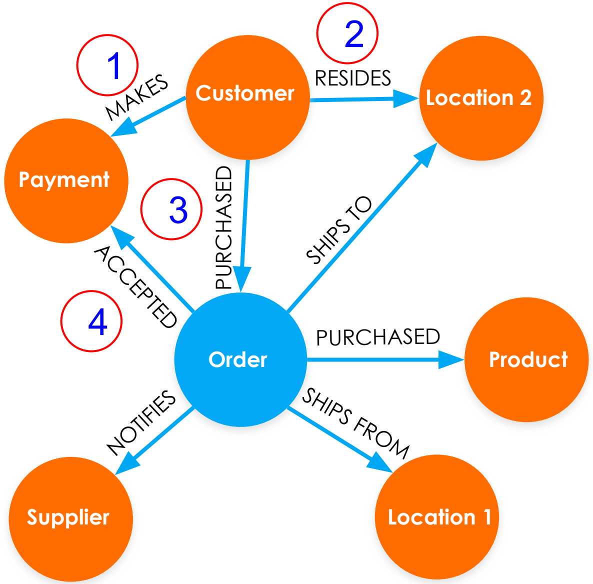


- **Definition** - Graph analytics is a set of analytic techniques that allows for the **exploration of relationships** between entities of interest such as organizations, people and transactions.
- **Forecasted growth** - 100% annually through 2022
- **What's driving the growth**
  - Need to ask **complex questions across complex data**, which is **not always practical or even possible at scale using SQL queries.** (RDBMS requires time-consuming & expensive table joins!)
- **What's needed for broad adoption of graph data stores**
  - Graph store can efficiently model, explore and query data with complex interrelationships across data silos, but the **need for specialized skills has limited their adoption to date.**

A Standard Graph Query Language (GQL) Comes To Rescue!

Source: Gartner - Top 10 data and analytics Trends for 2019

Graph Analytics Need Graph Query Language

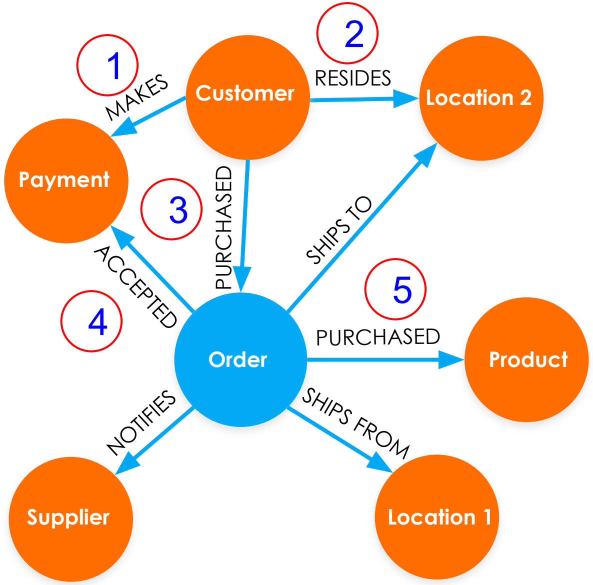


- **Definition** - Graph analytics is a set of analytic techniques that allows for the **exploration of relationships** between entities of interest such as organizations, people and transactions.
- **Forecasted growth** - 100% annually through 2022
- **What's driving the growth**
  - Need to ask **complex questions across complex data**, which is **not always practical or even possible at scale using SQL queries.** (RDBMS requires time-consuming & expensive table joins!)
- **What's needed for broad adoption of graph data stores**
  - Graph store can efficiently model, explore and query data with complex interrelationships across data silos, but the **need for specialized skills has limited their adoption to date.**

A Standard Graph Query Language (GQL) Comes To Rescue!

  Source: Gartner - Top 10 data and analytics Trends for 2019

Graph Analytics Need Graph Query Language

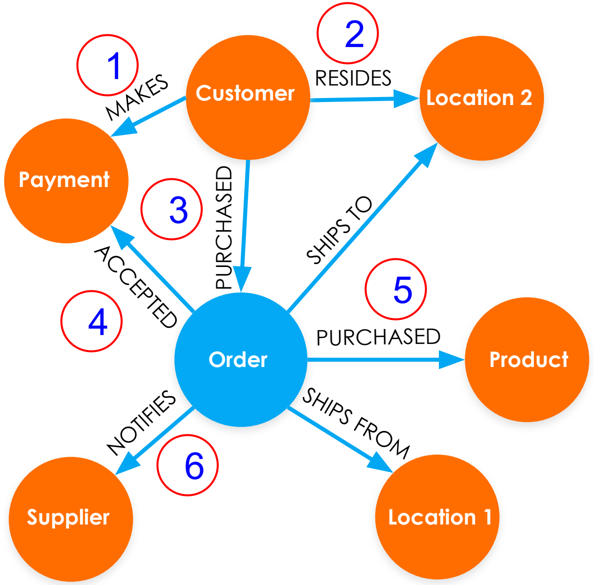


- **Definition** - Graph analytics is a set of analytic techniques that allows for the **exploration of relationships** between entities of interest such as organizations, people and transactions.
- **Forecasted growth** - 100% annually through 2022
- **What's driving the growth**
  - Need to ask **complex questions across complex data**, which is **not always practical or even possible at scale using SQL queries.** (RDBMS requires time-consuming & expensive table joins!)
- **What's needed for broad adoption of graph data stores**
  - Graph store can efficiently model, explore and query data with complex interrelationships across data silos, but the **need for specialized skills has limited their adoption to date.**

A Standard Graph Query Language (GQL) Comes To Rescue!

Source: Gartner - Top 10 data and analytics Trends for 2019

Graph Analytics Need Graph Query Language

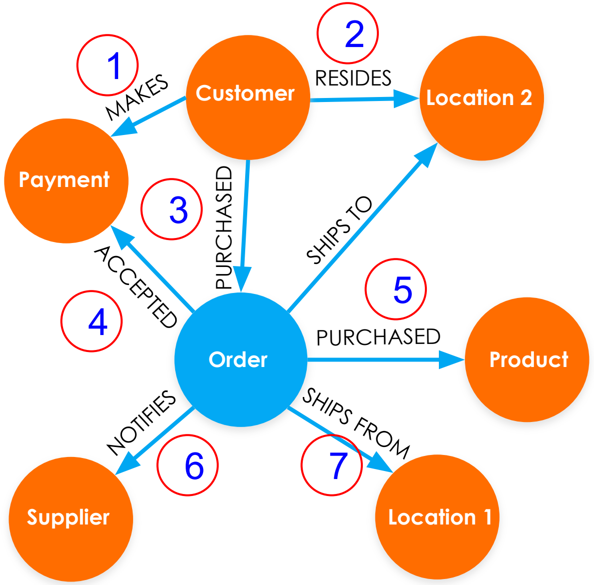


- **Definition** - Graph analytics is a set of analytic techniques that allows for the **exploration of relationships** between entities of interest such as organizations, people and transactions.
- **Forecasted growth** - 100% annually through 2022
- **What's driving the growth**
  - Need to ask **complex questions across complex data**, which is **not always practical or even possible at scale using SQL queries.** (RDBMS requires time-consuming & expensive table joins!)
- **What's needed for broad adoption of graph data stores**
  - Graph store can efficiently model, explore and query data with complex interrelationships across data silos, but the **need for specialized skills has limited their adoption to date.**

A Standard Graph Query Language (GQL) Comes To Rescue!

Source: Gartner - Top 10 data and analytics Trends for 2019

Graph Analytics Need Graph Query Language

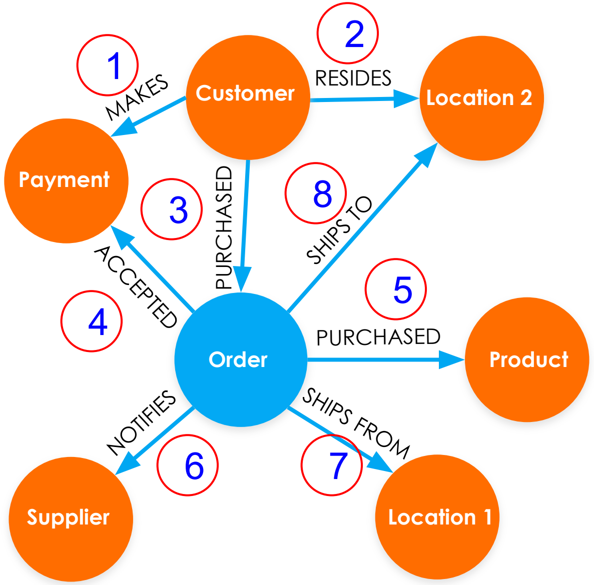


- **Definition** - Graph analytics is a set of analytic techniques that allows for the **exploration of relationships** between entities of interest such as organizations, people and transactions.
- **Forecasted growth** - 100% annually through 2022
- **What's driving the growth**
  - Need to ask **complex questions across complex data**, which is **not always practical or even possible at scale using SQL queries.** (RDBMS requires time-consuming & expensive table joins!)
- **What's needed for broad adoption of graph data stores**
  - Graph store can efficiently model, explore and query data with complex interrelationships across data silos, but the **need for specialized skills has limited their adoption to date.**

A Standard Graph Query Language (GQL) Comes To Rescue!

Source: Gartner - Top 10 data and analytics Trends for 2019

Graph Analytics Need Graph Query Language



- **Definition** - Graph analytics is a set of analytic techniques that allows for the **exploration of relationships** between entities of interest such as organizations, people and transactions.
- **Forecasted growth** - 100% annually through 2022
- **What's driving the growth**
  - Need to ask **complex questions across complex data**, which is **not always practical or even possible at scale using SQL queries.** (RDBMS requires time-consuming & expensive table joins!)
- **What's needed for broad adoption of graph data stores**
  - Graph store can efficiently model, explore and query data with complex interrelationships across data silos, but the **need for specialized skills has limited their adoption to date.**

A Standard Graph Query Language (GQL) Comes To Rescue!

Source: Gartner - Top 10 data and analytics Trends for 2019

GQL Standard History

- Early-mid-90s: semi-structured (object exchange model) research was all the rage
  - data logically viewed as graph
  - initially motivated by modeling WWW (page=vertex, link=edge)
  - query languages expressing constrained reachability in graph
- Late 90s - 2000s: special case XML (graph restricted to tree shape, a single root element)
  - Mature: W3C standard ecosystem for modeling and querying (XQuery, XPath, XLink, XSLT, XML Schema, …)
- Since mid 2000s: JSON and friends (also restricted to tree shape, a single root element)
  - Mongodb, Couchbase, GraphQL, AsterixDB, …
- Present: back to unrestricted graphs (aka “property graphs”)
  - Cypher, Gremlin, SparQL, more recently TigerGraph’s GSQL
  - Two ANSI/ISO standards coming up: SQL/PGQ (SQL extension) & GQL

GQL Standard History

- GQL Standard formation
    - 2016 - now Oracle/TigerGraph/Neo4j started discussing extension to ISO SQL standard, support property graph model
        - Project name: SQL/PGQ
    - 2019, GQL standard project for property graph-- approved by ISO Sep 2019
        - See wikipedia https://en.wikipedia.org/wiki/GQL_Graph_Query_Language
    - Biweekly, ANSI meeting by TigerGraph/Oracle/Neo4j…
        - Overlap with SQL/PGQ on pattern match section
        - Many new proposals
            - TigerGraph's multiple graphs support is adopted
            - TigerGraph's GSQL pattern match syntax sugar is adopted
    - Main topics under discussion on GQL
        - Property Graph Model and DDL
        - Supported Type System
        - Catalog structure and its supported objects
        - Query composition
        - Pattern Match as the core
        - DML - CREATE/READ/UPDATE/DELETE

What GQL On The Market?

- **Cypher** (OpenCypher project)
  - Neo4j/MemGraph/SAP HANNA/RedisGraph
- **Gremlin** (Apache project)
  - JanusGraph/Amazon Neptune/DataStax/Microsoft Cosmos DB
- **Sparql** (W3C standard, **RDF model**, latest version 1.1)
  - Cambridge Semantics/Amazon Neptune
- **PGQL** (proprietary)
  - Oracle
- **GSQL** (proprietary)
  - TigerGraph

What They Really Are?

- Cypher **- Pattern match style,** SQL complete
    - ```
      MATCH (p:person)-[f:friend_of]->(p2:person)
      WHERE p.first_name =“Dan”
      RETURN p2.first_name, p2.age
      ```
- Gremlin **- Functional chain style,** Turing complete
    - ```
      g.V().hasLabel(“person”)
            .has(“first_name”,“Dan”)
            .out(“friend_of”).as(“p2”)
            .select(“p2”).by(“first_name”).by(“age”)
      ```
- Sparql **- Triplet Pattern match style,** SQL complete
    - ```
      PREFIX foaf: <http://xmlns.com/foaf/0.1/>
      SELECT  ?p2_first_name, ?p2_age
      WHERE { ?person     is_a       foaf:Person .
              ?person     foaf:first_name   “Mingxi” .
              ?person     foaf:friend_of    ?person2 .
              ?person2    foaf:first_name  ?p2_first_name .
              ?person2    foaf:age ?p2_age }
      ```
      ```

What They Really Are?

- PGQL **- Pattern match style,** SQL complete
    - **SELECT** p2.first_name, p2.age
      FROM MATCH p:person-[f:friend_of]->p2:person
      WHERE p.first_name =“Dan”

- GSQL **- SQL Stored Procedure style,** Turing complete
    - ```
      CREATE QUERY Q() FOR GRAPH G {

         Result = SELECT p2
                     FROM person:p - (friend_of>) - person:p2
                     WHERE p.first_name ="Dan";

          PRINT Result.first_name, Result.age;
      }
      ```
      ```

Sparql - Jan 15 2008 -

- Sparql - declarative, triplet pattern match, SQL-complete

- Language Model
  - RDF Graph G + conjunction/disjunction of triplet table functions
  - Result : table output

- Match style:
  - PREFIX foaf: <http://xmlns.com/foaf/0.1/>
    SELECT ?name ?email
    WHERE { ?person **a** foaf:Person .
    ?person **foaf:name** ?name .
    ?person **foaf:mbox** ?email . }
- Branching:
  - Very limited, if-then-else, loop is hard.

- Runtime Attribute flow: just as in SQL, create graph view or use subquery

Sparql- Pros and Cons

- Pros
  - Easy for RDF characteristic
  - Borrow many from SQL (WHERE, GROUP BY, ORDER BY)
- Cons
  - Not too expressive - SQL complete
  - Flow control support very limited
  - Query Composability is not in native syntax
  - Not for property graph
  - Fine control of graph (hard)

Gremlin- Apache TinkerPop, Nov 2009-

- Gremlin - functional language, Turing complete
  - https://arxiv.org/pdf/1508.03843.pdf

- Language Model
  - Property Graph G (data) + Traversal Tao (instructions) + Set of Traversers T (read/write heads)
  - Result : the halted Traversers' locations.

- Traversal style: g.V().hasId("2").outE().inV()

- Match style:
  - g.V().match(**as("a").out("teach").as("b"),**
    as("a").out("registered").as("c")).dedup(a).select("a").by("name")

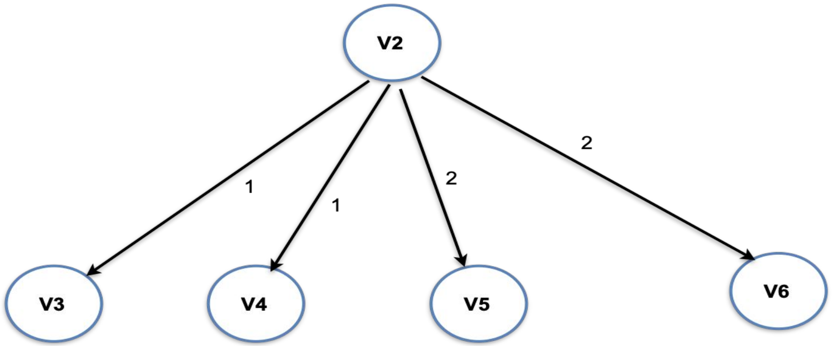
- Branching:
  - g.V().hasLabel('stock').**choose**(values('ticker')).
    option('AMZN', values('price')).
    option('FB', values('30Day-Avg'))

- Runtime Attribute flow: each traverser carry a "sack", local variable

Gremlin- Pros and Cons

- Pros
  - Expressive - Turing complete
  - Apache interactive shell - easy to start
- Cons
  - Thinking complexity is high - exponential runtime tree in developers' brain
  - Hard to do simple runtime computation when multiple passes is needed
  - Not SQL user-friendly
  - Query Calling Query is not native syntax
  - No flexible loading language

Simple Question: sum(v5+v6)-sum(v3+v4)



g.V(2).union(outE().has('weight',1).inV().sack(assign).by('vvalue').sack(mult).
by(constant(-1)).sack().sum(), outE().has('weight', 2).inV().values('vvalue').sum()).sum()

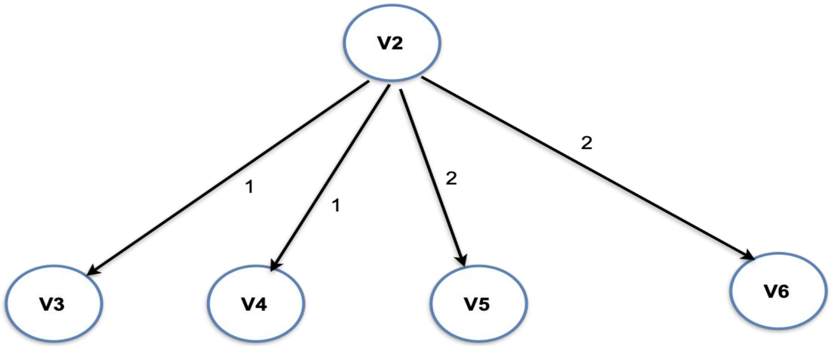
Cypher - Neo4j, early 2011-

- Cypher - declarative, pattern match, SQL-complete
  - *Cypher: An Evolving Query Language for Property Graphs*, SIGMOD 2018
- Language Model
  - Property Graph G + sequential or composition of Table functions
  - Result : table output
- Match style:
  - **MATCH** (a:teacher)-[r:teach]-(b:subject)
    RETURN a.name, count(distinct b) as subjCnt
- Tuple Flow style:
  - **MATCH** (a:teacher) -[r:teach]-> (b:subject)
    WITH a, count(distinct b) as subjCnt
    MATCH (a) -[t:has_title]-> (c:title)
    RETURN a.name, subjCnt, c.title_name
- Branching:
  - Very limited, if-then-else, loop is hard.
- Runtime Attribute flow: just as in SQL, augment output and flow to next table function

Cypher- Pros and Cons

- Pros
  - Easy for relational-mind transition to graph
  - Borrow many from SQL (WHERE, GROUP BY, ORDER BY)
- Cons
  - Not too expressive for graph - SQL complete
  - Flow control support very limited
  - Query composability is not in native syntax
  - Data dependent (schema free)
  - Iterative algorithm of graph (hard)

Simple Question: sum(v5+v6)-sum(v3+v4)

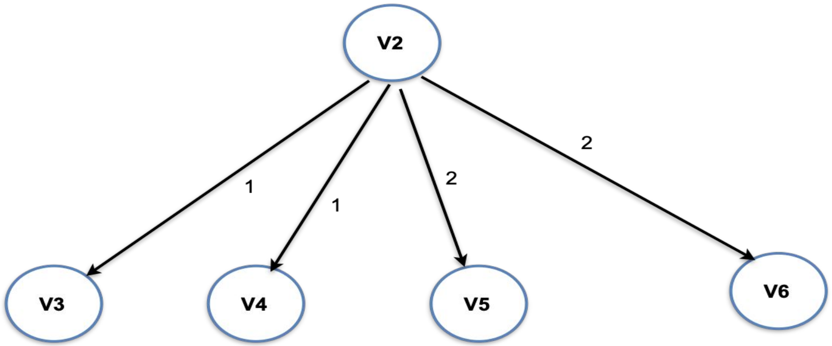



```
MATCH (a:V) - [e:E]- (b:V)
WHERE a.id = "v2" AND e.weight = 2
WITH  a, SUM(b.value) as sum1
MATCH (a) - [e:E]- (d:V)
WHERE  e.weight = 1
RETURN a, sum1 - SUM(d.value)
```

GSQL - Oct 2014 -

- GSQL - declarative, PL/SQL style or Stored Procedure style
  - *Aggregation Support for Modern Graph Analytics in TigerGraph, SIGMOD 2020*
- GSQL - turing complete
- Language Model
  - Property Graph G + DAG of GSQL query blocks
  - Result : graph or table format
- Language style:
  - composed by many single SQL block, each query block is essentially A Pattern + Accumulation
- Support Branching for different query blocks:
  - If-then-else, While-loop, Foreach
- Runtime Attribute flow: accumulator attached to vertices, accumulator storage complexity is O(V).
  - Developer’s mind will not explode.

Simple Question: sum(v5+v6)-sum(v3+v4)

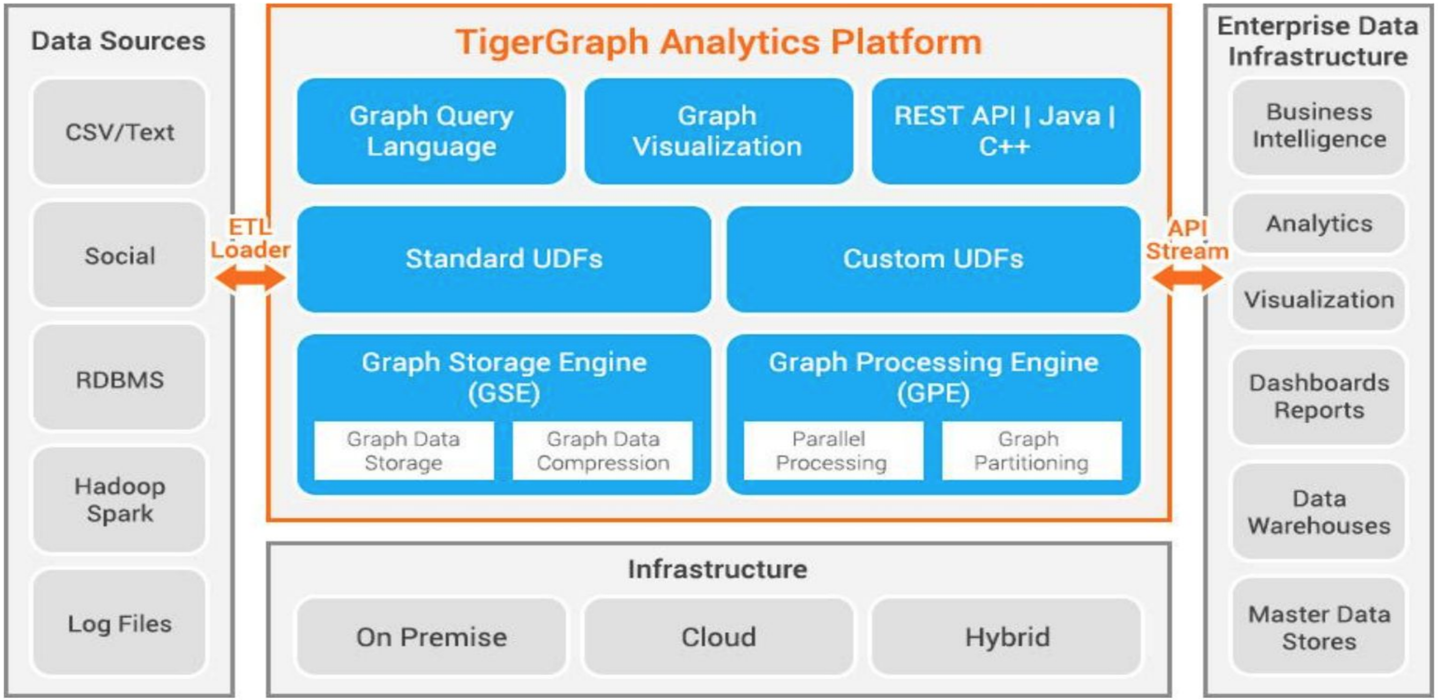



```
Start = {v2};

Result = SELECT v
         FROM Start-(:e>)-:tgt
         ACCUM
           CASE WHEN e.w == 1 THEN
                Start.@sum1 += tgt.val;
           CASE WHEN e.w == 2 THEN
                  Start.@sum2 += tgt.val;
           END;
          POST-ACCUM @@result = Start.@sum2 - Start.@sum1;

PRINT @@result;
```

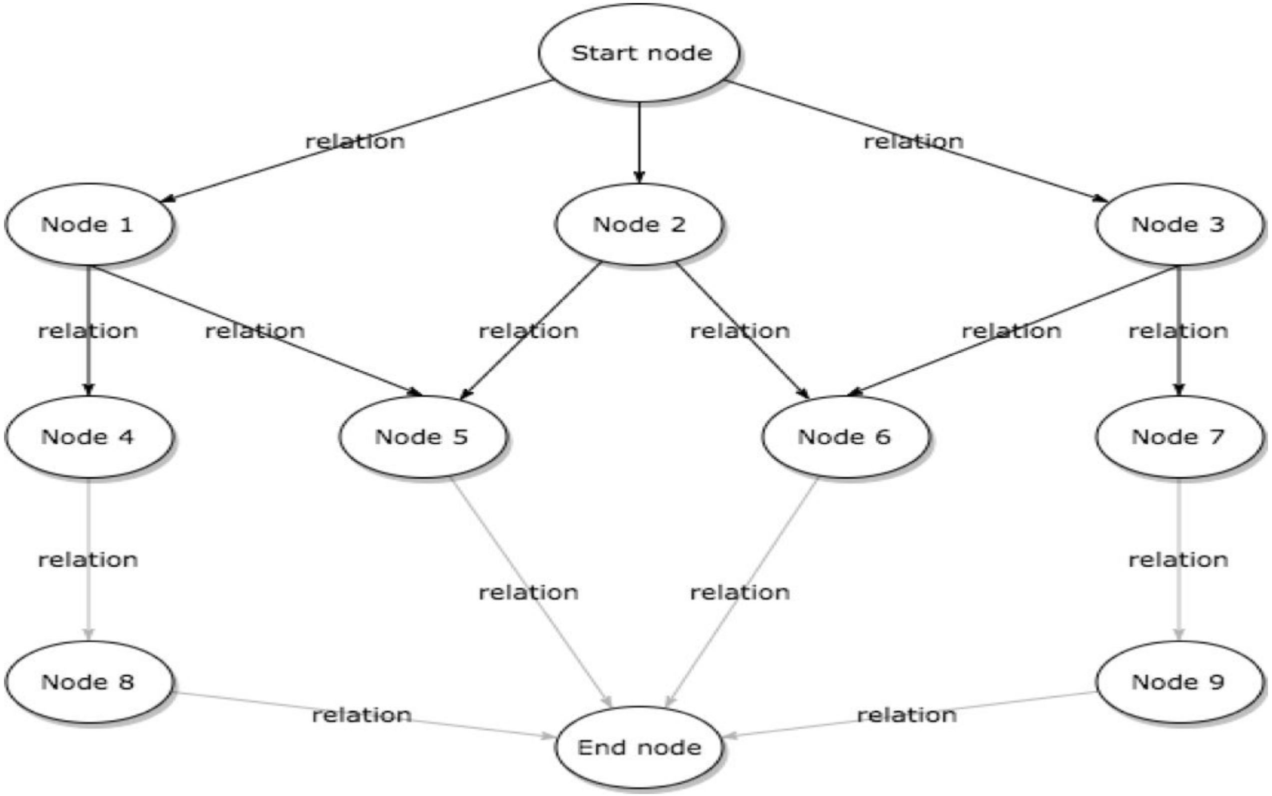
GSQL- Pros and Cons

- Pros
  - Expressive - Turing complete
  - Flow control support
  - Query Composability is in native syntax
  - Fine control of graph with accumulators
  - Expressive and elegant loading language
- Cons
  - Learning curve is a little higher than Cypher for beginners (GQL will smooth this)

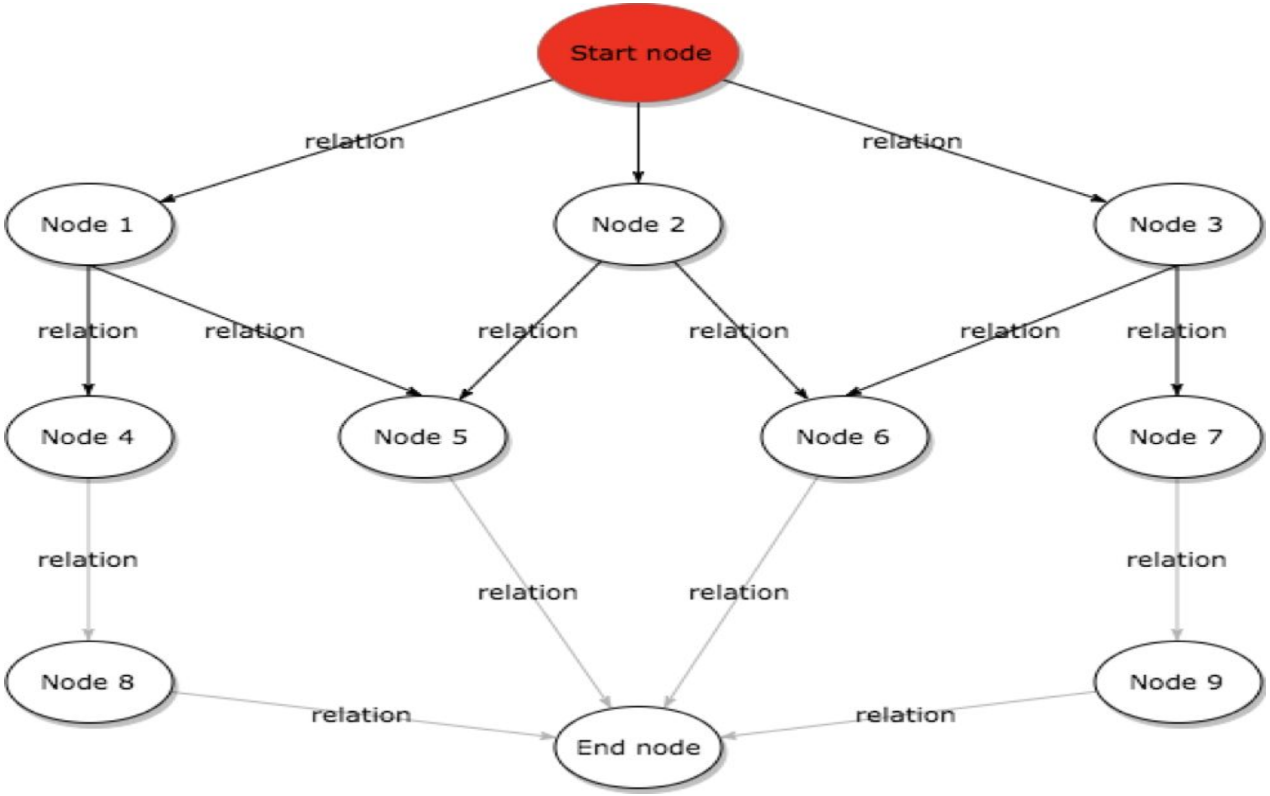
TigerGraph Architecture -- **Native Graph Storage + MPP**



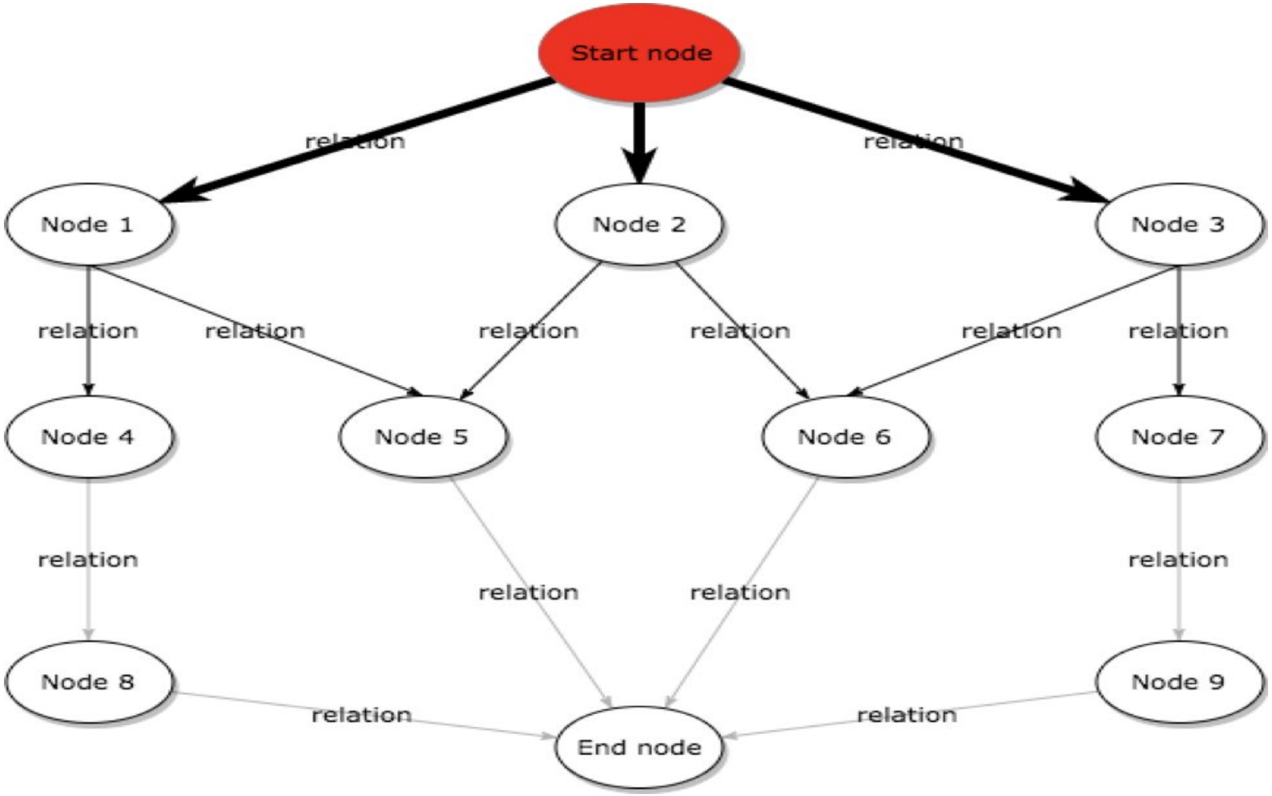
TigerGraph Parallel Processing



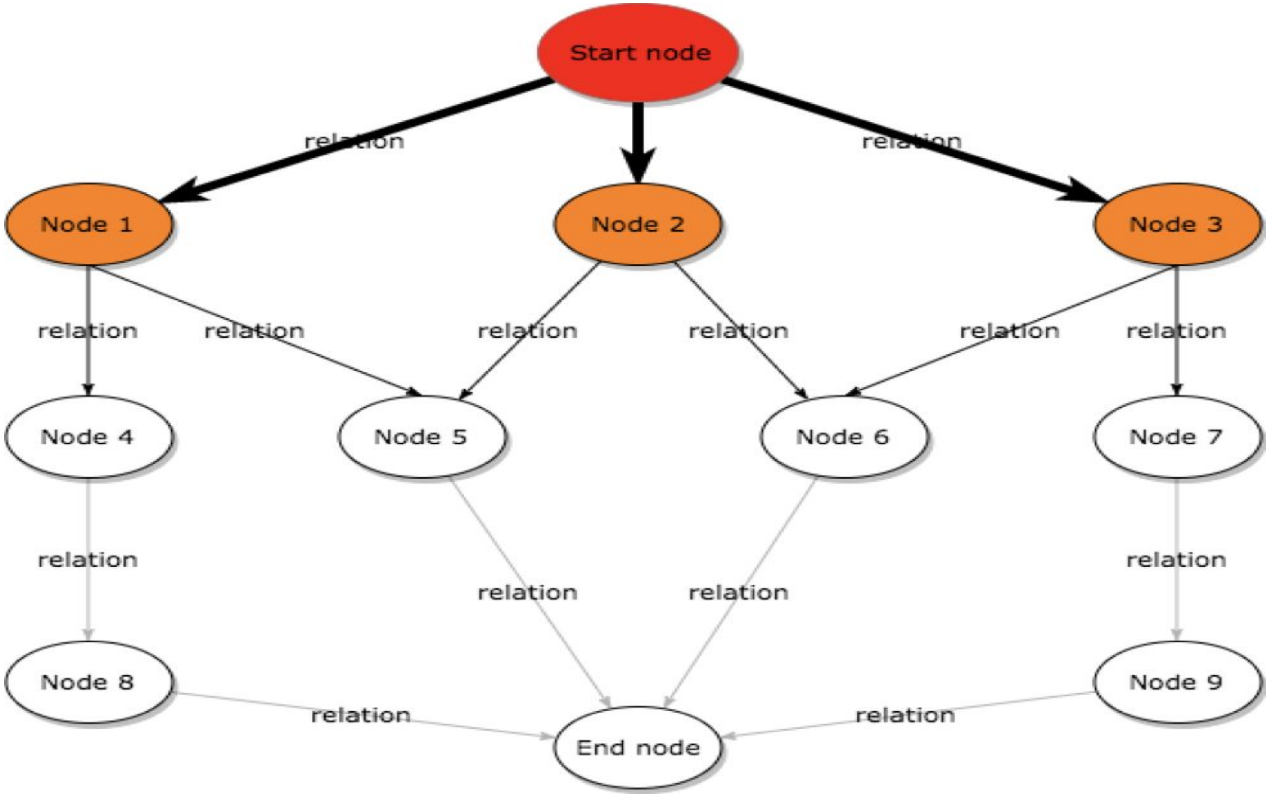
TigerGraph Parallel Processing



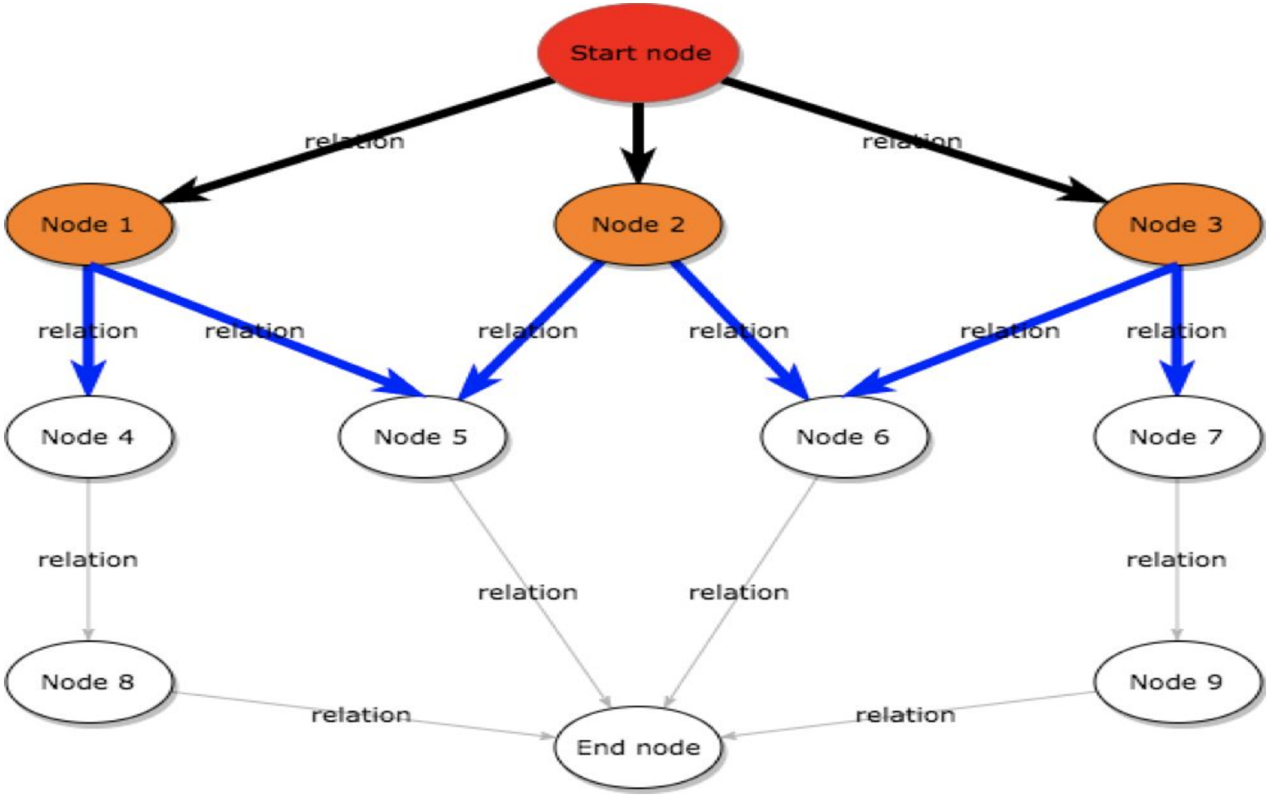
TigerGraph Parallel Processing



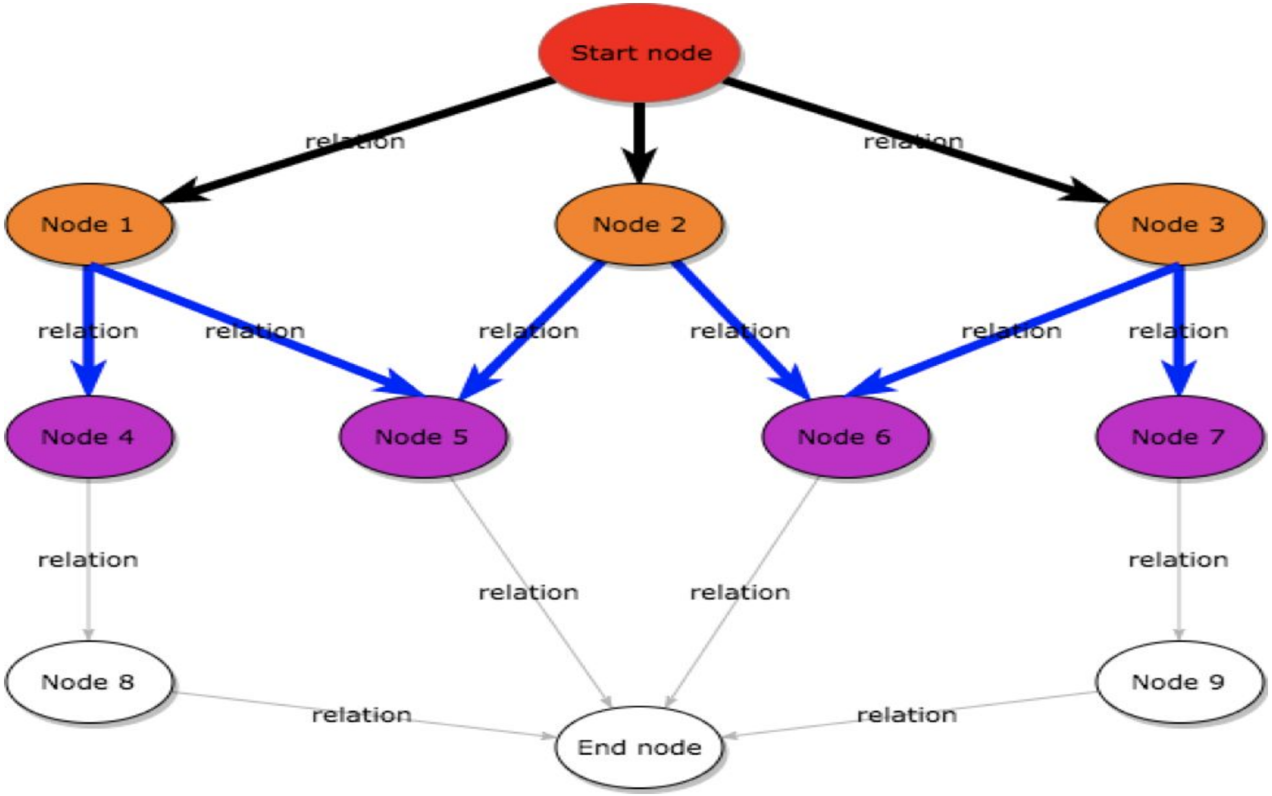
TigerGraph Parallel Processing



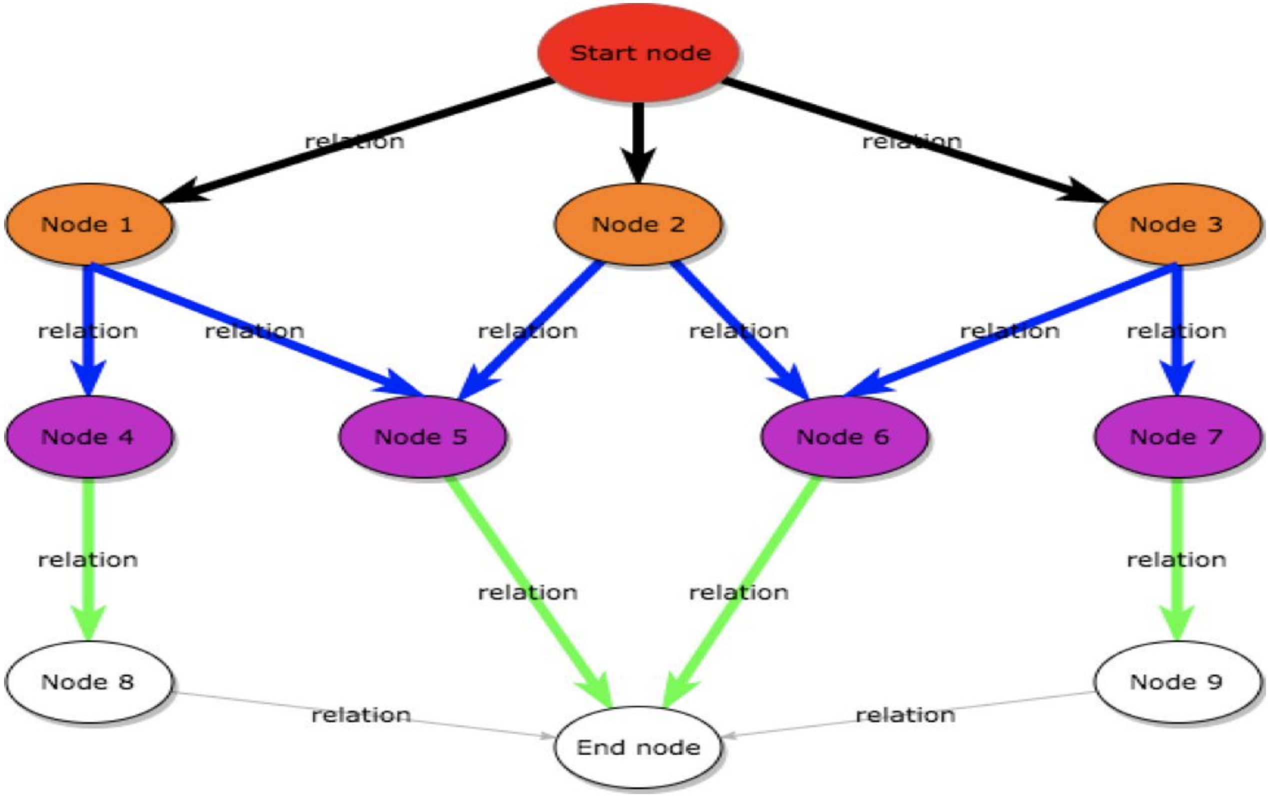
TigerGraph Parallel Processing



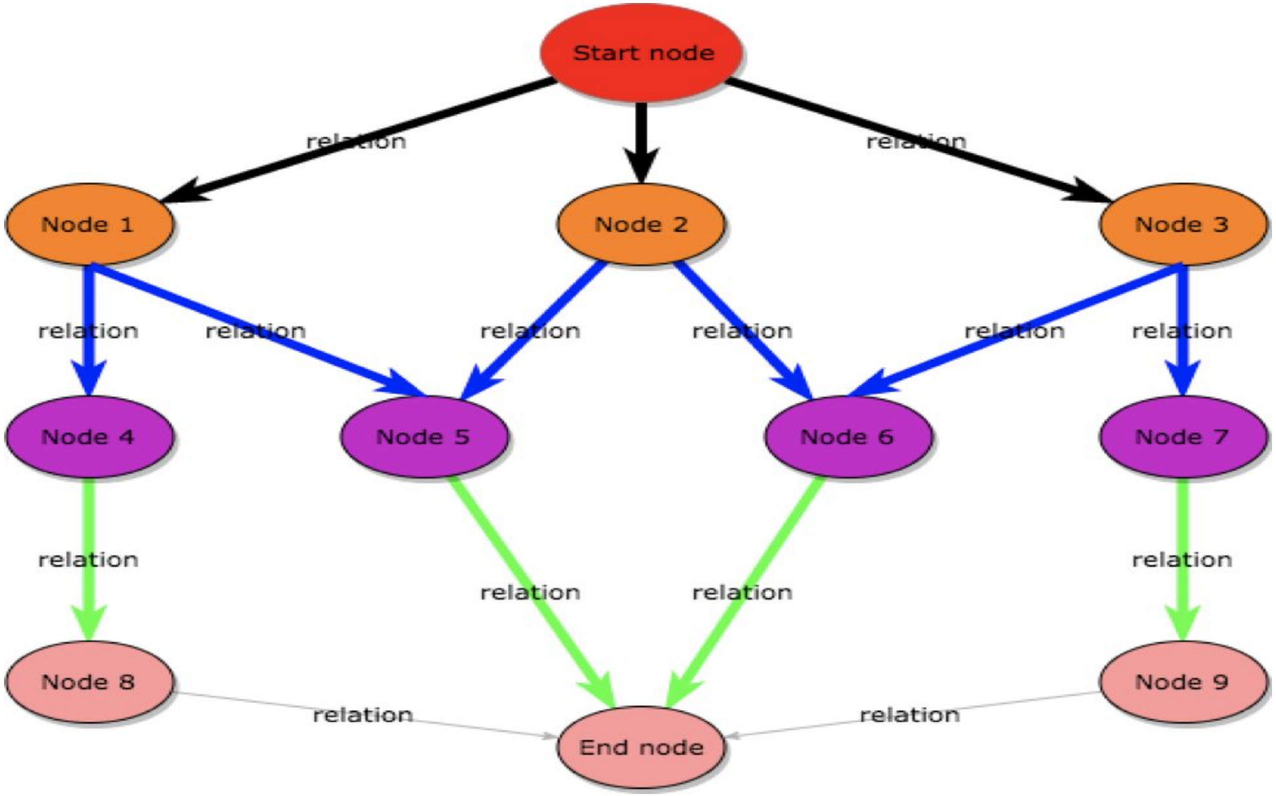
TigerGraph Parallel Processing



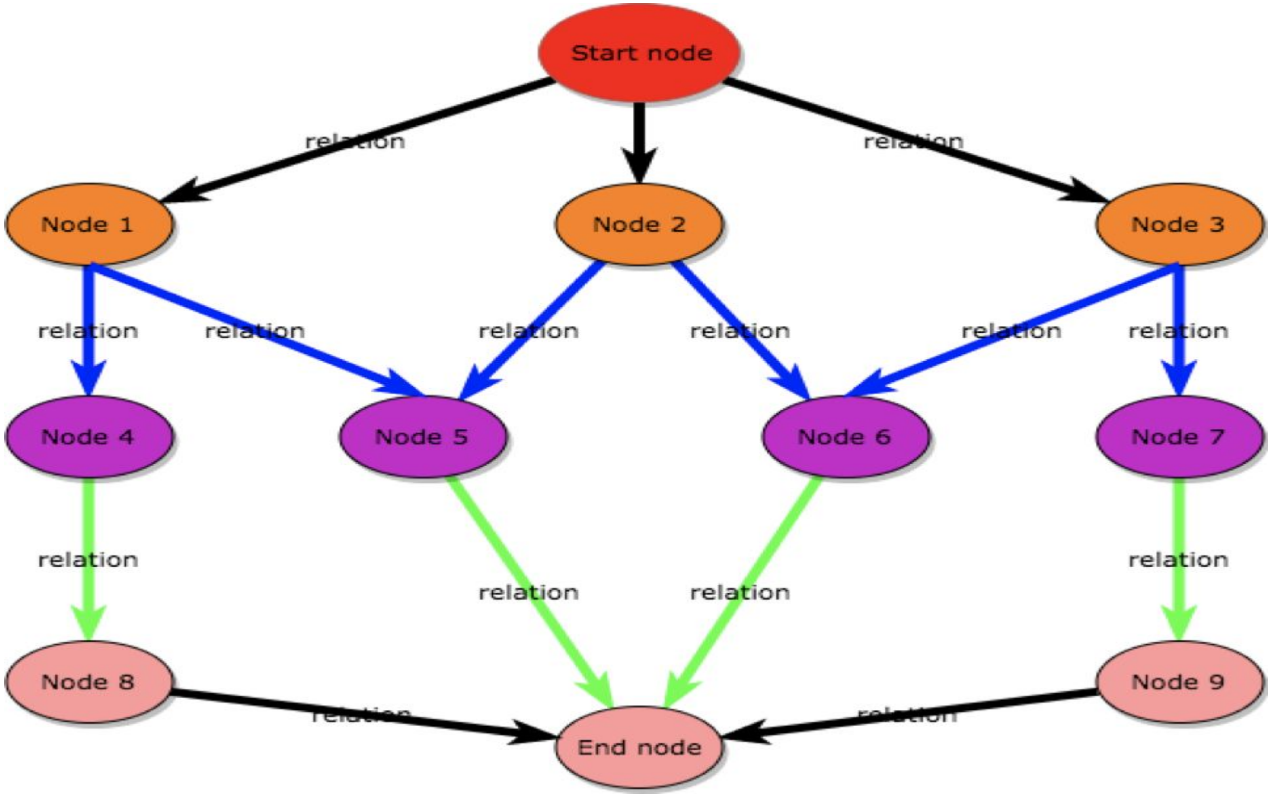
TigerGraph Parallel Processing



TigerGraph Parallel Processing



TigerGraph Parallel Processing



GSQL History

- 2015, designed from groundup based on customers requirement
  - Strong type
  - Borrow as much as we can from SQL keyword and structure
  - Introduced Accumulator (run-time state), parallel accumulation
  - Introduced flow-control (While-loop, ForEach, If-Then-Else)
  - Invented declarative data loading language

- 2017
  - Query calling query
  - Industry-first to support multiple graphs in one database.
  - Support online schema change
  - Support DML (CRUD)

- 2020
  - Linear pattern match (release 2.5)
  - Conjunctive pattern match (release 3.0)

- **Property Graph Definition**
- How to use Accumulator to do aggregation and bookkeeping
- Flow-control
- Pattern match

Property Graph Model

- Nodes correspond to entities
- Edges correspond to binary relationships
- Edges may be directed or undirected
- Nodes and edges may be labeled/typed
- Nodes and edges annotated with data
  - both have sets of attributes (key-value pairs)

- Property Graph Definition
- **How to use Accumulator to do aggregation and bookkeeping**
- Flow-control
- Pattern match

GSQL Aggregation

- Conventional (SQL-style):
    - Compute table of pattern matches, next group it
    - PGQL, Gremlin and SparQL use explicit GROUP BY clause
    - Cypher’s implicit GROUP BY has same syntax as
    - aggregation-extended conjunctive queries

- GSQL: alternate paradigm based on aggregating containers called “accumulators”
    - advantages for both naturality of specification and performance
    - Support conventional style as syntactic sugar, but accumulators remain strictly more versatile

- GSQL traversals collect and aggregate data by writing it into accumulators
- Accumulators are containers (data types) that
  - hold a data value
  - accept inputs
  - aggregate inputs into the data value using a binary operator

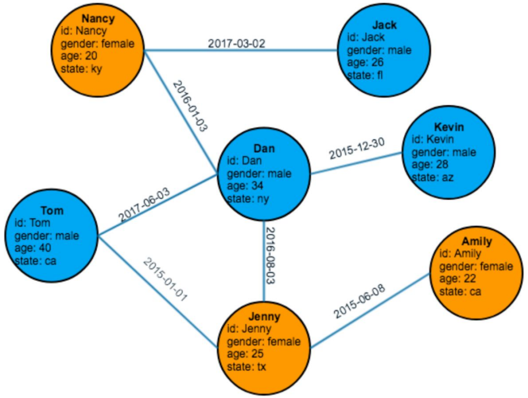
- May be built-in (sum, max, min, etc.) or user-defined
- May be
  - global (a single container per query)
  - vertex-attached (one container per vertex)

GSQL Accumulators -- Global Accumulator

```
CREATE QUERY accumulators(/* Parameters here */) FOR GRAPH MyGraph {
  // global accumulators
  SumAccum<INT> @@sum_accum;
  MinAccum<INT> @@min_accum;
  MaxAccum<INT> @@max_accum;
  OrAccum @@or_accum;
  AndAccum @@and_accum;
  ListAccum<INT> @@list_accum;

  @@sum_accum += 1;
  @@sum_accum += 2;
  PRINT @@sum_accum;

  @@min_accum += 1;
  @@min_accum += 2;
  PRINT @@min_accum;

  @@max_accum += 1;
  @@max_accum += 2;
  PRINT @@max_accum;

  @@or_accum += TRUE;
  @@or_accum += FALSE;
  PRINT @@or_accum;

  @@and_accum += TRUE;
  @@and_accum += FALSE;
  PRINT @@and_accum;

  @@list_accum += 1;
  @@list_accum += 2;
  @@list_accum += [3, 4];
  PRINT @@list_accum;
}
```

```
[
  {
    "@@sum_accum": 3
  },
  {
    "@@min_accum": 1
  },
  {
    "@@max_accum": 2
  },
  {
    "@@or_accum": true
  },
  {
    "@@and_accum": false
  },
  {
    "@@list_accum": [
      1,
      2,
      3,
      4
    ]
  }
]
```

GSQL Accumulators -- Local Accumulator



Friendship Social Graph

```
CREATE QUERY global_and_vertex_accumulators (VERTEX<person> p ) FOR GRAPH social
{
  SumAccum<INT> @@global_edge_cnt = 0;
  SumAccum<INT> @vertex_cnt = 0;

 Persons = {person.*};

  Neighbors = SELECT tgt
                        FROM Persons:src- () - :tgt
                        WHERE src == p
                        ACCUM tgt.@vertex_cnt += 1, @@global_edge_cnt +=1;
 PRINT @@global_edge_cnt;
 PRINT Neighbors[Neighbors.@vertex_cnt];



}
```

- Property Graph Definition
- How to use Accumulator to do aggregation and bookkeeping
- **Flow-control**
- Pattern match

Necessity of Flow Control - Page Rank In GSQL

```
CREATE QUERY pageRank (float maxChange, int maxIteration, float dampingFactor) {

  MaxAccum<float> @@maxDifference = 9999;  // max score change in an iteration
  SumAccum<float> @received_score = 0;     // sum of scores received from neighbors
  SumAccum<float> @score = 1;              // initial score for every vertex is 1.

  AllV = {Page.*};                         // start with all vertices of type Page
  WHILE @@maxDifference > maxChange LIMIT maxIteration DO
   @@maxDifference = 0;

    S= SELECT        s
       FROM          AllV:s -(Linkto)-> :t
       ACCUM         t.@received_score += s.@score/s.outdegree()
       POST-ACCUM    s.@score = 1-dampingFactor + dampingFactor * s.@received_score,
                     s.@received_score = 0,
                     @@maxDifference +=   abs(s.@score - s.@score');
  END;
}
```

Necessity of Flow Control - Page Rank In GSQL

```
CREATE QUERY pageRank (float maxChange, int maxIteration, float dampingFactor) {

  MaxAccum<float> @@maxDifference = 9999;  // max score change in an iteration
  SumAccum<float> @received_score = 0;      // sum of scores received from neighbors
  SumAccum<float> @score = 1;               // initial score for every vertex is 1.

  AllV = {Page.*};                          // start with all vertices of type Page
  WHILE @@maxDifference > maxChange LIMIT maxIteration DO
    @@maxDifference = 0;

    S= SELECT       s
       FROM         AllV:s -(Linkto)-> :t
       ACCUM        t.@received_score += s.@score/s.outdegree()
       POST-ACCUM   s.@score = 1-dampingFactor + dampingFactor * s.@received_score,
                    s.@received_score = 0,
                    @@maxDifference +=   abs(s.@score - s.@score');
  END;
}
```

- Property Graph Definition
- How to use Accumulator to do aggregation and bookkeeping
- Flow-control
- **Pattern match**

- Support Pattern Match Using Shortest Path Semantics
  - Aggregation semantics: would ideally consider all paths that satisfy a pattern
  - Pattern match semantics: requires limiting which paths are “legal” to consider (there can be infinitely many because of cycles)
  - Tractability requires careful design of the compromise
  - GSQL’s default semantics: “sweet spot” features both aggregation friendliness and tractability for a large class of queries that cover most of our customer use cases.
  - See TigerGraph’s **SIGMOD 2020 paper** for formal proofs and an experiment showing that on the same graph family, with GSQL’s and Cypher’s default semantics yielding same result, TigerGraph’s running time is linear in graph size, Neo4j’s is exponential (as predicted by complexity analysis)

Illustration of Alternatives of Legal Path Semantics in Pattern Match

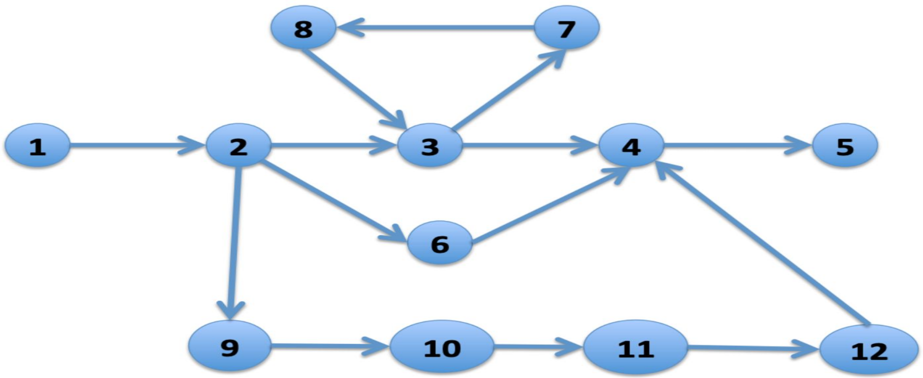


Figure 3. Shortest Path Illustration

For the pattern 1 - (_*) - 5 in Figure 3 above, you can see the following:

Illustration of Alternatives of Legal Path Semantics in Pattern Match

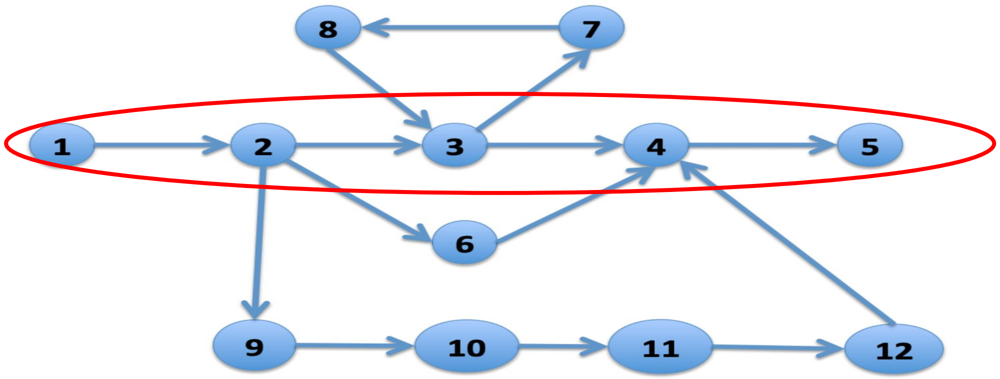


Figure 3. Shortest Path Illustration

For the pattern 1 - (_*) - 5 in Figure 3 above, you can see the following:

- **There are TWO shortest paths: 1-2-3-4-5 and 1-2-6-4-5**
  - **These have 4 hops, so we can stop searching after 4 hops. This makes the task tractable.**

Illustration of Alternatives of Legal Path Semantics in Pattern Match

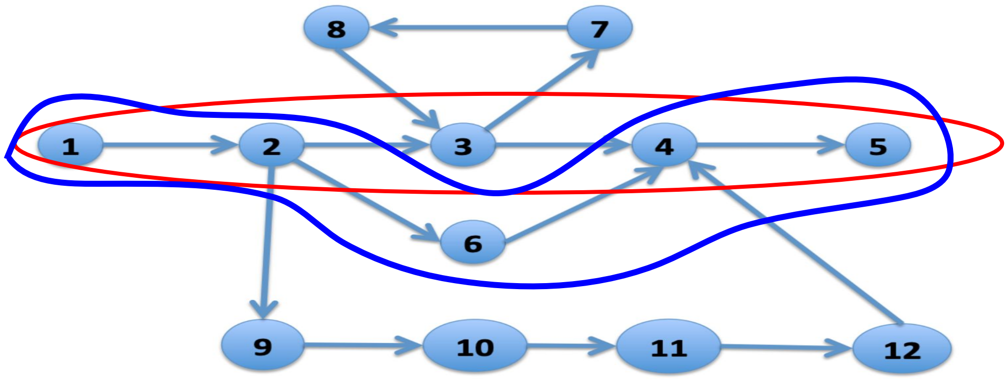
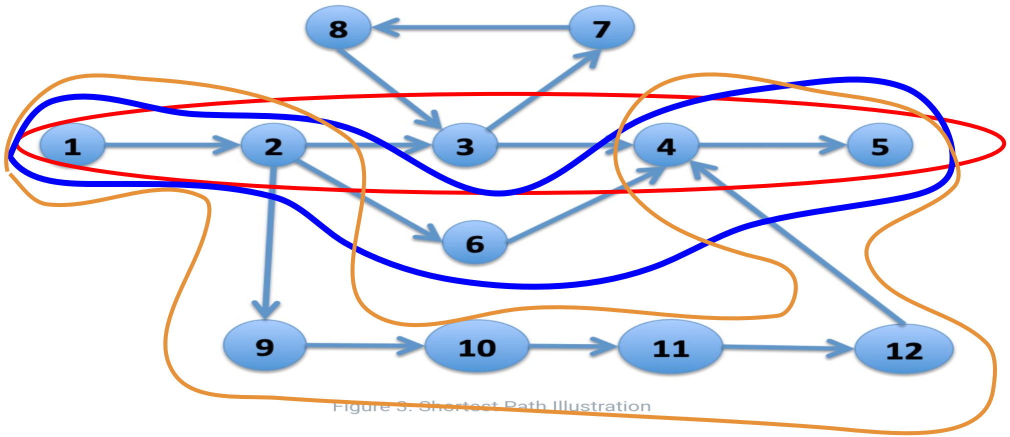


Figure 3. Shortest Path Illustration

For the pattern 1 - (_*) - 5 in Figure 3 above, you can see the following:

- **There are TWO shortest paths: 1-2-3-4-5 and 1-2-6-4-5**
    - **These have 4 hops, so we can stop searching after 4 hops. This makes the task tractable.**

Illustration of Alternatives of Legal Path Semantics in Pattern Match



For the pattern 1 - (_*) - 5 in Figure 3 above, you can see the following:

- **If we search for ALL paths which do not repeat any vertices**
  - **There are THREE non-repeated-vertex paths: 1-2-3-4-5, 1-2-6-4-5, and 1-2-9-10-11-12-4-5**
  - The actual number of matches is small, but the number of paths is theoretically very large.

Illustration of Alternatives of Legal Path Semantics in Pattern Match

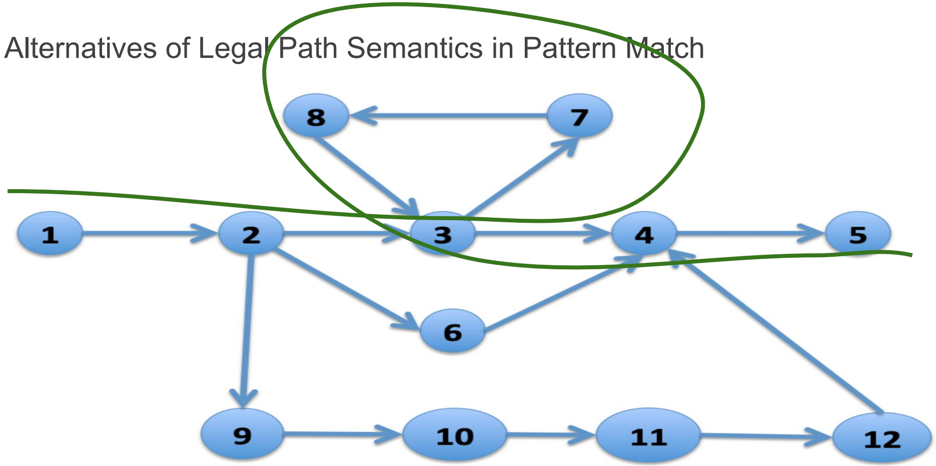


Figure 3. Shortest Path Illustration

For the pattern 1 - (_*) - 5 in Figure 3 above, you can see the following:

- If we search for ALL paths which **do not repeat any edges**:
  - **There are FOUR non-repeated-edge paths: 1-2-3-4-5, 1-2-6-4-5, 1-2-9-10-11-12-4-5, and 1-2-3-7-8-3-4-5**
  - The actual number of matches is small, but number of paths to consider is NP

Illustration of Alternatives of Legal Path Semantics in Pattern Match

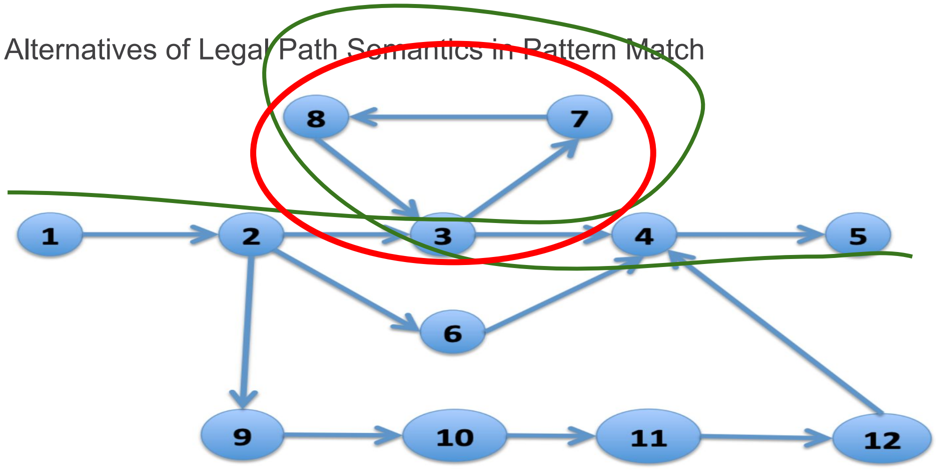


Figure 3. Shortest Path Illustration

For the pattern 1 - (_*) - 5 in Figure 3 above, you can see the following:

- If we search for **ALL paths** with no restrictions:
  - **There are an infinite number of matches, because we can go around the 3-7-8-3 cycle any number of times.**

Conjunctive Pattern Match + Accumulators

```

USE GRAPH ldbc_snb

CREATE QUERY bi_17(string cName) FOR GRAPH ldbc_snb SYNTAX v2 {
  TYPEDEF TUPLE <uint a, uint b, uint c> triplet;
  SetAccum<triplet> @@tripletSet;
  SumAccum<int> @@tripletCount;

  C =
    SELECT c
    FROM Country:c -(<IS_PART_OF.<IS_LOCATED_IN)- Person:p1,
         :c -(<IS_PART_OF.<IS_LOCATED_IN)- Person:p2,
         :c -(<IS_PART_OF.<IS_LOCATED_IN)- Person:p3,
         :p1 -(KNOWS)- :p2 -(KNOWS)- :p3 -(KNOWS)- :p1
    WHERE c.name == cName AND p1.id < p2.id AND p2.id < p3.id
    ACCUM @@tripletSet += triplet(p1.id, p2.id, p3.id);

  @@tripletCount = @@tripletSet.size();
  @@tripletSet.clear();
  PRINT @@tripletCount;
}
```

GSQL to GQL Timeline

- GQL roadmap
  - Property Graph Model and DDL
  - Pattern Match as the core
  - Catalog support multiple graphs
  - Support query composition
  - Support DML - CREATE/READ/UPDATE/DELETE
- GSQL evolvement & roadmap
  - Property Graph Model (GSQL started with this model back to 2012)
  - Catalog support multiple graphs (Mar 2016, GSQL supports multiple graphs in a catalog)
  - Support query composition (Sep 2016, GSQL support query calling query)
  - Pattern Match as the core (Mar 2020, GSQL start supporting single pattern match, by Jun 2020, start supporting Conjunctive Pattern Match)
  - Support DML - CREATE/READ/UPDATE/DELETE (Jun 2020, GSQL start supporting DML on Pattern Match)
  - Nov 2020-, closely follow GQL and add compatibility feature as standard forming
  - By end of 2021, GSQL should be fully compatible with GQL, the first draft that is subject to international circulation

Benchmark on Massive Graph

https://www.tigergraph.com/benchmark/

- Try TigerGraph Cloud https://www.tigergraph.com/cloud/
- Download free https://info.tigergraph.com/enterprise-free
- TigerGraph Certification https://www.tigergraph.com/certification/
- Join Community https://community.tigergraph.com/
- YouTube Channel https://www.youtube.com/c/TigerGraph/videos
- Starting Document
  - https://docs.tigergraph.com/start/gsql-101
  - https://docs.tigergraph.com/start/gsql-102

Hiring https://www.tigergraph.com/join-us/