LinkedIn’s Economic Graph
A digital representation of the global economy.

- 810M Members
- 58M Companies
- 120K Schools
- 38K Skills
Economic Graph Application Development
A view of a vicinity in the graph
Application Development: Graph Traversals

Members accessing the Economic Graph

- **250B** Edges
  All bi-directional relationship in the graph.

- **1.6M** QPS
  All LinkedIn application and web traffic.

- **XX ms** Latency
  Fast access to the graph.
Scale.
1. Create a Graph Index that Scales to Hundreds of Joins
2. Scale out with a Distributed Graph
3. Enable Fast Grow for New Data
4. Add and Modify Queries Fast
5. Simplify Operations
Graph Index Scales to xx Joins
Works at Scale, Concurrently

- In-memory relational system on hashed storage
- Wait-free data structures
- Single writer shared memory
- Writers: process isolation, read-only shared memory, pinned to core
- Memory bandwidth linear throughput to 50+ cores (current workload)
Log Structure

• Serialized graph; compaction on demand

• Branches: what-if queries

• Point in time queries

• → 1.5TB RAM graph shards
Scale out with a Distributed Graph
One Graph: multiple shards

- Graph distributed in a cluster (aka a graph replica)
- Eventual consistency: each shard updated independently
- Hash based sharding
- \( \rightarrow \) 250B bi-directional edges
HA through replication

- Scale-out throughput with multiple replicas
- Repair through periodic snapshot & spare nodes “at ready”
- Multi-zone deployment
- 99.99+% available
Enable Fast Grow for New Data
Declarative transformation

- Constant time schema evolution
- New datasets can be appended to an existing index
Add and Modify Queries Fast
Datalog: Declarative & Modular

• Constant time schema evolution

• New datasets can be appended to an existing index
Basic Ingredient: Triples

Edge("member:02", "name", "Bogdan").

Edge("member:01", "name", "George").

Edge("member:02", "name", "Bogdan").

Edge(x, "name", y)?

"member:01", "name", "George"

"member:02", "name", "Bogdan"
Dev Dream: Composable Rules

Member(id, name, date) :-
    Edge(id, "name", name), Edge(id, "joined", date).

Member("member:02", _, _)?
    "member:02", "Bogdan", 1995

Member(_, _, 1995)?
...

GoldMember(id, name, Date) :-
    Member(id, name, date), Edge(member, "status", "Gold").
N-ary relationships

Edge("SchoolAttendance", "liquid/compound_predicate", "student"),
Edge("SchoolAttendance", "liquid/compound_predicate", "school"),
Edge("SchoolAttendance", "liquid/compound_predicate", "matriculated"),
Edge("SchoolAttendance", "liquid/is_literal_compound", "false").

SchoolAttendance@((cid=x, matriculated="1984", school="school:01", student="member:03").

SchoolAttendance@((cid=_, matriculated=_, school=_, student=_)?
"{matriculated:1984,school:school:01,student:member:03}",
"matriculated", "1984"

DegreeGranted(d, m, sc, st) :-
    SchoolAttendance@(cid=x, matriculated=m, school=sc, student=st),
    Edge(x, "degree", d).
Simplify Operations
Automate everything

- Throughput management
- State-machine controller
- Self-repair
- Image and snapshot lifecycle
- Continuous correctness checks
- Continuous utilization metrics

... monitor toil and automate it.
Use the LinkedIn app, make our day!

100% in production since Nov 2020

99.99x% available
Summary: A Scale-out Graph Index System

- Build a global view graph: Index heterogeneous data as a homogeneous graph with xxxB edges
- Build one-query apps: declarative, composable queries with yy joins
- Scale your application to millions of QPS.
- Serve it with 99.99+% availability
- Iterate quickly
LinkedIn Systems Lab designs, develops, and evaluates novel technologies in the areas of distributed systems, high-performance computing, and databases.

We focus on identifying and applying the most innovative ideas to LinkedIn's data systems infrastructure while engaging with the academic and research communities.
Thank you