gMark: Schema-Driven Generation of Graphs and Queries

Guillaume Bagan* Angela Bonifati* Radu Ciucanu† George Fletcher§ Aurélien Lemay¶ Nicky Advokaat§

*Univ Lyon 1 & CNRS LIRIS †Univ Clermont Auvergne & CNRS LIMOS §TU Eindhoven ¶Univ Lille 3 & Inria

1. gMark: motivation and goals

Graph-structured data collections are increasingly common in many application domains.

Domain- and application-independent instance and query workload generators are important for the experimental study of data-intensive systems.

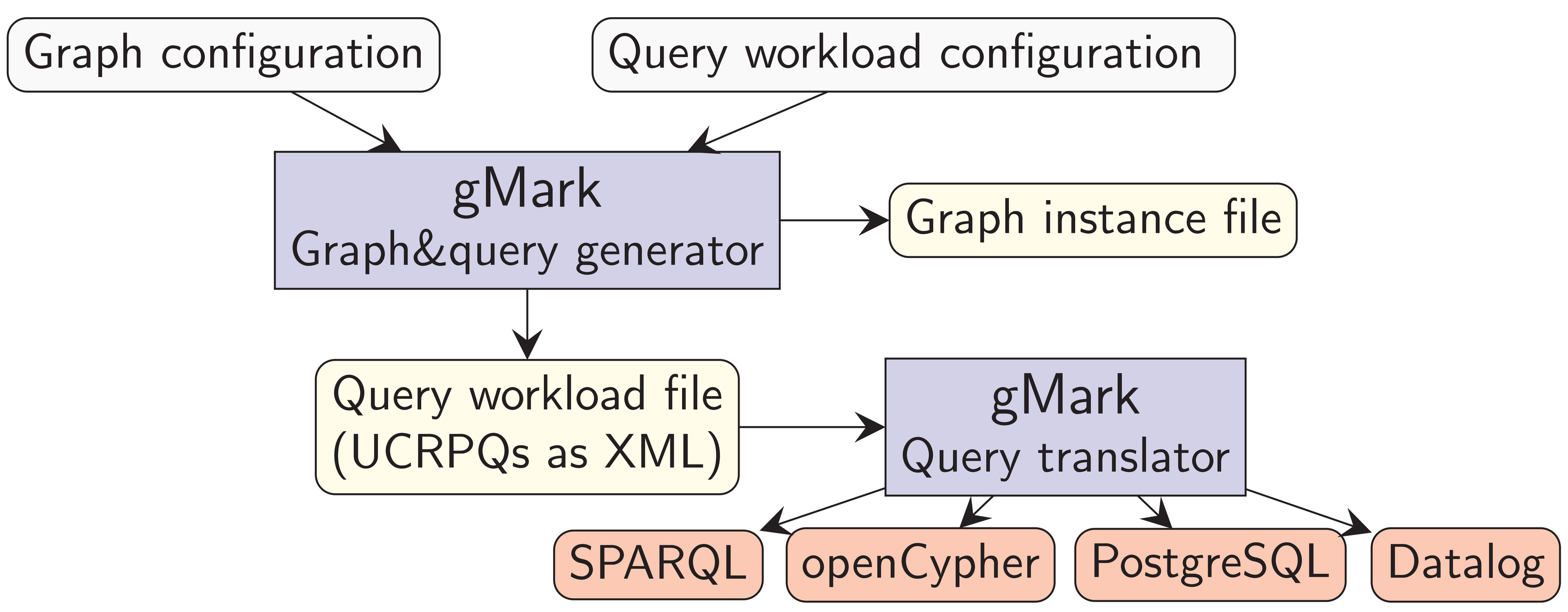
Currently there is no such framework for graph databases.
We have developed gMark, the first generation framework that

- is schema-driven, providing user-tailored instance and workload diversity, including flexible generation of recursive path queries; and
- provides control of query selectivities, in a purely instance-independent schema-driven fashion.

gMark is open-source and ready to use!

https://github.com/graphMark/gmark

2. gMark workflow



3. gMark graph configurations

Parameter	Description
Size	# of nodes to generate
Node types	finite set of node types ▷ e.g., `researcher`, `paper`, `conference`, `city`
Edge predicates	finite set of edge predicates ▷ e.g., `authors`, `publishedIn`, `heldIn`
Schema constraints	proportion of node types and edge predicates ▷ e.g., 50% of all nodes are `researchers`
Degree distributions	on the in- and out-degree of edge predicates ▷ e.g., the number of authors on papers follows a Gaussian distribution, whereas the number of papers authored by a researcher follows a Zipfian

source type *predicate* *target type*	*In-distr.*	*Out-distr.*
`researcher` `authors` `paper`	Gaussian	Zipfian

4. gMark query workload configurations

Parameter	Description
Workload size	# of queries to generate
Arity	min/max arity of generated queries ▷ e.g., min query arity = 0 and max = 5
Shapes (of join graphs)	chain, star, cycle, and/or star-chain ▷ e.g., generate star and star-chain queries
Selectivity	constant, linear, and/or quadratic ▷ e.g., generate only linear queries
Probability of recursion	percentage of conjuncts having a Kleene star ▷ e.g., 10% occurrence of recursion
Query size	# of conjuncts, # of disjuncts, etc. ▷ e.g., generate only triangles (i.e., queries with exactly 3 conjuncts)

gMark generates unions of conjunctions of regular path queries (UCRPQ)
e.g., the query $(?x, ?y, ?z) \leftarrow (?x, (a \cdot b + c)^*, ?y), (?y, a, ?w), (?w, b^-, ?z)$
has 3 conjuncts, having 1 or 2 disjuncts, having path length 1 or 2.

5. gMark evaluation

The problems of *graph* and *query generation* are theoretically intractable
⇒ We developed *best-effort* algorithms that work in *linear time*.

(a) gMark scalability

Instance generation. We adapted the scenarios of several popular use cases into meaningful gMark configurations:

- `Bib`: our default bibliographical use-case
- `LSN`: LDBC social network benchmark
- `WD`: WatDiv e-commerce benchmark
- `SP`: SP2Bench DBLP benchmark

	100K	1M	10M	100M
`Bib`	0m0.057s	0m0.638s	0m8.344s	1m28.725s
`LSN`	0m0.225s	0m1.451s	0m23.018s	3m11.318s
`WD`	0m2.163s	0m25.032s	4m10.988s	113m31.078s
`SP`	0m0.638s	0m7.048s	1m28.831s	15m23.542s

Graph instance generation times, with varying graph sizes (# nodes)

Generation time depends heavily on density of instances (e.g., `WD` has 100x number of edges than `Bib`).

Query generation. gMark *generates* workloads of *1000 queries* for `Bib` in $\sim 0.3s$; `LSN` and `SP` in $\sim 1.5s$; and, for the richer `WD` scenario in $\sim 10s$.

Query translation of the 1000 queries into all four supported syntaxes for each of the four scenarios required $\sim 0.1s$.

(b) gMark accuracy for the query selectivity estimation

Given a binary query Q and a graph G, we assume that $|Q(G)| = |G|^\alpha$, where α is the **selectivity value** (0–constant, 1–linear, 2–quadratic).

- Experiments confirmed the assumption and the estimation quality.

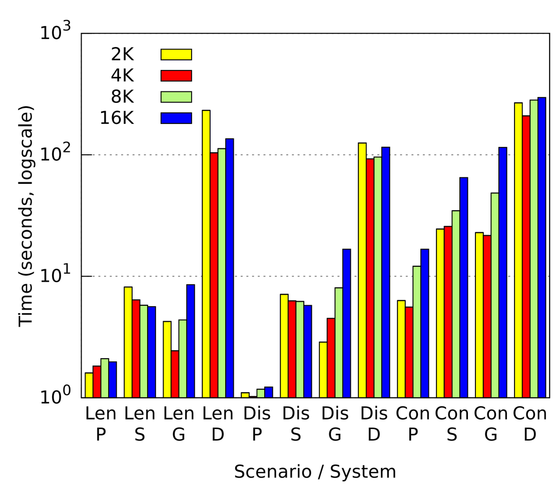
6. Evaluation of current graph DBMSs with gMark

We studied query evaluation performace of four mainstream graph DBMSs:

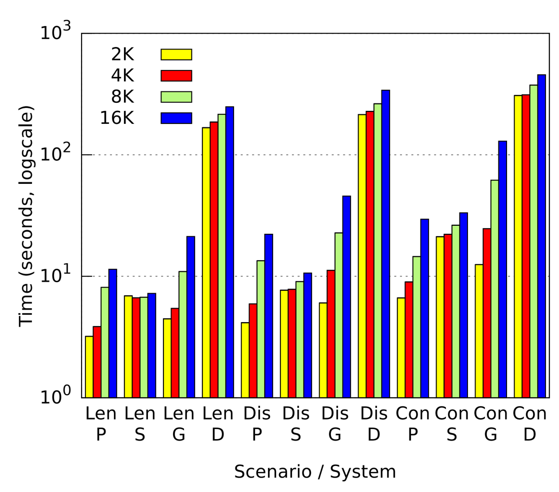
- **P**: PostgreSQL v9.3.9 (SQL:1999 recursive views)
- **S**: a popular SPARQL query engine (SPARQL 1.1)
- **G**: a native graph database (openCypher)
- **D**: a modern Datalog engine (Datalog)

(a) Non-recursive queries

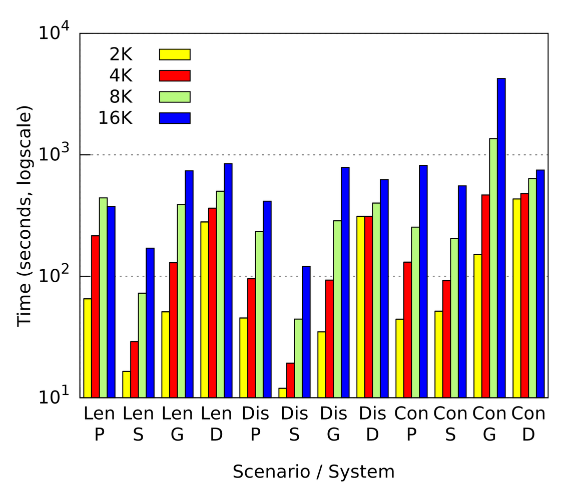
Query execution times for various graph sizes and diverse query workloads: *Len* (varying path lengths, 1 disjunct, 1 conjunct), *Dis* (multiple disjuncts, 1 conjunct), *Con* (multiple conjuncts and disjuncts):



Constant queries



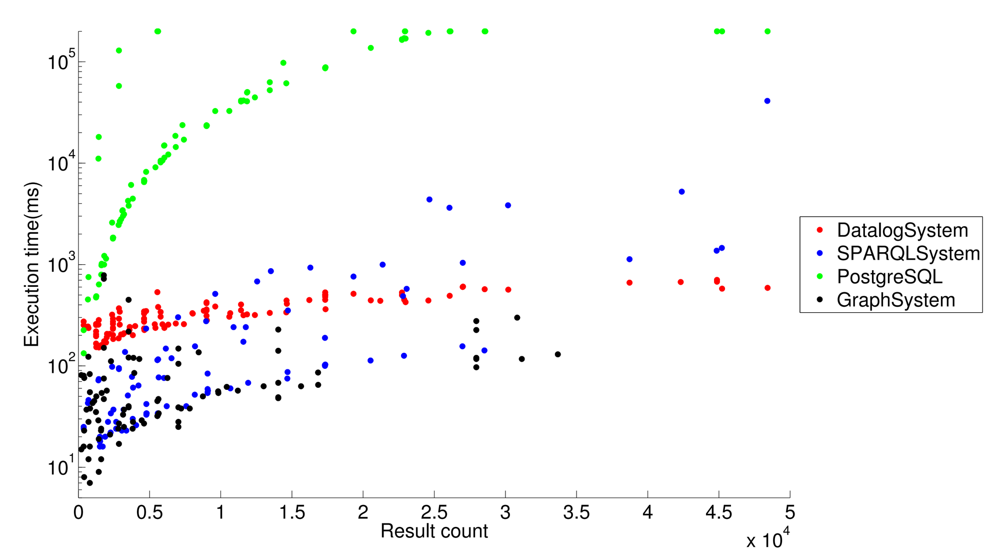
Linear queries



Quadratic queries

(b) Recursive queries

Query execution times for simple recursive queries on various small graph sizes (from 2K to 32K nodes):



References

[1] G. Bagan, A. Bonifati, R. Ciucanu, G. Fletcher, A. Lemay, N. Advokaat. gMark: Schema-Driven Generation of Graphs and Queries. IEEE TKDE, 2017. http://arxiv.org/abs/1511.08386

[2] G. Bagan, A. Bonifati, R. Ciucanu, G. Fletcher, A. Lemay, N. Advokaat. Generating flexible workloads for graph databases. VLDB Demo, 2016.