Photo: Gephi, flickr

NoSQL Frankfurt

co-located to ICOODB 2010
28. September 2010
Frankfurt, Germany

# The GraphDB Landscape and sones

Achim Friedland <achim@sones.de>

Photo: litlnemo, flickr

Photo: litlnemo, flickr

- 43 years Object Orientation / OOP
- 35 years Relational Databases / SQL
- 21 years Semantic Web / RDF

Photo: litlnemo, flickr

- 43 years Object Orientation / OOP
- 35 years Relational Databases / SQL
- 21 years Semantic Web / RDF

…but still we do not know where our files have gone!

Photo: Chris Westermeyer, flickr

# Our wish list…

Photo: litlnemo, flickr

# Simple management of structured, semi-structured and unstructured information

Photo: litlnemo, flickr

# Simple management of structured, semi-structured and unstructured information

SQL

XML/JSON

binaries

Photo: Robbert van der Steeg, flickr

# Simple recursively linked information models

Photo: Robbert van der Steeg, flickr

# Simple recursively linked information models

- <a href = "…">
- metadata e.g. EXIF

Photo: Khem, flickr

# Traversing linked information, finding shortest-paths, do semantic partitions

Photo: NASA, flickr

# Recommendation and discovery of potentially interessting linked information

Photo: NASA, flickr

personal social item-related

# Recommendation and discovery of potentially interessting linked information

Photo: squacco, flickr

# Consistency criteria and indices for simple attributes up to complex subgraph structures

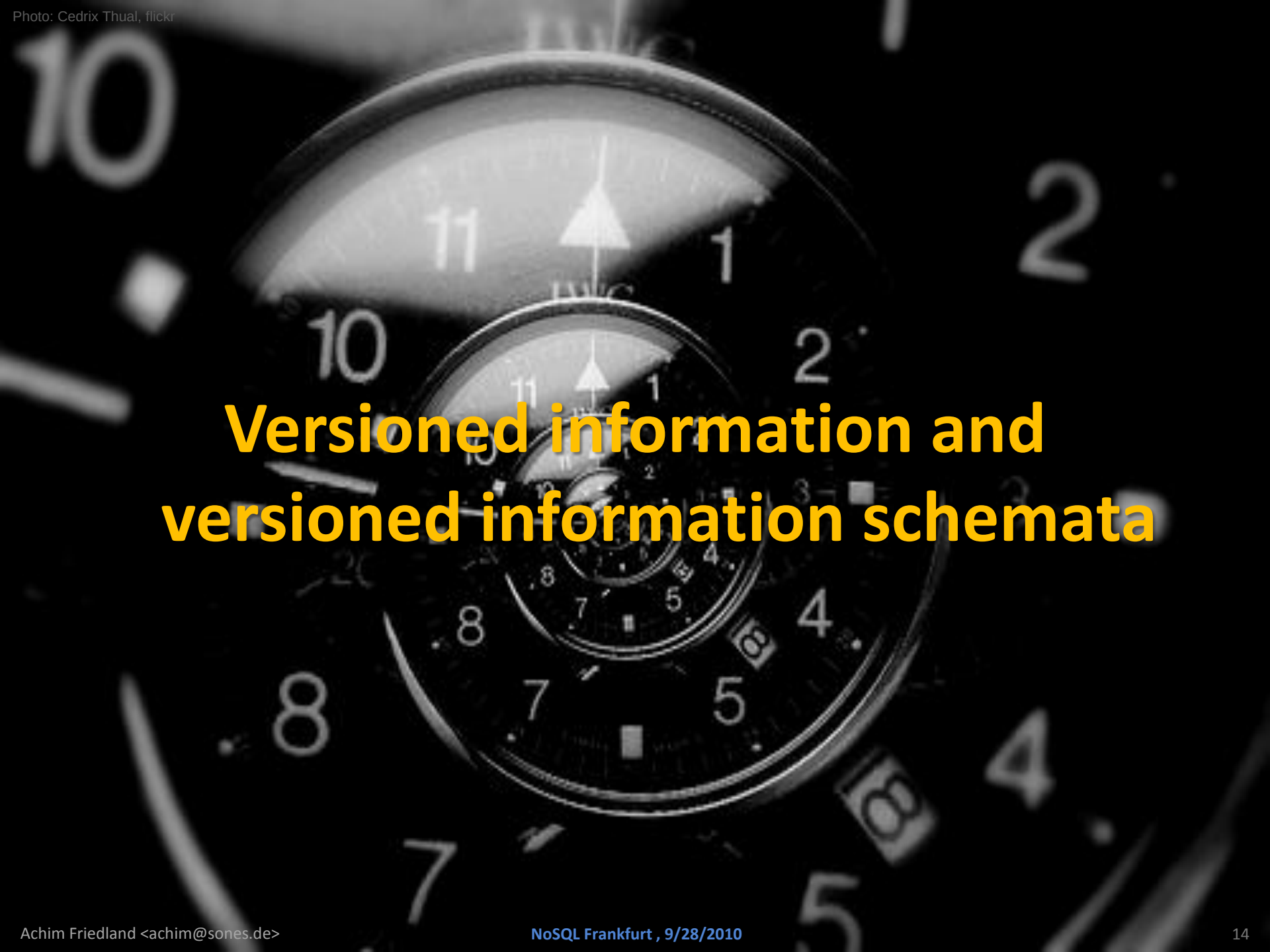


Photo: Cedrix Thual, flickr

# Versioned information and versioned information schemata

Photo: Birger Hoppe, flickr

# Good integration into state-of-the-art programming concepts (no Object-Relational-Mappers!)

Photo: Chuck “Caveman” Coker, flickr

# Can graph-databases help?

Photo: Jeffrey Beall, flickr

# Graph-Databases?

**Graph = ( Vertics, Edges )**

A graph database is a database that uses graph structures with nodes, edges, and properties to represent and store information. General graph databases that can store any graph are distinct from specialized graph databases such as triple stores and network databases.

http://en.wikipedia.org/wiki/Graph_database

Photo: Jeffrey Beall, flickr

# Graph-Databases?

**Graph = ( Vertics, Edges )**

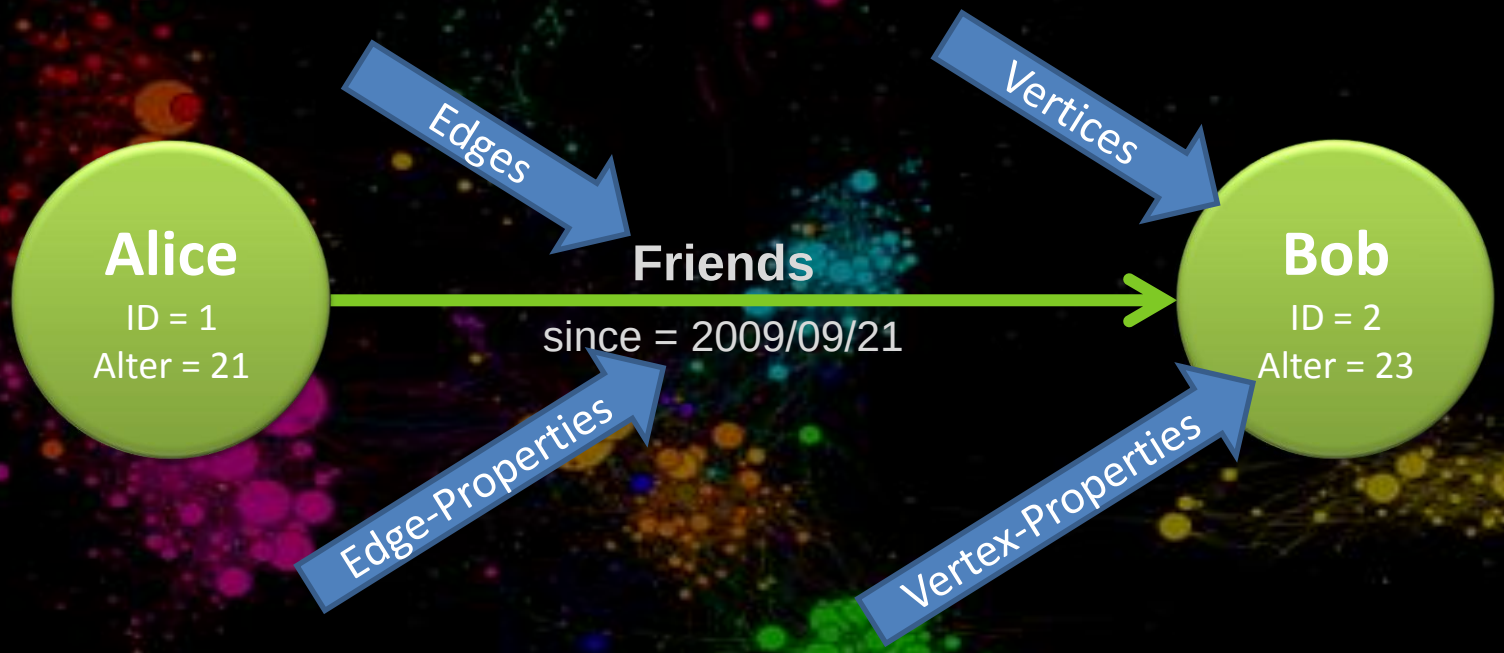
R. Angles, C. Gutierrez: Survey of graph database models, *ACM Computing Surv. 40, 1* (Feb. 2008), 1-39

Adrian Silvescu, Donia Caragea, Anna Atramentov: Graph Databases, Technical Report. May 2002

Mark Gemis, Jan Paredaens, Inge Thyssens, und Jan Van den Bussche: GOOD: A Graph-Oriented Object Database System, 1993; In: *Proceedings of the1993 ACM SIGMOD Intern. Conf. on Management of Data (Washington D.C., US, May 1993), 505-510.*

Photo: Gephi, flickr

# The Property-Graph

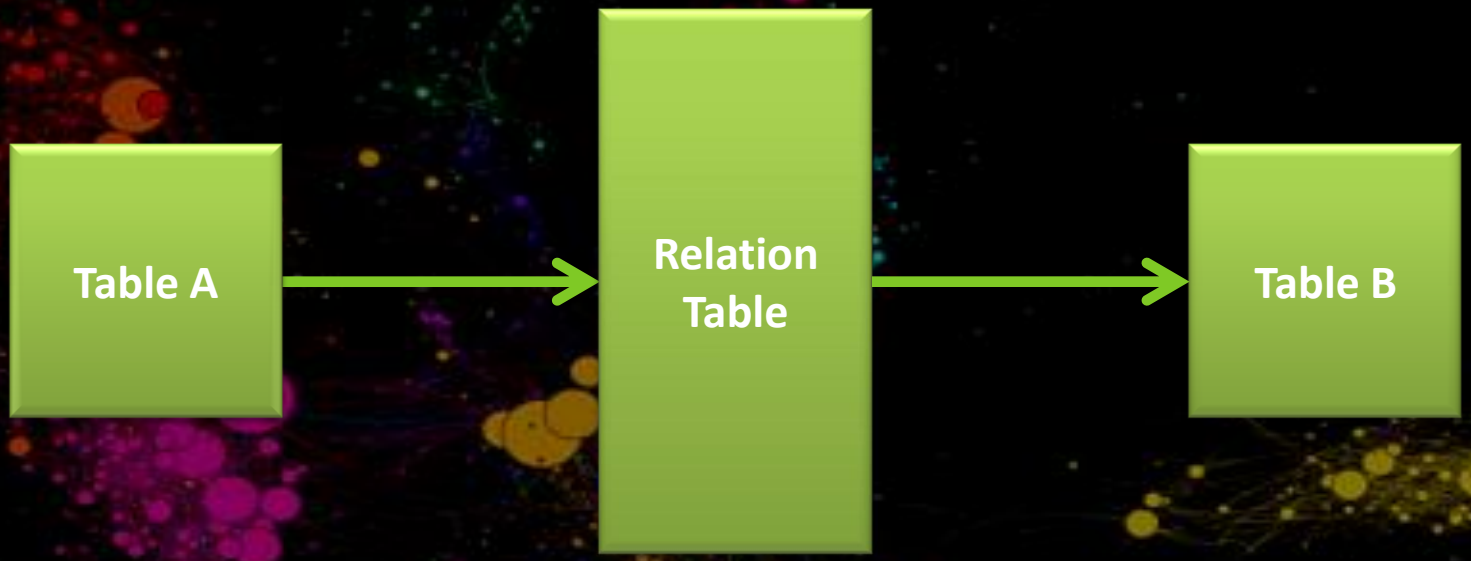



## Properties are key-values pairs

### e.g. <String, Object> or <AttributeKey, Object>

Photo: Gephi, flickr

# The Property-Graph
## vs. Relational Databases




**Focus on structured tables and their relational consistency constraints. Does not auto-scale.**

Photo: Gephi, flickr

# The Property-Graph
## vs. Object-Oriented Model


**Focus on structured or dynamic objects.**
**No (simple way for) edge properties.**

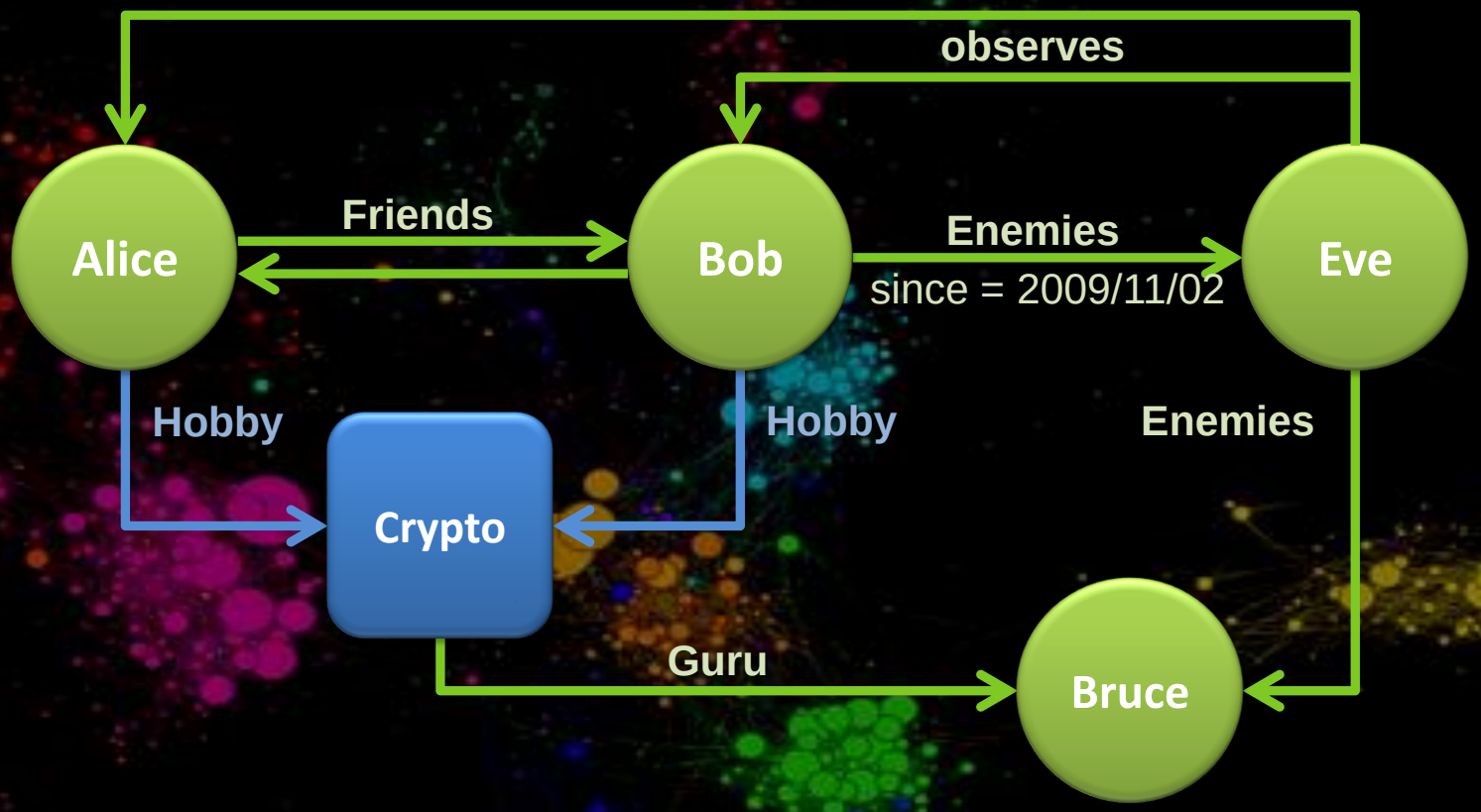
Photo: Gephi, flickr

# The Property-Graph
## vs. Semantic Web / RDF


**Focus on triples or quads.**
**No (simple way for) edge properties.**

Photo: Gephi, flickr

# The Property-Graph




**A directed, attributed, multi-relational graph as central data structure for graph-databases.**

Photo: Jeffrey Beall, flickr

# The Property-Graph

**Currently no academically founded or well-defined graph data model, but…**

The Property-Graph Model
*http://wiki.github.com/tinkerpop/blueprints/property-graphmodel*

Rodriguez, M.A., Neubauer, P., “Constructions from Dots and Lines,” Bulletin of the American Society for Information Science and Technology, American Society for Information Science and Technology, volume 36, August 2010.

…much more at http://markorodriguez.com ;)

# The Property-Graph

```
// Neo4J example

Node _alice = graphdb.createNode();
                    _alice.setProperty("Name", "Alice");
                    _alice.setProperty("Age", 21);

Node _bob = graphdb.createNode();
                    _alice.setProperty("Name", „Bob");
                    _alice.setProperty("Age", 23);

_alice.createRelationshipTo(_bob, Friends);
_bob.createRelationshipTo(_alice, Friends);
```

Photo: Gephi, flickr

# The Property-Graph

```
// sones gql example

CREATE VERTEX User
        ADD ATTRIBUTES (String Name, SET<User> Friends)
        INDICES (Name)

INSERT INTO User VALUES (Name = "Alice", Age = 21)
INSERT INTO User VALUES (Name = "Bob", Age = 23)

LINK User(Name = ‘Alice') VIA Friends TO User(Name = ‘Bob')
LINK User(Name = ‘Bob') VIA Friends TO User(Name = ‘Alice‘)
```

Photo: Gephi, flickr

# Benefits of Graph-Databases

- Explicit graph data model
  - No implicit model like within relational dbs
  - In the spirit of NoSQL:
    "One database for one data model"
  - GraphDB gets a change to understand the semantics and our intention

Photo: Gephi, flickr

# Benefits of Graph-Databases

- Index-free adjacency
  - No global adjacency index as each vertex acts as its own “mini-index”
  - The speed for traversing from one node to another is independed of the size of the graph ( Data locality and improved scalability )
  - Optimized for random access patterns

Photo: Gephi, flickr

# Graph-Database landscape

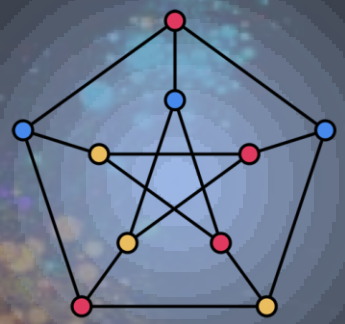
Neo4J

HyperGraphDB

InfoGrid

InfiniteGraph

Filament



OrientDB

DEX

VertexDB

sones

Marklogic?

Photo: Gephi, flickr

# Data model

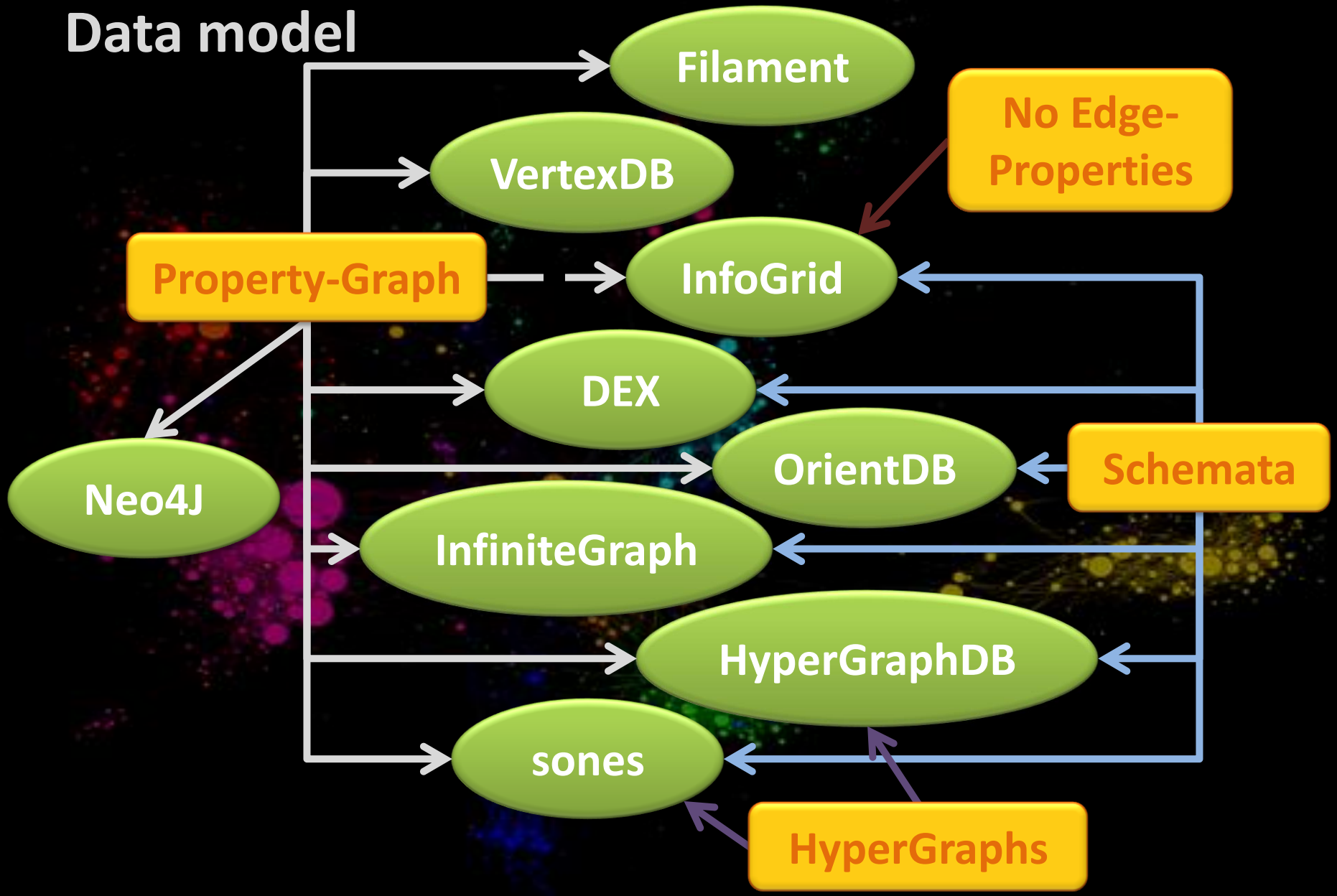


Photo: Gephi, flickr

# Query methods

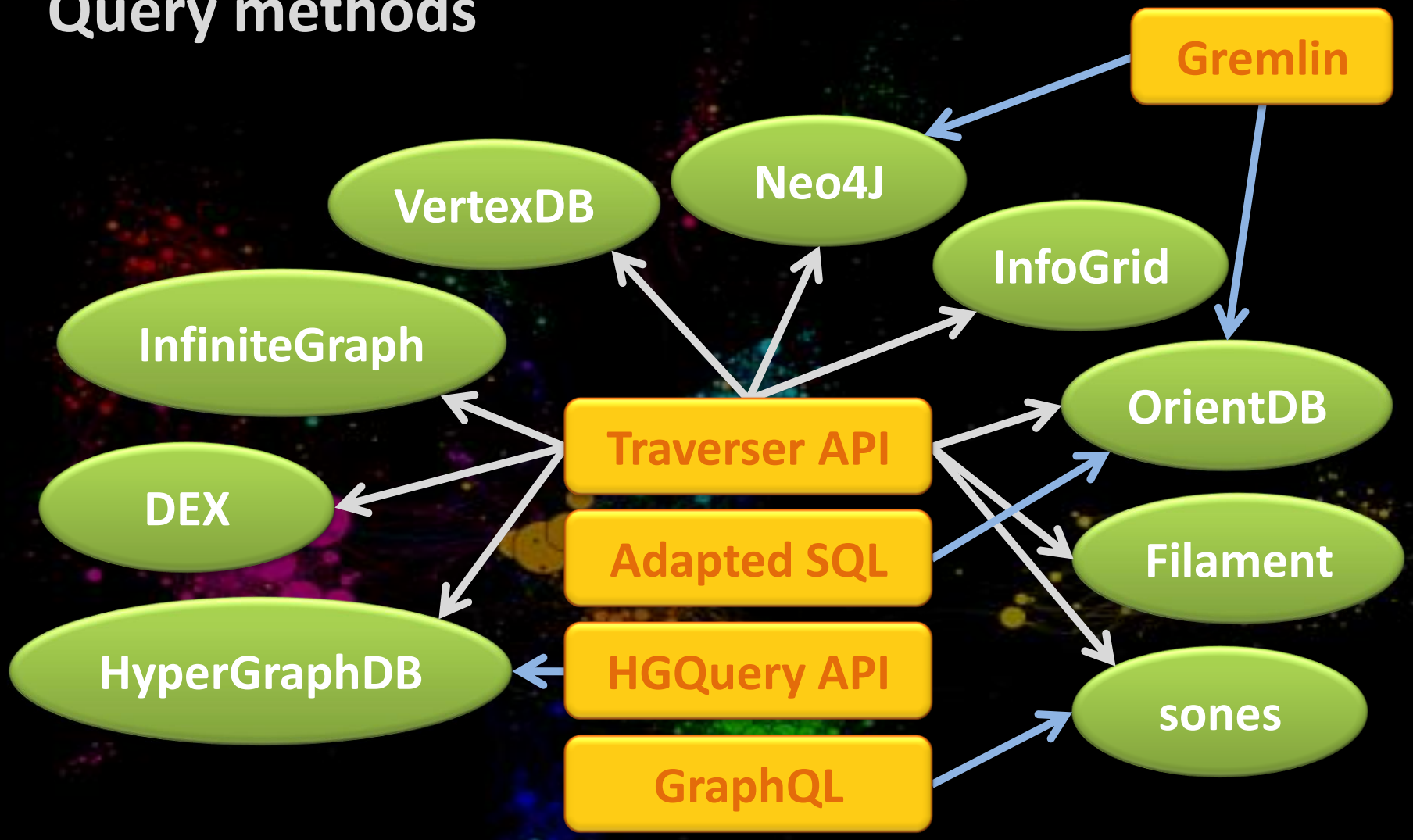


Photo: Gephi, flickr

# Programming languages

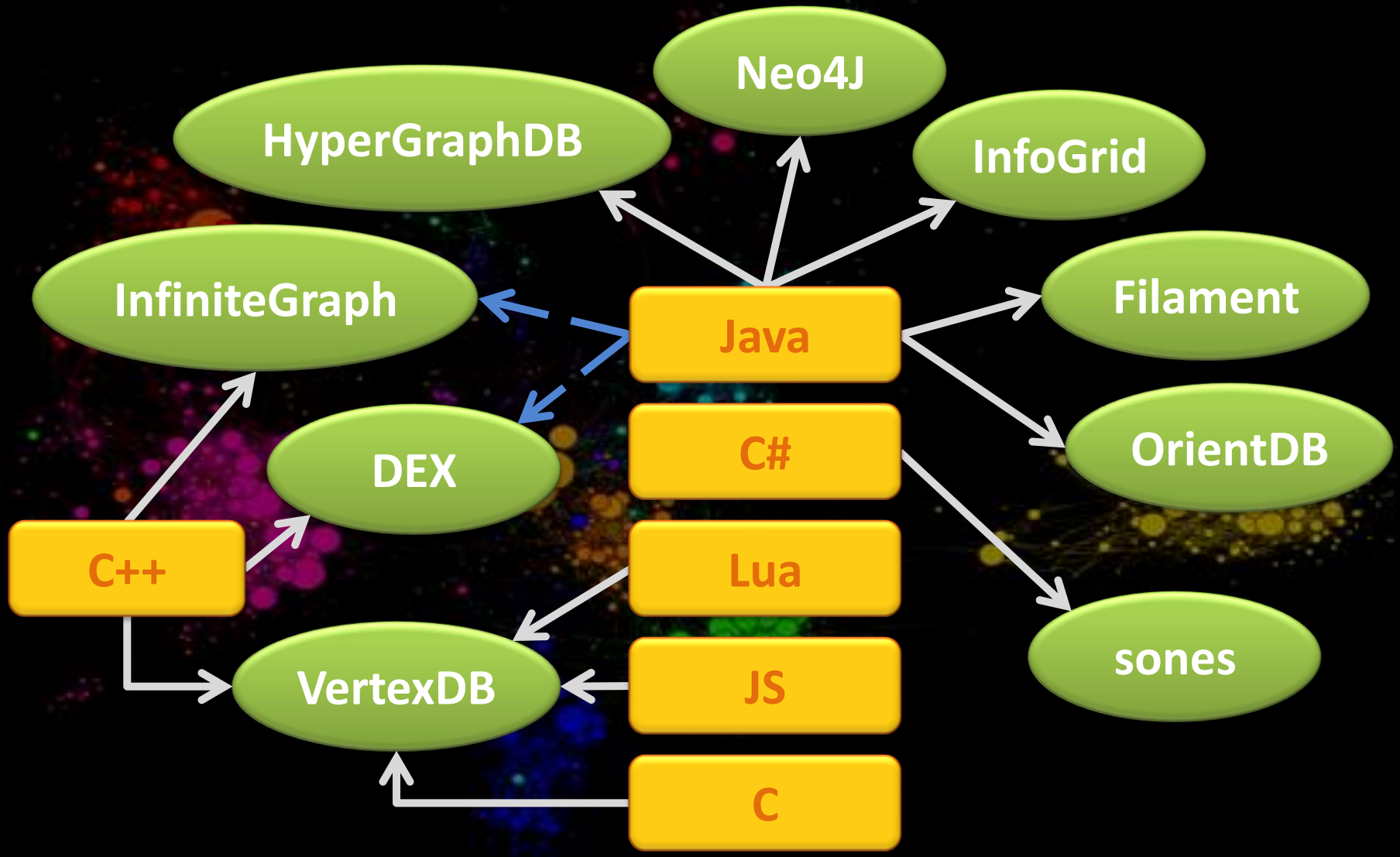


Photo: Gephi, flickr

# License models

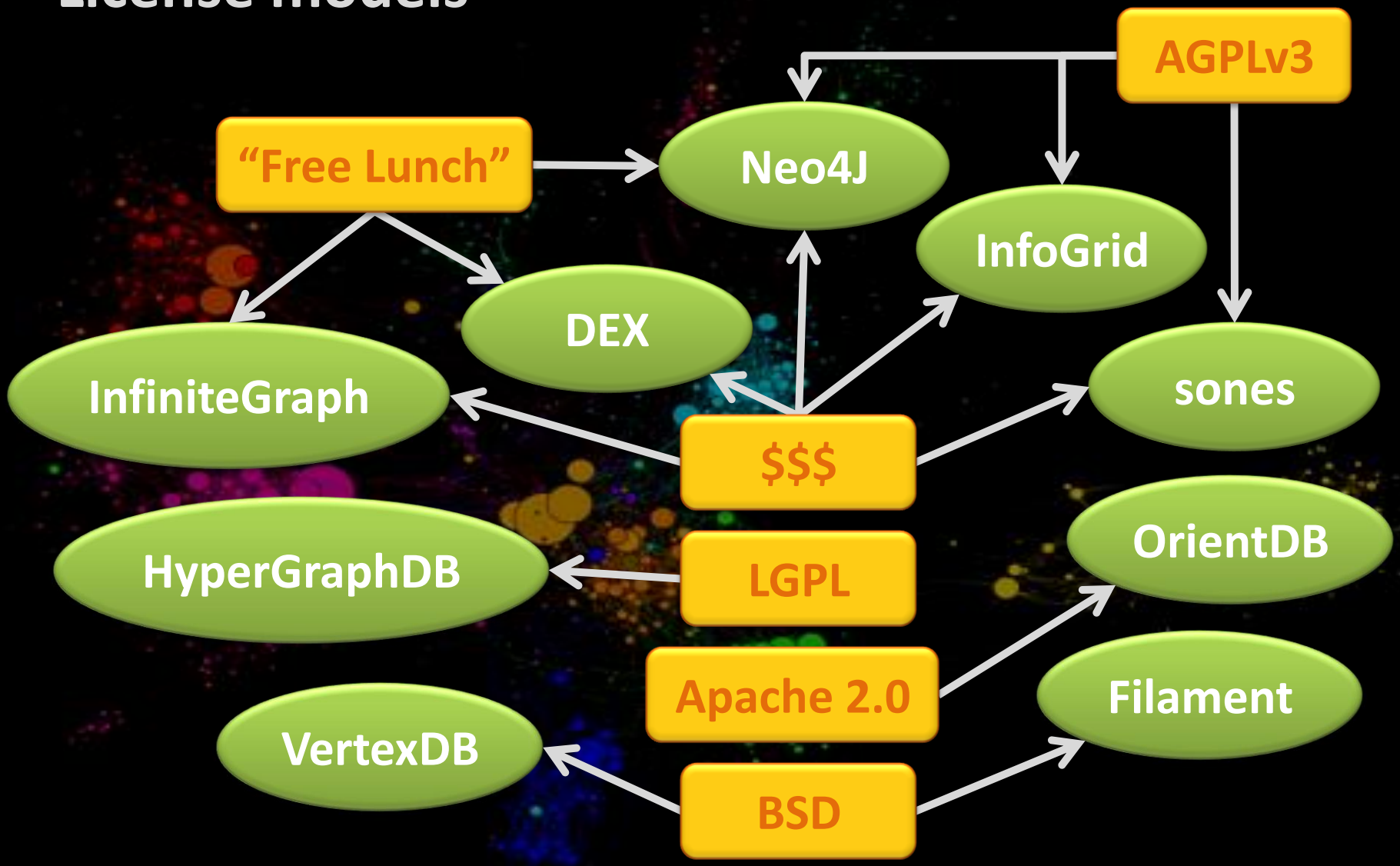


Photo: Gephi, flickr

# Protocols

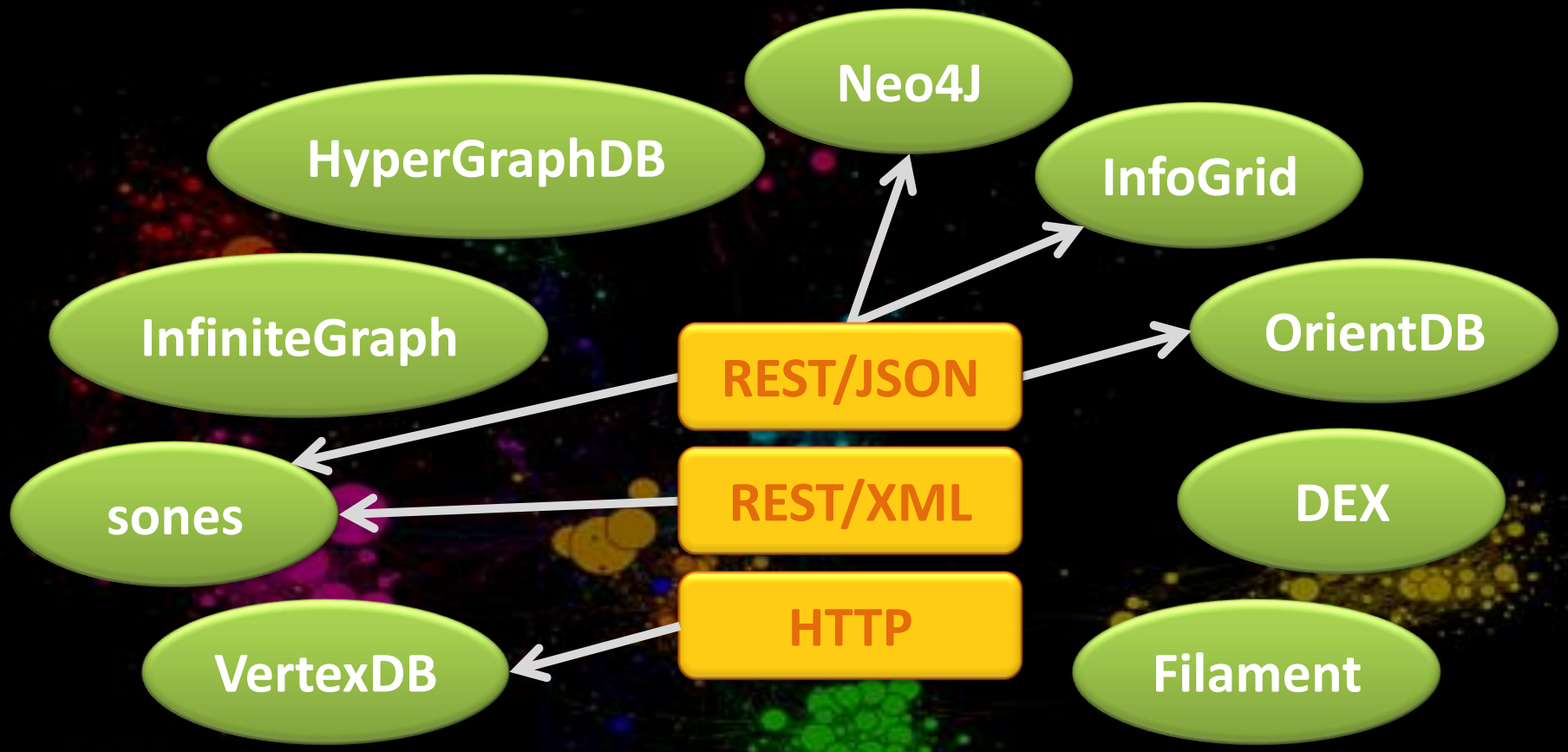


Photo: Gephi, flickr

# Neo4J

| | |
|---|---|
| URL | http://www.neo4j.org |
| Goals | Embedded and spatial DBMS |
| APIs | Java, JRuby, Ruby, Python, Scala, Clojure, C#, Erlang |
| Transactions | Concurrent reads, Sync on nodes |
| Repl./Scaling | Master-Slave, Master Failover |
| Persistency | Proprietary ondisk format |

# InfoGrid

- URL — http://www.infogrid.org
- Goals — Distributed GraphDB and Web-Application Framework
- Transactions — Yes
- Repl./Scaling — p2p-replication, partitioning
- Persistency — Proprietary, RDBMS, Hadoop

```
MeshBase _GDB = StoreMeshBase.create(_MySQLStore);
MeshObject _xkcd = _GDB.getMeshObjectLifecycleManager().createMeshObject();
_xkcd.setProperty("Name", "xkcd");
_xkcd.setProperty("Url", "http://www.xkcd.com");
_xkcd.relate(_good);
```

# DEX

- URL http://www.dama.upc.edu
- Goals High performance and integration of various information sources
- Transactions n/a
- Repl./Scaling n/a
- Persistency Proprietary ondisk format

```
long xkcd = _GDB.newNode(Resource);
_GDB.setAttribute(xkcd, Name, "xkcd");
_GDB.setAttribute(xkcd, Url, "http://xkcd.com");
int Tags = _GDB.newUndirectedEdgeType("Tags");
long e2 = _GDB.newEdge(xkcd, good, Tags);
```

# HyperGraphDB

- URL http://www.kobrix.com
- Goals AI, NLP, Semantic Web
- Transactions ACI(D), STM
- Repl./Scaling p2p
- Persistency Berkeley DB

```
// Note: Website(), Tag() are POJO
HGHandle _xkcd = _GDB.add(new Website("xkcd", "http://www.xkcd.com"));
HGHandle _good = _GDB.add(new Tag("good");
HGHandle _xkcd2good = _GDB.add(new HGPlainLink(_xkcd, _good));
```

# Infinite Graph

- URL — http://www.infinitegraph.com
- Goals — Very large graph databases
- Transactions — Distributed transactions
- Repl./Scaling — Objectivity/DB
- Persistency — Objectivity/DB

```
// Note: Website(), Tag() extend BaseVertex; Edge() extends BaseEdge
Website xkcd = new Website("xkcd", "http://xkcd.com/");  _GDB.addVertex(xkcd);
Tag good = new Tag("good"); _GDB.addVertex(good);
xkcd.addEdge (new Edge(), good, EdgeKind.BIDIRECTIONAL);
```

Photo: Gephi, flickr

# OrientDB

- URL http://www.orienttechnologies.com
- Goals Providing a KV-Store, DocumentDB and a GraphDB in one solution
- Transactions ACID, MVCC
- Repl./Scaling n/a
- Persistency Proprietary ondisk format

```
OGraphVertex _xkcd = _GDB.createVertex().
                     set("Name", "xkcd").
                     set("Url", "http://www.xkcd.com");
OGraphVertex _good = _GDB.createVertex().set("Name", "good");
_xkcd.link (_good);
```

Photo: Gephi, flickr

# sones

- URL — http://www.sones.de
- Goals — Management of linked (binary) data
- Transactions — ACID, MVCC (alpha)
- Repl./Scaling — p2p (alpha)
- Persistency — Proprietary file system

Photo: Shayne Kaye, flickr

# sones Property-Hypergraph

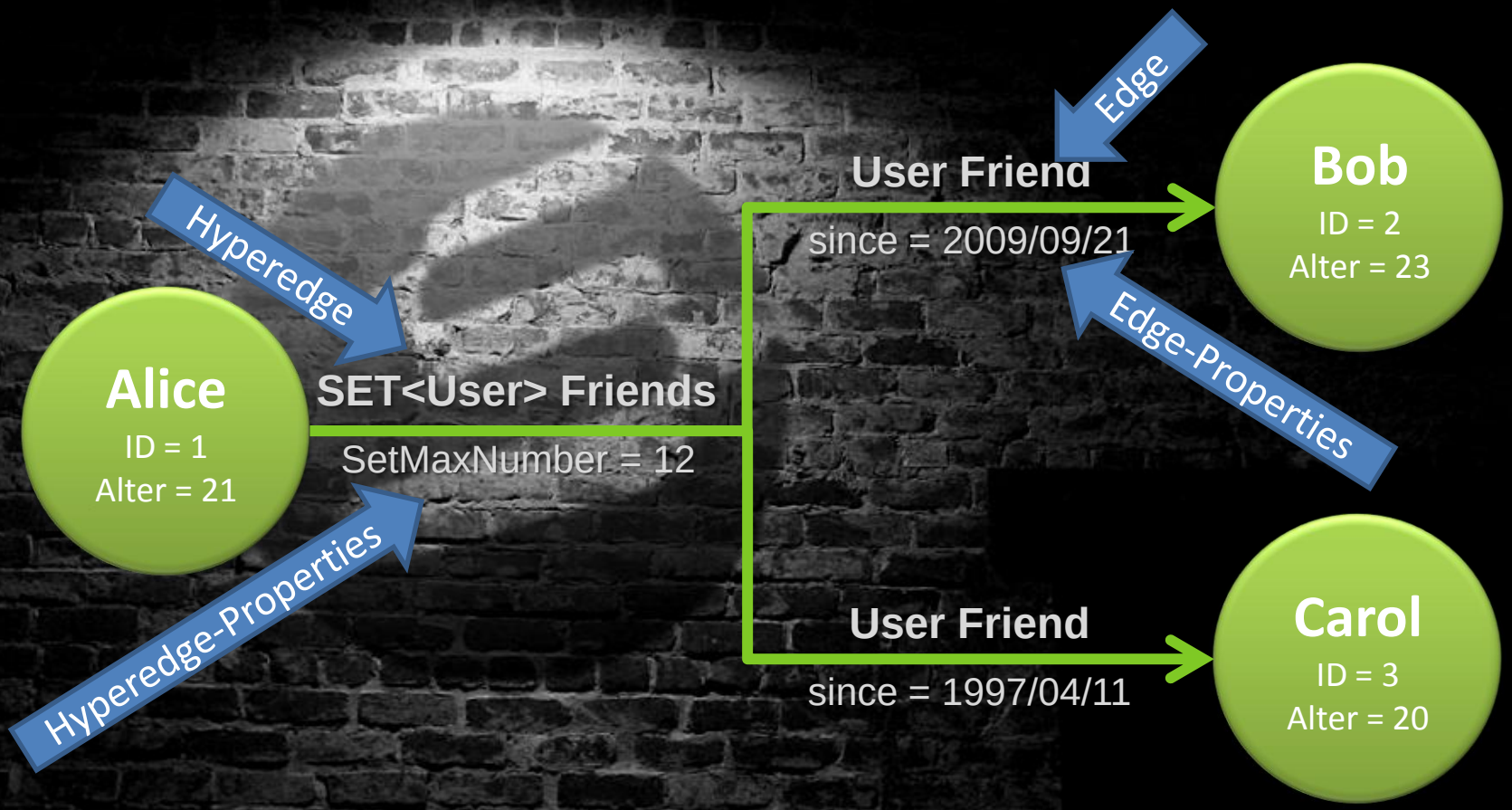
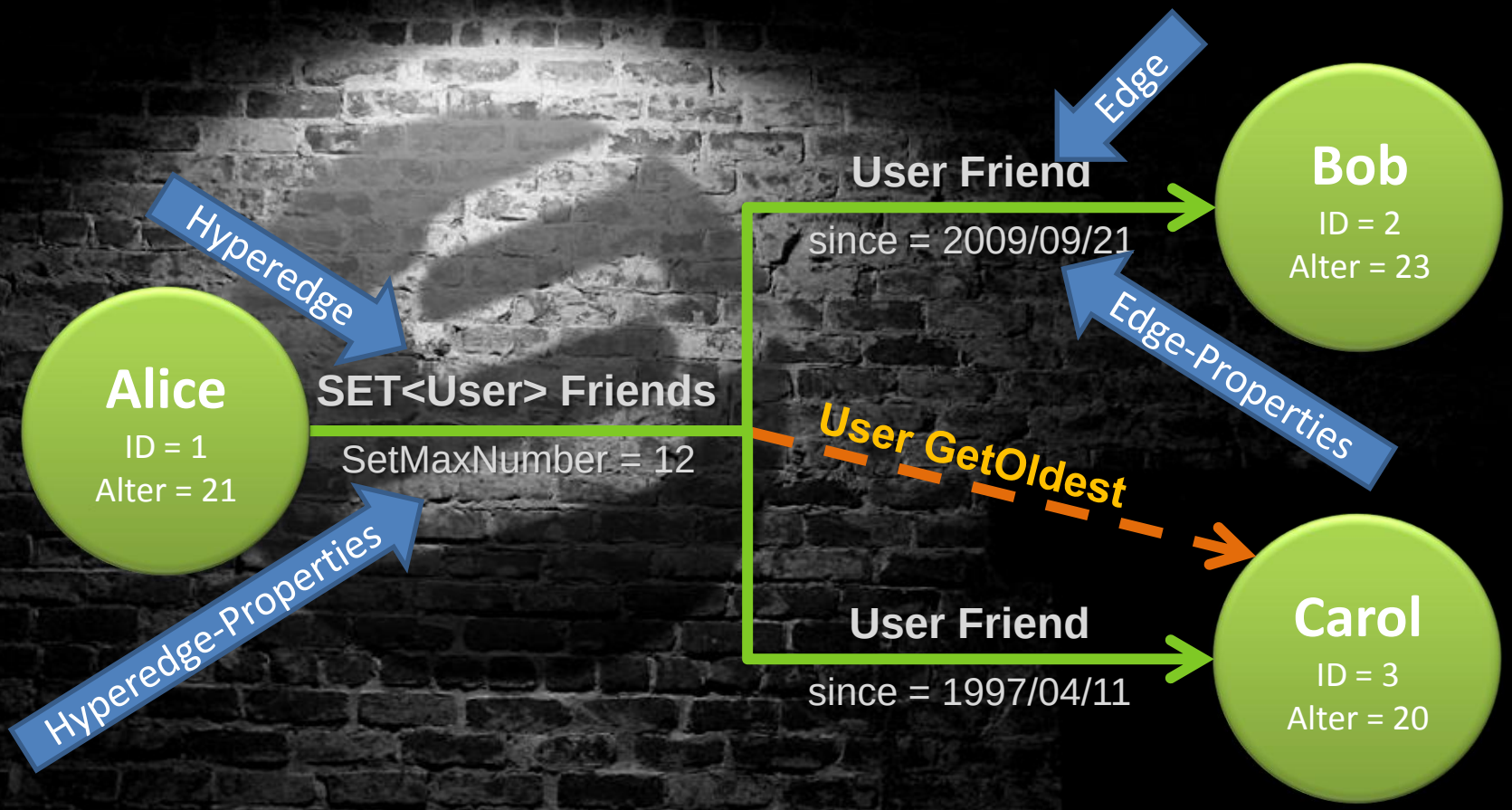


Photo: Shayne Kaye, flickr

# sones Property-Hypergraph

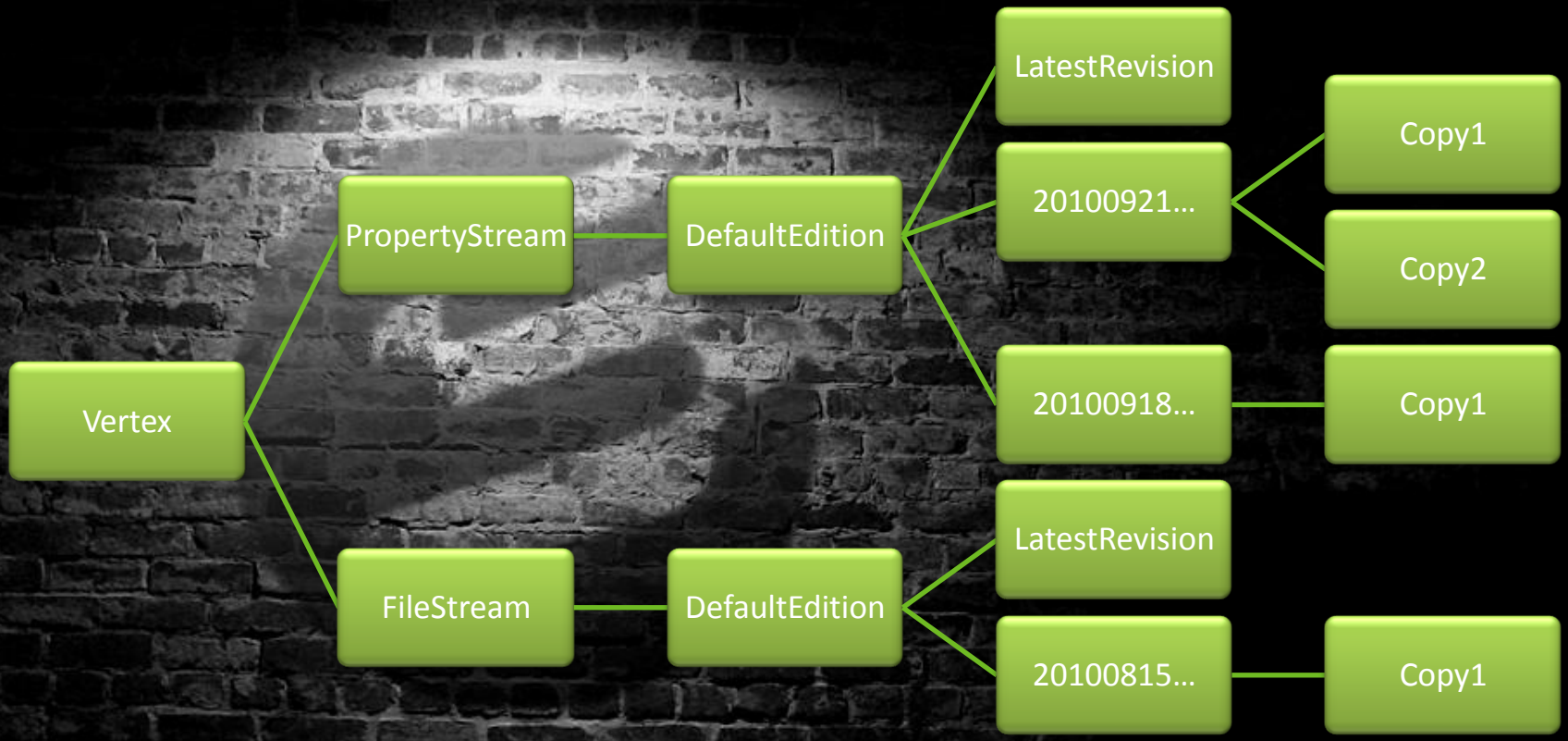


# sones Property-Hypergraph

- Extensions to the Property Graph model
- Multiple edges are be grouped by a hyperedge
- Hyperedge may have their own properties
- Properties may include code as data
  (e.g. C#: Func<…>, ExpressionTrees)
- Allows calculations be done among the set of edges (GetMinWeight, SetMaxNumber, …)

Photo: Shayne Kaye, flickr

# sones Special Edges and Properties



# sones Graph Query Language

```
FROM User SELECT User.Friends.Friends.Name
```

- "SQL for graphs"
- Its goal is to provide a very user-friendly language for ad-hoc graph queries
- Functions and aggregates are type-safe and can be extended by your own plug-ins, e.g.
  - `SELECT COUNT(User.Friends)`
  - `SELECT User.Friends.Random(2)`
  - `SELECT User.Friends.Name.Substring(2,5)`

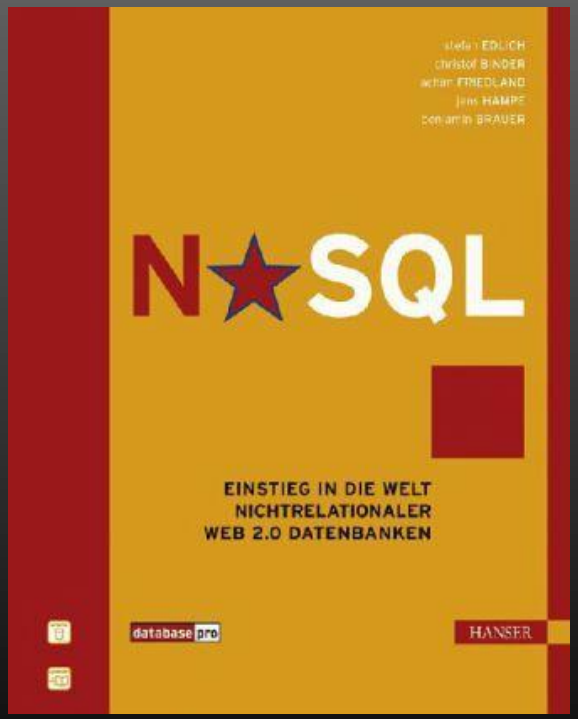
Photo: jonathanpercy, flickr

# Gremlin

```
./outE[@label='created']/inV/inE[@label=
        'created']/outV[g:except($_)]/@name
```

- Get outgoing edges, where their label is equal to 'created'
- Get their incoming vertices
- Get incoming edges, where their labels equals 'created'
- Get outgoing vertices, except the starting vertex
- Get their name property

# For more information…




**achim@sones.de**
**http://www.twitter.com/ahzf**
**http://www.twitter.com/graphdbs**