The GraphDB Landscape and sones
• 43 years Object Orientation / OOP
• 35 years Relational Databases / SQL
• 21 years Semantic Web / RDF
• 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!
Our wish list...
Simple management of structured, semi-structured and unstructured information
Simple management of structured, semi-structured and unstructured information

SQL

XML/JSON

binaries
Simple recursively linked information models
Simple recursively linked information models

- `<a href = “…”>`
- metadata e.g. EXIF
Traversing linked information, finding shortest-paths, do semantic partitions
Recommendation and discovery of potentially interesting linked information
Recommendation and discovery of potentially interesting linked information
Consistency criteria and indices for simple attributes up to complex subgraph structures
Versioned information and versioned information schemata
Good integration into state-of-the-art programming concepts (no Object-Relational-Mappers!)
Can graph-databases help?
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
Graph-Databases?

Graph = (Vertics, Edges)


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

The Property-Graph

Properties are key-values pairs
e.g. <String, Object> or <AttributeKey, Object>
The Property-Graph vs. Relational Databases

Focus on structured tables and their relational consistency constraints. Does not auto-scale.
The Property-Graph vs. Object-Oriented Model

Focus on structured or dynamic objects.
No (simple way for) edge properties.
The Property-Graph vs. Semantic Web / RDF

Focus on triples or quads.
No (simple way for) edge properties.
The Property-Graph

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


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

Achim Friedland <achim@sones.de>
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);
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')
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
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 independent of the size of the graph (Data locality and improved scalability)
  – Optimized for random access patterns
Graph-Database landscape

- Neo4J
- HyperGraphDB
- InfiniteGraph
- DEX
- VertexDB
- InfoGrid
- Filament
- OrientDB
- sones
- Marklogic?
Query methods

- VertexDB
- Neo4J
- InfoGrid
- OrientDB
- Filament
- sones
- InfiniteGraph
- DEX
- HyperGraphDB
- Gremlin
- Traverser API
- Adapted SQL
- HGQuery API
- GraphQL
Programming languages

HyperGraphDB

InfiniteGraph

DEX

C++

VertexDB

C#

Java

JS

Lua

C

Neo4J

InfoGrid

Filament

OrientDB

sones
License models

Neo4J

DEX

InfiniteGraph

HyperGraphDB

VertexDB

“Free Lunch”

InfoGrid

sones

Filament

OrientDB

AGPLv3

$$$

LGPL

Apache 2.0

BSD

Apache 2.0

BSD

AGPLv3

“Free Lunch”
Protocols

HyperGraphDB

InfiniteGraph

sones

VertexDB

Neo4J

InfoGrid

OrientDB

DEX

Filament

REST/JSON

REST/XML

HTTP
# 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](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

```java
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
GHHandle _xkcd = _GDB.add(new Website("xkcd", "http://www.xkcd.com");
GHHandle _good = _GDB.add(new Tag("good");
GHHandle _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);
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

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

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

Alice
ID = 1
Alter = 21

SET<User> Friends
SetMaxNumber = 12

User Friend
since = 2009/09/21

Bob
ID = 2
Alter = 23

User Friend
since = 1997/04/11

Carol
ID = 3
Alter = 20

Photo: Shayne Kaye, flickr
sones Property-Hypergraph

- **Alice**
  - ID = 1
  - Alter = 21

- **Carol**
  - ID = 3
  - Alter = 20

- **Bob**
  - ID = 2
  - Alter = 23

**SET<User> Friends**
- **SetMaxNumber** = 12

**Hyperedge:**
- Between Alice and Carol

**User Friend**
- **since** = 2009/09/21
- **since** = 1997/04/11

**User GetOldest**
- Edge

**Edge-Properties**
- Connections between users

Photo: Shayne Kaye, flickr
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

- Vertex
  - PropertyStream
    - DefaultEdition
      - LatestRevision
        - 20100921
          - Copy1
          - Copy2
      - LatestRevision
        - 20100918
          - Copy1
    - DefaultEdition
      - LatestRevision
        - 20100815
          - Copy1

Achim Friedland <achim@sones.de>  
NoSQL Frankfurt, 9/28/2010
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)
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