Database Implementation

HypergraphDB

17.03.2015

Presentation outline

- Introduction -> DONE
- Use case + Usage -> Janosch
- Example -> Frank
- Technological Background (What is behind HDB/What is a Hypergraph?) -> Roland (DONE)
- Difference between HDB and Neo4J -> Tilman
- Conclusion -> All

Introduction

- General purpose & open-source
- Backed by BerkeleyDB
- Designed for knowledge management, AI and semantic web
- Can also be used as an embedded
  - Object-oriented database
  - Graph database
  - (non-SQL) relational database

Key features

- Allows edges to point to other edges and makes every node or edge carry an arbitrary value as payload. (E + N = Atom)
- Platform independent storage scheme accessible by any platform and language
- No software size limits
- Automatic mapping of POJO's
- Embedded in-process

Use cases

- Semantic web
- Bioinformatics
- Desktop application configuration storage
- Server-side Java applications
  - ➡ move to object-oriented DBs

Create DB

```
HyperGraph graph = new HyperGraph("/path/");
```

- Easy to use
- No management of other databases
- `HGEnvironment` class for more management

Storing / loading (fast)

```
graph.add(Object)
graph.get(HGHandle)
```

- No check for duplicates
- Stores any object, returns Handle for direct access
- Custom objects need to meet Java Beans convention

Querying

- Query package provides conditional expressions

```
hg.getOne(HyperGraph, HGQueryCondition)

hg.getAll(HyperGraph, HGQueryCondition)
```

➡ Returns list of normal Java Objects

```
hg.findAll(HyperGraph, HGQueryCondition)
```

➡ Returns list of handles

Querying (conditions)

- Classes for:
  - Logical expressions
  - Type matching
  - Regex string matching
  - Value matching
  - and more

Querying (conditions)

```
new And(
   new AtomTypeCondition(Book.class),
   new AtomPartCondition(
      new String[]{"author"},
      "George Bush",
      ComparisonOperator.EQ
    )
);
```

What else?

- A lot!
    - Links/relations (to make it a real hypergraph)
    - Indexing
    - Transactions
    - Caching
    - P2P framework for distributed processing

Live demo

https://github.com/steilerDev/HypergraphDBProject

HypergraphDB Model

- atom: has value, target set, incidence set and value
  - atom with |target set| > 0: link
  - atom with |target set| = 0: node
- value: typed data
- type: atom
- Definition of hypergraph structure by atoms
- No influence on structure by values and types

HypergraphDB Model

- 2-Layer Architecture

- Primitive storage layer
  - LinkStore: ID -> List < ID >
  - DataStore: ID -> List < byte >

- Model layer
  - AtomID -> [TypeID; ValueID; TargetID; ...; TargetID]
  - TypeID -> AtomID
  - TargetID -> AtomID
  - ValueID -> List < ID > | List < byte >

Typing

- Types are useful:
    - constraints for DB integrity and consistency
    - define data semantics
- Types are atoms:
    - construction of new types at runtime
    - domain model part of data model
- Predefined types

Differences between HypergraphDB and Neo4j

- Storage
- Query language
- License
- Data types
- Integrity model
- Graph model
- Similarites

Differences

- **Storage**
    - HypergraphDB:
        - Only in volatile memory (JDOs)
            - Can be serialised to disk
    - Neo4j in memory and on disk
- **Query language**
    - HypergraphDB: hgdbquery-api
    - Neo4j: API calls, REST, many more

Differences - License

- HypergraphDB:
  - LGPL (embeddable in non-GPL applications)
- Neo4j:
  - GPL, AGPL (community edition) or commercial license
    - ➡ own application has to be (A)GPL, or one needs a commercial license
    - ➡ reduced functionality compared to commercial version

Differences

- **Datatypes**
  - HypergraphDB: POJO's
  - Neo4j: (Array of) Java primitives, Strings
- **Integrity model**
  - HypergraphDB: MVCC (Multiversion concurrency control)
    - ➡ lock free, snapshots, gc
  - Neo4j: ACID, Log replication

Differences - Graph model

- HypergraphDB
  - Hypergraph with 'n-ary hyperedges'
    - Hyperedge: Connect n nodes to m nodes, n,m >=0
    - ➡ Ability to have edges from and to other edges
- Neo4j
  - Property Graph (directed, non- hypergraph)

Similarities

- Graph-oriented storage (as name suggests)
- Embeddable
- Allow transactions

Conclusion