
# SQL OR NOSQL?

## (WHAT AM I GIVING UP/GAINING?)

Andy Miller / amiller@objectpartners.com

# HOW DID THIS START?

- After hearing EMC state *"We recommend mongoDB for all new development."*

- And after using Mongo on a couple client projects

- I wanted to see if it would work for MTM.
  - Was this app best suited for SQL?
  - for noSQL?
  - or a combination?
- Exporting projects as json seemed like a good match for mongo, so why not try it and see what's up...

# QUESTIONS

1. Is mongo a danger to you and your data?
2. Is mongo a replacement for mysql? (or...)
3. Is noSQL (mongo) better than sql?
4. Should all new projects start with mongo?

# NOSQL DEFINED

- ...modern web-scale databases
- ...schema-free
- ...easy replication support
- ...simple API
- ...eventually consistent / BASE (not ACID)
  - **B**asically **A**vailable **S**oft state **E**ventual consistency
  - **A**tomicity **C**onsistency **I**solation **D**urability

# SQL ≡ RELATIONAL DATABASE

- ...a database that has a collection of formally described tables and organized according to the relational model
- ...each table identifies a primary key
- ...tables can relate by using foreign keys
- ...ACID

# NOSQL DATABASE CATEGORIES...

from nosql-database.org

- Wide Column Store / Column Families
- Document Store
- Key Value / Tuple Store
- Graph Databases
- Multimodel Databases
- Object Databases
- Grid & Cloud Database Solutions
- XML Databases
- Multidimensional Databases
- Multivalue Databases
- Event Sourcing
- "Other"
- "unresolved" and "uncategorized"

# EXPERIENCE WITH

- Wide Column Store / Column Families
- Document Store — **mongo**
- Key Value / Tuple Store — **redis, S3**
- Graph Databases
- Multimodel Databases
- Object Databases — **objectstore, poet, versant**
- Grid & Cloud Database Solutions
- XML Databases
- Multidimensional Databases
- Multivalue Databases
- Event Sourcing
- "Other"
- "unresolved" and "uncategorized"

# MONGODB.ORG

- open-source document database
- JSON-style documents with dynamic schemas
- full index support
- replication & high availability
- auto-sharding
- document-based queries
- atomic modifiers
- map/reduce
- GridFS file storage
- professional support (10gen.com)

# MTM

github.com/onetribeyoyo/mtm

*(live)*

# CONVERTING THE APP

- BuildConfig.groovy
- converting the datasource
- try it...
- It works. I must have missed something?

# RUN THE TESTS

*(live)*

...I must have missed something

# WHAT'S SO AWESOME ABOUT GRAILS/GORM?

- 80/20 rule for queries
- Trivial schema generation
- 80/20 rule for schema evolution
  - 80% from GORM
  - 20% liquibase (80/20 of that 20%!)
- RDBMS vendor independence
  - I'm happy to make a recommendation, but...
  - ...let's just leave this in the hands of our client's DBAs

# GRAILS MONGODB PLUGIN

grails.org/plugin/mongodb

- Story.list()
- Story.get(42), Story.getAll(5, 6, 7)
- Story.list(max: 5, sort: "first", order: "desc"), Story.listOrderBySpecialty()
- Story.findByEstimateGreaterThan(8)
- ...
- criteria queries
- projections
- query by example

# GRAILS MONGODB PLUGIN

grails.org/plugin/mongodb

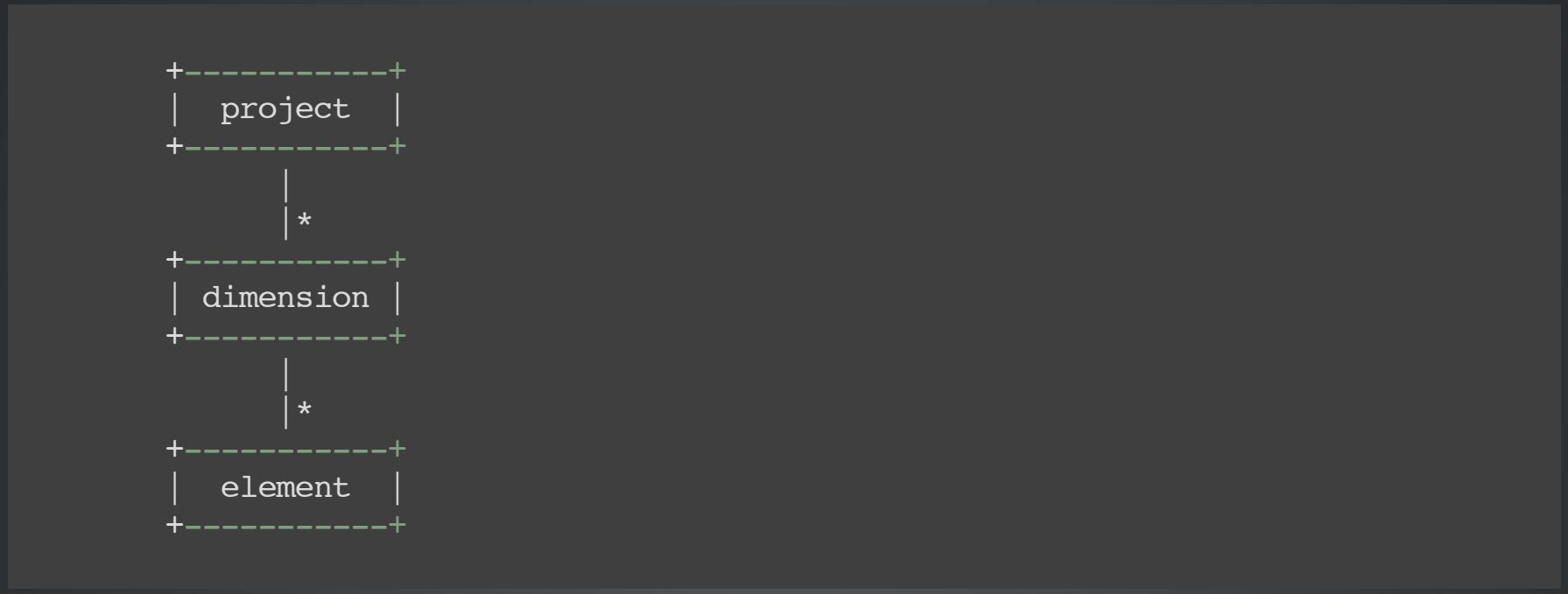
Not supported...

- Criteria queries on associations
- HQL
- Groovy SQL

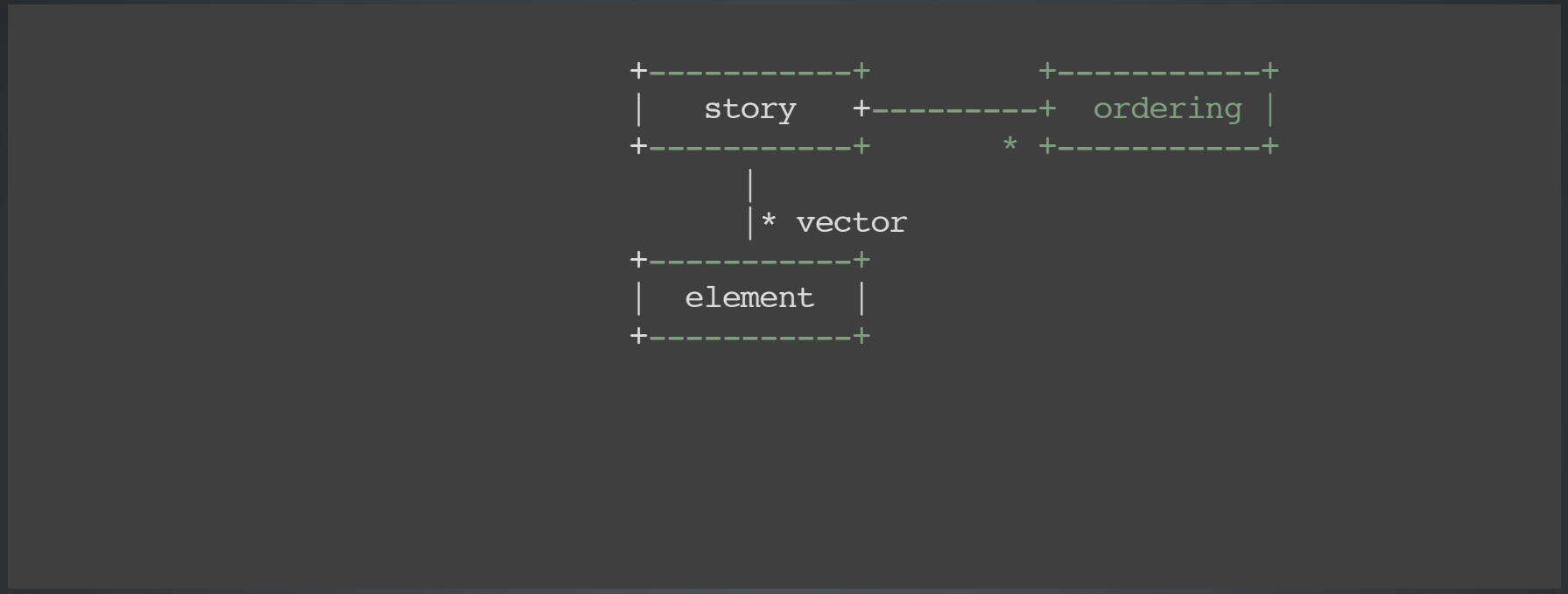
The big one...

TRANSACTIONS!

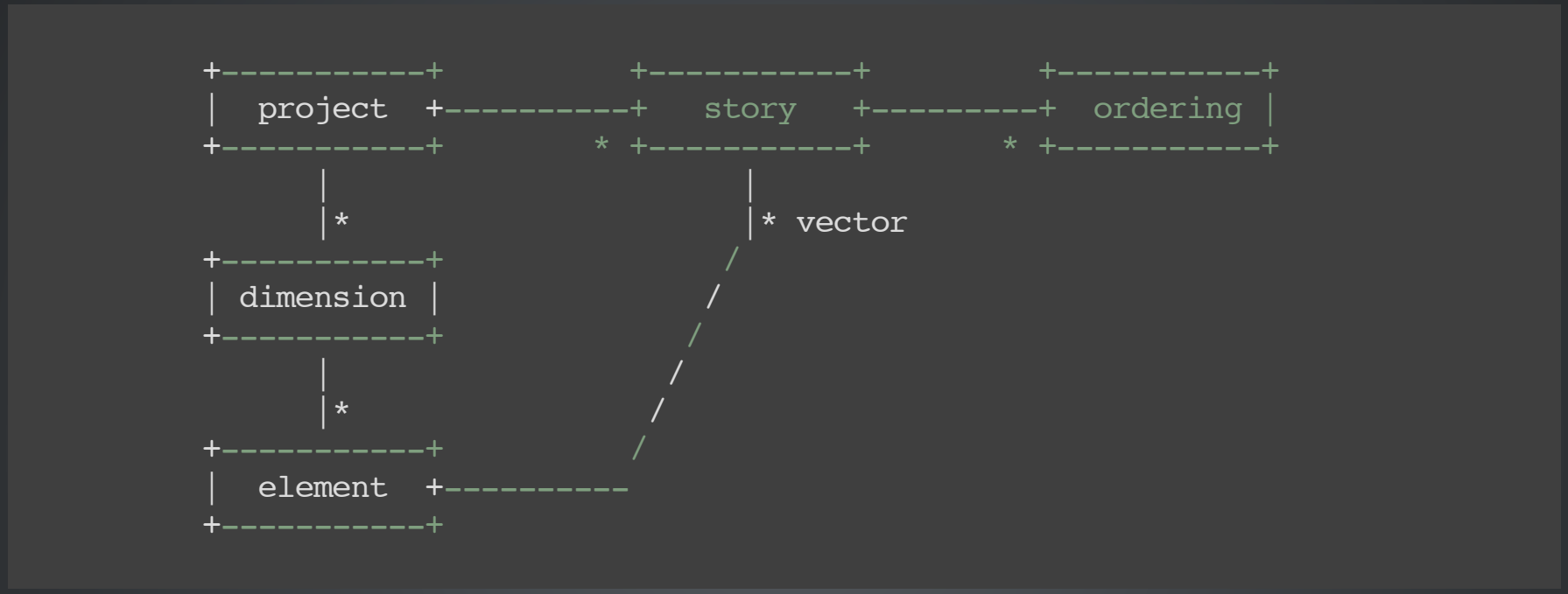
# MTM DOMAIN MODEL



# MTM DOMAIN MODEL



# MTM DOMAIN MODEL



# TESTING WITHOUT TRANSACTIONS

- `MongoUtils`
- `MongoISpec` classes
- *(live)*

# WHAT ARE WE MISSING?

- transactions
- in memory mongo
- ID:Long vs ID:String
- `beforeInsert()`
- ...?
  jira.grails.org/browse/GPMONGODB
- familiar query tools

# QUERY TOOLS

- mongohub
- *(live)*
- rockmongo

# ID:LONG VS ID:STRING

- RDBMS use sequences to generate ID values
  dimension.id = 1, dimension.id = 2, and so on...
- MongoDB can use *"sequences"* to generate ID values
  - collection for domain object
  - ...plus collection for next_id
  - e.g. dimension + dimension.next_id
- OR MongoDB can use UUIDs as ID values
  - dimension.id = "51ec81f7744e7c5811f6cfc8"
  - cleaner, but numeric ID's might be easier to integrate with legacy systems?

# ARE THE ADVANTAGES REAL?

- simplified deployment (mongo is easier to install than mysql)
- master/slave is trivial to setup and configure
- trivial backup and recovery (collecitons are files, just copy them to/from s3)
- easy sharding (yes, I know it can be done with mysql but lets try this on your dev box...)
- don't worry about mapping fields using sql reserved words (e.g. "order")
- embedded collections saved in mongo preserve order. why? 'cause a json "list" is written in and then read back out in the same order.
- ...?

# DISADVANTAGES?

- schema evolution
  - no liquibase, but there is mongeez
  - grails.org/plugin/mongeez
- embedded objects are not domain objects (they don't have constraints, no addToXyzzy, ...
- embedded objects don't have ID's
- schemaless mongoDB doesn't implement field data types
- schemaless mongoDB doesn't implement field constraints (nullable, size limits, etc.) Because of this the data is not "portable" without the executble application.

# SCHEMA EVOLUTION VS. SELF-HEALING DATA

- self-healing domain objects ensure default values can be determined or derived from existing data

or

- evolve the data in place with **github.com/secondmarket/mongeez**

# SCHEMA EVOLUTION WITH DYNAMIC ATTRIBUTES

```
def foo = new Story(summary: "xyzzy", ...)

// add fields dynamically...
foo["eta"] = new Date() + 3
foo.save()
```

# CONCLUSIONS

1. Is mongo a danger to you and your data?
   No.

2. Is mongo a replacement for mysql?
   Yes.

3. Is noSQL (mongo) better than sql?
   Well... it depends.
4. Should all new projects start with mongo?
   No.

5. But how are we gonna sell this to the *data stewards*?
   *object database deja vu!*

http://nosql-database.org/

http://www.mongodb.org/

http://grails.org/plugin/mongodb

http://www.10gen.com/

https://github.com/onetribeyoyo/sql-or-nosql

https://github.com/onetribeyoyo/mtm

https://github.com/secondmarket/mongeez

- - - -

Andy Miller / amiller@objectpartners.com