Making MongoDB Accessible to All

Brody Messmer
Product Owner
DataDirect On-Premise Drivers
Progress Software

PROGRESS EXCHANGE 2014

Agenda

- Intro to MongoDB
  - What is MongoDB?
  - Benefits
  - Challenges and Common Criticisms
  - Schema Design Comparison
- ODBC/JDBC & SQL access to MongoDB
  - Industry Approach
  - DataDirect's Solution
  - Demo

Introduction to MongoDB

PROGRESS EXCHANGE 2014

What is mongoDB?

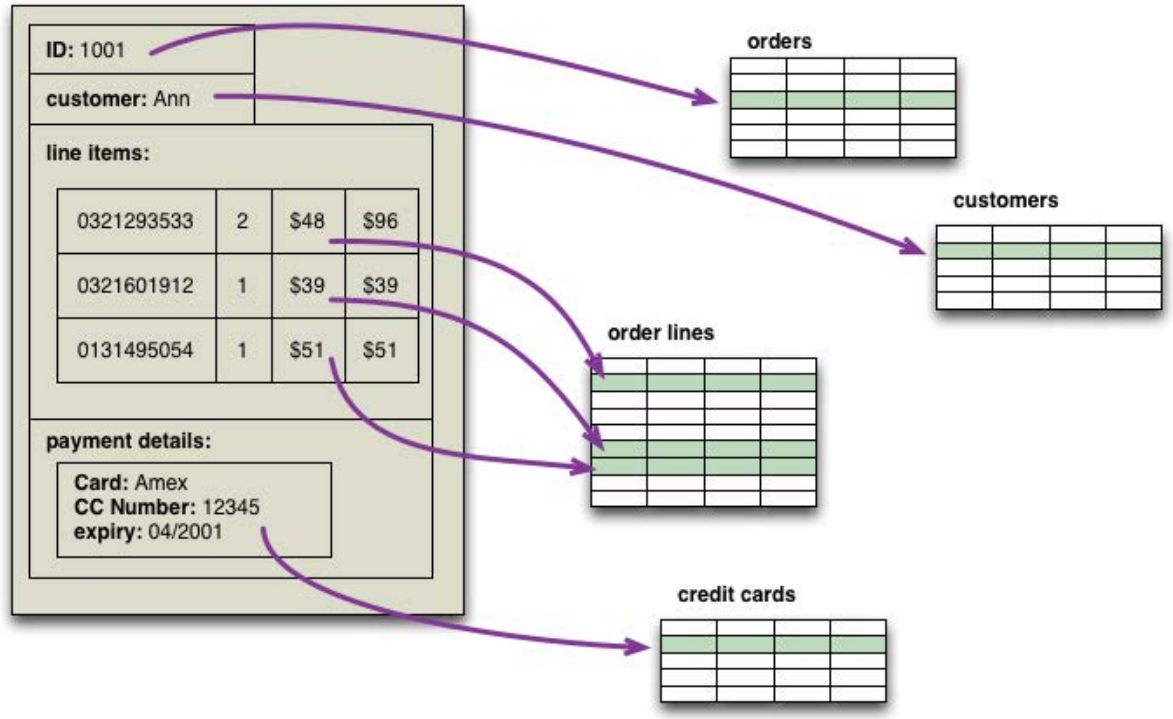
MongoDB: an OpenSource, NoSQL Document (JSON) Database

Sample JSON Document:

{ name: “sue”, age: 26, status: “A”, groups: [“news”, “sports”]}

Relational database design
focuses on **data storage**

NoSQL document database design
focuses on **data use**

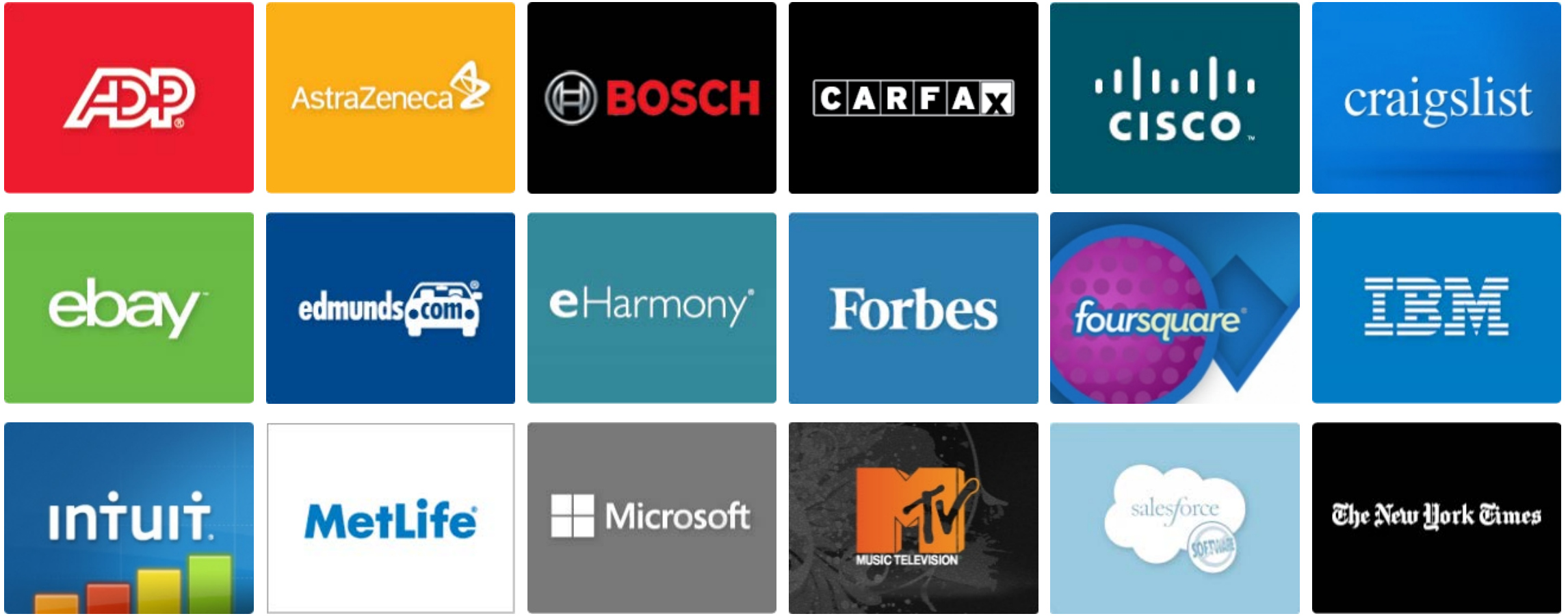


Benefits of MongoDB

- High Performance
  - Embedded documents and arrays (Denormalized schema design) reduce need for expensive joins
  - Supports indexes, even on fields within embedded documents and arrays
  - Minimal conversions with Node.js apps
- Dynamic schema
  - Fields are polymorphic (types can vary)
  - Fields within document can vary
- High Availability & Scaling
  - Replica sets provide automatic failover and data redundancy
  - Sharded environments distribute data across a cluster of machines
  - Commodity hardware
- Cost

Benefits of MongoDB

Given these benefits, MongoDB caters well to web & social application use cases.



Challenges & Common Criticisms of MongoDB

- Radically different schema design
- Database level locking
- No ACID transactions
  - Atomic – guaranteed at document/row level only
  - Consistent – no constraints
  - Isolation – read committed behavior if connected to the primary server
  - Durability – default “write concern” is not durable
- Field names affect database size
- Dynamic schema
- No native join support
- Lack of support for ODBC/JDBC APIs and SQL limit compatibility with analytic, ETL and reporting tools

Schema Design Comparison

Relational Design

Table: users

user_id 🔑	first	last	…
123456	Brody	Messmer	…
…			

Table: purchases

user_id	symbol	date	price	qty	…
123456	PRGS	2013-02-13	23.50	100	…
123456	PRGS	2012-06-12	20.57	100	…
…					

VS

NoSQL Document Design

Collection: users

```
{ user: {
        first: “Brody,
        last: “Messmer”, ...
        }
  purchases: [
        { symbol: “PRGS”, date: “2013-02-13”, price: 23.50, qty: 100, ...},
        { symbol: “PRGS”, date: “2012-06-12”, price: 20.57, qty: 100, ...},
        ...
        ]
}
...
```

ODBC/JDBC & SQL Access to MongoDB

PROGRESS EXCHANGE 2014

Connectivity to MongoDB and NoSQL Data Is Hard

- Non-standard query language
  - Queries are written as BSON, but exposed in Mongo's "drivers" as an API.
- Lack of common RDBMS functionality
  - No support for joins
  - Lack of Implicit conversions for query filters (where clause)
  - Sorting is not ANSI SQL compliant
  - No ACID transactions
  - Unique Authentication
- Non-relational schema
  - Documented oriented data model with nesting (denormalized)
  - Self-describing schema -- Can only discover columns by selecting data
  - Primary / Foreign Keys maintained by apps, not the database and no Joins

The Industry Approach – Flatten

Collection Name: stock

```
{ symbol: “PRGS”,
  purchases:[
     {date: ISODate(“2013-02-13T16:58:36Z”), price: 23.50, qty: 100},
     {date: ISODate(“2012-06-12T08:00:01Z”), price: 20.57, qty: 100,
      sellDate: ISODate(“2013-08-16T12:34:58Z”)}, sellPrice: 24.60}
  ]
}
```

Table Name: stock

_id	symbol	purchases_0_date	purchases_0_price	purchases_0_qty	purchases_1_date	purchases_1_price	purchases_1_qty	purchases_1_sellDate	purchases_1_sellPrice
1	PRGS	2013-02-13 16:58:36	23.50	100	2012-06-12 08:00:01	20.57	100	2013-08-30 12:34:58	24.60

The Progress DataDirect Approach – Normalize

The Benefits:

- Re-use of existing skills (SQL, Joins, etc)
  - Exposing complex types using concepts familiar to those savvy with RDBMS
- As arrays grow, table definitions remain constant
- Simplified / Narrower table definitions
- Joins across parent/child tables result in a single query to MongoDB. In other words, there's no performance penalty.
- Data in Arrays can be sorted and/or aggregated via SQL

	16:58:36				
1	2012-06-12 08:00:01	20.57	100	2013-08-16 12:34:58	24.60

Introducing the Schema Tool

- Creates a contract of the schema the driver will expose to ODBC/JDBC apps
- As the MongoDB schema changes (new fields added), an application will have to opt-in to these changes

  This ensures MongoDB schema changes don't break your app!
- This "contract" is stored as an XML file on the client
  - Alter column/table names
  - Hide Columns/Tables/Databases
  - Add Columns

- View statistics about your MongoDB data
  - Schema consistency (ie data type consistency for a field/column)
  - Max String length per field/column
  - Min and Max elements in an array object

Demo

Background:

Data generated by Node.js app connected to MongoDB,
all hosted by Modulus

The Problem:

How do I connect my expensive BI / analytics app to MongoDB?

Summary

- MongoDB brings significant benefits, but it’s not an RDBMS replacement

- Progress DataDirect solves several challenges for MongoDB users:
  - ODBC/JDBC access to MongoDB
  - Normalized, relational mapping to MongoDB’s denormalized schema
  - Performs cross table/collection joins

PROGRESS

PROGRESS EXCHANGE 2014

Visit the Resource Portal

- Get **session details** & presentation **downloads**
- Complete a **survey**
- Access the latest Progress **product literature**

www.progress.com/exchange2014