gophercon.ardanstudios.com

BUILDING a Relevancy ENGINE

using

mongoDB & GO

William Kennedy

@goinggodotnet

ArdanStudios
ardanstudios.com

OutCast
outcast.io

Going Go Programming
goinggo.net

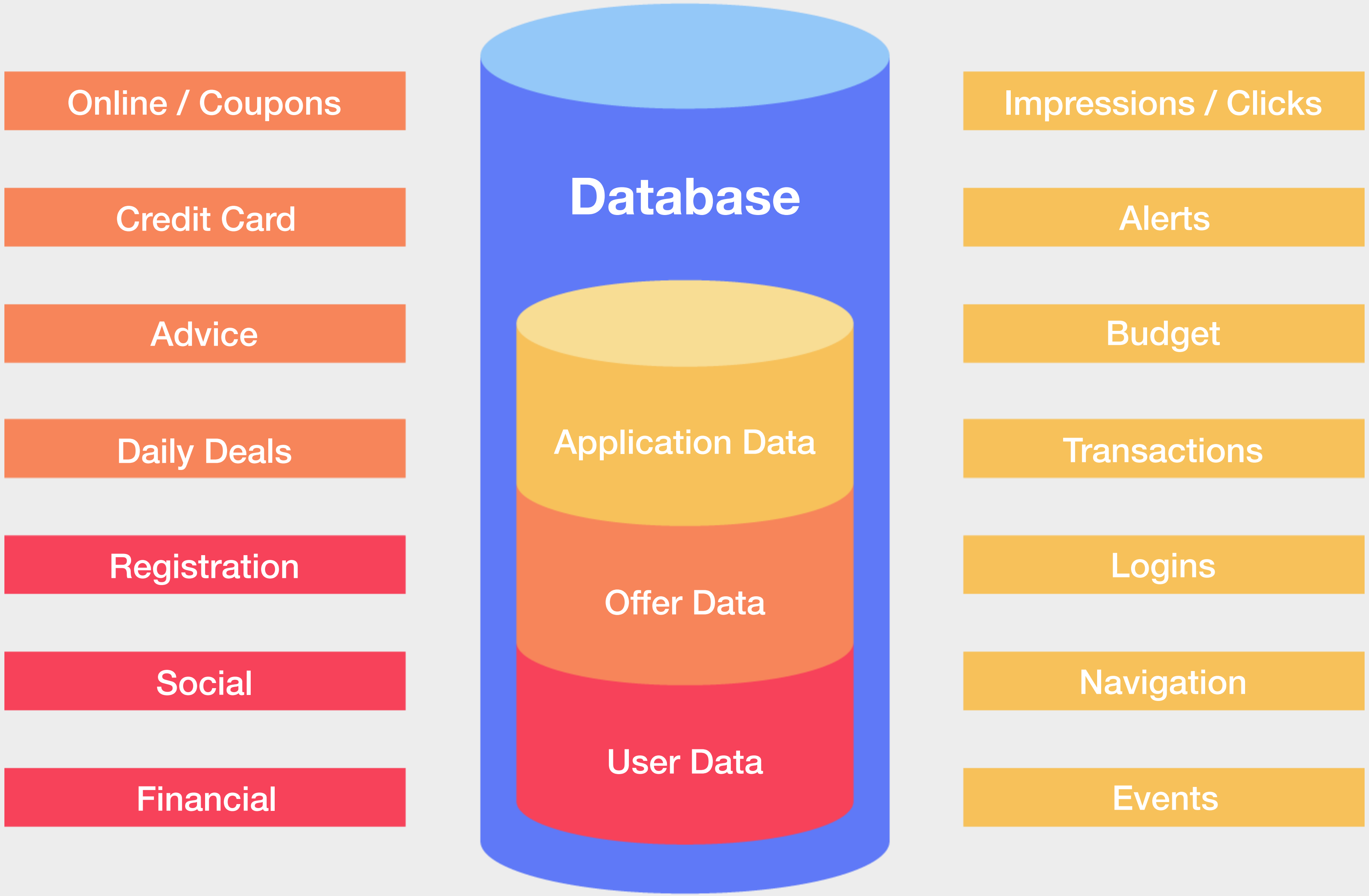
Go In Action
goinactionbook.com

Financial Apps
financialapps.com

http://vimeo.com/ardanstudios/gophercon

BUSINESS PROBLEM

Many applications have lots of great actionable data on users. How can applications use that data to provide users offers, advice and help?



BUSINESS PROBLEM

If we had a system that could do the following, we would have something:

- Dynamic Feed Management
  - Add new offer feeds, advice or help at runtime.
  - No builds required to add new feeds.
- Dynamic Rule Management
  - Add, edit and execute rules at runtime.
  - Write rules against the actionable data that targets users.
  - Have all the user and application data be available for writing rules.
  - No builds required to add new rules.

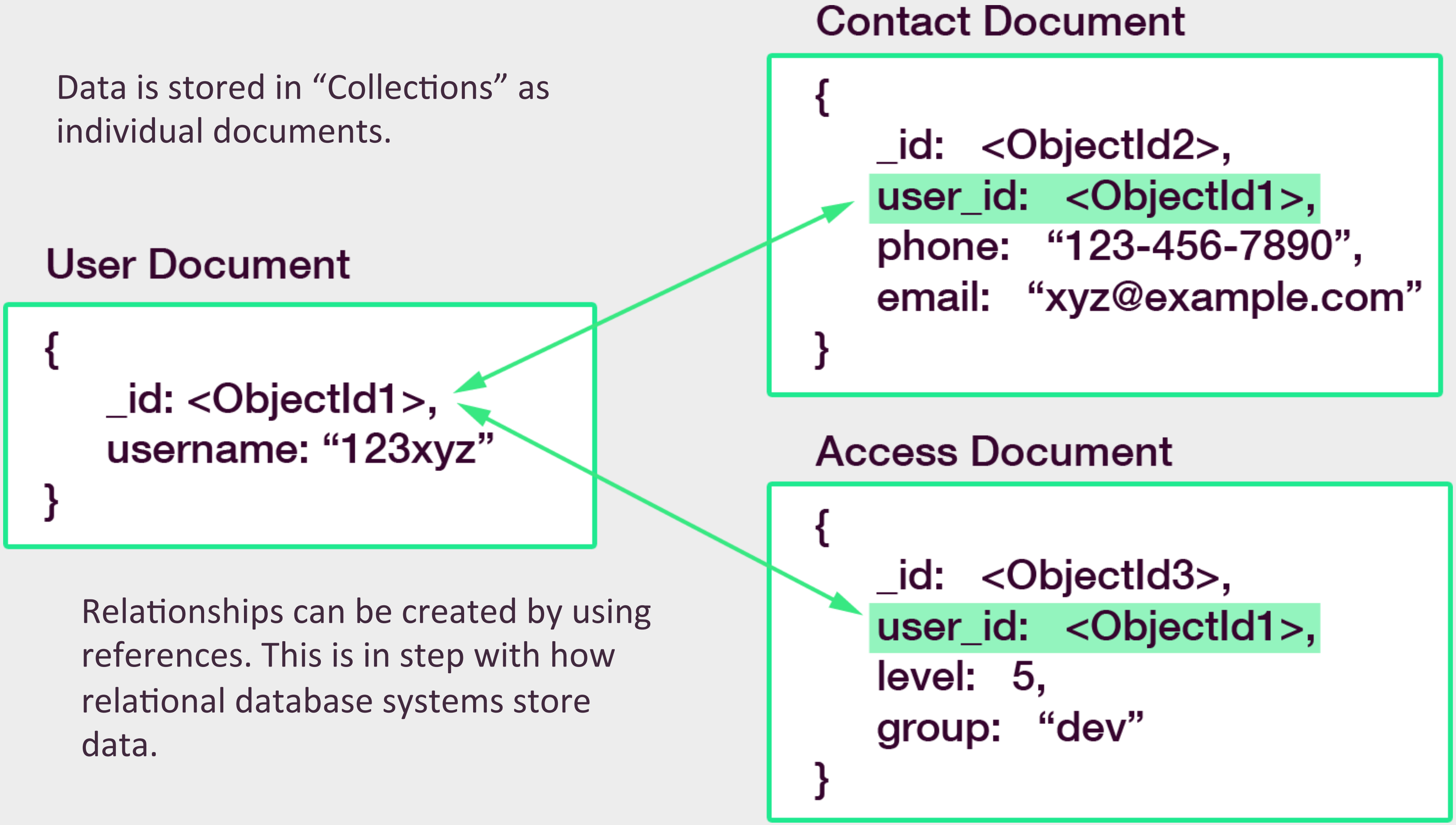
Our Answer

Go Language **MongoDB** **Mgo/Beego**

WHY MONGODB - DYNAMIC FEED MANAGEMENT

MongoDB’s schemaless database provides great flexibility.

Data is stored in “Collections” as individual documents.



User Document

```
{
    _id: <ObjectId1>,
    username: “123xyz”
}
```

Contact Document

```
{
    _id:   <ObjectId2>,
    user_id:   <ObjectId1>,
    phone:   “123-456-7890”,
    email:   “xyz@example.com”
}
```

Access Document

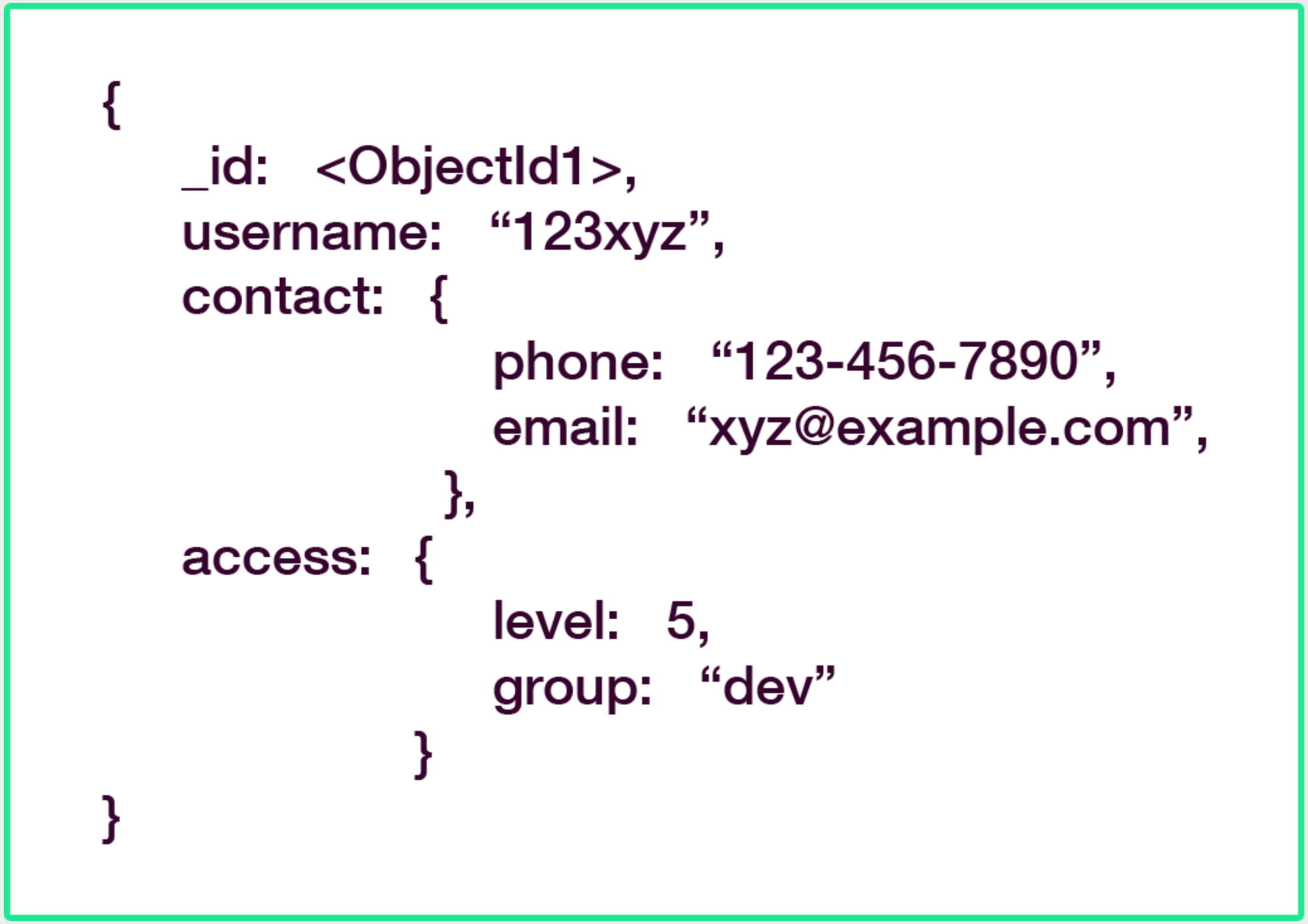
```
{
    _id:   <ObjectId3>,
    user_id:   <ObjectId1>,
    level:   5,
    group:   “dev”
}
```

Relationships can be created by using references. This is in step with how relational database systems store data.

http://docs.mongodb.org/manual/core/data-modeling-introduction/

WHY MONGODB - DYNAMIC FEED MANAGEMENT

Embedding data allows the greatest flexibility and eliminates the need for “joins” by keeping all the data organized in a single document. This allows for dynamic feed management.



```
{
    _id:  <ObjectId1>,
    username:  “123xyz”,
    contact:  {
                  phone:  “123-456-7890”,
                  email:  “xyz@example.com”,
                },
    access:  {
                  level:  5,
                  group:  “dev”
               }
}
```

Embedded sub-document

Embedded sub-document

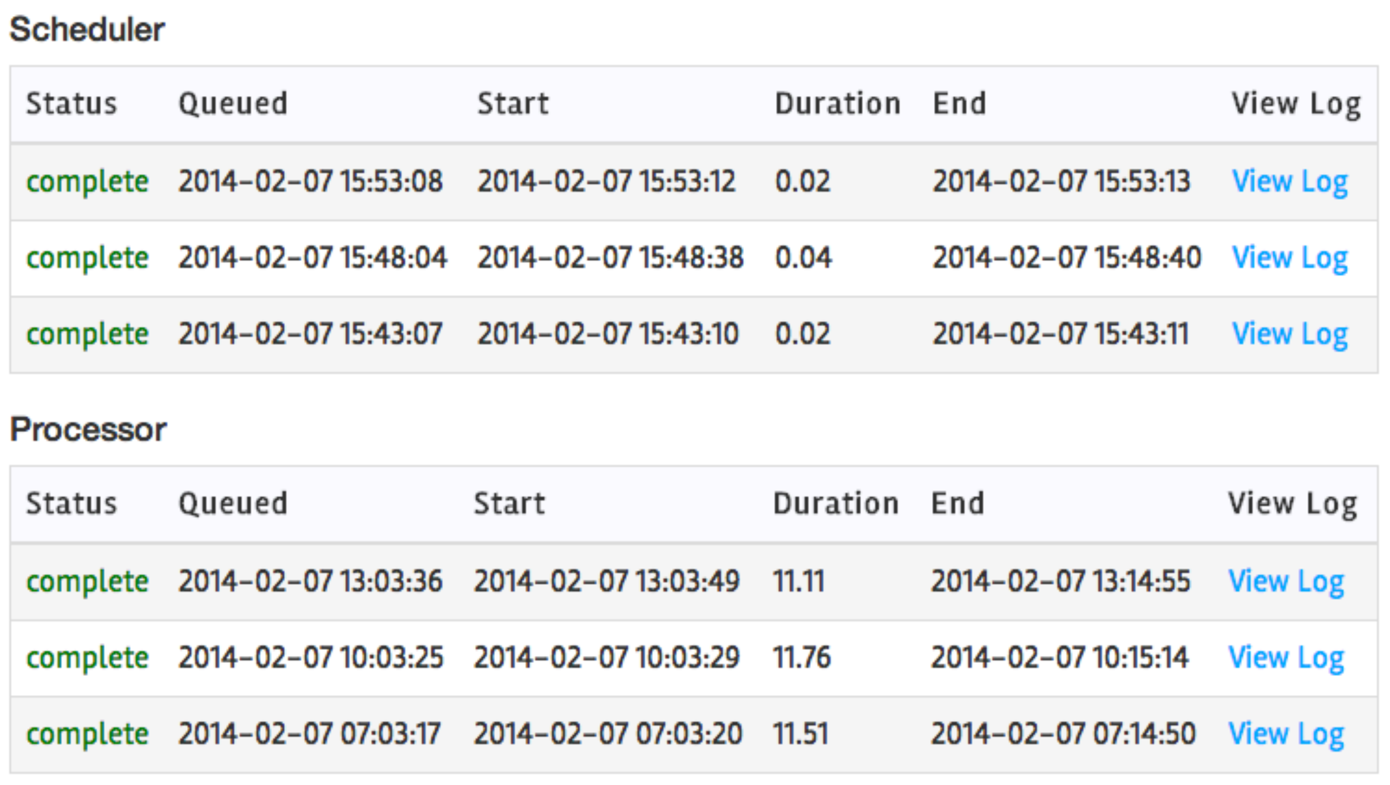
http://docs.mongodb.org/manual/core/data-modeling-introduction/

WHY GO - DYNAMIC FEED MANAGEMENT

Go supports cross platform application development. This allows for building programs that can be used in high-scale cloud platforms like Iron.IO.

Iron.io

Two Go programs perform all the feed management work



Scheduler

Status	Queued	Start	Duration	End	View Log
complete	2014-02-07 15:53:08	2014-02-07 15:53:12	0.02	2014-02-07 15:53:13	View Log
complete	2014-02-07 15:48:04	2014-02-07 15:48:38	0.04	2014-02-07 15:48:40	View Log
complete	2014-02-07 15:43:07	2014-02-07 15:43:10	0.02	2014-02-07 15:43:11	View Log

Processor

Status	Queued	Start	Duration	End	View Log
complete	2014-02-07 13:03:36	2014-02-07 13:03:49	11.11	2014-02-07 13:14:55	View Log
complete	2014-02-07 10:03:25	2014-02-07 10:03:29	11.76	2014-02-07 10:15:14	View Log
complete	2014-02-07 07:03:17	2014-02-07 07:03:20	11.51	2014-02-07 07:14:50	View Log

Scheduler
Checks the schedules and posts inventory jobs in queue.

Processor
Performs the data retrieval and insertion.

The Feed System Is Driven By Data

DYNAMIC RULE MANAGEMENT

Building rules is a core piece of the system. If we had a rule system that could do the following, we would have something.

- Write Rules that Can be Updated and Applied at Runtime
- Pass Variables to Filter and Pin Point Relevance
- Use Data Aggregation Techniques to Filter and Group Data
- Build Tests Against Aggregated Datasets
- Build Tests Against Multiple Aggregated Datasets
- Publish Data from Offer and Internal Feeds

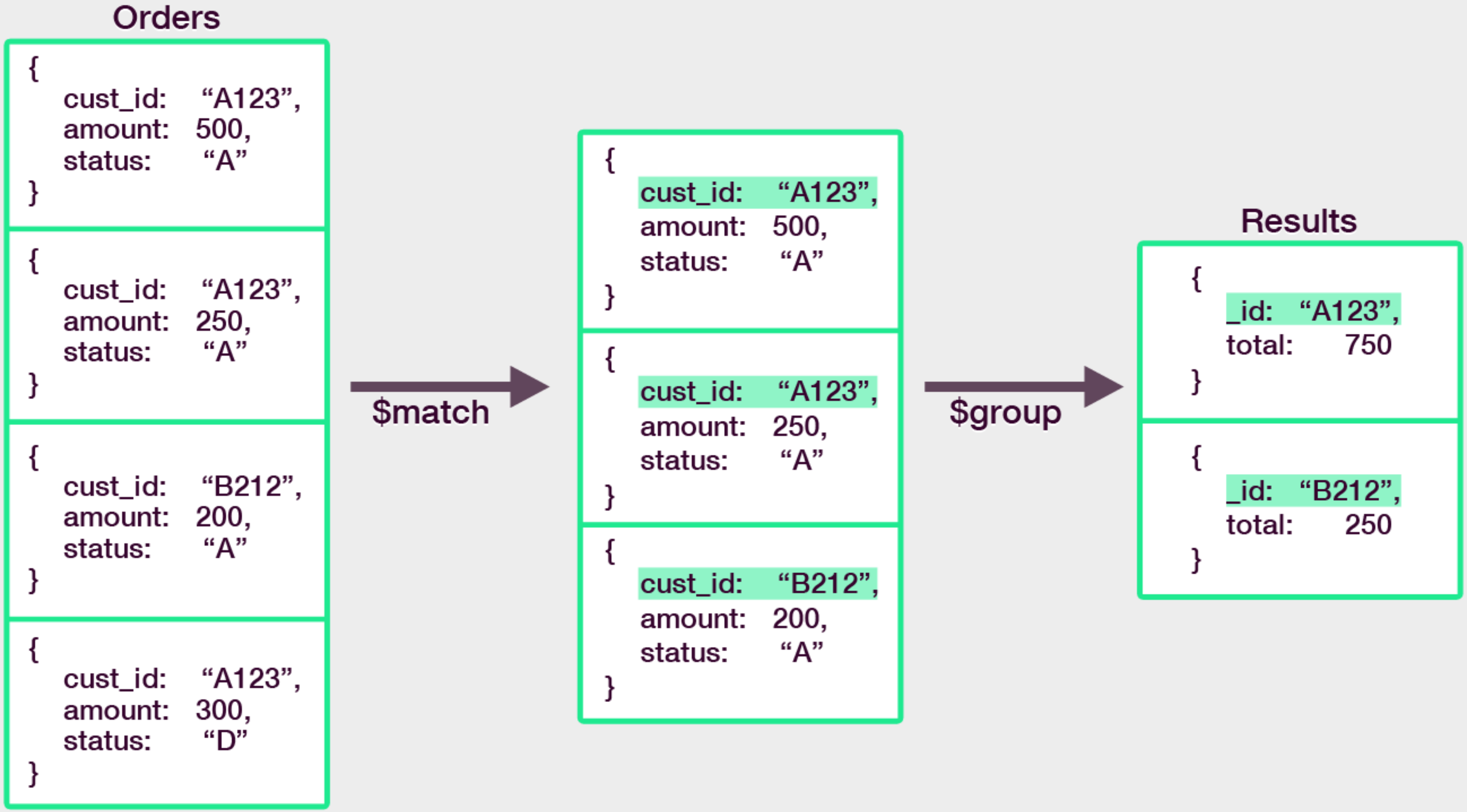
Our Answer

Go Language **MongoDB** **Mgo/Beego**

WHY MONGODB - DYNAMIC RULE MANAGEMENT

We can leverage the aggregation pipeline for writing rules.

```
Collection
    ↓
db.orders.aggregate (
      $match phase        { $match: { status: “A” } },
      $group phase        { $group: { _id: “$cust_id”, total: { $sum: “amount” } } }
                     )
```



```
{
   cust_id:   “A123”,
   amount:   500,
   status:    “A”
}
{
   cust_id:   “A123”,
   amount:   250,
   status:    “A”
}
{
   cust_id:   “B212”,
   amount:   200,
   status:    “A”
}
{
   cust_id:   “A123”,
   amount:   300,
   status:    “D”
}
```

```
{
   cust_id:   “A123”,
   amount:   500,
   status:    “A”
}
{
   cust_id:   “A123”,
   amount:   250,
   status:    “A”
}
{
   cust_id:   “B212”,
   amount:   200,
   status:    “A”
}
```

```
{
   _id:   “A123”,
   total:     750
}
{
   _id:   “B212”,
   total:     250
}
```

http://docs.mongodb.org/manual/core/aggregation-pipeline/

VARIABLE RULES - DYNAMIC RULE MANAGEMENT

Variable rules take variable data that is passed into the rule from the application. Variables are similar to query string parameters on an URL. They can be tested for specific values.

For Variable Substitution

- **Substitute userId for this value in any rule being applied.**
- **Substitute screen-tab for this value in any rule being applied.**

Rule Type: variable

```
{
   "userId" : "12345",
   "screen-tab" : "user"
}
```

Pass Variables to Filter & Pin Point Relevance

- **When the user is on the user tab.**
- **When the budget item is for entertainment expenses.**
- **When the transaction record is a grocery transaction.**

PIPELINE RULES - DYNAMIC RULE MANAGEMENT

Pipeline rules are run through the MongoDB aggregation pipeline. Datasets can be checked for specific conditions or saved for use in later Go Template rules.

User Demographics : (User Collection)

- **When the user is younger than 40 years old.**
- **When the user is single.**
- **When the user has a Gmail account.**

Rule Type: pipeline

```
{ "$match" : { "user_id" : "#userId#" }}
{ "$match" : { "age" : { "$lt" : 40}}}
```

PIPELINE RULES - DYNAMIC RULE MANAGEMENT

Pipeline rules are run through the MongoDB aggregation pipeline. Datasets can be checked for specific conditions or saved for use in later Go Template rules.

User Transactions : (User_Transactions Collection)

- **When the user has spent more than $20 this month on movies.**
- **When the user has over 5 transactions for less than $5 this week.**
- **When the user has spent over $5,000 in the past 30 days.**

Rule Type: pipeline

```
{ "$match" : { "user_id" : "#userId#", "category" : "movies" }}

{ "$group" : { "_id" : { "category" : "$category" }, "amount" :
    { "$sum" : "$amount" }}}

{ "$match" : { "amount" : { "$gt" : 20.00}}}
```

GO TEMPLATE RULES - DYNAMIC RULE MANAGEMENT

Template rules are used to compare multiple datasets generated by aggregation pipeline queries. Multiple datasets can be compared and checked for specific conditions.

Compare Multiple Datasets

- **When the user spends over $100 on groceries and entertainment.**
- **When the user has a G+ account and is over 40.**
- **When the user is married and spent over $1,000 last week.**

Rule Type: template

```
{{$movies := index .Movies "amount"}}

{{$dining := index .Dining "amount"}}

{{if gt $dining $movies}}
   VALID
{{end}}
```

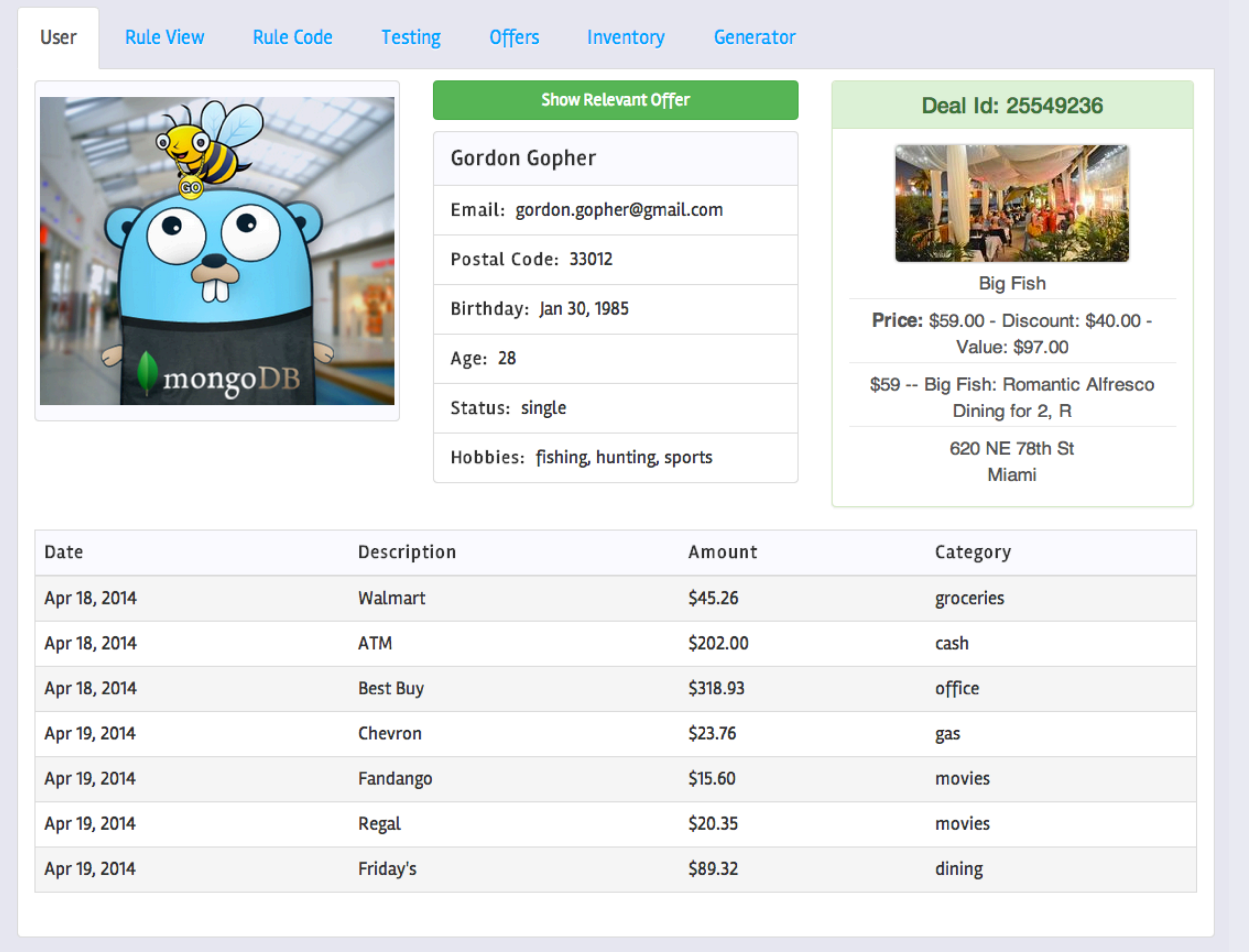
WHY GO, MONGODB and BEEGO - SOLUTION

By combining the data flexibility and aggregation capabilities of MongoDB with the Go language and template framework, we have a scalable, redundant and feature rich solution.

- **Go Language**
  - Systems programming language
  - Compiles to binary code for target OS/Architectures
  - Cross compile on many operating systems
  - Access to scalable cloud computing environments
  - MGO driver for Go provides excellent MongoDB support

- **MongoDB**
  - Scalability and redundancy out of the box
  - Great MongoDB hosting providers
  - Schemaless database that provides great flexibility
  - Aggregation pipeline to build rules and datasets
  - Can search against text with good performance

- **Beego Web Framework**
  - MVC based web framework
  - Enough framework to be productive without being handcuffed
  - Sits on top of the Go HTTP package
  - Excellent web service and view support



Gordon Gopher

Email: gordon.gopher@gmail.com

Postal Code: 33012

Birthday: Jan 30, 1985

Age: 28

Status: single

Hobbies: fishing, hunting, sports

Deal Id: 25549236

Big Fish

Price: $59.00 - Discount: $40.00 - Value: $97.00

$59 -- Big Fish: Romantic Alfresco Dining for 2, R

620 NE 78th St
Miami

Date	Description	Amount	Category
Apr 18, 2014	Walmart	$45.26	groceries
Apr 18, 2014	ATM	$202.00	cash
Apr 18, 2014	Best Buy	$318.93	office
Apr 19, 2014	Chevron	$23.76	gas
Apr 19, 2014	Fandango	$15.60	movies
Apr 19, 2014	Regal	$20.35	movies
Apr 19, 2014	Friday's	$89.32	dining

CONCLUSION

MongoDB and Go are technologies that work incredibly well together. The communities for both technologies are vibrant and expanding. Now is the time to consider these technologies.

- Go, MGO, MongoDB and Beego are powerful technologies.
- Create data driven and dynamic systems with Go and MongoDB.
- Leverage powerful cloud systems like Iron.IO and Heroku.
- The MongoDB aggregation pipeline is fast and powerful.
- Go templates provide a seamless way to process datasets.
- These technologies can be used to create highly scalable systems.

gophercon.ardanstudios.com

BUILDING a Relevancy ENGINE using

mongoDB & GO

William Kennedy

@goinggodotnet

ArdanStudios

ardanstudios.com

OutCast

outcast.io

Going Go Programming

goinggo.net

Go In Action

goinactionbook.com

Financial Apps

financialapps.com