PHP
fwdays

Getting the Most out of MongoDB

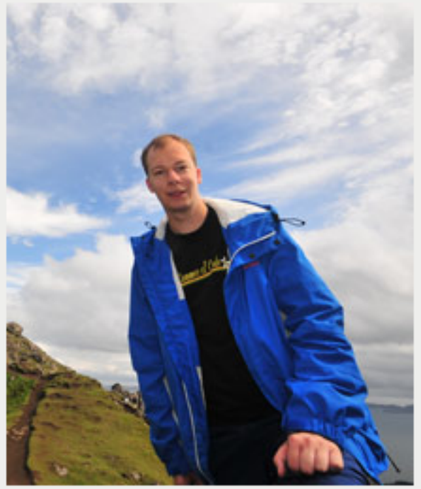
Derick Rethans

derick@mongodb.com—@derickr

Svitla EVOPLAY AMADEUS Magento stfalcon.com ERGO PLACE www.ergo.place ZONE 3000

About Me

Derick Rethans



- I'm European, living in London
- I work on the **MongoDB** PHP driver
- **Xdebug** & PHP's Date/Time support
- I ♥🌍 maps, I ♥🍺 beer, I ♥🥃 whisky
- twitter: @derickr

Terminology

- **Document**: the data (row)
- **Collection**: contains documents (table, view)
- **Index**
- **Embedded Document** (~join)
- **Sharding** (partitioning)

2.

Nota previa.

El libro primitivo en q se asentaban con la debida distincion y claridad las partidas de las Informaciones Patrimoniales y casamientos de esta Mision de S.n Luis Obispo desde su principio, pereció en el Incendio q la abrasó en gran parte en la Madrugada del dia 29 de Noviembre del año proxime vencido de 1776. por cuya causa los Patrim.s hasta dho dia celebrados se hallaran acá notados sin la formalidad de los propios dias en q se celebraron y de los particulares testigos q á cada uno de ellos asistieron, si solo en forma de Padron q espresse una á una las familias de cuyos Patrimonios ta, ya de los Padrones, ya de [illegible] celebraron, y celebraren despues de dho incendio se continuarán con la correspte distincion; y p.a la claridad asi lo noté.

Fr Junipero Serra

Vicente Briones Natural de S.t Luys Potosi Casado

4472.

Documents: Simple

```
{
  "_id" : ObjectId("569c19f3b8c96e09421b2926"),
  "slug" : "ardbeg",
  "name" : "Ardbeg",
  "created_at" : 1453070835,
  "region_slug" : "scotland-islay",
  "region" : "Scotland, Islay",
  "location" : "Ardbeg, Scotland, UK",
  "closed" : false,
  "description" : "Ardbeg Distillery (Scottish Gaelic: Taigh-stail Àirde Beaga)
is a Scotch whisky distillery in Ardbeg on the south coast of the isle of
Islay, Argyll and Bute, Scotland, in the Inner Hebrides group of islands. The
distillery is owned by Louis Vuitton Moët Hennessy, and produces a heavily
peated Islay whisky.[2] The distillery uses malted barley sourced from the
maltings in Port Ellen. (Wikipedia)"
}
```

- `_id`: immutable unique key of document
- All strings are UTF-8

Documents: Complex

```
{
  "_id" : "derick@localhost",
  "fullname" : "Derick Rethans",
  "slug" : "derick-rethans",
  "created_at" : 1452546141,
  "timezone" : "Europe/London",
  "confirmed" : true,
  "confirmed_at" : 1452546148,
  "location" : "London, UK",
  "words" : [ "derick", "rethans", "london", "uk" ],
  "count" : 16,
  "count_unique" : 13,
  "badges" : [
    { n: "unique1", l: 1 },
    { n: "age21", l: 3 }
  ],
  "isAdmin" : true
}
```

- `_id`: does not have to be an Object ID
- Values can be arrays, documents, or arrays of documents

Collections

Collections

```
{
  "_id" : ObjectId("56b4f6dfb8c96e5c2e5121c7"),
  "slug" : "ancnoc",
  "name" : "anCnoc",
  "region_slug" : "scotland-highlands",
  "region" : "Scotland, Highlands",
  "url" : "http://ancnoc.com/",
}
{
  "_id" : ObjectId("56a6a6dab8c96e35ea7b3e24"),
  "slug" : "suntory",
  "name" : "Suntory",
  "region_slug" : "japan",
  "region" : "Japan",
  "location" : "Osaka, Japan",
  "description" : "Suntory Holdings Limited (サントリーホールディングス株式会社
Santorī Hōrudingusu Kabushiki-Gaisha?) is a Japanese brewing and distilling
company group. Established in 1899, it is one of the oldest companies in the
distribution of alcoholic beverages in Japan, and makes Japanese whisky."
}
```

- Common fields
- Disparate fields

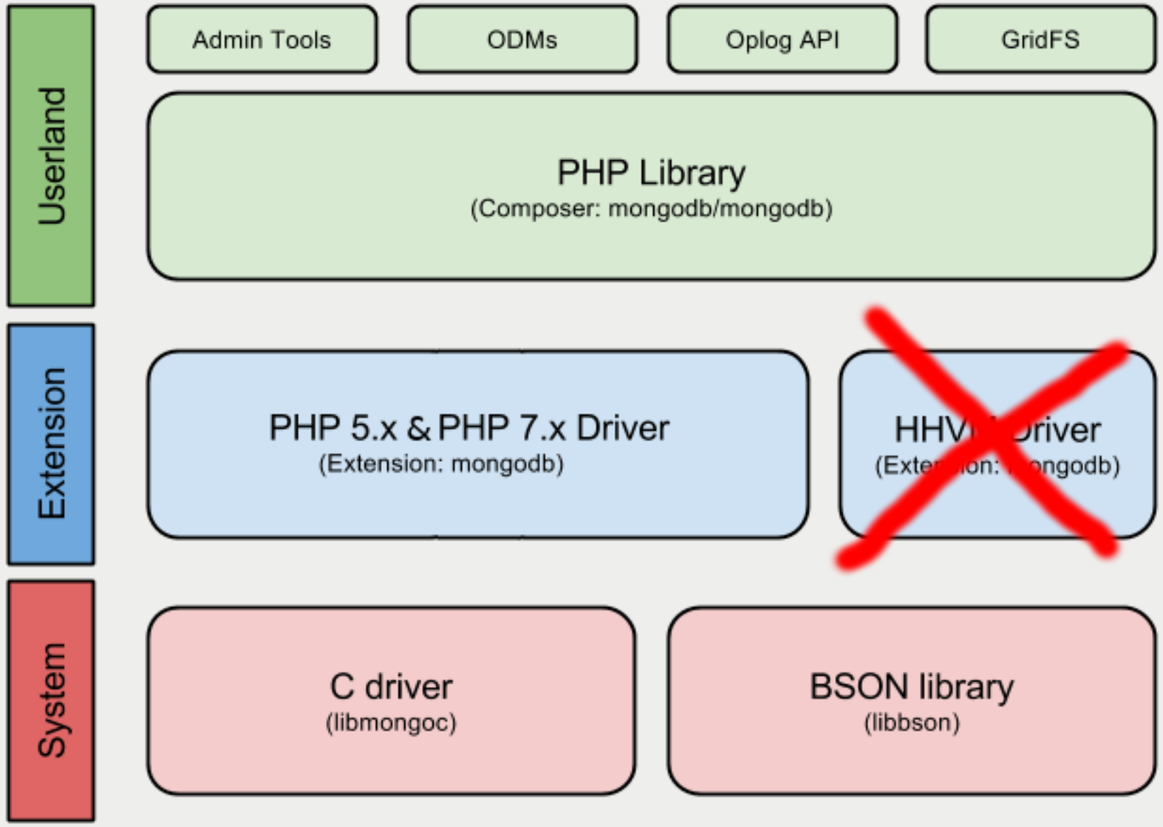
Indexes

MongoDB: Indexes

```
{
  "_id" : ObjectId("56a7f861b8c96e06585e8195"),
  "name" : "Glen Grant-Glenlivet 20",
  "style_slug" : "single-cask",
  "description" : "Distilled in 1995 and bottled in October 2005.",
  "attributes": {
    "age" : 20,
    "cask_strength" : true,
  },
  "words" : [ "glen", "grant", "glenlivet", "20", "cadenheads" ],
}
```

- `_id`: automatic unique key
- `name`: unique key
- `style_slug`: category lookups
- `attributes.age`: on nested fields
- `words`: on array values

Libraries, Extensions, and Drivers



MongoDB & PHP

- Extension: `pecl/mongodb` (PHP 5.4 and higher)
- Library: `phplib` (`mongodb/mongodb` on composer)

Installation:

```
pecl install mongodb
echo "extension=mongodb.so" > /etc/php5/cli/conf.d/20-mongodb.ini"

mkdir new-project
cd new-project
composer require mongodb/mongodb
```

In your script:

```
<?php
require 'vendor/autoload.php';

$client = new \Mongo\Client;
$collection = $client->selectCollection( 'databasename', 'collectionname' );
?>
```

MongoDB & PHP: CRUD

```php
<?php
require 'vendor/autoload.php';

$whiskies = (new MongoDB\Client)->dramio->whisky;

// Find one document:
$whisky = $whiskies->findOne( [ 'name' => 'Cambus 26' ] );

// Find multiple documents:
$cursor = $whiskies->find( [ 'style' => 'single-cask' ] );

// Insert one document:
$whiskies->insertOne( [ 'name' => 'Glenfiddich 15' ] );

// Insert multiple documents:
$whiskies->insertMany( [
  [ 'name' => 'Glenlivet 12' ],
  [ 'name' => 'Glenlivet 15' ]
] );

// Update document:
$whiskies->updateOne( [ 'slug' => 'cambus-26' ], [ '$set' => [ 'age' => 26 ] ] );

// Remove document:
$whiskies->deleteOne( [ 'name' => 'Jack Daniels' ] );
?>
```

Querying

```
SELECT * FROM whisky;

$whisky->find( [] );
```

```
SELECT name, age FROM whisky;

$whisky->find( [], [ 'name' => 1, 'age' => 1 ] );
```

```
SELECT * FROM whisky WHERE age = 42;

$whisky->find( [ 'age' => 42 ] );
```

```
SELECT * FROM whisky WHERE name = 'Macallan' AND age > 18;

$whisky->find( [ 'a' => 'Macallan', 'age' => [ '$gt' => 18 ] ] );
```

```
SELECT * FROM whisky WHERE name = 'Macallan' OR age > 18;

$whisky->find( [ '$or' => [
        [ 'name' => 'Macallan' ],
        [ 'age' => [ '$gt' => 18 ] ]
] ] );
```

```
SELECT * FROM whisky LEFT JOIN distillery ON (whisky.distillery_id = distillery.id);

???
```

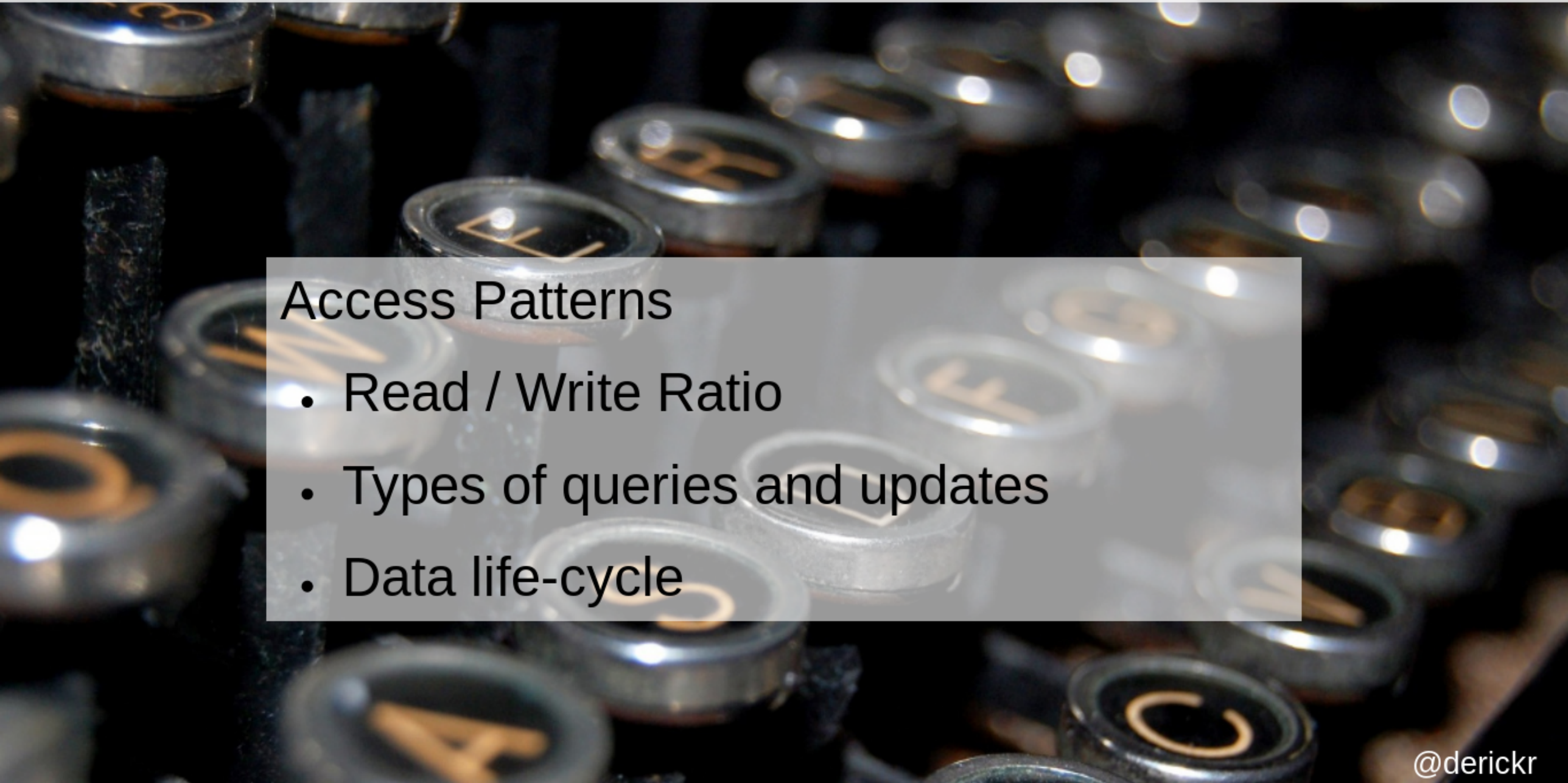
Data Model

Data Model

- No relations between collections
- No foreign keys
- No transactions

- Use nested documents and arrays
- Duplicate data
- Use atomic operations

Schema Considerations



Access Patterns

- Read / Write Ratio
- Types of queries and updates
- Data life-cycle

EAV: Entity Attribute Value

```
SELECT entity_id, attribute_code, value
FROM catalog_product_entity_text cpev
JOIN eav_attribute ea ON cpev.attribute_id = ea.attribute_id;

| entity_id | attribute_code       | value              |
+-----------+----------------------+--------------------+
|         1 | description          | Cute elephpant     |
|         1 | short_description    | It&#39;s cute      |
|         1 | meta_keyword         | NULL               |

SELECT entity_id, attribute_code, value
FROM catalog_product_entity_int cpev
JOIN eav_attribute ea ON cpev.attribute_id = ea.attribute_id;

| entity_id | attribute_code        | value |
+-----------+-----------------------+-------+
|         1 | status                |     1 |
|         1 | visibility            |     4 |
|         1 | tax_class_id          |     2 |
```

In MongoDB

```
{
        '_id': 1,
        'name' : 'Elephpant',
        'url_key': 'elephpant',
        'description': 'Cute elephpant',
        'short_description': "It's cute",
        'status': 1,
        'visibility': 4,
        'tax_class_id': 2,
}
```

Enforcing Schema

Allows you to set a **validator**, a query that every new or updated document must match

Example:

```
db.runCommand( {
        collMod: "whisky",
        validator: {
                name: { $type: "string" },
                age: { $type: "integer" },
        },
        validationLevel: "moderate", // off, strict
        validationAction: "warn", // error,
} )
```

Enforcing Schema

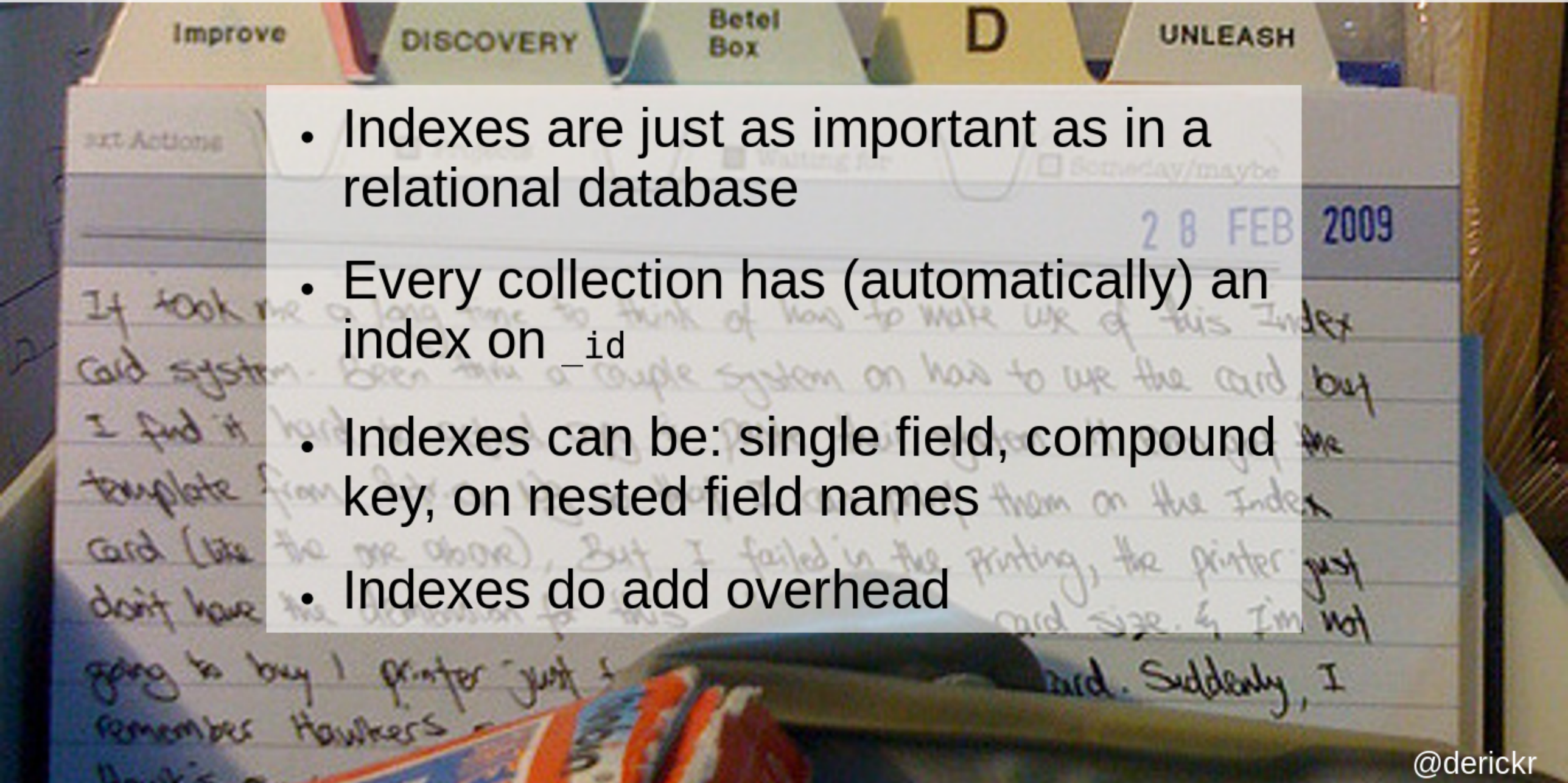
Allows you to set a **validator**, a query that every new or updated document must match

Example with JSON schema:

```
db.runCommand( {
    collMod: "whisky",
    validator: {
        $jsonSchema: {
            bsonType: "object",
            required: [ "name", "age" ],
            properties: {
                name: {
                    bsonType: "string",
                    description: "The name of the whisky.",
                },
                age: {
                    bsonType: "int",
                    minimum: 3,
                    description: "The age in years of the whisky".
                }
            }
        }
    },
} )
```

Indexes

Indexes



- Indexes are just as important as in a relational database
- Every collection has (automatically) an index on `_id`
- Indexes can be: single field, compound key, on nested field names
- Indexes do add overhead

Creating Indexes

Shell

```
db.checkins.createIndex( { 'clinked_by.slug': 1 } );
```

Driver

```
<?php
$cmd = new \MongoDB\Driver\Command( [
    'createIndexes => 'checkins',
    'indexes' => [
        [ 'key' => [ 'user_slug' => 1, 'friend_slug => 1 ], 'unique' => true ],
    ]
] );
$manager->executeCommand( 'dramio', $cmd );
```

Library

```
<?php
require 'vendor/autoload.php';
$collection = (new MongoDB\Client)->dramio->user;
$collection->createIndex(
    [ 'validation_code' => 1 ],
    [ 'sparse' => true ]
);
```

Explain (Index)

```
> db.whisky.find( { name: 'Ord 10' } ).explain();
{
    "queryPlanner" : {
        "namespace" : "dramio.whisky",
        "indexFilterSet" : false,
        "parsedQuery" : { "name" : { "$eq" : "Ord 10" } },
        "winningPlan" : {
            "stage" : "FETCH",
            "inputStage" : {
                "stage" : "IXSCAN",
                "keyPattern" : {
                    "name" : 1
                },
                "indexName" : "name_1",
                "isMultiKey" : false,
                "direction" : "forward",
                "indexBounds" : {
                    "name" : [
                        "[\"Ord 10\", \"Ord 10\"]"
                    ]
                }
            }
        },
    },
}
```

Special Indexes

Geospatial Indexes (2d, and 2dsphere)

- Find objects near a pair of coordinates
- Find objects intersecting other objects
- GeoJSON: Point, Line, Polygon, GeometryCollection

Text Indexes

- "Full Text Search"
- Stemming, Weighing, Ranking

Aggregation Framework

Aggregation Pipeline

- Process a stream of documents
  - Original input is a collection
  - Final output is a cursor
- Series of operators
  - Filter or transform data
  - Input/output chain

```
ps ax | grep mongod | head -n 1
```

Our Example data


```
{
    "_id" : {
        "k" : "untappd_20140706213817",
        "u" : "derick"
    },
    "beer_name" : "Aurora Australis",
    "brewery_name" : "Bridge Road Brewers",
    "beer_type" : "Belgian Quad",
    "comment" : "Red wine flavours. Brewed in Australia but packaged in Norway.",
    "venue_name" : "Olympen",
    "venue_city" : "Oslo",
    "venue_state" : "Oslo",
    "checkin_url" : "https://untappd.com/c/97860912",
    "beer_url" : "https://untappd.com/beer/300283",
    "brewery_url" : "https://untappd.com/brewery/3174",
    "brewery_country" : "Australia",
    "beer_abv" : 11,
    "beer_ibu" : 30,
    "rating_score" : 4.5,
    "l" : {
        "type" : "Point",
        "coordinates" : [
            10.7643,
            59.9121
        ]
    }
}
```

Aggregation Pipeline Stages

- `$match`
- `$project`
- `$group`
- `$unwind`
- `$sample`
- `$sortByCount`
- `$bucket` / `$bucketAuto`
- `$geoNear`
- `$sort`, `$limit`, `$skip`

What is your favourite *German* beer?

Logical Approach:

- **find** all *German* beers
- **sort** by rating **descendingly**
- **limit** by **1**

Pipeline:

```
db.beer.aggregate( [
    { '$match' : { 'brewery_country' : 'Germany' } },
    { '$sort' : { 'rating_score' : -1 } },
    { '$limit' : 1 },
] ).pretty()
```

What is your favourite *German* beer?

```
db.beer.aggregate( [
    { '$match' : { 'brewery_country' : 'Germany' } },
    { '$sort' : { 'rating_score' : -1 } },
    { '$limit' : 1 },
] ).pretty()
```

Result:


```
{
    "_id" : { "k" : "untappd_20141115152335", "u" : "derick" },
    "beer_name" : "Aventinus Eisbock",
    "brewery_name" : "Weisses Bräuhaus G. Schneider & Sohn",
    "beer_type" : "Eisbock",
    "comment" : "This is the stuff. I can't say why I like it so much though",
    "venue_name" : "Tap East",
    "venue_city" : "Stratford",
    "venue_state" : "Greater London",
    "checkin_url" : "https://untappd.com/c/129666676",
    "beer_url" : "https://untappd.com/beer/67996",
    "brewery_url" : "https://untappd.com/brewery/1023",
    "brewery_country" : "Germany",
    "beer_abv" : 12,
    "beer_ibu" : 15,
    "rating_score" : 4.5,
    "l" : { "type" : "Point", "coordinates" : [ -0.00878692, 51.5443 ] }
}
```

Give an overview of number of beers per *x*

Pipeline:

```
db.beer.aggregate( [
    { '$facet' : {
        'beer_type' :       [ { '$sortByCount' : '$beer_type' } ],
```

```
        'brewery_country' : [ { '$sortByCount' : '$brewery_country' } ],
```

```
        'abv' : [
            { '$match' : { 'beer_abv' : { '$gte' : 0 } } },
            { '$bucket' : {
                groupBy: '$beer_abv',
                boundaries: [ 0, 2, 4, 6, 8, 10 ],
                default: 'way-too-much',
            } }
        ],
    } },
] ).pretty()
```

Give an overview of number of beers per x

```
db.beer.aggregate( [
    { '$facet' : {
        'abv' : [
            { '$match' : { 'beer_abv' : { '$gte' : 0 } } },
            { '$bucket' : {
                groupBy: '$beer_abv',
                boundaries: [ 0, 2, 4, 6, 8, 10 ],
                default: 'way-too-much',
            } }
        ],
        'beer_type' :       [ { '$sortByCount' : '$beer_type' } ],
        'brewery_country' : [ { '$sortByCount' : '$brewery_country' } ],
    } },
] ).pretty()
```

Result:

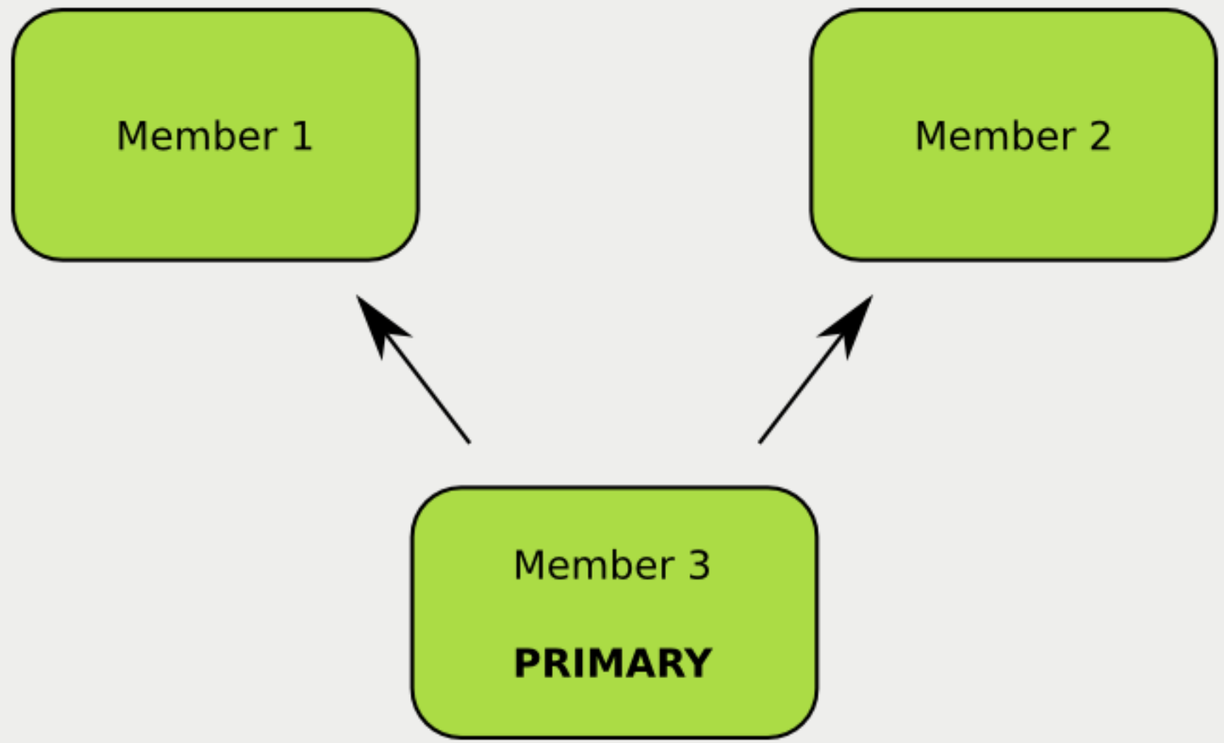
```
{
    "abv" : [
        { "_id" : 0, "count" : 64 },
        { "_id" : 2, "count" : 240 },
        { "_id" : 4, "count" : 2248 },
        { "_id" : 6, "count" : 966 },
        { "_id" : 8, "count" : 435 },
        { "_id" : "way-too-much", "count" : 187 }
    ],

    "beer_type" : [
        { "_id" : "Cider", "count" : 730 },
        { "_id" : "English Bitter", "count" : 164 },
        { "_id" : "Stout", "count" : 139 },
        …
    ],
    "brewery_country" : [
        { "_id" : "England", "count" : 1971 },
        { "_id" : "United States", "count" : 501 },
        { "_id" : "Belgium", "count" : 405 },
        …
    ],
}
```

Replication

MongoDB Replica Set Features

- A cluster of ***n*** servers
- All writes to **primary**
- Reads can be from primary (default) or a secondary
- Any **(one)** node can be primary
- Consensus **election** of primary
- Automatic failover and recovery

How MongoDB replication works



- Election establishes the PRIMARY
- Data replication from PRIMARY to SECONDARY

Eventual Consistency

- Read Preference (except `RP_PRIMARY`)
  - Driver will always send writes to primary
  - Driver will send read requests to secondaries
  - Options to prefer primary, secondary, or nearest
- Warning!
  - Secondaries may be out of date
  - Failover may happen while you're trying to write

MongoDB Atlas

MongoDB's Database as a Service

- Automated Operations
- Easy to scale up and down
- Managed Upgrades
- Continuous Backups
- Performance Monitoring
- **Free Tier**

Any Queries?

Slides & Resources

Contact

derick@mongodb.com—@derickr