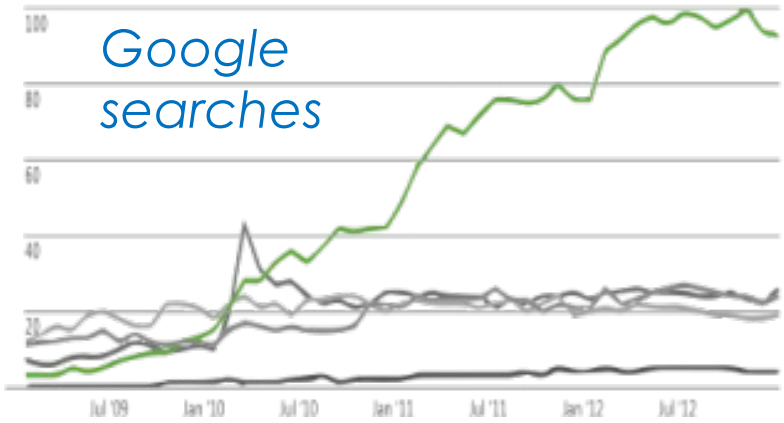
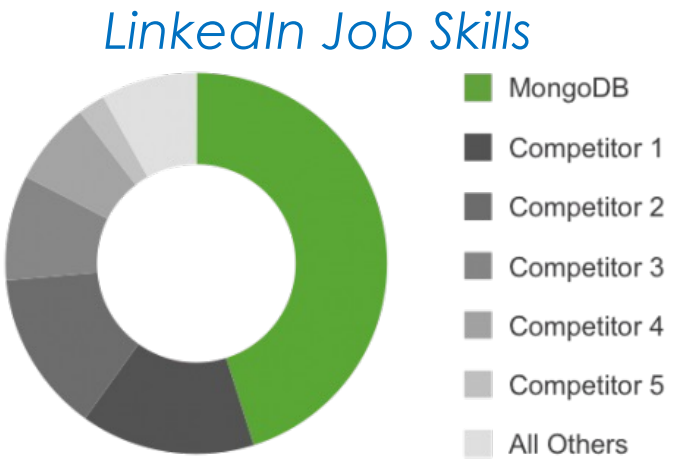
Εισαγωγή στη MongoDB

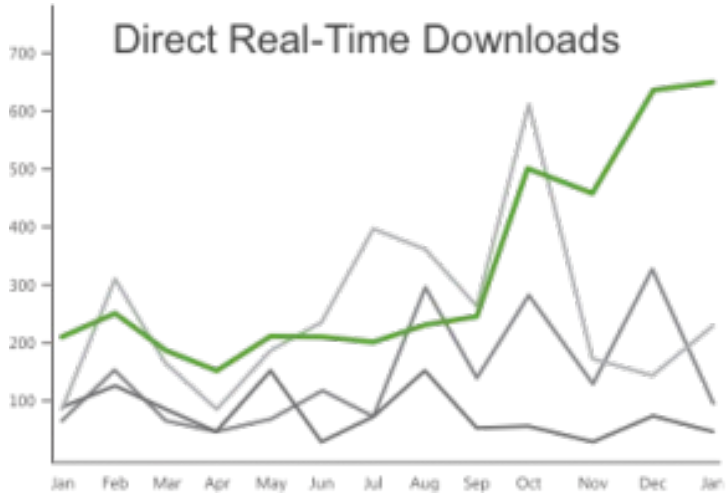
Μάθημα: «Διαχείριση Μεγάλων Δεδομένων»
ΠΜΣ «Κυβερνοασφάλεια και Επιστήμη Δεδομένων»

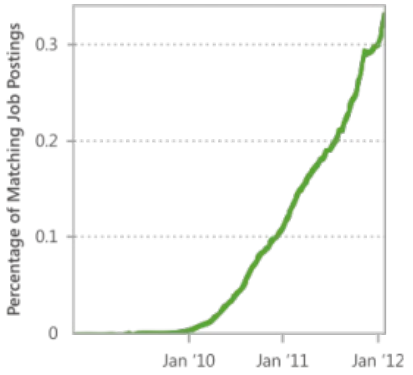
Αναπληρωτής Καθηγητής Χρήστος Δουλκερίδης
Τμήμα Ψηφιακών Συστημάτων
Πανεπιστήμιο Πειραιώς

MongoDB









Top Job Trends

1. HTML 5
2. **MongoDB**
3. iOS
4. Android
5. Mobile Apps
6. Puppet
7. Hadoop
8. jQuery
9. PaaS
10. Social Media

Πηγή: https://www.mongodb.com/leading-nosql-database

Περίγραμμα Διάλεξης

- Σύντομη Εισαγωγή στην Έννοια του Document
- Μοντελοποίηση Δεδομένων στη MongoDB
- Ευρετήρια (Indexing)
- Κατάτμηση (Partitioning/Sharding)

Τι **δεν** θα δούμε στο σημερινό μάθημα:

Εντολές MongoDB → ***Εργαστήριο***

Σύντομη Εισαγωγή στην Έννοια του Document

Έγγραφο (Document)

- Ένα διατεταγμένο σύνολο (ordered set) κλειδιών (keys) με σχετιζόμενες τιμές (value)
  - {"greeting" : "Hello, world!"}
  - {"greeting" : "Hello, world!", "foo" : 3}

- Η MongoDB είναι type-sensitive και case-sensitive
  - {"foo": 3} != {"foo": "3"}
  - {"foo": 3} != {"Foo": 3}

Συλλογές (Collections)

- Ένα collection είναι το αντίστοιχο του πίνακα σε μια σχεσιακή ΒΔ
  - Όπως και το document είναι το αντίστοιχο του row ή tuple
- Ένα collection έχει *δυναμικό σχήμα*
  - Μπορεί να περιέχει documents διαφορετικών μορφών
  - Π.χ. τα documents:
    - {"greeting" : "Hello, world!"}
    - {"foo" : 5}

    μπορούν να αποθηκευτούν στο ίδιο collection

Σχέση με JSON Documents – Τύποι Δεδομένων

- Τα documents της MongoDB μοιάζουν με JSON
  - Υπάρχουν περισσότεροι τύποι δεδομένων, από τους 6 που υποστηρίζει το JSON:
    - null, boolean, numeric, string, array, object
  - Η MongoDB εισάγει νέους τύπους δεδομένων, διατηρώντας την έννοια των key-value ζευγαριών
    - Null, boolean, number (NumberInt, NumberLong), date, string, regular expression, array, embedded object, objected, binary data, code

Εμφώλευση Εγγράφων (Embedding)

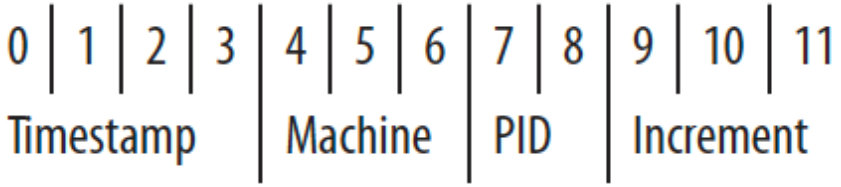
- Ένα έγγραφο μπορεί να χρησιμοποιηθεί σαν τιμή ενός κλειδιού
  - Ονομάζεται εμφωλευμένο έγγραφο (embedded document)
- Χρησιμεύει στην οργάνωση των δεδομένων σε ιεραρχική μορφή (μη επίπεδη μορφή)

```
{
  "name" : "John Doe",
  "address" : {
     "street" : "123 Park Street",
     "city" : "Anytown",
     "state" : "NY"
   }
}
```

- Σε μια σχεσιακή ΒΔ, θα είχαμε δύο πλειάδες σε δύο διαφορετικούς πίνακες
- Αντίθετα, ο τρόπος αυτός είναι πιο «φυσική» αναπαράσταση της πληροφορίας

_id και ObjectId

- Κάθε έγγραφο στη MongoDB πρέπει να έχει ένα πεδίο: **_id**
  - Ο τύπος δεδομένων μπορεί να είναι οποιοσδήποτε
  - Από default είναι **ObjectId**
- Σε μια συλλογή, κάθε έγγραφο έχει μοναδική τιμή στο πεδίο **_id**
- **ObjectId**: *καθολικά* μοναδικό αναγνωριστικό (*ακόμη και μεταξύ μηχανημάτων*)
  - Auto-generated
  - Μέγεθος: 12 bytes (ή 24 ψηφία σε δεκαεξαδική αναπαράσταση)



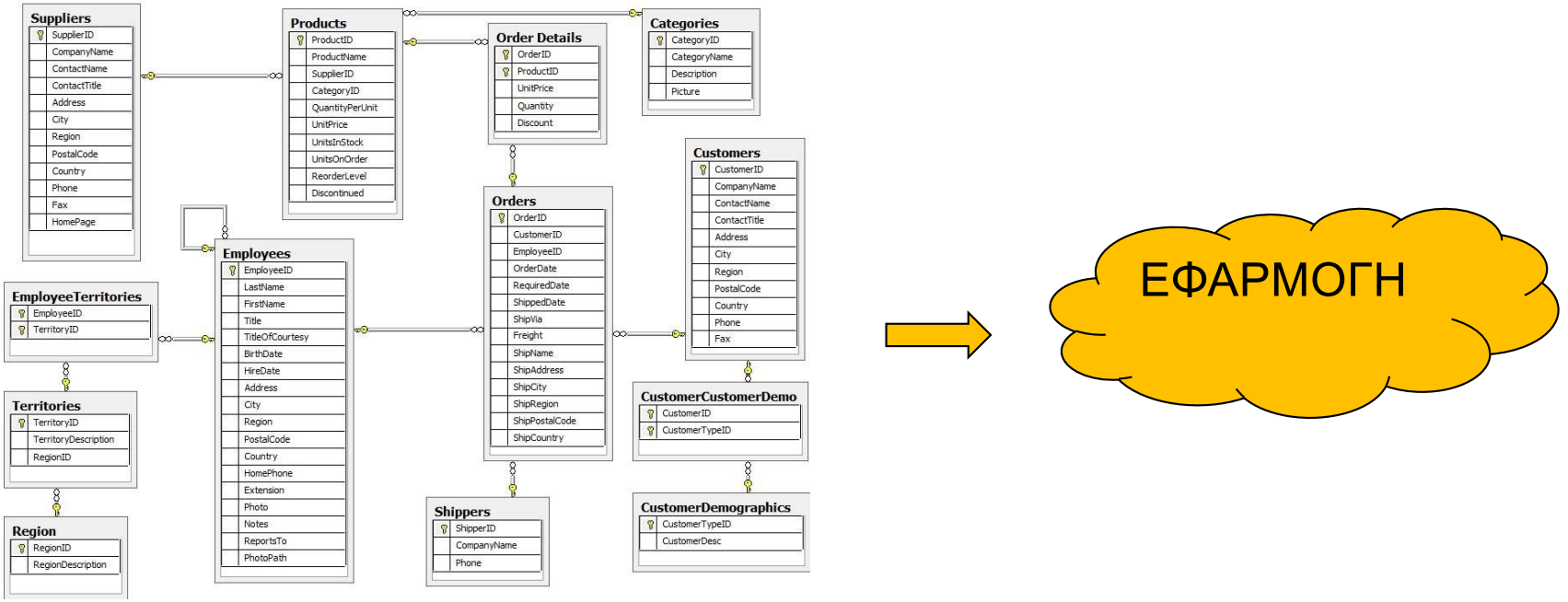
Ανασκόπηση

- Ένα έγγραφο (document) είναι η βασική μονάδα δεδομένων στην MongoDB
  - Είναι αντίστοιχο της έννοιας της πλειάδας σε σχεσιακές ΒΔ (όμως πιο εκφραστική αναπαράσταση)
- Ομοίως, μια συλλογή (collection) είναι αντίστοιχη της έννοιας του πίνακα/σχέσης σε μια σχεσιακή ΒΔ (όμως με δυναμικό σχήμα)
- Σε μια εγκατάσταση της MongoDB μπορούν να φιλοξενηθούν πολλές ανεξάρτητες ΒΔ, και καθεμιά να περιέχει τις συλλογές της
- Κάθε έγγραφο περιέχει ένα ειδικό πεδίο-κλειδί (_id) που είναι **μοναδικό** εντός της συλλογής

Μοντελοποίηση Δεδομένων στη MongoDB

Σχεδίαση σε Σχεσιακές ΒΔ

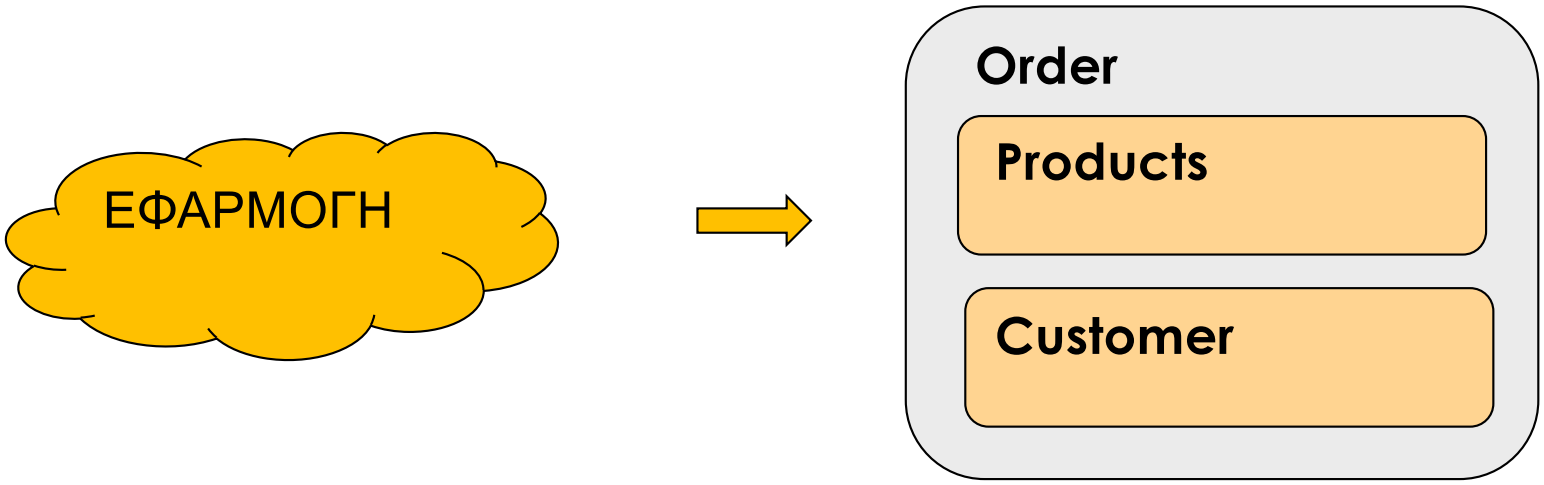
- Βήμα 1: Καθορισμός σχήματος (με κανονικοποίηση)
- Βήμα 2: Ανάπτυξη εφαρμογής



- Η εφαρμογή προσαρμόζεται στο σχήμα της ΒΔ

Σχεδίαση στη MongoDB

- Βήμα 1: Ανάπτυξη εφαρμογής
- Βήμα 2: Καθορισμός μοντέλου δεδομένων, βάσει αναγκών εφαρμογής



- Το μοντέλο δεδομένων **καθορίζεται** από την εφαρμογή

Παράδειγμα

Orders

OrderId	CustomerId	Date	ShipCity	ShipRegion
O1	C1	20/11/2020	Duckburg	Calisota

Customers

CustomerId	Name	Phone	Address
C1	Donald Duck	131-3131	1313 Webfoot Walk

Order Details

OrderId	ProductId	Quantity	UnitPrice
O1	P1	3	$ 3.99
O1	P2	1	$ 5.99

Παράδειγμα

Orders

OrderId	CustomerId	Date	ShipCity	ShipRegion
O1	C1	20/11/2020	Duckburg	Calisota

Customers

CustomerId	Name	Phone	Address
C1	Donald Duck	131-3131	1313 Webfoot Walk

Order Details

OrderId	ProductId	Quantity	UnitPrice
O1	P1	3	$ 3.99
O1	P2	1	$ 5.99

```
{
  OrderId : "O1",
  CustomerId : "C1",
  Date: "20/11/2020",
  ShipCity: "Duckburg",
  ShipRegion: "Calisota",
}
```

Παράδειγμα

Orders

OrderId	CustomerId	Date	ShipCity	ShipRegion
O1	C1	20/11/2020	Duckburg	Calisota

Customers

CustomerId	Name	Phone	Address
C1	Donald Duck	131-3131	1313 Webfoot Walk

Order Details

OrderId	ProductId	Quantity	UnitPrice
O1	P1	3	$ 3.99
O1	P2	1	$ 5.99

```
{
  OrderId : "O1",
  CustomerId : "C1",
  Date: "20/11/2020",
  ShipCity: "Duckburg",
  ShipRegion: "Calisota",
  OrderDetails: [“P1”, “P2”]
}
```

Παράδειγμα

Orders

OrderId	CustomerId	Date	ShipCity	ShipRegion
O1	C1	20/11/2020	Duckburg	Calisota

Customers

CustomerId	Name	Phone	Address
C1	Donald Duck	131-3131	1313 Webfoot Walk

Order Details

OrderId	ProductId	Quantity	UnitPrice
O1	P1	3	$ 3.99
O1	P2	1	$ 5.99

```
{
  OrderId : "O1",
  CustomerId : "C1",
  Date: "20/11/2020",
  ShipCity: "Duckburg",
  ShipRegion: "Calisota",
  OrderDetails: [
        {
          ProductId: "P1",
          Quantity: 3,
          UnitPrice: 3.99
        },
        {
          ProductId: "P2",
          Quantity: 1,
          UnitPrice: 5.99
        }
  ]
}
```

Παράδειγμα

Orders

OrderId	CustomerId	Date	ShipCity	ShipRegion
O1	C1	20/11/2020	Duckburg	Calisota

Customers

CustomerId	Name	Phone	Address
C1	Donald Duck	131-3131	1313 Webfoot Walk

Order Details

OrderId	ProductId	Quantity	UnitPrice
O1	P1	3	$ 3.99
O1	P2	1	$ 5.99

```
{
  OrderId : "O1",
  CustomerId : "C1",
  Date: "20/11/2020",
  ShipCity: "Duckburg",
  ShipRegion: "Calisota",
  Customer: {
    Name: “Donald Duck”,
    Phone: “131-3131”,
    Address: “1313 Webfoot Walk”
  }
}
```

Βασικά Σημεία

- Μεγάλη ευελιξία στη σχεδίαση του μοντέλου δεδομένων (=σχήματος δεδομένων)
- Δεν υπάρχει προτυποποιημένη μεθοδολογία
  - Σε αντίθεση με σχεσιακές ΒΔ, κανονικοποίηση
- *Βασικοί παράγοντες σχεδίασης*
  - Μέγεθος συνόλου δεδομένων
  - Ανάλυση/κατηγοριοποίηση αιτημάτων (query workload)
  - Απόδοση αιτημάτων (performance)
  - Αποθήκευση δεδομένων βάσει απαιτήσεων εφαρμογής

Εμφώλευση (Embedding)

- Ένα έγγραφο μπορεί να περιέχει στο εσωτερικό του άλλα έγγραφα (εμφώλευση)
- Ομοιότητες με σύζευξη (join)

Orders

OrderId	CustomerId	Date	ShipCity	ShipRegion
O1	C1	20/11/2020	Duckburg	Calisota

Order Details

OrderId	ProductId	Quantity	UnitPrice
O1	P1	3	$ 3.99
O1	P2	1	$ 5.99

```
{
  OrderId : "O1",
  CustomerId : "C1",
  Date: "20/11/2020",
  ShipCity: "Duckburg",
  ShipRegion: "Calisota",
  OrderDetails: [
        {
          ProductId: "P1",
          Quantity: 3,
          UnitPrice: 3.99
        },
        {
          ProductId: "P2",
          Quantity: 1,
          UnitPrice: 5.99
        }
  ]
}
```

Πλεονεκτήματα Εμφώλευσης

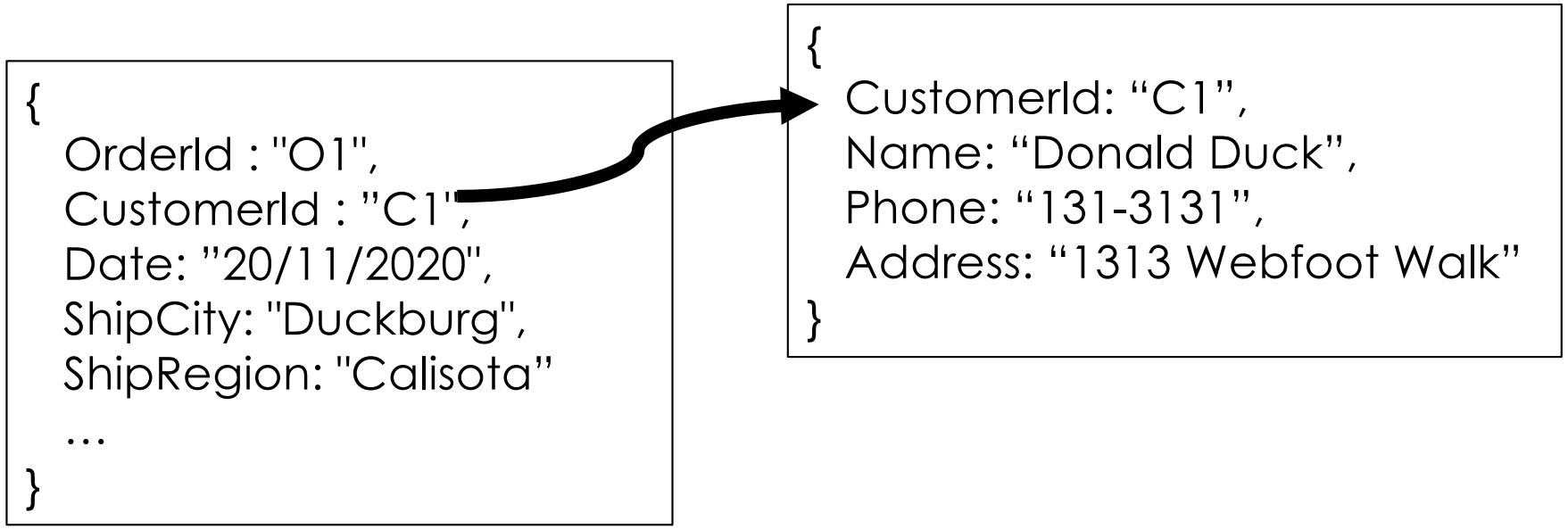
- Ταιριάζει καλύτερα στις ανάγκες των εφαρμογών
  - Με πρόσβαση στο ίδιο μπλοκ δίσκου ανακτώνται τα δεδομένα περισσότερων του ενός «πινάκων»
- Αποφεύγεται η ακριβή πράξη σύζευξης (join) ή το $lookup
- Η ενημέρωση ενός εγγράφου γίνεται με μια λειτουργία πρόσβασης στο δίσκο

Μειονεκτήματα Εμφώλευσης

- Δημιουργείται πλεονασμός στα δεδομένα
  - Επαφίεται στην εφαρμογή να διασφαλίσει τη συνέπεια (consistency)
- Τα έγγραφα έχουν μεγαλύτερο μέγεθος
  - Μπορεί να επιδρά αρνητικά στο χρόνο ανάκτησης
- Υπάρχει άνω όριο (16MB) στο μέγεθος ενός εγγράφου

Διασύνδεση (Linking/Referencing)

- Διασύνδεση εγγράφου με άλλο έγγραφο
  - Με χρήση μοναδικού αναγνωριστικού
  - Μοιάζει με το μηχανισμό αναφορικής ακεραιότητας (ξένου κλειδιού) σε σχεσιακές ΒΔ



Διασύνδεση (Linking/Referencing)

- Συσχετίσεις: 1-1, 1-N, M-N

```
Author = {
  "_id" : 1,
  "first" : "Rick",
  "last": "Catell",
  "paper": 1
}
```

```
Paper = {
  "_id": 1,
  "title": "Scalable SQL and NoSQL stores",
  "venue": "SIGMOD Record",
  "author": 1
}
```

Διασύνδεση (Linking/Referencing)

- Συσχετίσεις: 1-1, 1-N, M-N

```
Paper = {
   "_id": 1,
   "title": "A Survey on NoSQL Stores",
   "venue": “Communications ACM",
   "authors": [1, 2, 3]
}
```

```
Author = {
   "_id": 1,
   "first": "Ali",
   "last": "Davoudian"
}
```

```
Author = {
   "_id": 2,
   "first": "Liu",
   "last": "Chen"
}
```

```
Author = {
   "_id": 3,
   "first": "Mengchi",
   "last": "Liu"
}
```

Διασύνδεση (Linking/Referencing)

- Συσχετίσεις: 1-1, 1-N, M-N εναλλακτικά

```
Paper1 = {
   "_id": 1,
   "title": "A Survey on NoSQL Stores",
   "venue": "Communications ACM",
   "author": 1
}
```

```
Paper1 = {
   "_id": 2,
   "title": "A Survey on NoSQL Stores",
   "venue": "Communications ACM",
   "author": 2
}
```

```
Paper1 = {
   "_id": 3,
   "title": "A Survey on NoSQL Stores",
   "venue": "Communications ACM",
   "author": 3
}
```

Διασύνδεση (Linking/Referencing)

- Συσχετίσεις: 1-1, 1-N, M-N

```
Paper = {
   "_id": 1,
   "title": "A Survey on NoSQL Stores",
   "venue": "Communications ACM"
   "authors": [1, 27, 53]
}
```

```
Author = {
   "_id": 1,
   "first": "Ali",
   "last": "Davoudian",
   "papers": [1, 22, 43, 74, 155]
}
```

Πλεονεκτήματα Διασύνδεσης

- Μοιάζει περισσότερο με κανονικοποιημένη σχεδίαση
  - Δεν υπάρχει πλεονασμός στα δεδομένα
- Δεδομένα που δε χρησιμοποιούνται συχνά από την εφαρμογή, δε χρειάζεται να ανακτώνται σε κάθε αίτημα (query)
  - Για κάθε έγγραφο (αφού δεν είναι εμφωλευμένα)
- Τα έγγραφα είναι μικρότερα σε μέγεθος

Μειονεκτήματα Διασύνδεσης

- Απαιτείται η πρόσβαση σε διαφορετικά μπλοκ δίσκου (με χρήση του $lookup), για να ανακτήσουμε τα δεδομένα που χρειάζονται οι εφαρμογές
  - Άρα επιπτώσεις στην απόδοση(!)

Ζητήματα Σχεδίασης: 1-1

- Όταν η συσχέτιση είναι 1-1, τότε είναι εύκολο να γίνει εμφώλευση (embedding) των έξτρα πεδίων στο έγγραφο
  - Παράδειγμα: Παραγγελία – Πελάτης

```
{
  OrderId : "O1",
  CustomerId : "C1",
  Date: "20/11/2020",
  ShipCity: "Duckburg",
  ShipRegion: "Calisota",
  CustName: "Donald Duck",
  CustPhone: "131-3131",
  CustAddress: "1313 Webfoot Walk",
  …
}
```

*Ως **γενικός κανόνας**: προτιμάται η εμφώλευση, όταν αυτό δεν παραβιάζει κάποιο άλλο στόχο*

Ζητήματα Σχεδίασης: 1-Ν

- Όταν η συσχέτιση είναι 1-Ν, τότε πάλι μπορούμε να κάνουμε εμφώλευση
- Ειδικότερα, όταν τα δεδομένα που θα εμφωλευθούν είναι **λίγα σε πλήθος**, προτιμάμε την εμφώλευση

```
{
   OrderId : "O1",
   CustomerId : "C1",
   Date: "20/11/2020",
   ShipCity: "Duckburg",
   ShipRegion: "Calisota",
   OrderDetails: [
            {ProductId: "P1", Quantity: 3, UnitPrice: 3.99},
            {ProductId: "P2", Quantity: 1, UnitPrice: 5.99}
   ]
}
```

Ζητήματα Σχεδίασης: 1-Ν

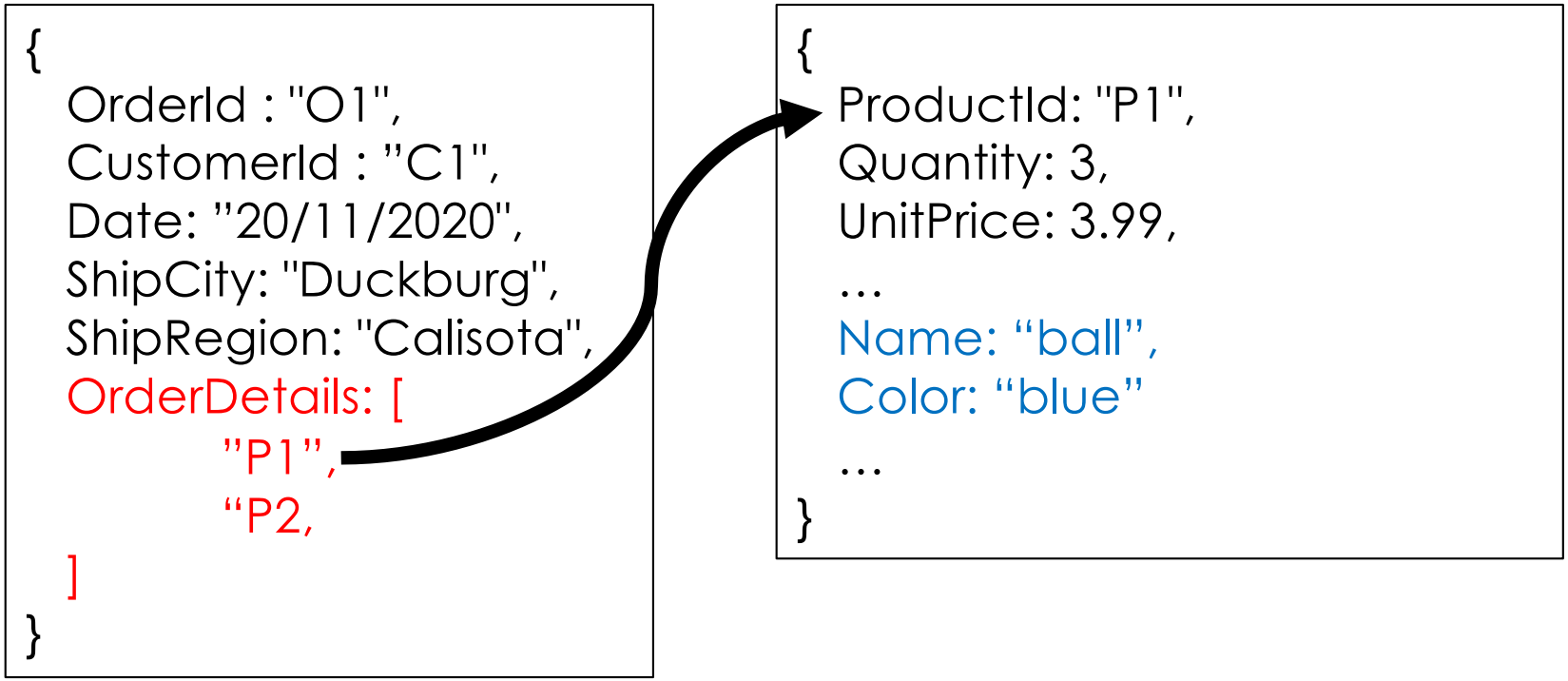
- Όταν η συσχέτιση είναι 1-Ν, τότε πάλι μπορούμε να κάνουμε εμφώλευση
- Ειδικότερα, όταν τα δεδομένα που θα εμφωλευθούν είναι **λίγα σε πλήθος**, προτιμάμε την εμφώλευση

```
{
   OrderId : "O1",
   CustomerId : "C1",
   Date: "20/11/2020",
   ShipCity: "Duckburg",
   ShipRegion: "Calisota",
   OrderDetails: [
            {ProductId: "P1", Quantity: 3, UnitPrice: 3.99},
            {ProductId: "P2", Quantity: 1, UnitPrice: 5.99}
   ]
}
```

Ένας λόγος να αποφύγουμε την εμφώλευση είναι εάν υπάρχουν ανάγκες αυτόνομης πρόσβασης στα εμφωλευμένα δεδομένα

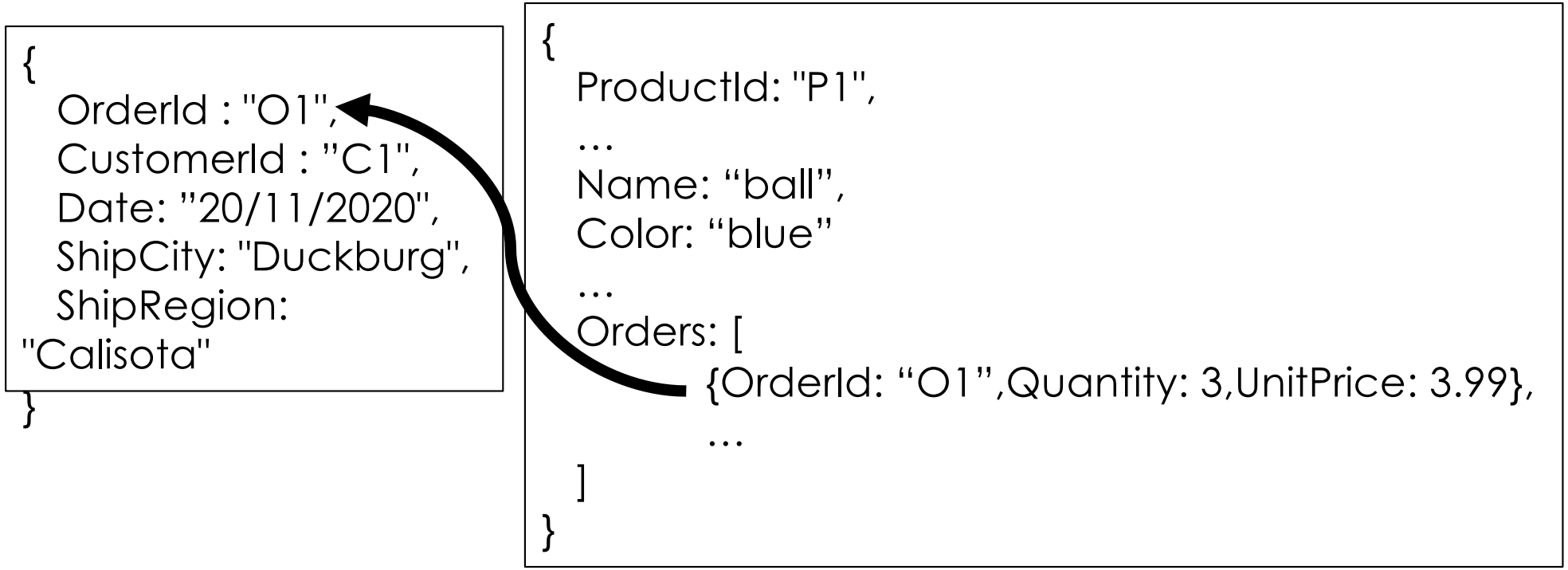
Ζητήματα Σχεδίασης: 1-N

- Όταν η συσχέτιση είναι 1-N, και τα δεδομένα που συσχετίζονται με το 1 **είναι αρκετά**, τότε πρέπει να αναλογιστούμε *εάν αξίζει να εμφωλευθεί* ολόκληρο το αντικείμενο



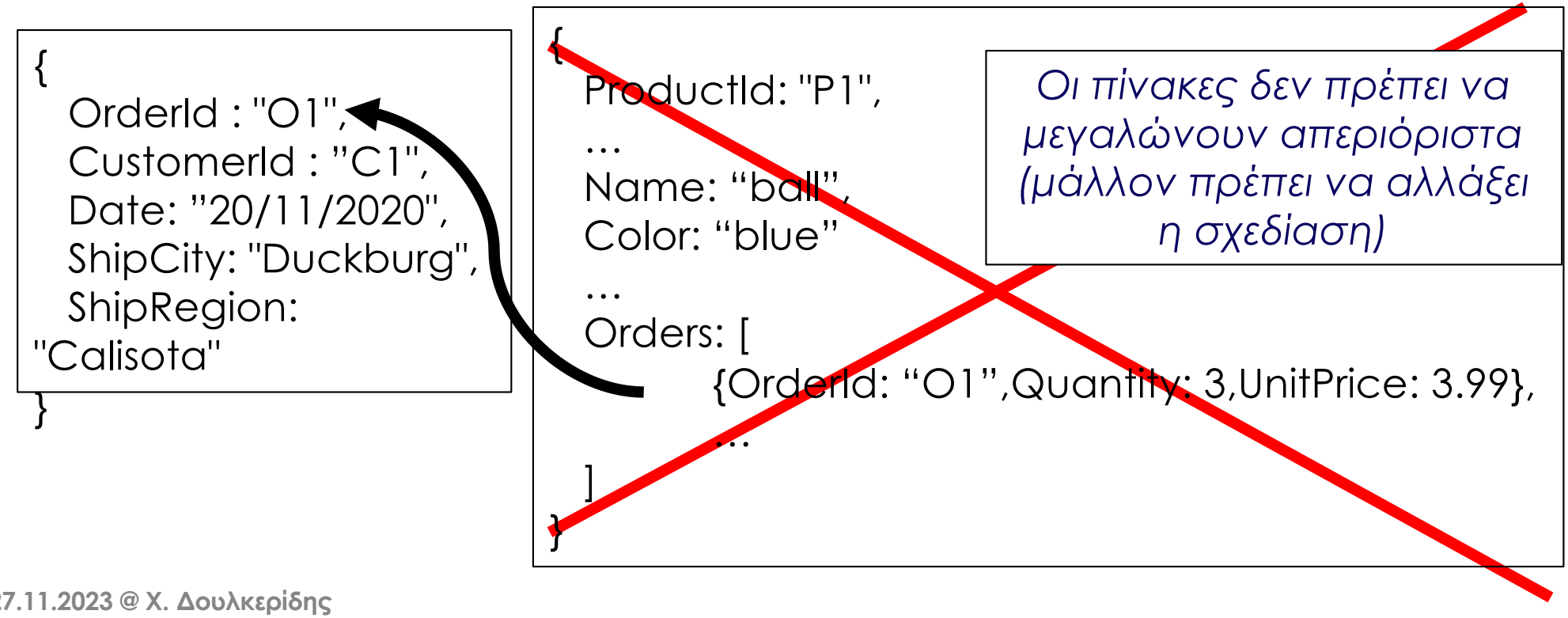
Ζητήματα Σχεδίασης: 1-Ν

- Όταν η συσχέτιση είναι 1-Ν, αλλά τα δεδομένα που θα εμφωλευθούν μπορεί να είναι πάρα πολλά και δυναμικά αυξανόμενα σε πλήθος, τότε θέλει προσοχή ποιο αντικείμενο εμφωλεύεται σε ποιο



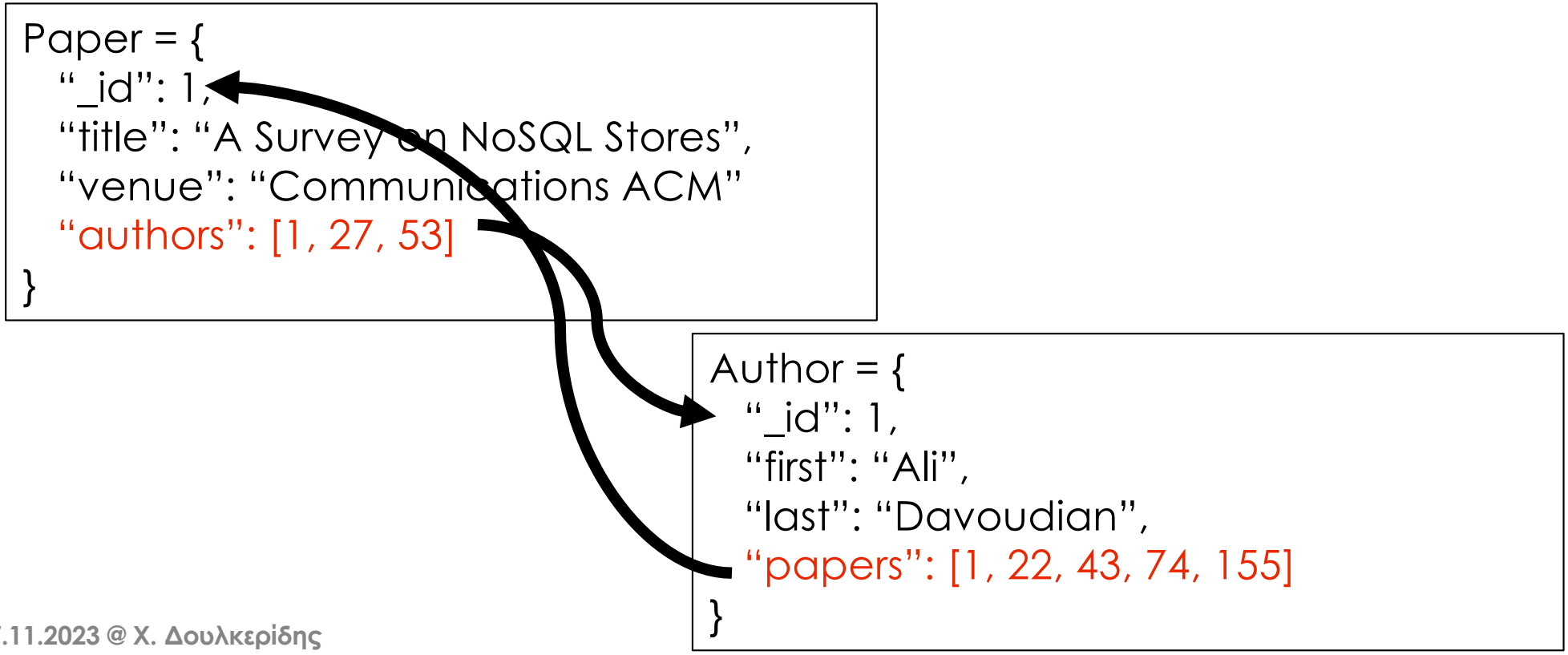
Ζητήματα Σχεδίασης: 1-Ν

- Όταν η συσχέτιση είναι 1-Ν, αλλά τα δεδομένα που θα εμφωλευθούν μπορεί να είναι **πάρα πολλά** και **δυναμικά αυξανόμενα σε πλήθος**, τότε θέλει προσοχή ποιο αντικείμενο εμφωλεύεται σε ποιο



Ζητήματα Σχεδίασης: Μ-Ν

- Όταν η συσχέτιση είναι Μ-Ν, τότε πρέπει να **χρησιμοποιηθεί διασύνδεση** (ή αναφορά) από και προς τα δύο αντικείμενα



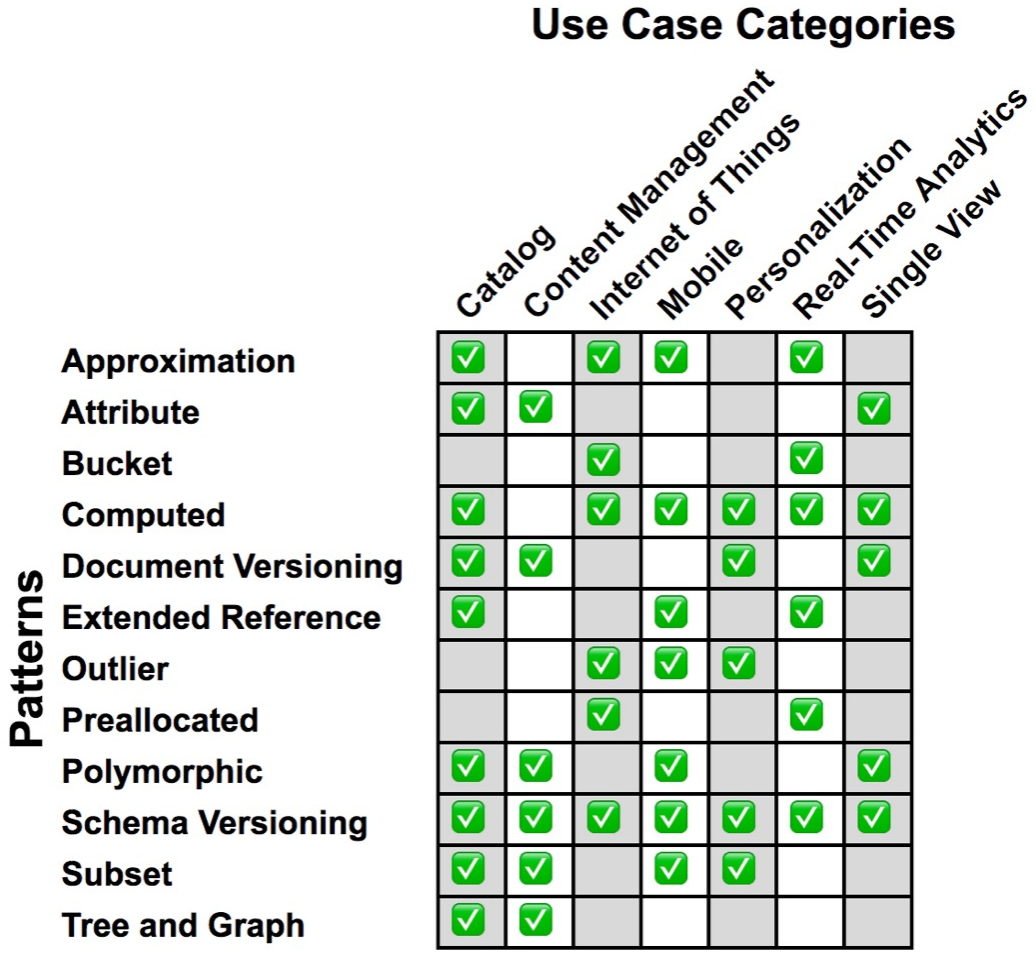
Βασικά Σημεία Σχέσεων στη MongoDB

- 1-1: προσθήκη πεδίων κλειδί-τιμή
- 1-N: εμφωλευμένο έγγραφο (ειδικά για λίγα αντικείμενα)
- 1-N: χρήση διασύνδεσης (για πολλά αντικείμενα)
- 1-N: αντίστροφη διασύνδεση (εάν το N αυξάνει δυναμικά)
- M-N: χρήση διασύνδεσης

Βασικά Σημεία Σχεδίασης (Κανόνες) στη MongoDB

- Γενικά, προτιμάται η εμφώλευση
- Αν υπάρχουν στόχοι που αντιβαίνουν στην εμφώλευση, τότε μπορεί να χρησιμοποιηθεί διασύνδεση
  - Για παράδειγμα, εάν κάποιο αντικείμενο απαιτείται να ανακτάται μόνο του από αιτήματα
- Αποφεύγονται τα joins και η χρήση $lookup, εκτός κι αν είναι απαραίτητο
- Προσοχή στο μέγεθος των εγγράφων
  - Να μην αυξάνεται χωρίς όριο
- Οι ανάγκες της εφαρμογής καθορίζουν τη σχεδίαση του σχήματος της ΒΔ

Πρότυπα/Μοτίβα Σχεδίασης Σχήματος (Schema Design Patterns)



Πηγή: https://www.mongodb.com/blog/post/building-with-patterns-a-summary

Bucket Pattern

Όταν έχουμε δεδομένα ροής (streaming data): IoT, sensors, positions of moving objects,…

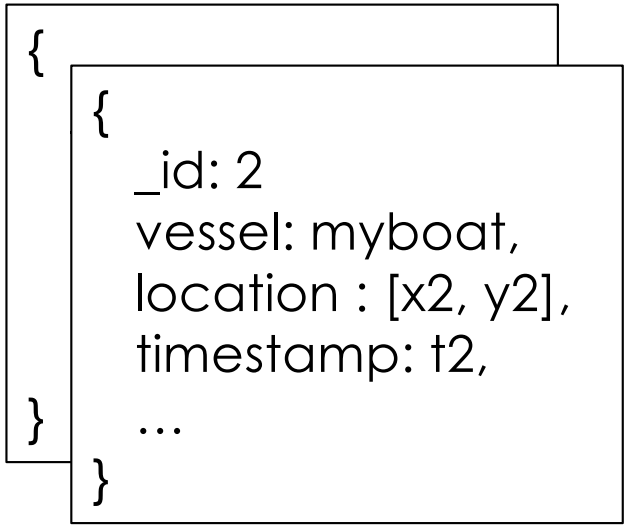
- Σε σχεσιακή ΒΔ, θα είχαμε μια πλειάδα για κάθε νέα μέτρηση
- Σε document-oriented ΒΔ, μπορούμε να παράγουμε ένα έγγραφο ανά αντικείμενο (sensor, moving object,…) ανά μονάδα χρόνου

```
{
   _id: 1
   vessel: myboat,
   location : [x1, y1],
   timestamp: t1,
   …
}
```

Bucket Pattern

Όταν έχουμε δεδομένα ροής (streaming data): IoT, sensors, positions of moving objects,…

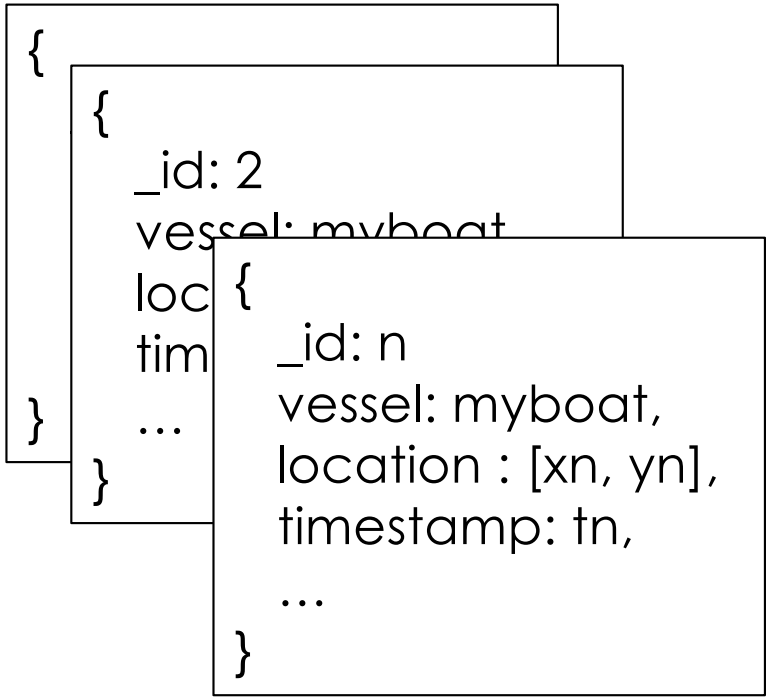
- Σε σχεσιακή ΒΔ, θα είχαμε μια πλειάδα για κάθε νέα μέτρηση
- Σε document-oriented ΒΔ, μπορούμε να παράγουμε ένα έγγραφο ανά αντικείμενο (sensor, moving object,…) ανά μονάδα χρόνου



Bucket Pattern

Όταν έχουμε δεδομένα ροής (streaming data): IoT, sensors, positions of moving objects,…

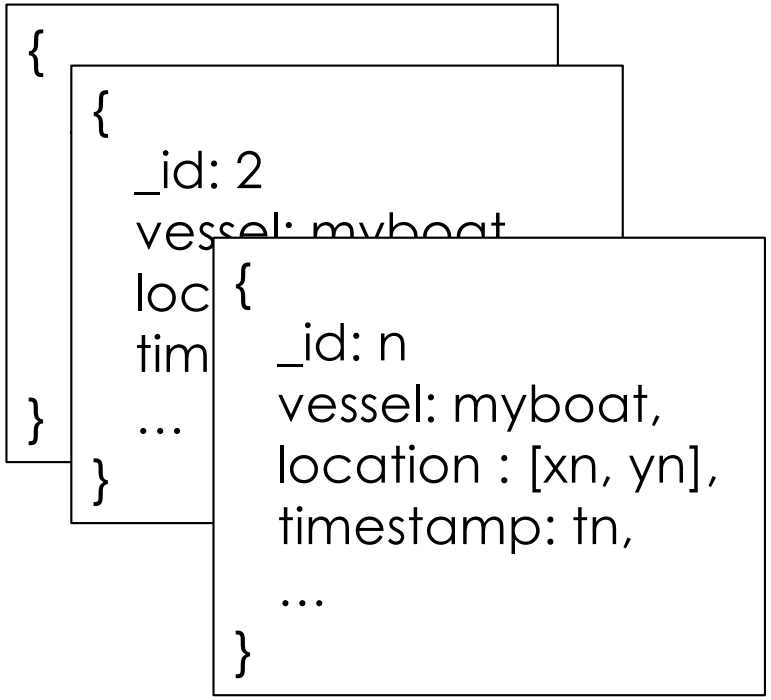
- Σε σχεσιακή ΒΔ, θα είχαμε μια πλειάδα για κάθε νέα μέτρηση
- Σε document-oriented ΒΔ, μπορούμε να παράγουμε ένα έγγραφο ανά αντικείμενο (sensor, moving object,…) ανά μονάδα χρόνου



Bucket Pattern

Όταν έχουμε δεδομένα ροής (streaming data): IoT, sensors, positions of moving objects,…

- Σε σχεσιακή ΒΔ, θα είχαμε μια πλειάδα για κάθε νέα μέτρηση
- Σε document-oriented ΒΔ, μπορούμε να παράγουμε ένα έγγραφο ανά αντικείμενο (sensor, moving object,…) ανά μονάδα χρόνου




```
{
  _id: 1
  vessel: myboat,
  startDate: t1,
  endDate: tn,
  positions: [
     {location : [x1, y1], timestamp: t1},
     {location : [x2, y2], timestamp: t2},
     …
     {location : [xn, yn], timestamp: tn},
  ],
  nrPositions: n
}
```

Bucket Pattern – Πλεονεκτήματα

- Μειώνει το πλήθος των εγγράφων που αποθηκεύονται
  - Μειώνεται το μέγεθος του ευρετηρίου
  - Άρα ταχύτερη ανάκτηση
- Το κάθε έγγραφο περιέχει σχετιζόμενη πληροφορία (ανάλογα με την εφαρμογή)

Computed Pattern

- Μπορούμε να προϋπολογίσουμε διάφορες ποσότητες, αντί να τις υπολογίζουμε όταν η εφαρμογή τις ζητάει
- Πλεονεκτήματα:
  - Δε χρειάζεται να υπολογίζεται ξανά και ξανά η ίδια ποσότητα
  - Σε εφαρμογές όπου τα reads είναι πολύ περισσότερα από τα writes, το κέρδος είναι πολύ μεγαλύτερο
  - Το κόστος ενημέρωσης είναι μικρό, καθώς ούτως ή άλλως ενημερώνεται το έγγραφο με τη νέα μέτρηση/θέση

```
{
  _id: 1
  vessel: myboat,
  startDate: t1,
  endDate: tn,
  positions: [
     {location : [x1, y1], timestamp: t1},
     {location : [x2, y2], timestamp: t2},
     …
     {location : [xn, yn], timestamp: tn},
  ],
  nrPositions: n,
  sumx: (x1+x2+…+xn),
  sumy: (y1+y2+…+yn)
}
```

Outlier Pattern

Μοντελοποίηση ενός
χρήστη στο Twitter

```
{
   _id: “cdoulk”,
   name: “cdoulk”,
   displayName : “Christos Doulkeridis”,
   numFollowers: 93,
   followers: [
            “TrackandKnow”,
            “dimosthenis777”,
            “noervaag”,
            …
   ]
}
```

Outlier Pattern

Μοντελοποίηση ενός λίγο πιο δημοφιλούς χρήστη στο Twitter

```
{
   _id: “Cristiano”,
   name: “Cristiano”,
   displayName: “Cristiano Ronaldo”,
   numFollowers: 89.2M,
   followers: [
              “fan1”,
              “fan2”,
              “fan3”,
              …
   ]
}
```

Outlier Pattern

Μοντελοποίηση ενός λίγο πιο δημοφιλούς χρήστη στο Twitter

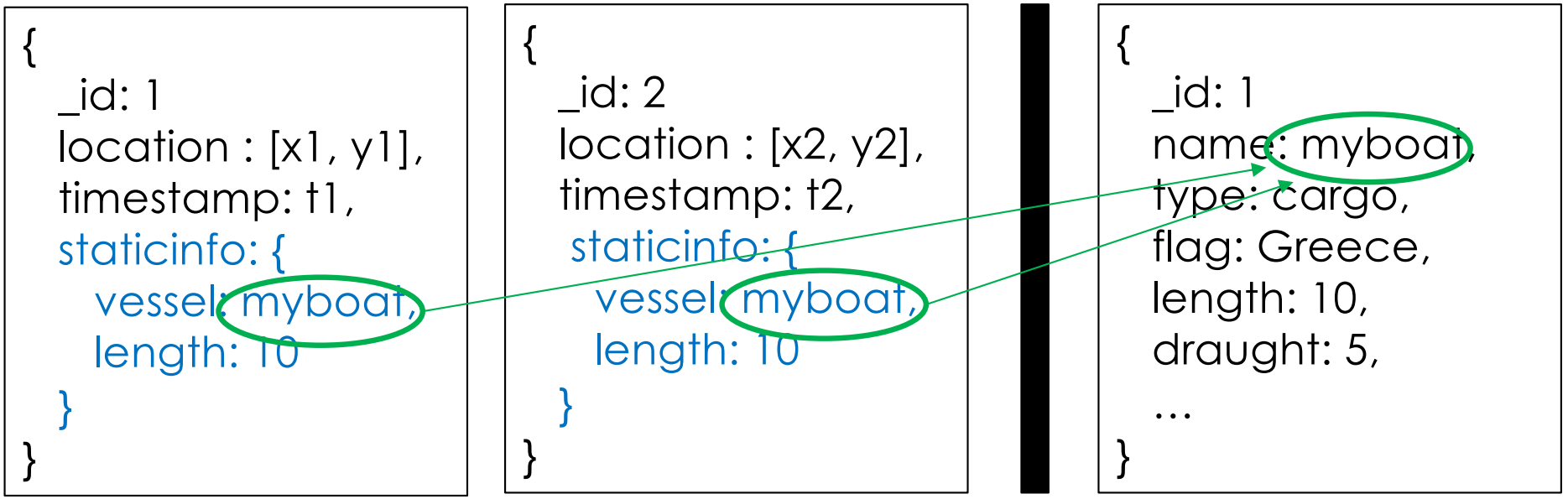
```
{
   _id: “Cristiano”,
   name: “Cristiano”,
   displayName: “Cristiano Ronaldo”,
   numFollowers: 89.2M,
   followers: [
            “fan1”,
            “fan2”,
            “fan3”,
            …
   ],
   has_extras: true
}
```

Έγγραφο υπερχείλισης (overflow document)

```
{
   _id: “Cristiano_1”,
   twitterId: “Cristiano”,
   isOverflow: true,
   followers: [
            “fan1001”,
            “fan1002”,
            “fan1003”,
            …
   ],
   has_extras: true
}
```

Extended Reference Pattern

- Αντιγραφή (εμφώλευση) *μόνο* των δεδομένων που απαιτούν από κοινού πρόσβαση
  - Αντί για αντιγραφή όλων των δεδομένων
  - Γεγονός που θα είχε ως συνέπεια υψηλό κόστος για συχνές ενημερώσεις



Indexing (Ευρετηρίαση)

Βασικές Έννοιες

- Τα βασικά σημεία που αφορούν την ευρετηρίαση είναι κοινά με τις σχεσιακές ΒΔ
  - Πότε χτίζουμε ένα ευρετήριο;
    - Επιτάχυνση των reads, επιβράδυνση των writes
  - Σε ποιο/α πεδίο/α χτίζουμε ευρετήρια;
    - Σε πεδία που χρησιμοποιούνται στην πλειονότητα των queries στο αναμενόμενο query workload
    - Σε πεδία που χρησιμοποιούνται σε queries που είναι κρίσιμο να απαντώνται γρήγορα

Compound Indexes

- Ένα compound index είναι ένα ευρετήριο σε περισσότερα από ένα πεδία
  - Η σειρά των πεδίων είναι κρίσιμη
  - Παράδειγμα:
    - db.users.ensureIndex({"age" : 1, "username" : 1})
  - Αιτήματα που χρησιμοποιούν το age μπορούν να αξιοποιήσουν το ευρετήριο
  - Αιτήματα που χρησιμοποιούν μόνο το username **δεν** μπορούν να αξιοποιήσουν το ευρετήριο
  - Τα δεδομένα επιστρέφονται με τη σειρά που είναι αποθηκευμένα στο ευρετήριο

```
[0, "user100309"] -> 0x0c965148
[0, "user100334"] -> 0xf51f818e
[0, "user100479"] -> 0x00fd7934
...
[0, "user99985" ] -> 0xd246648f
[1, "user100156"] -> 0xf78d5bdd
[1, "user100187"] -> 0x68ab28bd
[1, "user100192"] -> 0x5c7fb621
...
[1, "user999920"] -> 0x67ded4b7
[2, "user100141"] -> 0x3996dd46
[2, "user100149"] -> 0xfce68412
[2, "user100223"] -> 0x91106e23
```

Η Σειρά των Πεδίων στο Compound Index

- Όταν χτίζουμε ένα compound index σε πεδία {Α, Β, Γ, Δ}, έχουμε και τα ευρετήρια (implicit indexes) εμμέσως: {Α}, {Α,Β}, {Α,Β,Γ}
- Μεταξύ δύο πεδίων προτιμούμε πρώτο το πεδίο στο οποίο γίνεται ακριβές ταίριασμα (exact matching) αντί για πεδίο στο οποίο γίνονται αιτήματα διαστήματος τιμών
  - db.users.find(

    {"age" : 47, "username" : {"$gt" : "user5", "$lt" : "user8"}}

).explain()

Τελεστές που δεν Αξιοποιούν Ευρετήρια

- Ο έλεγχος ύπαρξης κάποιου πεδίου **δεν** μπορεί να χρησιμοποιήσει ευρετήριο, και αναγκάζεται να σαρώσει όλο το collection
  - {"key" : {"$exists" : true}}
- Το διάφορο ($ne) δεν είναι αποδοτικός τελεστής, καθώς δεν μπορεί να χρησιμοποιήσει καλά το ευρετήριο
- Η άρνηση ($not) μερικές φορές μπορεί να αξιοποιήσει το ευρετήριο, αλλά πολύ συχνά όχι
- Ο έλεγχος μη-ύπαρξης τιμής ($nin) πάντα κάνει σειριακή σάρωση
- Η διάζευξη ($or) μπορεί να αξιοποιήσει δύο ευρετήρια, αλλά κατόπιν πρέπει να κάνει συγχώνευση αποτελεσμάτων
  - Πάντα προτιμάται το $in από το $or

Πληθικότητα Τιμών Πεδίου

- Προτιμάται πεδίο που έχει πολλές διακριτές τιμές, από πεδίο με λίγες τιμές
  - Διότι μπορεί να περιορίσει γρηγορότερα την αναζήτηση
  - Παράδειγμα:
    - Email: διαφορετική τιμή για κάθε document
    - Gender: μόνο δύο διακριτές τιμές
- Επομένως, προτιμούμε να χτίζουμε ευρετήρια σε πεδία με πολλές διακριτές τιμές
- Σε compound indexes, προτιμούμε να βάζουμε πρώτα τα πεδία με πολλές διακριτές τιμές

Βελτιστοποίηση

- Ο query optimizer (βελτιστοποιητής) της MongoDB λειτουργεί διαφορετικά από ότι σε μια παραδοσιακή ΒΔ
  - Έστω αίτημα για το πεδίο Α
  - Έστω ότι υπάρχει ευρετήριο στο πεδίο Α
  - Ο βελτιστοποιητής **συνήθως** χρησιμοποιεί το ευρετήριο
  - Αν υπάρχουν και άλλα ευρετήρια, ο βελτιστοποιητής εκτελεί το αίτημα με χρήση αυτών (παράλληλα), και το πρώτο που επιστρέφει 100 αποτελέσματα είναι αυτό που θα χρησιμοποιηθεί
    - Και θα χρησιμοποιείται στο εξής – cached – για το συγκεκριμένο αίτημα, μέχρι να αλλάξει η συλλογή αρκετά ή αφού εκτελεστούν 1.000 αιτήματα

Τύποι Ευρετηρίων: Unique

- Εγγυώνται ότι κάθε τιμή κλειδιού στο στιγμιότυπο της συλλογής εμφανίζεται το πολύ μια φορά στο ευρετήριο
- Παράδειγμα:
  - Το ευρετήριο στο πεδίο _id
  - db.users.ensureIndex({"username" : 1}, {"unique" : true})
- Και το null θεωρείται τιμή → μόνο ένα έγγραφο με τιμή null μπορεί να υπάρχει

Τύποι Ευρετηρίων: Sparse

- Όταν θέλουμε ο περιορισμός μοναδικότητας να εφαρμοστεί ακόμη και εάν πολλά έγγραφα δεν έχουν τιμή στο πεδίο
- Παράδειγμα:
  - db.ensureIndex({"email" : 1}, {"unique" : true, "sparse" : true})
  - Το πεδίο email είναι προαιρετικό, όμως όταν παίρνει τιμή, αυτή είναι μοναδική
- Τα sparse ευρετήρια δεν είναι απαραίτητο να είναι unique
  - Αφαιρώντας το: "unique" : true
- Το ίδιο αίτημα (query) μπορεί να επιστρέψει διαφορετικά αποτελέσματα ανάλογα με το αν χρησιμοποιεί ένα sparse ευρετήριο
  - Αυτό συμβαίνει διότι ένα έγγραφο χωρίς καθορισμένη τιμή πεδίου στο sparse index δεν εισάγεται στο ευρετήριο

Partitioning/Sharding (Κατάτμηση)

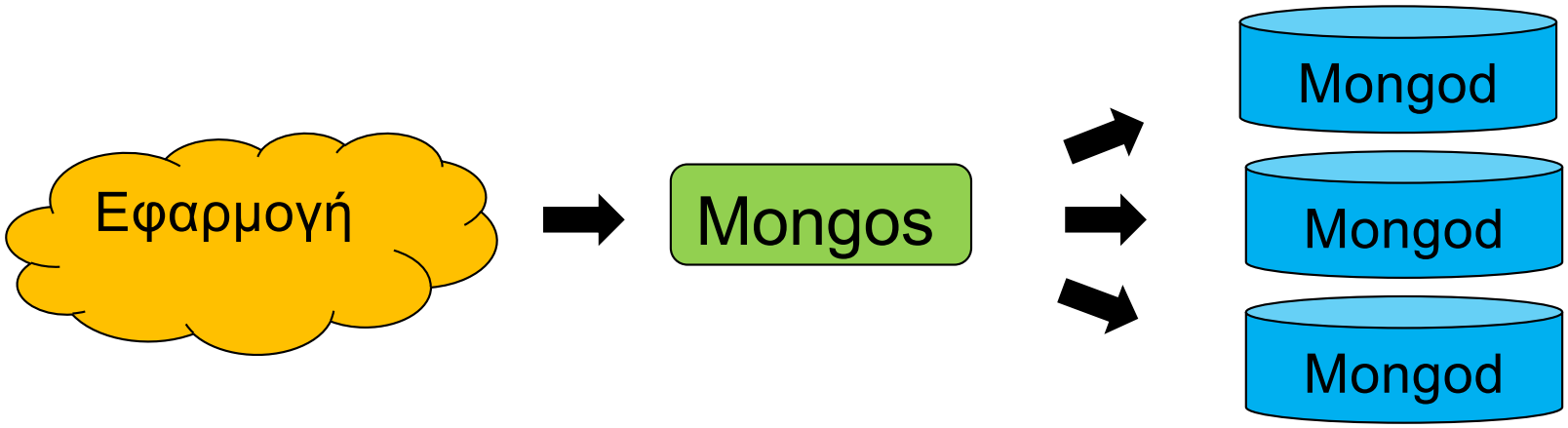
Βασικές Έννοιες

- Το sharding (κατάτμηση) χρησιμοποιείται για να αναφερθούμε στη διαδικασία διαχωρισμού των δεδομένων σε μηχανήματα
  - Γνωστό και ως partitioning
- Με αυτόν τον τρόπο μπορούμε
  - Να αποθηκεύουμε περισσότερα δεδομένα
  - Να διαχειριζόμαστε υψηλότερο φόρτο

    δίχως να απαιτούνται μηχανήματα υψηλότερων προδιαγραφών (vertical vs. horizontal scaling)

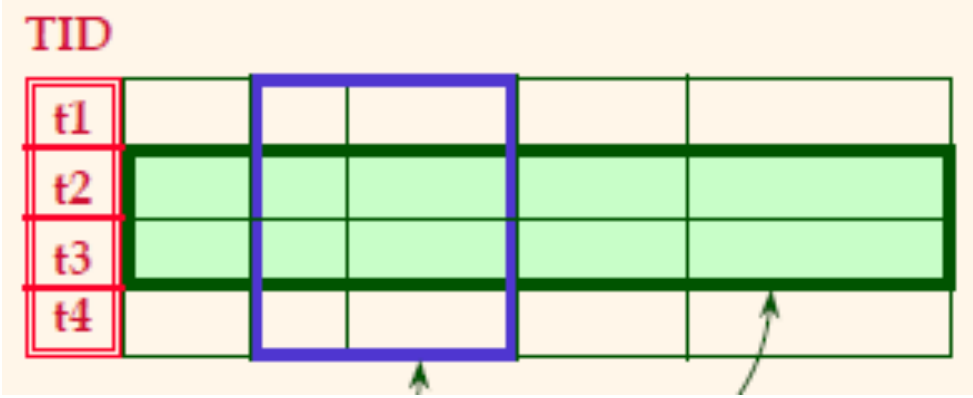
Βασικές Έννοιες

- Ένας στόχος του sharding είναι να αποκρύψει από την εφαρμογή την ύπαρξη πολλών μηχανημάτων
  - Η εφαρμογή είναι σαν να «βλέπει» ένα μηχάνημα
- Για το σκοπό αυτό χρησιμοποιείται μια διαδικασία δρομολόγησης (mongos) «μπροστά» από τα shards



Κατάτμηση σε Κατανεμημένες ΒΔ

- Ο διαχωρισμός μιας σχέσης σε μικρότερες σχέσεις ή τεμάχια και η αποθήκευσή τους (αντί της ίδιας της σχέσης) σε διαφορετικές τοποθεσίες
  - Οριζόντια κατάτμηση
    - Κάθε τεμάχιο απαρτίζεται από ένα υποσύνολο γραμμών της αρχικής σχέσης (αποτέλεσμα της πράξης *επιλογής*)
  - Κατακόρυφη κατάτμηση
    - Κάθε τεμάχιο απαρτίζεται από ένα υποσύνολο στηλών της αρχικής σχέσης (αποτέλεσμα της πράξης *προβολής*)



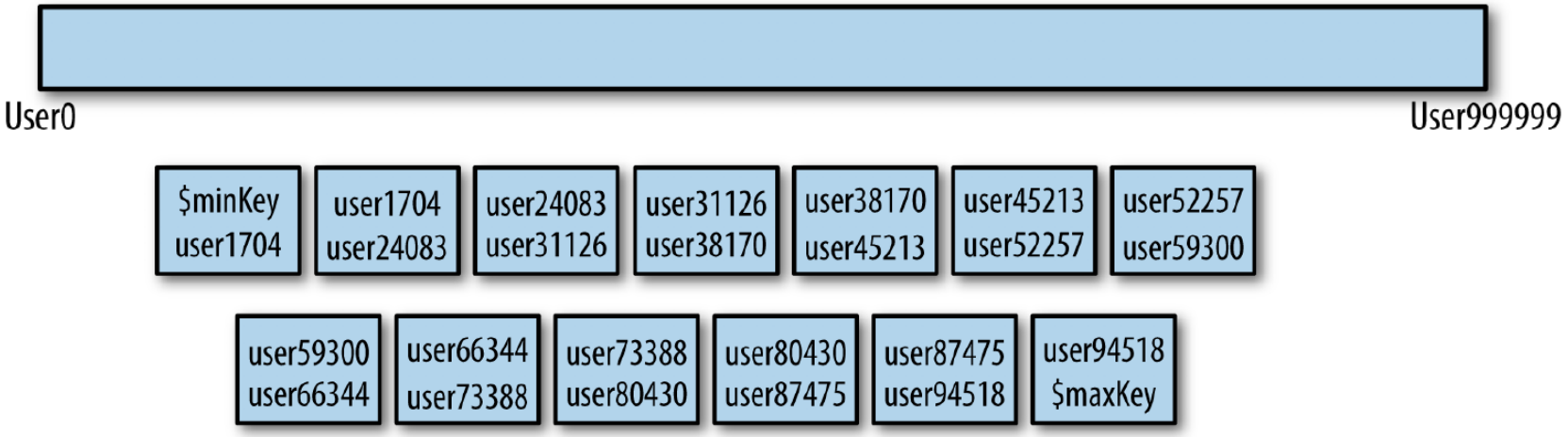
Κλειδί Κατάτμησης (Shard Key)

- Η MongoDB υιοθετεί οριζόντια κατάτμηση
- Επιλέγεται πεδίο πάνω στο οποίο θα γίνει το sharding
  - Η επιλογή του πεδίου είναι πολύ σημαντική
    - Καθορίζει το πώς ακριβώς θα μοιραστούν τα δεδομένα στα μηχανήματα
    - Παράδειγμα: username → [aaa-def), [def-ghj), …
- Το πεδίο αυτό πρέπει να έχει ευρετήριο

Παράδειγμα

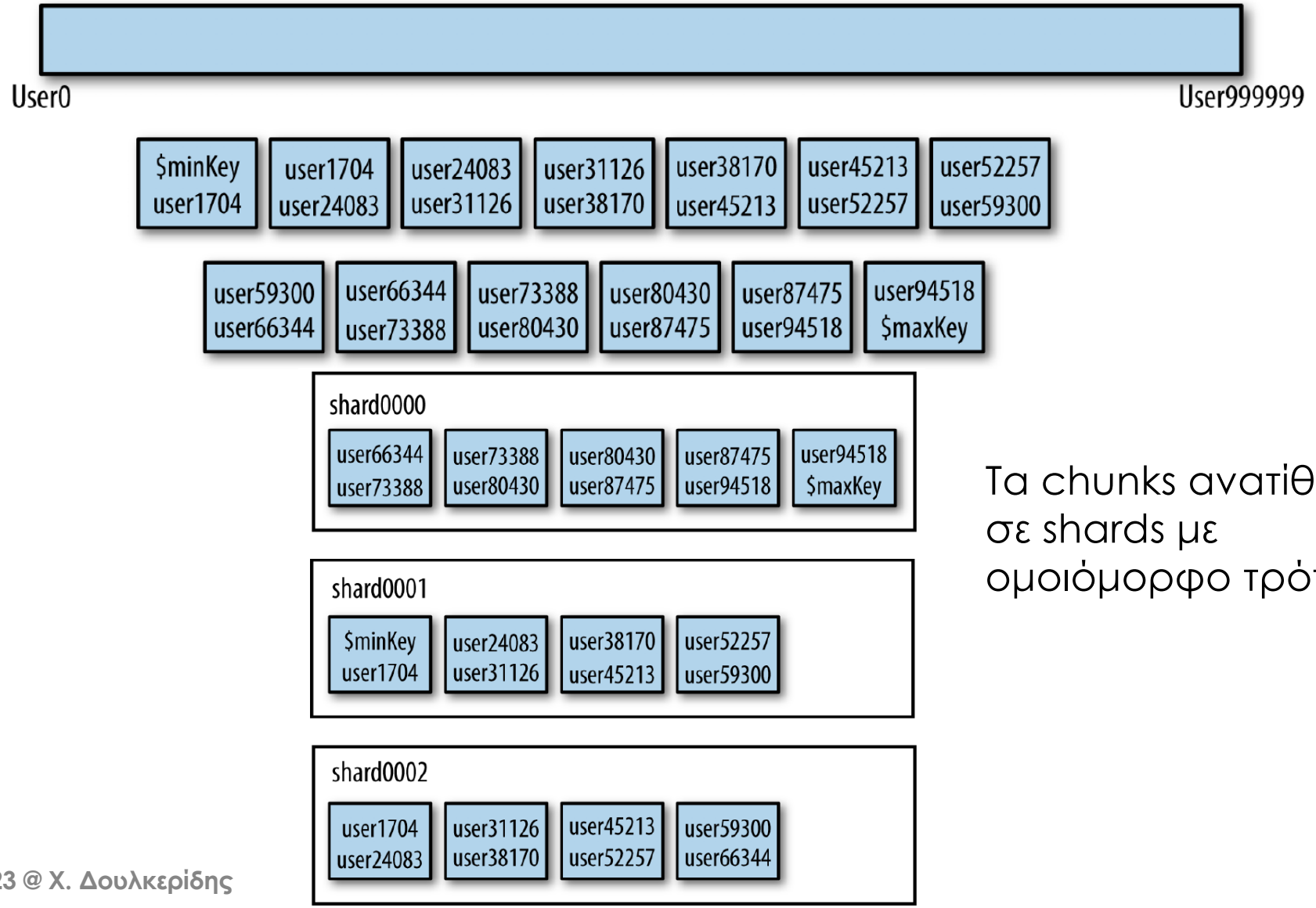
Ολόκληρο το collection (ως ένα ***chunk***), εμφανίζεται το shard key

Παράδειγμα



Το collection διασπαται σε ***chunks*** με βάση το shard key

Παράδειγμα



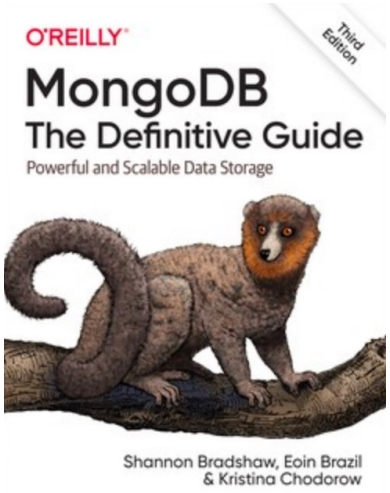
Τα chunks ανατίθενται σε shards με ομοιόμορφο τρόπο

Επεξεργασία Αιτημάτων

- Αιτήματα που επιβάλουν φίλτρο στο shard key
  - Θα κατευθυνθούν μόνο στο(α) shard(s) που περιέχουν τα αντίστοιχα δεδομένα
  - Τέτοια αιτήματα ονομάζονται: targeted queries
- Εάν το αίτημα δεν περιέχει το shard key
  - Τότε θα κατευθυνθεί σε όλα τα shards
  - Τέτοια αιτήματα ονομάζονται: scatter-gather queries

Βιβλιογραφικές Αναφορές

- Shannon Bradshaw, Eoin Brazil, Kristina Chodorow. **MongoDB: The Definitive Guide, 3rd Edition.** O'Reilly Media (ISBN: 9781491954461), December 2019



- **https://www.mongodb.com/**
- **https://docs.mongodb.com/manual/**
- **https://www.mongodb.com/blog**
- **https://developer.mongodb.com/**