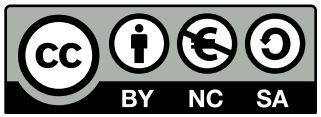
Veri Tabanı Sistemleri

NoSQL

H. Turgut Uyar Şule Öğüdücü

2005-2016

License



You are free to:

- Share – copy and redistribute the material in any medium or format
- Adapt – remix, transform, and build upon the material

Under the following terms:

- Attribution – You must give appropriate credit, provide a link to the license, and indicate if changes were made.
- NonCommercial – You may not use the material for commercial purposes.
- ShareAlike – If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.

For more information:
`https://creativecommons.org/licenses/by-nc-sa/4.0/`

Read the full license:
`https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode`

Konular

Konular

Bağıntı Modeli

- bağıntı modeli her türlü problem için en iyi çözüm olmayabilir

- kullanıcı tercihlerini saklama
- Wikipedia sayfalarındaki veriyi işleme
- sosyal ağ oluşturma

Örnek: Kullanıcı Tercihleri

- kullanıcı, tercih tipi, seçilen seçenek
- örnek iş:
  belirli bir kullanıcının bildirim ayarını sorgulamak
- SQL kullanıldığında yapılacak karmaşık sorgulamalara gerek yok

Örnek: Kullanıcı Tercihleri

- kullanıcı, tercih tipi, seçilen seçenek
- örnek iş:
  belirli bir kullanıcının bildirim ayarını sorgulamak
- SQL kullanıldığında yapılacak karmaşık sorgulamalara gerek yok

Örnek: Wikipedia Sayfaları

Casino Royale (2006 film)

From Wikipedia, the free encyclopedia

This article is about the 2006 film. For the 1967 film, see Casino Royale (1967 film). For other uses, see Casino Royale (disambiguation).

Casino Royale (2006) is the twenty-first film in the Eon Productions *James Bond* film series and the first to star Daniel Craig as the fictional MI6 agent James Bond. Directed by Martin Campbell and written by Neal Purvis & Robert Wade and Paul Haggis, the film marks the third screen adaptation of Ian Fleming's 1953 novel of the same name. *Casino Royale* is set at the beginning of Bond's career as Agent 007, just as he is earning his licence to kill. After preventing a terrorist attack at Miami International Airport, Bond falls in love with Vesper Lynd, the treasury employee assigned to provide the money he needs to bankrupt a terrorist financier, Le Chiffre, by beating him in a high-stakes poker game. The story arc continues in the following *Bond* film, *Quantum of Solace* (2008), with explicit references to characters and events in *Spectre* (2015).

Casino Royale reboots the series, establishing a new timeline and narrative framework not meant to precede or succeed any previous *Bond* film,[3][4] which allows the film to show a less experienced and more vulnerable Bond.[5] Additionally, the character Miss Moneypenny is, for the first time in the series, completely absent.[6] Casting the film involved a widespread search for a new actor to portray James Bond, and significant controversy surrounded Craig when he was selected to succeed Pierce Brosnan in October 2005. Location filming took place in the Czech Republic, the Bahamas, Italy and the United Kingdom with interior sets built at Pinewood Studios. Although part of the storyline is set in

Casino Royale

British cinema poster for *Casino Royale*, designed by Empire Design

Directed by	Martin Campbell
Produced by	Michael G. Wilson Barbara Broccoli
Screenplay by	Neal Purvis Robert Wade Paul Haggis
Based on	*Casino Royale* by Ian Fleming
Starring	Daniel Craig Eva Green Mads Mikkelsen Giancarlo Giannini Jeffrey Wright Judi Dench

- yapısal ve yapısal olmayan verinin birleşimi
- örnek iş:
  Daniel Craig'in rol aldığı bütün James Bond filmlerinin
  ilk paragrafı
- bağıntı olarak temsil etmek zor

Örnek: Wikipedia Sayfaları

Casino Royale (2006 film)

From Wikipedia, the free encyclopedia

This article is about the 2006 film. For the 1967 film, see Casino Royale (1967 film). For other uses, see Casino Royale (disambiguation).

Casino Royale (2006) is the twenty-first film in the Eon Productions *James Bond* film series and the first to star Daniel Craig as the fictional MI6 agent James Bond. Directed by Martin Campbell and written by Neal Purvis & Robert Wade and Paul Haggis, the film marks the third screen adaptation of Ian Fleming's 1953 novel of the same name. *Casino Royale* is set at the beginning of Bond's career as Agent 007, just as he is earning his licence to kill. After preventing a terrorist attack at Miami International Airport, Bond falls in love with Vesper Lynd, the treasury employee assigned to provide the money he needs to bankrupt a terrorist financier, Le Chiffre, by beating him in a high-stakes poker game. The story arc continues in the following *Bond* film, *Quantum of Solace* (2008), with explicit references to characters and events in *Spectre* (2015).

Casino Royale reboots the series, establishing a new timeline and narrative framework not meant to precede or succeed any previous *Bond* film,[3][4] which allows the film to show a less experienced and more vulnerable Bond.[5] Additionally, the character Miss Moneypenny is, for the first time in the series, completely absent.[6] Casting the film involved a widespread search for a new actor to portray James Bond, and significant controversy surrounded Craig when he was selected to succeed Pierce Brosnan in October 2005. Location filming took place in the Czech Republic, the Bahamas, Italy and the United Kingdom with interior sets built at Pinewood Studios. Although part of the storyline is set in

Casino Royale

British cinema poster for *Casino Royale*, designed by Empire Design

Directed by	Martin Campbell
Produced by	Michael G. Wilson Barbara Broccoli
Screenplay by	Neal Purvis Robert Wade Paul Haggis
Based on	*Casino Royale* by Ian Fleming
Starring	Daniel Craig Eva Green Mads Mikkelsen Giancarlo Giannini Jeffrey Wright Judi Dench

- yapısal ve yapısal olmayan verinin birleşimi
- örnek iş:
  Daniel Craig'in rol aldığı bütün James Bond filmlerinin
  ilk paragrafı
- bağıntı olarak temsil etmek zor

Örnek: Sosyal Ağ

- kullanıcılar: ID, isim, yaş, cinsiyet, ...
- arkadaşlar: ID1, ID2
- örnek iş:
  bir kullanıcının bütün arkadaşlarını bul
  bir kullanıcının bütün arkadaşlarının arkadaşlarını bul
  bir kullanıcının erkek arkadaşlarının kadın arkadaşlarını bul
  bir kullanıcının arkadaşlarının arkadaşlarının ... arkadaşlarını bul
- çok fazla sayıda karmaşık katma işlemi

Örnek: Sosyal Ağ

- kullanıcılar: ID, isim, yaş, cinsiyet, ...
- arkadaşlar: ID1, ID2
- örnek iş:
  bir kullanıcının bütün arkadaşlarını bul
  bir kullanıcının bütün arkadaşlarının arkadaşlarını bul
  bir kullanıcının erkek arkadaşlarının kadın arkadaşlarını bul
  bir kullanıcının arkadaşlarının arkadaşlarının ... arkadaşlarını bul
- çok fazla sayıda karmaşık katma işlemi

Problemler: Temsil

- yapısal ve yarı yapısal veriyi işlemek zor
- hiyerarşi ve yakınlığı temsil etmek zor
- katı şema: her satırda her sütun için değer olmalı
- uygulanamaz olsa bile
- önceden sabitlenmiş
- değişiklik yapmak için: kapat, tabloyu değiştir, yeniden başlat

Problemler: Temsil

- yapısal ve yarı yapısal veriyi işlemek zor
- hiyerarşi ve yakınlığı temsil etmek zor
- katı şema: her satırda her sütun için değer olmalı
- uygulanamaz olsa bile
- önceden sabitlenmiş
- değişiklik yapmak için: kapat, tabloyu değiştir, yeniden başlat

Problemler: Ölçekleme

- veri miktarı artınca:
- büyütme (scale up): daha hızlı işlemci
- bir yere kadar çalışır
- genişletme (scale out): daha fazla işlemci
- sıradan donanım

Problemler: Ölçekleme

- veri miktarı artınca:
- büyütme (scale up): daha hızlı işlemci
- bir yere kadar çalışır
- genişletme (scale out): daha fazla işlemci
- sıradan donanım

NoSQL

- NoSQL $\neq$ "SQL kullanma"
- Not Only SQL
- bazı parçalar için bağıntı diğer parçalarda farklı veri yapıları
- anahtar-değer depoları
- sütun ailesi depoları
- belge depoları
- çizge veri tabanları

NoSQL

- NoSQL $\neq$ "SQL kullanma"
- Not Only SQL
- bazı parçalar için bağıntı diğer parçalarda farklı veri yapıları
- anahtar-değer depoları
- sütun ailesi depoları
- belge depoları
- çizge veri tabanları

NoSQL İlkeleri

- esnek şemalar
- performans odaklı
- katma işlemi yok
- büyük kapsamlı ölçeklenebilirlik
- ulaşılabilirliğe odaklı
- güncellemelere her zaman izin verilir

NoSQL İlkeleri

- esnek şemalar
- performans odaklı
- katma işlemi yok
- büyük kapsamlı ölçeklenebilirlik
- ulaşılabilirliğe odaklı
- güncellemelere her zaman izin verilir

NoSQL İlkeleri

- esnek şemalar
- performans odaklı
- katma işlemi yok
- büyük kapsamlı ölçeklenebilirlik
- ulaşılabilirliğe odaklı
- güncellemelere her zaman izin verilir

NoSQL İlkeleri

- esnek şemalar
- performans odaklı
- katma işlemi yok
- büyük kapsamlı ölçeklenebilirlik
- ulaşılabilirliğe odaklı
- güncellemelere her zaman izin verilir

Ulaşılabilirlik / Tutarlılık

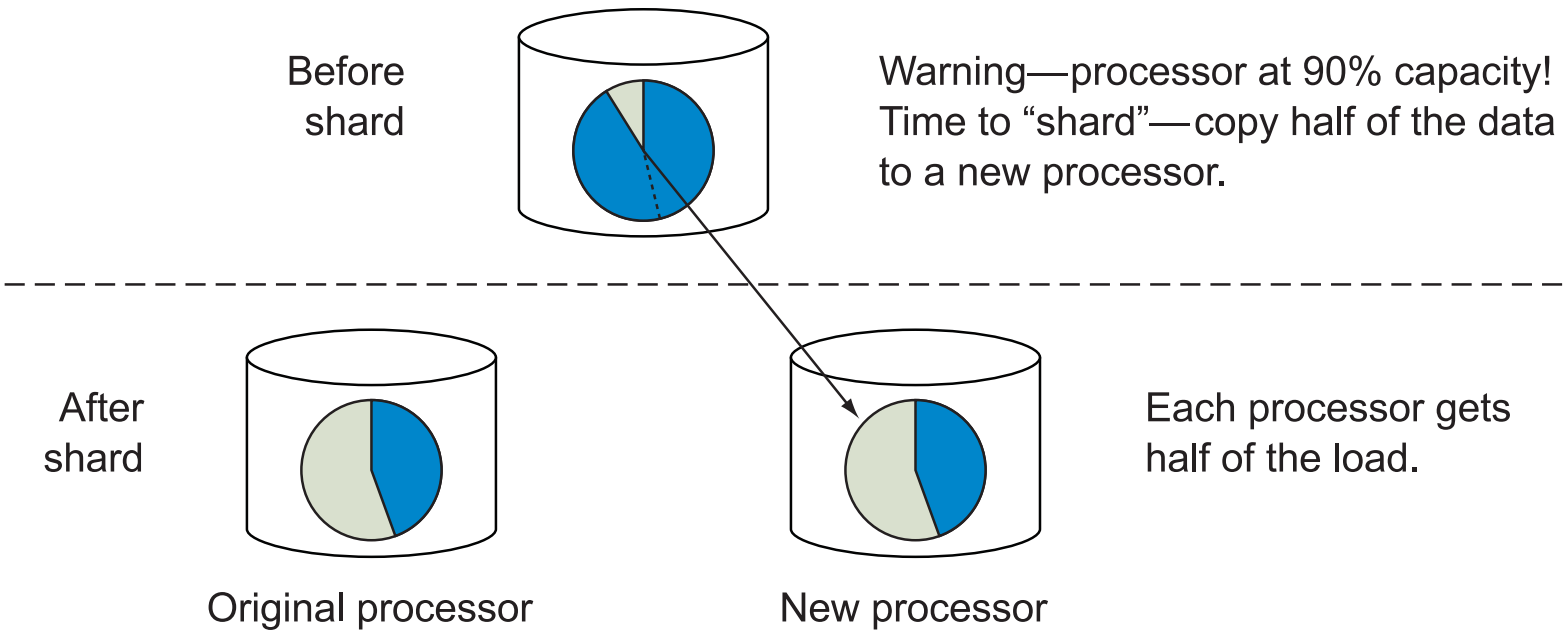
- ulaşılabilirliğe odaklı $\rightarrow$ tutarlılık gevşek
- daha az hareket yönetimi garantisi

- ACID yerine BASE
- Basic availability (temel ulaşılabilirlik)
- Soft state (gevşek durum)
- Eventual consistency (er geç tutarlılık)

Parçalama

- bir sunucu veri depolama kapasitesinin sınırına gelirse
- parçalama: veriyi parçalara ayır (sharding)
- parçaları dağıtık sunuculara paylaştır
- verimi artırır
- daha fazla sunucu $\rightarrow$ daha fazla risk noktası

Parçalama


Before shard
Warning—processor at 90% capacity! Time to “shard”—copy half of the data to a new processor.
After shard
Each processor gets half of the load.
Original processor
New processor

Çoğaltma

- veriyi farklı sunucularda çoğalt
- hata toleransını artırır
- kopyalar farklı olabilir
- er geç tutarlı: anlık tutarsızlıklara izin var
- sistem durduğunda bütün kopyalar aynı

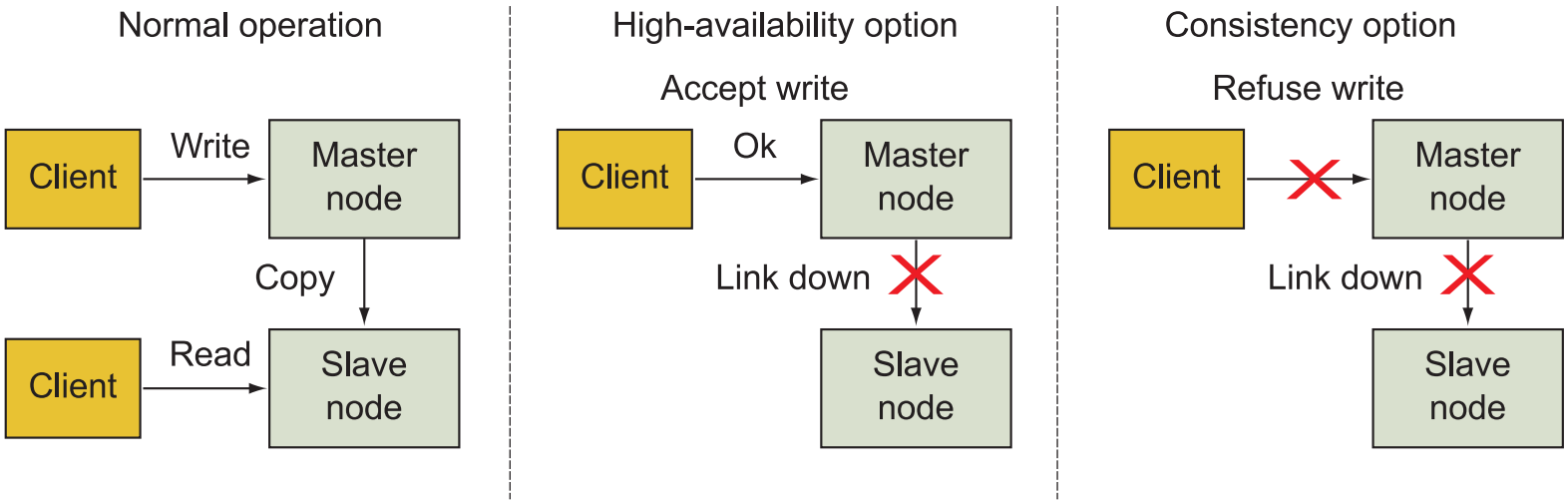
CAP Özellikleri

- tutarlılık (Consistency):
  bütün istemciler dağıtık sistemden gelen verinin
  aynı güncel versiyonuna erişebilir
- ulaşılabilirlik (Availability):
  çoğaltılmış veri parçaları arasındaki iletişim hataları
  güncellemeyi engellemez
- bölümleme toleransı (Partition tolerance):
  bölümler arasında bağlantı hatası olsa bile
  sistem çalışmaya devam eder

CAP Teoremi

- Dağıtık bir veri tabanı CAP özelliklerinin en fazla ikisine sahip olabilir.
  (Eric Brewer - 2000)

CAP


Normal operation
Client
Write
Master node
Copy
Client
Read
Slave node
High-availability option
Accept write
Client
Ok
Master node
Link down
Slave node
Consistency option
Refuse write
Client
Master node
Link down
Slave node

Serileştirme

- bir nesne hangi biçimde kaydedilebilir?
- basit çözüm: katar
- seri biçim
- yazma sırasında: nesne $\rightarrow$ seri biçim (serileştirme)
- okuma sırasında: seri biçim $\rightarrow$ nesne (geri oluşturma)

Serileştirme

- bir nesne hangi biçimde kaydedilebilir?
- basit çözüm: katar
- seri biçim
- yazma sırasında: nesne $\rightarrow$ seri biçim (serileştirme)
- okuma sırasında: seri biçim $\rightarrow$ nesne (geri oluşturma)

Serileştirme Biçimleri

- yaygın biçimler: XML, JSON
- insanlar okuyabilir
- veri değişimi için elverişli
- yarı yapısal veriyi tutmak için elverişli

Konular

JSON

- JavaScript Object Notation
- taban değerler: sayı, katar, ...
- nesneler: anahtar-değer çifti kümeleri
- değer dizileri
- içiçe geçmiş yapı

JSON Örneği

```
{
  "title": "The Usual Suspects",
  "year": 1995,
  "score": 8.7,
  "votes": 35027,
  "director": "Bryan Singer",
  "cast": [
    "Gabriel Byrne",
    "Benicio Del Toro"
  ]
}
```

JSON Example

```
[
  {
    "title": "The Usual Suspects",
    "year": 1995,
    ...
  },
  ...
  {
    "title": "Being John Malkovich",
    "year": 1999,
    ...
  }
]
```

Geçerli Belgeler

- JSON Schema

Sorgulama Dili

- yaygın kullanılan, bildirim temelli sorgulama dili yok
- programlama ile veri işleme

Konular

Anahtar-Değer Deposu

- model: (anahtar, değer) çiftleri
- anahtarlar üzerinden dizinleniyor
- anahtarlar eşsiz
- değerler herhangi bir büyük veri olabilir (BLOB gibi)
- çok basit arayüz: put, get, delete
- değerler üzerinden sorgulama yok
- ürünler: Redis, Riak, Memcache, Amazon DynamoDB

Anahtar-Değer Deposu

- model: (anahtar, değer) çiftleri
- anahtarlar üzerinden dizinleniyor
- anahtarlar eşsiz
- değerler herhangi bir büyük veri olabilir (BLOB gibi)
- çok basit arayüz: put, get, delete
- değerler üzerinden sorgulama yok
- ürünler: Redis, Riak, Memcache, Amazon DynamoDB

Anahtar-Değer Deposu

- model: (anahtar, değer) çiftleri
- anahtarlar üzerinden dizinleniyor
- anahtarlar eşsiz
- değerler herhangi bir büyük veri olabilir (BLOB gibi)
- çok basit arayüz: put, get, delete
- değerler üzerinden sorgulama yok
- ürünler: Redis, Riak, Memcache, Amazon DynamoDB

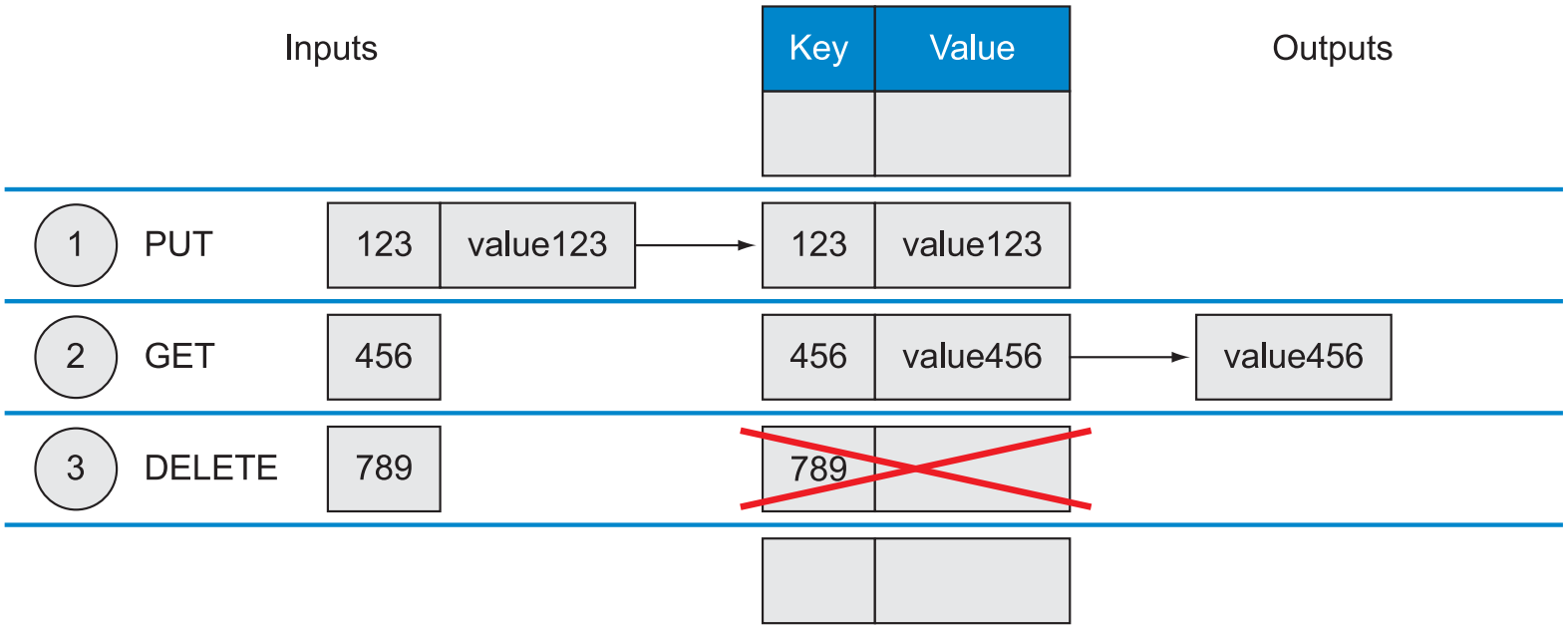
Anahtar-Değer Deposu Örnekleri

- web sayfası ön belleğe alma
- anahtar: URL, değer: web sayfası
- görüntü deposu
- anahtar: görüntünün yolu, değer: görüntü

Anahtar-Değer Deposu Örnekleri

- web sayfası ön belleğe alma
- anahtar: URL, değer: web sayfası
- görüntü deposu
- anahtar: görüntünün yolu, değer: görüntü

Anahtar-Değer Deposu



Anahtar-Değer Depoları

- kayıtların anahtarlara göre sunuculara dağıtılması
- gelişmiş: değer alanında veri yapısı
- veri sadece blob tipinde değil

Sütun Ailesi Depoları

- anahtar bir (satır, sütun) çifti
- seyrek matris
- gelişmiş anahtarlar: (satır, sütun ailesi, sütun, zaman damgası)
- sütun ailesi: sütun grubu
- zaman damgası: zaman içinde birden fazla değer saklanır
- ürünler: Apache Cassandra, Apache HBase, Google BigTable

Sütun Ailesi Depoları

- anahtar bir (satır, sütun) çifti
- seyrek matris
- gelişmiş anahtarlar: (satır, sütun ailesi, sütun, zaman damgası)
- sütun ailesi: sütun grubu
- zaman damgası: zaman içinde birden fazla değer saklanır
- ürünler: Apache Cassandra, Apache HBase, Google BigTable

Sütun Ailesi Depo Örneği

- kullanıcı tercihleri
- gizlilik ayarları, iletişim bilgileri, bildirimler, ...
- genellikle 100 alandan az, 1 KB

- sadece kullanıcı değişiklik yapıyor: ACID özelliklerine gerek yok
- çoğunlukla okuma işlemi
- hızlı ve ölçeklenebilir olması gerekiyor

Konular

Belge Depoları

- model: (anahtar, belge) çifti
- belge: JSON biçimli veri

- belge içeriğine göre sorgu
- belgeler otomatik olarak dizinlenir
- belgeler gruplanır: hiyerarşik yapı

- ürünler: MongoDB, CouchDB
- uygulama örneği: içerik yönetim sistemleri

MongoDB Ekleme Örneği

```
itucsdb.movies.insert(
  {
    "title": "Ed Wood",
    "year": 1994,
    "score": 7.8,
    "votes": 6587,
    "director": "Tim Burton",
    "cast": [
      "Johnny Depp"
    ]
  }
)
```

MongoDB Ekleme Örneği

```
itucsdb.movies.insert(
  {
    "title": "Three Kings",
    "year": 1999,
    "score": 7.7,
    "votes": 10319,
    "cast": [
      "George Clooney",
      "Spike Jonze"
    ]
  }
)
```

MongoDB Bulma Örneği

```
itucsdb.movies.find()

itucsdb.movies.find(
  {"year": 1999}
)

itucsdb.movies.find(
  {"year": {$gt 1999}}
)
```

Konular

XML

- XML kendisi bir dil değil
- dil tanımlama kuralları
- XML ile tanımlanmış diller:
  XHTML, DocBook, SVG, ...
- XML ile bağlantılı diller:
  XPath, XQuery, XSL Transforms, ...

XML Yapısı

- ağaç yapısı
- düğümler: *elemanlar*
- elemanlar açılış-kapanış takıları ile gösterilir
- içiçe geçen elemanlar ağaç yapısı oluşturur
- boş elemanlar: kendi kapanan takılar
- elemanların nitelikleri olabilir
- elemanların karakter verileri olabilir (CDATA)

XML Örneği: XHTML

```
<html>
<head>
<title>...</title>
<meta charset="utf-8" />
</head>
<body>
<h1>...</h1>
<p>...</p>
<img src="..." alt="..." />
</body>
</html>
```

XML Örneği: Film

```
<movie color="Color">
<title>The Usual Suspects</title>
<year>1995</year>
<score>8.7</score>
<votes>35027</votes>
<director>Bryan Singer</director>
<cast>
<actor>Gabriel Byrne</actor>
<actor>Benicio Del Toro</actor>
</cast>
</movie>
```

XML Örneği: Filmler

```
<movies>
<movie color="Color">
<title>The Usual Suspects</title>
<year>1995</year>
...
</movie>
<movie color="Color">
<title>Being John Malkovich</title>
<year>1999</year>
...
</movie>
...
</movies>
```

İyi Biçimlendirilmiş Belge

- iyi biçimlendirilmiş: XML kurallarına uygun
- sözdizimsel olarak doğru
- tek bir kök düğüm
- elemanlar düzgün biçimde içiçe geçmiş: uygun takılar
- elemanların nitelikleri tekil
- XML ayrıştırıcılar iyi biçimlendirilmiş XML dökümanlarını DOM (Document Object Model) nesnelerine çevirirler

İyi Biçimlendirilmiş Belge

- iyi biçimlendirilmiş: XML kurallarına uygun
- sözdizimsel olarak doğru
- tek bir kök düğüm
- elemanlar düzgün biçimde içiçe geçmiş: uygun takılar
- elemanların nitelikleri tekil
- XML ayrıştırıcılar iyi biçimlendirilmiş XML dökümanlarını DOM (Document Object Model) nesnelerine çevirirler

Geçerli Belge

- geçerli: uygulama alanı kurallarına uygun
- anlamsal olarak doğru
- DTD, XML şema
- XML ayrıştırıcılar geçerliliği de kontrol ediyorlar

Geçerli Belge

- geçerli: uygulama alanı kurallarına uygun
- anlamsal olarak doğru
- DTD, XML şema
- XML ayrıştırıcılar geçerliliği de kontrol ediyorlar

XML Veri Tabanları

- belge depolarının bir türü
- belge: XML biçimlendirilmiş veri
- XPath ile sorgulama
- ürünler: Oracle Berkeley DBXML, BaseX, eXist

XPath

- XPath: XML belgelerinden düğüm ve veri çekme
- aranılan düğümlerin yolu: yol adımları zinciri
- kök düğümünden başlanarak (mutlak)
- bulunulan düğümden başlanarak (bağıl)

XPath Örnekleri

- bütün filmler: `/movies/movie`
- bu filmin oyuncuları: `./cast/actor`
- `../../year`

Yol Adımları

- yol adımı yapısı:
  `axis::node_selector[predicate]`
- eksen: nerede aranacak?
- seçici: ne aranacak?
- yüklem: hangi koşula uyanlar aranacak?

Eksenler

- `child`: bütün çocuklarda, bir düzey (varsayılan eksen)
- `descendant`: bütün çocuklarda, rekürsif olarak (kısa gösterilim: `//`)
- `parent`: anne düğümde, bir düzey
- `ancestor`: bütün anne düğümlerde, köke kadar
- `attribute`: nitelikler (kısa gösterilim: `@`)
- `following-sibling`: sonra gelen kardeşlerde
- `preceding-sibling`: önce gelen kardeşlerde
- `...`

Düğüm Seçiciler

- düğüm takısı
- düğüm niteliği
- düğüm metni: `text()`
- düğümün bütün çocukları: `*`

XPath Örnekleri

- bütün yönetmenlerin isimleri:
  ```
  /movies/movie/director/text()
  //director/text()
  ```
- bu filmdeki bütün oyuncular:
  ```
  ./cast/actor
  .//actor
  ```
- bütün filmlerin renkleri:
  ```
  //movie/@color
  ```
- bundan sonraki filmlerin puanları:
  ```
  ./following-sibling::movie/score
  ```

XPath Örnekleri

- bütün yönetmenlerin isimleri:
  `/movies/movie/director/text()`
  `//director/text()`
- bu filmdeki bütün oyuncular:
  `./cast/actor`
  `.//actor`
- bütün filmlerin renkleri:
  `//movie/@color`
- bundan sonraki filmlerin puanları:
  `./following-sibling::movie/score`

XPath Örnekleri

- bütün yönetmenlerin isimleri:
  `/movies/movie/director/text()`
  `//director/text()`
- bu filmdeki bütün oyuncular:
  `./cast/actor`
  `.//actor`
- bütün filmlerin renkleri:
  `//movie/@color`
- bundan sonraki filmlerin puanları:
  `./following-sibling::movie/score`

XPath Örnekleri

- bütün yönetmenlerin isimleri:
  `/movies/movie/director/text()`
  `//director/text()`
- bu filmdeki bütün oyuncular:
  `./cast/actor`
  `.//actor`
- bütün filmlerin renkleri:
  `//movie/@color`
- bundan sonraki filmlerin puanları:
  `./following-sibling::movie/score`

XPath Yüklemleri

- düğüm sırası sınaması: `[position]`
- çocuk varlığı sınaması: `[child_tag]`
- çocuk değeri sınaması: `[child_tag="value"]`
- nitelik varlığı sınaması: `[@attribute]`
- nitelik değeri sınaması: `[@attribute="value"]`

XPath Yüklemleri

- düğüm sırası sınaması: `[position]`
- çocuk varlığı sınaması: `[child_tag]`
- çocuk değeri sınaması: `[child_tag="value"]`
- nitelik varlığı sınaması: `[@attribute]`
- nitelik değeri sınaması: `[@attribute="value"]`

XPath Yüklemleri

- düğüm sırası sınaması: `[position]`
- çocuk varlığı sınaması: `[child_tag]`
- çocuk değeri sınaması: `[child_tag="value"]`
- nitelik varlığı sınaması: `[@attribute]`
- nitelik değeri sınaması: `[@attribute="value"]`

XPath Örnekleri

- birinci filmin başlığı:
  `/movies/movie[1]/title`
- 1997 yılında çekilmiş filmler:
  `movie[year="1997"]`
- siyah-beyaz filmler:
  `movie[@color="BW"]`

XPath Örnekleri

- birinci filmin başlığı:
  `/movies/movie[1]/title`
- 1997 yılında çekilmiş filmler:
  `movie[year="1997"]`
- siyah-beyaz filmler:
  `movie[@color="BW"]`

XPath Örnekleri

- birinci filmin başlığı:
  `/movies/movie[1]/title`
- 1997 yılında çekilmiş filmler:
  `movie[year="1997"]`
- siyah-beyaz filmler:
  `movie[@color="BW"]`

Çizge Veri Tabanları

- model: düğümler ve ayrıtlar
- düğümlerin özellikleri var
- ayrıtların etiketleri var
- ilişki içerikli veri: sosyal ağlar, ...
- katma yerine geçiş
- ürünler: Neo4J

Çizge Veri Tabanları

- uygun işler:
  en kısa yol, arkadaşların arkadaşları,
  belli özellikteki komşu düğümler
- ölçeklenmesi zor
- bildirim temelli sorgulama dili: Cypher, Gremlin

Çizge Veri Tabanları

- uygun işler:
  en kısa yol, arkadaşların arkadaşları,
  belli özellikteki komşu düğümler
- ölçeklenmesi zor
- bildirim temelli sorgulama dili: Cypher, Gremlin

Çizge Veri Tabanları

- uygun işler:
  en kısa yol, arkadaşların arkadaşları,
  belli özellikteki komşu düğümler
- ölçeklenmesi zor
- bildirim temelli sorgulama dili: Cypher, Gremlin

Cypher

- başlangıç düğümlerini bul
- ilişkileri izle ve getir
- değerleri değiştir ve/veya getir

Cypher: Düğümler

- `(name)`
- `(name:Type)`
- `(name:Type {attributes})`

```
(matrix)
(matrix:Movie)
(matrix:Movie {title: "The Matrix"})
(matrix:Movie {title: "The Matrix", released: 1997})
```

Cypher: İlişkiler

- yönsüz: `--`
- yönlü: `-->` `<--`
- detayları ile: `-[]-`

```
-[role]->
-[role:ACTED_IN]->
-[role:ACTED_IN {roles: ["Neo"]}]->
```

Cypher: Örüntüler

- Düğümleri ve ilişkileri birleştir
- örüntülere isim ver

```
(keanu:Person   {name:  "Keanu Reeves"} )
-[role:ACTED_IN {roles: ["Neo"] } ]->
(matrix:Movie   {title: "The Matrix"} )

acted_in = (:Person)-[:ACTED_IN]->(:Movie)
```

Cypher: Veri Yaratma

```
CREATE (:Movie {title: "The Matrix", released: 1997})

CREATE (p:Person {name: "Keanu Reeves", born: 1964})
RETURN p

CREATE (a:Person {name: "Tom Hanks", born:1956 })
  -[r:ACTED_IN {roles: ["Forrest"]}]->
  (m:Movie {title: "Forrest Gump", released: 1994})
CREATE (d:Person {name: "Robert Zemeckis", born: 1951})
  -[:DIRECTED]-> (m)
RETURN a, d, r, m
```

Cypher: Örüntüleri Eşleştirme

```
MATCH (m:Movie)
RETURN m

MATCH (p:Person {name:"Keanu Reeves"})
RETURN p

MATCH (p:Person {name:"Tom Hanks"})
  -[r:ACTED_IN]-> (m:Movie)
RETURN m.title, r.roles
```

References

Supplementary Reading

- Making Sense of NoSQL, by Dan McCreary and Ann Kelly, Manning Publications
- The Neo4J Manual: Tutorials
  `http://neo4j.com/docs/stable/tutorials.html`