JSON (JavaScript Object Notation)

JSON, özellikle yarı yapılandırılmış veriler için standart bir veri değişim formatı haline geldi. JSON veri tabanları, çeşitli veri türlerini depolamada esneklik sunan ve veri modeli veya proje gereksinimlerindeki değişiklikleri kolayca karşılayan NoSQL veri tabanları ailesinin bir parçasıdır. Bir JSON veri tabanının esnekliği, verilerin katı tablolar yerine belgeler olarak saklanma biçiminden gelir.

JSON, **J**ava **S**cript **O**bject **N**otation anlamına gelir

JSON, verileri depolamak ve taşımak için kullanılan bir **metin biçimidir.**

JSON "kendi kendini tanımlar" ve anlaşılması kolaydır

```
'{"isim":"Merve", "yaş":29, "araba":null}'
```

3 özelliği olan bir nesne tanımlar:

- isim
- yaş
- araba

Her özelliğin bir değeri vardır.

JSON dizesini bir JavaScript programıyla ayrıştırırsanız, verilere bir nesne olarak erişebilirsiniz:

```
let personAd = obj.isim;
let personYas = obj.yas;
```

JSON'u Neden Kullanmalı?

JSON formatı, sözdizimsel olarak JavaScript nesneleri oluşturmak için kullanılan koda benzer. Bu nedenle, bir JavaScript programı, JSON verilerini kolayca JavaScript nesnelerine dönüştürebilir.

Biçim yalnızca metin olduğundan, JSON verileri bilgisayarlar arasında kolayca gönderilebilir ve herhangi bir programlama dili tarafından kullanılabilir.

JavaScript, JSON dizelerini JavaScript nesnelerine dönüştürmek için yerleşik bir işleve sahiptir:

```
JSON.parse()
```

JavaScript ayrıca bir nesneyi JSON dizisine dönüştürmek için yerleşik bir işleve sahiptir:

```
JSON.stringify()
```

Bir sunucudan saf metin alabilir ve onu bir JavaScript nesnesi olarak kullanabilirsiniz.

Salt metin biçiminde bir sunucuya bir JavaScript nesnesi gönderebilirsiniz.

Karmaşık ayrıştırma ve çeviriler olmadan verilerle JavaScript nesneleri olarak çalışabilirsiniz.

Veri depolama

Verileri saklarken, verilerin belirli bir formatta olması gerekir ve onu nerede saklamayı seçerseniz seçin, *metin* her zaman uygun formatlardan biridir.

JSON, JavaScript nesnelerini metin olarak depolamayı mümkün kılar.

JSON Söz Dizimi Kuralları

JSON sözdizimi, JavaScript nesne notasyonu sözdiziminden türetilmiştir:

- Veriler ad/değer çiftlerindedir
- Veriler virgülle ayrılır
- Kaşlı ayraçlar nesneleri tutar
- Köşeli parantezler dizileri tutar

JSON Verileri - Bir Ad ve Değer

JSON verileri, ad/değer çiftleri (diğer adıyla anahtar/değer çiftleri) olarak yazılır.

Bir ad/değer çifti, bir alan adından (çift tırnak içinde), ardından iki nokta üst üste ve ardından bir değerden oluşur:

```
"isim":"Ozal"
```

JSON Değerleri

JSON'da değerler aşağıdaki *veri* türlerinden biri olmalıdır:

- dizi
- bir sayı
- bir obje
- bir dizi
- bir boole
- hükümsüz

JSON Dosyaları

- JSON dosyaları için dosya türü ".json"dur.
- JSON metni için "application/json"dur

Aşağıdaki JSON ve XML örneklerinin her ikisi de, 3 çalışandan oluşan bir dizi ile bir çalışanlar nesnesini tanımlar:

JSON

```
{"employees":[
                { "firstName":"John", "lastName":"Doe" },
                { "firstName":"Anna", "lastName":"Smith" },
                { "firstName":"Peter", "lastName":"Jones" }
]}
```

XML

```
<employees>
  <employee>
    <firstName>John</firstName> <lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName> <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName> <lastName>Jones</lastName>
  </employee>
</employees>
```

Nesnelerdeki Diziler

Nesneler diziler içerebilir:

```
{
"isim":"Merve",
"yaş":30,
"Arabalar":["Ford", "BMW", "Fiat"]
}
```

Dizi değerlerine dizine göre erişirsiniz:

```
myObj.Arabalar[0];
```

Veri Gönderme

Bir JavaScript nesnesinde saklanan verileriniz varsa, nesneyi JSON'a dönüştürebilir ve bir sunucuya gönderebilirsiniz:

```
const myObj = {name: "John", age: 31, city: "New York"};

const myJSON = JSON.stringify(myObj);

window.location = "demo_json.php?x=" + myJSON;
```

Veri alıyor

Verileri JSON biçiminde alırsanız, onu kolayca bir JavaScript nesnesine dönüştürebilirsiniz:

```
const myJSON = '{"name":"John", "age":31, "city":"New York"}';
const myObj = JSON.parse(myJSON);
document.getElementById("demo").innerHTML = myObj.name;
```

JSON veritabanları nelerdir?

JSON veri tabanı, yarı yapılandırılmış verileri depolamak için ideal olan belge tipi bir NoSQL veri tabanıdır. Küçük şema değişikliklerinin uygulanması söz konusu olduğunda sabit ve pahalı olan satır-sütun formatına kıyasla çok daha esnektir.

İlişkisel veri tabanlarında, JSON verilerinin NVARCHAR sütunu (LOB depolama) kullanılarak ayrıştırılması veya saklanması gerekir. Ancak MongoDB gibi belge veri tabanları, JSON verilerini insanlar ve makineler tarafından okunabilen doğal biçiminde depolayabilir.

JSON verilerini bir JSON veritabanında depolamanın iki yolu vardır :

1- Tüm nesneyi tek bir belgede depolama:

```
Kitap {
   baslik: 'Çalıkuşu',
   yazar: {
       adsoyad: Reşat Nuri Güntekin',
       dtarihi: 1889
   }
}
```

Burada, yazar ayrıntıları kitap belgesinin içindedir. Yazar alt belgesi kitap belgesine dahil edildiği için bu teknik gömme olarak da bilinir.

2- Nesnelerin parçalarını ayrı ayrı depolamak ve benzersiz tanımlayıcıları (referans) kullanarak bunları birbirine bağlamak.

```
yazar {
   _id: ObjectId(1),
    adsoyad: 'Reşat Nuri Güntekin',
    dtarih: 1889
      }

Kitap {
   _id: ObjectId(55),
   baslik: 'Çalıkuşu',
   yazar: ObjectId(1)
      }
```

Bir yazar birden fazla kitap yazabilir. Bu nedenle, tüm kitapların içinde yinelenen verilerden kaçınmak için ayrı bir yazar belgesi oluşturabilir ve bu belgeye `_id` alanına göre atıfta bulunabiliriz:

JSON veritabanlarının avantajları

Tıpkı geleneksel veri tabanları gibi, JSON belge veri tabanları da veri bölümleme, indeksleme, kümeleme, çoğaltma ve veri erişimini kendi başlarına yönetir. Bunun dışında JSON veri tabanları birçok avantaj sunar.

JSON veritabanları daha hızlıdır ve daha fazla depolama esnekliğine sahiptir

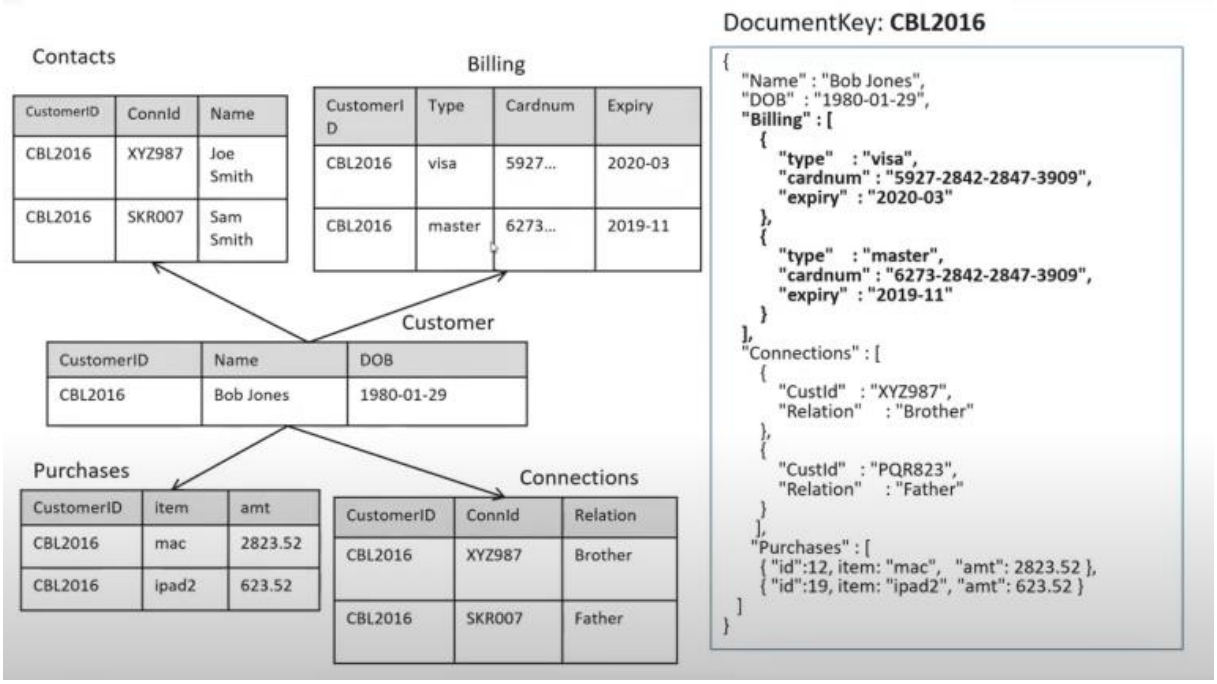
NoSQL veritabanları genel olarak daha fazla depolama esnekliğine sahiptir ve daha iyi indeksleme yöntemleri sunar. Bir belge veritabanında, her belge ayrı bir nesne olarak ele alınır ve sabit bir şema yoktur, bu nedenle her belgeyi en kolay erişilebileceği ve görüntülenebileceği şekilde saklayabilirsiniz. Ek olarak, değişen uygulama gereksinimlerinize uyarlamak için veri modelinizi geliştirebilirsiniz. Şema sürüm oluşturma modeli , tam da buna izin vermek için esnek belge modelini kullanır.

JSON veritabanları daha iyi şema esnekliği sağlar

Bir JSON belge veritabanının en iyi yanı, şema esnekliğidir ; yani, verilerinizi depolamak istediğiniz her yolu destekler. Erişmeniz gereken tüm bilgilere tek bir belgede birlikte sahip olabilirsiniz (gömülü) veya ayrı belgeler oluşturma ve ardından bunları bağlama (referans verme) özgürlüğünü kullanabilirsiniz. İç içe diziler veya katıştırılmış belgeler gibi bir belgenin içindeki iç içe nesneleri sorgulamak bile çok basittir.

JSON veritabanları, SQL yapılarıyla kolayca eşlenebilir

Birçok geliştirici SQL'e aşinadır. Geliştiriciler, verileri bir JSON veritabanında depolayarak SQL sütunlarını ve JSON belge anahtar adlarını kolayca eşleyebilir. Örneğin, bir belgenin kitapAdı anahtarı, kitap tablosunun kitap_adı sütununa eşlenebilir. JSON veritabanlarının çoğu , geliştiricinin öğrenme eğrisinden tasarruf sağlayan ve geliştirme süresini azaltan bu eşlemeyi otomatikleştirir .



JSON veritabanları farklı dizin türlerini destekler

Çeşitli indeks türlerinin mevcudiyeti nedeniyle, arama sorguları oldukça hızlıdır. Örneğin, MongoDB'nin sabit bir şeması olmadığından, bir alan veya alan kümesi üzerinde o alanın sorgulanmasını desteklemek için bir joker karakter dizini oluşturabilirsiniz. O2-tree ve T-tree gibi NoSQL veritabanlarını yüksek performanslı yapan birçok başka dizin türü vardır.

JSON veritabanları, büyük veri analitiği için daha uygundur

JSON veritabanlarının esnek bir şeması vardır ve dikey ve yatay olarak iyi ölçeklenebilir, bu da onları büyük hacimleri ve çeşitli büyük verileri depolamaya uygun hale getirir. MongoDB gibi belge veritabanları , veri işleme ve dönüştürme için ETL sistemlerine olan ihtiyacı ortadan kaldıran zengin bir sorgu diline (MQL) ve toplama hattına sahiptir. Ayrıca bu veritabanları, ek kodlama adımları olmadan verileri Python ve R gibi popüler veri analizi programlama dillerine kolayca aktarabilir .

JSON veritabanı örnekleri

MongoDB	MongoDB is the most popular document database used by companies like Google, Facebook, and Forbes. MongoDB stores data in a binary encoded JSON format (BSON) to offer better data types support, improved indexing and querying.
Cosmos DB	Cosmos DB is a serverless NoSQL database by Azure. It offers high scalability and enterprise-grade security, amongst other benefits of a document database.
CouchDB	Apache CouchDB natively uses JSON to store data and is an open source database. It supports binary data.
Couchbase	Couchbase is a memory-first database that stores data as JSON documents, and provides good performance and scalability.
Firestore	Firestore is a cloud-hosted NoSQL database that requires little to no maintenance and is a popular choice for mobile applications.