Databaseleri düzenlemenin 2 yolu var

* **Row oriented**
* **Column oriented** (also known as **columnar** or **C-store**)

**Row oriented:**

Klasik databaselerim row oriented şekilde tutuyor yani kayıtlar satır satır tutuluyor. row orinted tutmanın avantajı kayıt yazma (row yazma) kayıt okumak(row okuma) için elverişli yani OLTP işleri Bunlar klasik databaselerim örnek Oracle postresql sql server mysql

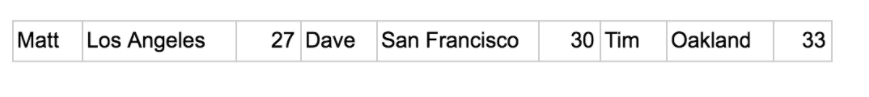
**Column oriented:**datalar kolon bazında tutuluyor bir analitik sorgu atıyorsak çok hızlı bir şekilde getirir

Kolonları okuma ve hesaplamakta iyidir

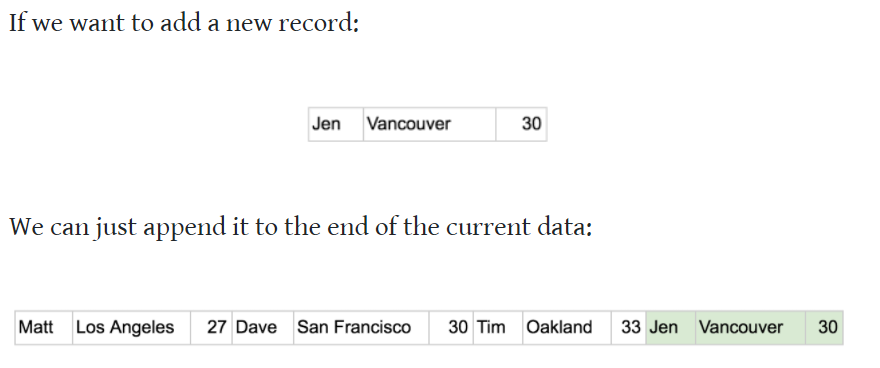
Redshift, BigQuery , Snowflake

**Row oriented database**

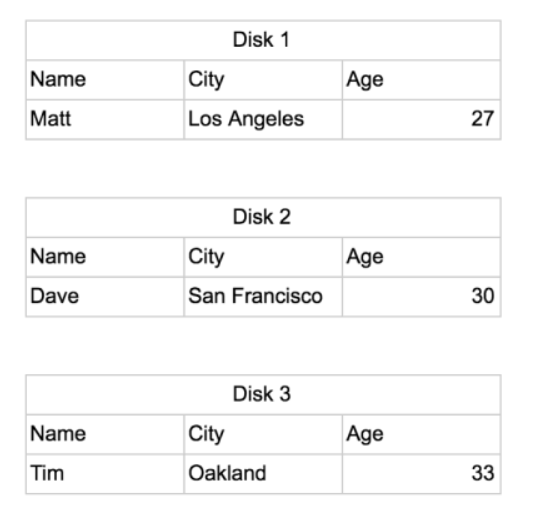
benim yukarıdaki gibi bir tablom olsun bu data diskte row oriented tutuluyor







Her diskin 1 satırlık veri tutacağını varsayalım

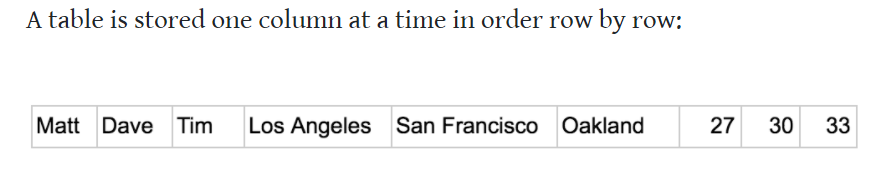


Ben herhangi bir satırı okumak istediğimde direkt diskten okuyabilirim

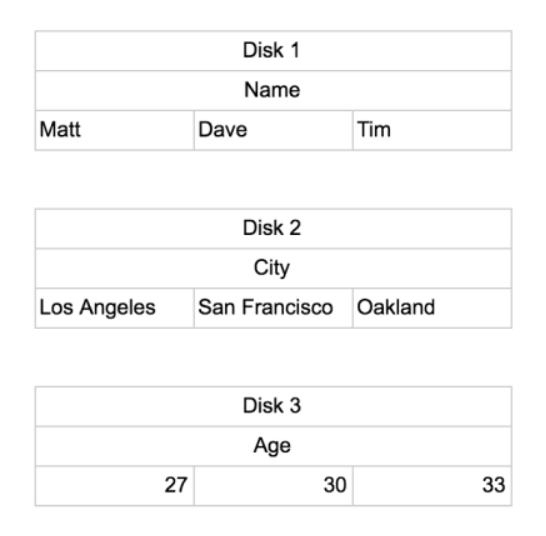
**Column Oriented Database**



**Datalar column oriented tutuyor**



**Her bir kolondaki veriler bir diskte tutuluyor**



**Şimdi ben toplam yaşları istiyorum bunu row oriented bir databasede yaparsam 3 disktende okumam gerekir**

**Ama column oriented bir database de sadece yaşarın tutulduğu disk yani disk3 den getirebilirim**

**Column oriented row orienteda göre daha iyi sıkıştırma yapıyor çünkü benzer veriler bir arada column oriented da**

**Şimdi elimde boyutu 100 Gb bir tablo olsun 100 milyon satırım 100 tane kolonum olsun her kolondada 1 gb data olsun şimdi ben cinsiyeti erkek olanların ortalama yaşını istiyorum**

**Şimdi ben bu sorguyu row oriented bir databasede yazarsam 100 gb datanın tamamına bakmam lazım**

**Ama column oriented bir databasede bir sorgu atarsam o zman benim 2 kolona bakmam yani gender ve age kolonlarına bakmam yeterli olcak bu şekilde 2 gb datayı okuyarak sorgu sonucuna ulaşcam**

**AYNI datayı farklı formatlarda tutma**

**Şema her kolonun ismi ve veri tipi**

**DATA SOURCE FILE**

**CSV(comma seperated values)**

**Virgülle ayrılmış veriler diyebiliriz**

**Direkt csv dosyasını açtığında datayı anlayabilirsin**

**Avantajları : iyi bir sıkıştırma oranına sahip**, **kolay okunabilir**,**batch ve real time işleri destekler**

**Dezavantajları:null ve empty değerleri desteklemiyor schema evolutionu desteklemiyor**

eski verilerinizin şemasıyla geriye dönük uyumluluğu korurken yeni verileri yazmak için kullanılan şemayı güncellemenize izin verir.

Şema: kolonun(attribute,field) ın adı ve veri tipinin tutulduğu yer

**Csv de metadata yok header yok**

**split edilmesi garanti değil**

**Hadoop spark kafka tarafından desteklenir**



JSON(JavaScript object notation): json çeşitli tiplerde veriyi(string, integer, booleans, array, object, key-value, nested data) içerebilen bir kayıt biçimidir)

Herhangi bir metin düzenleyicisi tarafından açabilirim windowsda notepad den açıyorum

Batch ve realtime işleri destekler

Metadatayı verilerle birlikte depolar

Split edilemez ama json line formatında ise split edilebilir

JSON, MongoDB, Couchbase ve Azure Cosmos DB gibi NoSQL veritabanları için yaygın olarak kullanılan bir dosya biçimidir.



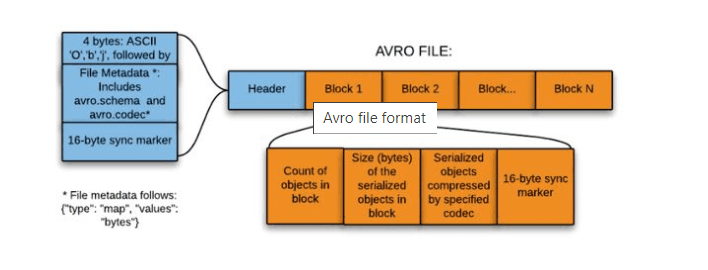
APACHE AVRO

lişkisel veritabanlarındaki gibi satır bazlı saklar. Avro’nun ana amacı, verileri sıkıştırmak ve bunu şema esnekliğini kaybetmeden yapabilmektir. Örneğin, Hadoop’u belge deposu olarak kullanmak ve tüm verilerinizi JSON olarak sıkıştırmak için Avro dosyalarında tutmak isteyebilirsiniz. Çalışmak istediğiniz bazı karmaşık şemalara sahip olabilirsiniz ve tümü Avro ile de çalışabilir. Avro’nun esnekliği, istediğiniz sayıda şema çalışmanıza ve hala düzgün bir sıkıştırma elde etmenize olanak sağlar.

dezavantaj

Verileri insan tarafından okunamaz.(binary formatında olduğu için)

Her dile entegre değil.



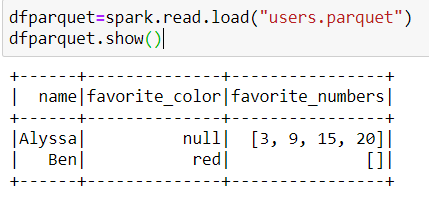
APACHE parquet

Verileri column oriented depolar

Parquet, Avro’dan farklı bir amaca sahiptir. Maksimum şema esnekliğine izin vermek yerine, sorgu hızlarını artırmak ve disk i/o’sunu azaltmak için kullanabileceğiniz şema türlerini optimize etmeye çalışır. Parquet, sütunlarda iç içe tiplere izin vererek geleneksel sütun bazlı tutmanın bazı zayıflıklarının üstesinden gelmeye çalışır. Böylece teknik olarak bir dizi olan bir sütuna ya da aslında birkaç sütun olan bir sütuna sahip olabilirsiniz.

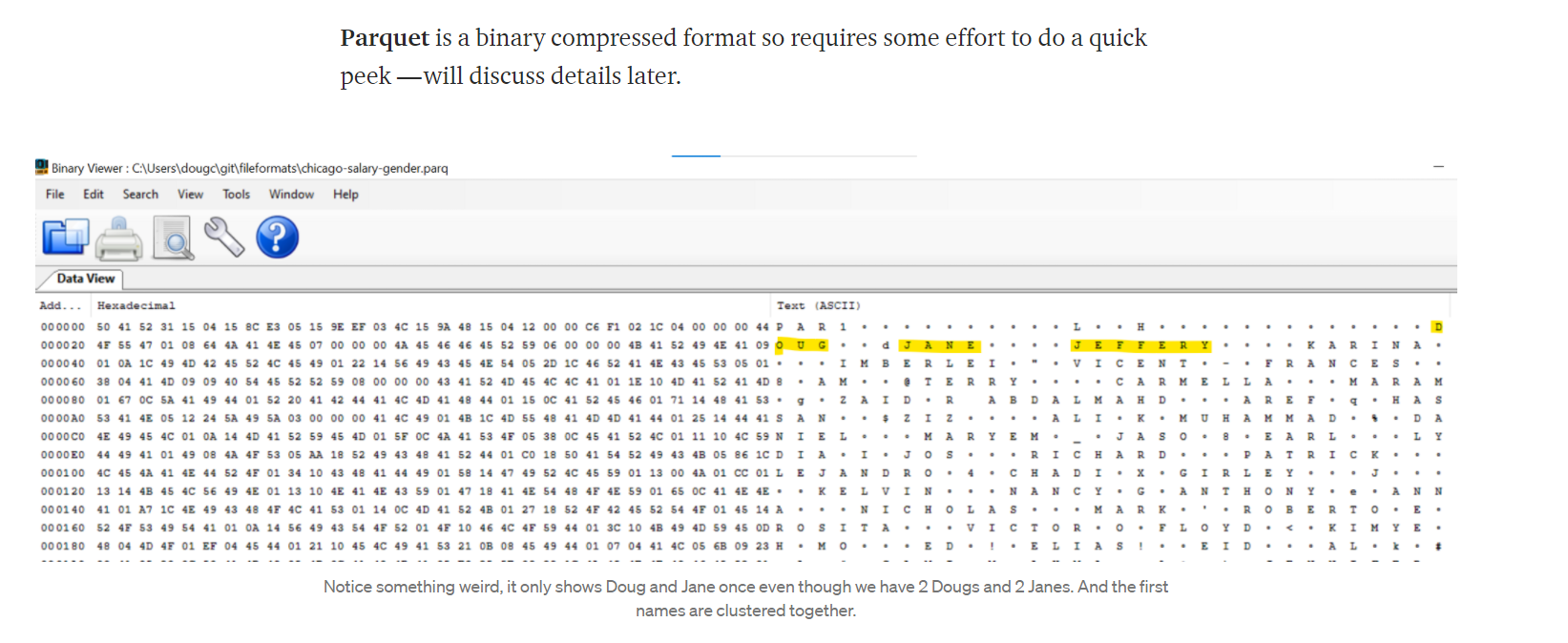
Sadece batch processing yapabiliirm

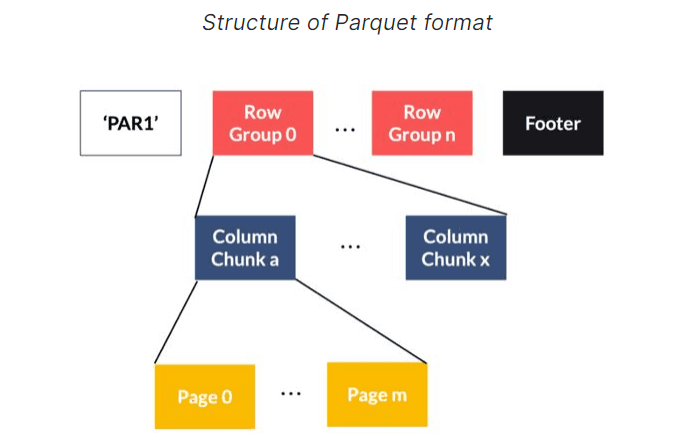
Spark ve impalada kullanabilirim



Dezavantajları:

İnsan tarafından okunamaz





ORC:

ORC (Optimized Row Columnar)

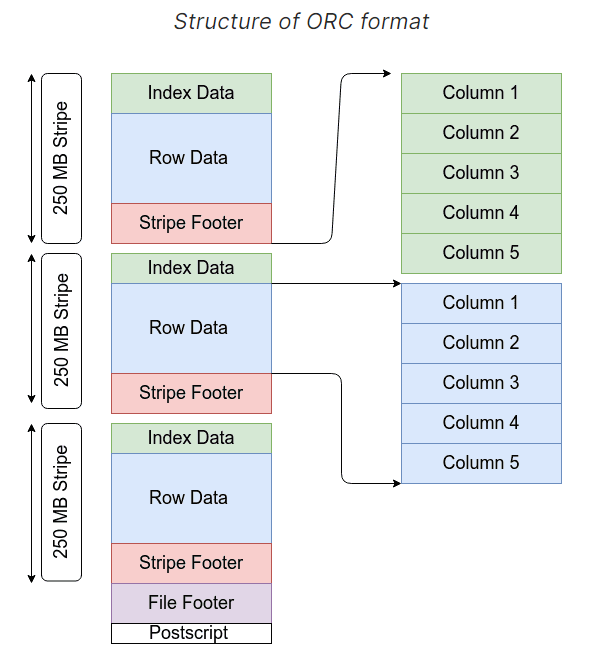
ORC, Parquet’nin sütun biçimini paylaşır ancak bazı farklılıklar vardır. Başlıca farklılıklardan biri iç içe geçen veri türlerine izin veren listeler ve haritalar gibi karmaşık türleri destekler. Sıkıştırma söz konusu olduğunda, ORC’nin verileri Parquet’den daha verimli bir şekilde sıkıştırdığı söylenir, ancak bu verilerinizin nasıl yapılandırıldığına bağlıdır.

*Avantaj*

* Yüksek oranda sıkıştırılabilir, orijinal verilerin boyutunu %75'e kadar azaltır.

Batch processing yapabilir sadece

ORC, Hive'da performans okuma, yazma ve işlemeyi iyileştirir.

s

