**1. Imperative Programming ve Declarative Programming kavramlarını kısaca açıklayıp farklarını belirtiniz.**

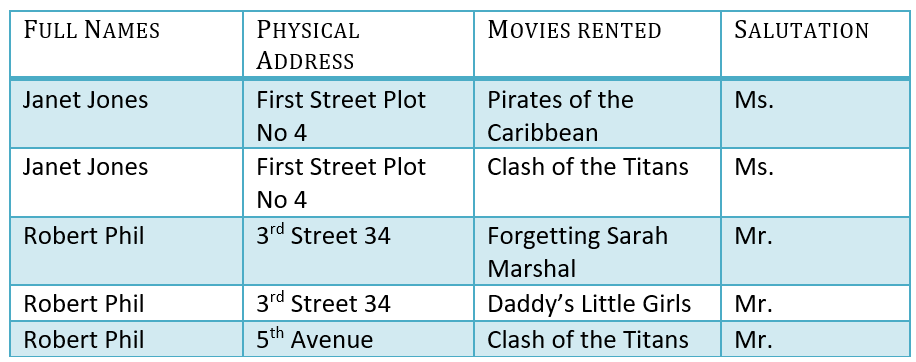
Ne yapılacağını içeren, nasıl yapılacağınının mantığını pek de içermeyen paradigmadır. Imperative ise yapılan bir işin nasıl yapılacağınıda gösterir.

**3. İlişkisel veritabanları için normalizasyon kavramı neyi ifade etmektedir ? İlk 3 normal formu örnek üzerinden açıklayınız.**

Normalizasyon kavramı, ilişkisel veritabanı modelinin kurucusu olan Edgar F. Codd tarafından oluşturulmuş bir kavramdır. Tabloların tasarımı sırasında normalizasyonu uygun seviyede kullanarak performans artışı ile birlikte veri tutarlılığının ve bütünlüğünün korunması sağlanır. Normalizasyon işlemleri belli seviyelerden oluşur. Bir seviyedeki işlemler tamamlanmadan diğer seviyeye geçilmez. Her seviyede tablo ve kolon bazlı işlemler yapılarak tabloların daha atomik veriler tutması hedeflenir. Temel olarak 6 normalizasyon seviyesi vardır.

**1. Normalizasyon Seviyesi:**

* Aynı tabloda tekrarlayan kolonların bulunmaması
* Her kolonda yalnızca bir değer bulunması
* Her satır için tanımlayıcı bir anahtar kolonun bulunması



**Yukarıdaki örnekte 1.** Normalizasyon **Seviyesi nin doğru kullanımı verilmiştir. Eğer Janet Jones in kiraladığı filmler tek bir satırda bulunsaydı, bir den çok veri kiraladığı için kolan tekrarı oluşacaktı. Kolon tekrarını önlemek için hepsi tek bir sütunda toplansaydı bu seferde sütunda birden fazla veri oluşacaktı.**

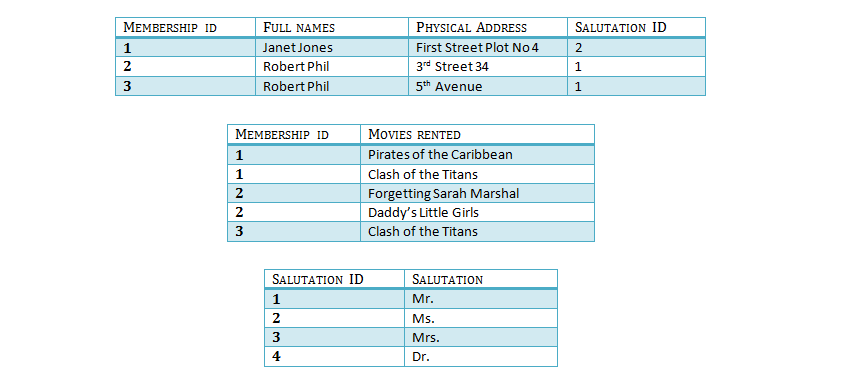
**2. Normalizasyon Seviyesi:**

* Belirli sayıda kolondan oluşan alt kümede tekrarlanan kayıtlar bulunmaması, bu alt küme için ayrı bir tablo oluşturulması.
* Alt kümeden oluşan tablo ile asıl tablo arasında ilişki tanımlayacak anahtarların (foreign key) tanımlanması.



Kiralanan filimler için yeni bir tablo oluşturuldu. Bu tablodaki member ıd üzerinden kullanıcının kiraladığı filmlere erişilebilmiş oldu. Böylece sürekli kullanıcının ismi, soyisminin yazılmasının önüne geçilmiş oldu. Buda veri tekrarını öndeli.

**3. Normalizasyon Seviyesi:**Bir tablonun 3. Normalizasyon seviyesinde olması için öncelikli olarak 2. Seviyede olması gerekir. Bu seviyede ayrıca birincil anahtar (primary key) olmayan her kolonun başka bir kolona bağlı olmaması da kontrol edilir.



Resimde de görüldüğü gibi en başta colonlarda tekrar eden veriler vardı. Bu veri tekrarının önüne geçilmiş oldu.

**4. ORM kütüphaneleri kullanmak her zaman avantajlı mıdır? ORM kütüphanelerinin ne gibi dezavantajları olabilir?**

Birincisi, bir ORM kitaplığı kullanarak, uygulamanız yalnızca en temel CRUD (Oluştur, Al, Güncelle, Sil) işlemlerini yapabilir. Birleşimler ve birleşimler, karmaşık veri türlerini işlemek, saklı yordamlar, tetikleyiciler ve işlevler kullanmak, vb. gibi uzaktan daha gelişmiş herhangi bir şey yapmak için, SQL kullanılmak zorunda kalınabilir ve bu, tüm kullanım amacını ortadan kaldırır.

İkinci büyük dezavantaj elbette hızdır. Kodunuz ve veritabanı arasında tamamen yeni bir ek katman olduğundan, performans, uygulamanızın ölçeklenmesi için iyi olmayabilecek bir yavaşlamaya sebep olabilir. Büyük projelerde ORM kullanmamanın daha avantajlı olduğunu düşünmekteyim.

**5. Domain Specific Language (DSL) kavramını açıklayınız.**

Etki Alanına Özgü Dil, belirli bir sorun sınıfı için optimize edilmiş daha yüksek bir soyutlama düzeyine sahip bir programlama dilidir. DSL, genellikle Java, C veya Ruby gibi genel amaçlı bir dilden daha az karmaşıktır. Genel olarak, DSL'ler, DSL'nin tasarlandığı alandaki uzmanlarla yakın koordinasyon içinde geliştirilir. Çoğu durumda, DSL'lerin yazılım çalışanları tarafından değil, programcı olmayan kişiler tarafından kullanılması amaçlanmıştır.

Apache Ant veya SQL DSL bazı örnekleridir.

**7. Thread Pool nedir ? Nerelerde kullanılır ?**

Threadler ayni zamanda Işletim sisteminin kaynaklarını kullandıkları için kontrolsüz bir şekilde kullanıldığında bu kaynakları hızlıca tüketmiş olurlar. örnek vermek gerekirse, donanımsal olarak 2 işlemci çekirdeğimiz mevcut olduğu bir ortamda 4 thread çalıştırmak gerekse bile önce ilk 2 sinin bitmesini bekleyip daha sonra diğer 2 sini oluşturup çalıştırmalıyız. Bu olayı bu şekilde takip etmek yerine Java nın bize sağladığı Thread havuzlarını kullanabiliriz. Thread sayısının ve çalışmasının düzenli ve kontrollü bir şekilde gerçekleştirilmesi için Java bize Executor adında bir sınıf sunmaktadır. Böylece aynı anda en fazla kaç adet thread çalışacağı belirlenmiş olur.

**8. Scalability nedir ? Horizontal ve Vertical Scalability kavramlarını açıklayınız.**

Sistemin/Veritabanının Yatayda ölçeklenebilir olması (horizontally scalable, scale out); ucuz ve çok sayıda makinenin aynı anda kullanılması anlamına gelir. Yatay ölçeklenebilirlik sayesinde yedeklilik de performans artışı da sağlanabilir.

Sistemin/Veritabanının Dikeyde ölçeklenebilir olması (dikey ölçeklenebilirlik, vertically scalable, scale up); bir tane çok güçlü aynı zamanda pahalı bir makine/donanım kullanılmasıdır. Dikey Ölçeklenebilir sistemlerde donanım kısıtları mevcuttur. Örneğin mevcut sisteminizin CPU frekansını 5 ghz yapamazsınız veya 1 tb ram yapamazsınız.

Yatay ölçeklenebilirlikte yüzlerce, binlerce makinelik server (sunucu) ağı kurulabilir. Yatay ölçeklenebilirliğin yönetimi dikey ölçeklenebilirliğe göre daha zordur.Sonuçta onlarca veya yüzlerce makinayı yönetmek, tabi ki tek bir makinayı yönetmekten daha zordur.

**9. Data replication ve sharding nedir ? Aralarında nasıl bir fark bulunmaktadır ?**

Replication, aynı veri kümesinin birkaç kopyasının ayrı makinelerde barındırıldığı bir veritabanı kurulumuna atıfta bulunur. Replicationın ana nedeni fazlalıktır. Tek bir veritabanı ana makinesi bozulursa, aynı veritabanının bir kopyasını barındıran diğer makinelerden biri devralabileceğinden kurtarma işlemi hızlıdır. İkincil bir makineye hızlı yük devretme, aksama süresini en aza indirir ve veritabanının aktif bir kopyasını tutmak, veri kaybını en aza indirmek için bir yedek görevi görür.

Sharding ( Horizontal Partitioning ) Sharding aksine entityleri bir bütün olarak farklı veritabanları üzerinde tutabilmeyi sağlıyor. Daha farklı bir ifadeyle büyük tek bir veritabanı kullanmak yerine, verileri vertical partitioning’in aksine aralarında hiçbir ilişki olmayacak şekilde küçük parçalara bölerek farklı ve nispeten daha ucuz sunuculardaki veritabanları üzerinde tutabilmeyi sağlıyor.