Sorting, Limiting, and Pagination

ORDER BY, LIMIT/OFFSET, ve NULLS FIRST/LAST -- Spring Data JPA kursundaki Pageable /Sort/Page<T>'in altta gerçekte hangi ham SQL'e dönüştüğü, bu projenin kendi gerçek topic tablosu üzerinde. PostgreSQL Foundations kategorisinin 8.'i.

Orta 20 dk

Son güncelleme: 25 Ağustos 2026
Çevrimiçi oku: https://www.learnforgex.com/tr/topics/sorting-limiting-and-pagination

---

Spring Data JPA kursundaki "Pagination, Sorting, and Projections" dersi, `Pageable`, `Sort`, ve `Page<T>` i kapsamıştı -- controller seviyesinde bir repository çağrısına çözülen bir Java API'si. Bu ders, o repository çağrısının PostgreSQL'e ulaştığında gerçekte neye dönüştüğüne bakıyor: bu projenin kendi gerçek `topic` tablosu üzerinde, ham `ORDER BY`, `LIMIT`, ve `OFFSET`.

ORDER BY: Satırları Sıralamak

Bir `ORDER BY` olmadan, PostgreSQL satır sırası hakkında hiçbir söz vermez -- "ekleme sırası" değil, "primary key sırası" değil, gerçekten belirsiz, ve birebir aynı sorgunun çalıştırmaları arasında değişebilir. `ORDER BY`, sırayı bir kaza yerine bir garanti yapan şeydir:

```
SELECT slug, sort_order FROM topic
WHERE category_id = 1
ORDER BY sort_order;
```

Bu projenin kendi gerçek `postgresql-foundations` kategorisine karşı çalıştırıldığında, bu şimdiye kadar kapsanan her topic'i, gerçekten `sort_order`da döndürür -- "Databases, Schemas, Tables, and Basic SQL Syntax"tan itibaren hepsinin sahip olduğu tam kolon, ve bu kursun kendi migration'larının her topic için açıkça ayarladığı tam değer (ilk ders için `sort_order = 1`, ikinci için `2`, ve böyle devam eder).

Birden Fazla Sıralama Anahtarı ve Yön

`ORDER BY`, her biri kendi yönüyle birden fazla kolonu kabul eder -- `ASC` (artan, varsayılan) ya da `DESC` (azalan):

```
SELECT slug, difficulty, sort_order FROM topic
ORDER BY difficulty ASC, sort_order DESC;
```

Sıralama soldan sağa uygulanır: satırlar önce `difficulty`ye göre gruplanır, ve yalnızca aynı `difficulty` değeri *içinde* sonra `sort_order`a göre sıralanırlar -- ikinci kolon yalnızca birinci tarafından çözülmemiş bırakılan eşitlikleri kırar, tüm sonucu bağımsız olarak yeniden sıralamaz.

LIMIT ve OFFSET

`LIMIT`, kaç satırın geri döneceğini sınırlar; `OFFSET`, onları döndürmeye başlamadan önce belirli sayıda satırı atlar:

```
SELECT slug FROM topic
ORDER BY sort_order
LIMIT 5 OFFSET 0;   -- ilk 5 satır

SELECT slug FROM topic
ORDER BY sort_order
LIMIT 5 OFFSET 5;   -- sonraki 5 satır
```

`LIMIT` / `OFFSET`, yalnızca bir `ORDER BY`yle eşleştirildiğinde anlam ifade eder -- olmadan, "ilk 5 satır" ve "sonraki 5 satır" baştan itibaren iyi tanımlanmış kavramlar değildir, çünkü satır sırasının kendisi garanti edilmez.

Sonuçlar Arasında Sayfalamak: Gerçek Bir Örnek

`ORDER BY`yi artan offset'lerle `LIMIT` / `OFFSET`le zincirlemek, bir sonuç sayfasının tam olarak nasıl, seferde bir sayfa, inşa edildiğidir:

```
-- Sayfa 1 (ilk sayfa, sayfa başına 3 topic)
SELECT slug FROM topic
ORDER BY sort_order
LIMIT 3 OFFSET 0;

-- Sayfa 2
SELECT slug FROM topic
ORDER BY sort_order
LIMIT 3 OFFSET 3;

-- Sayfa 3
SELECT slug FROM topic
ORDER BY sort_order
LIMIT 3 OFFSET 6;
```

Bu projenin kendi `postgresql-foundations` kategorisine karşı çalıştırıldığında, sayfa 1 bu kursun ilk üç dersini döndürür; sayfa 3 tam olarak bu dersin ve komşularının üzerine iner. Sayfa boyutu `s` olan sayfa *n* (sıfır-indeksli) için `OFFSET`, her zaman `n * s`dir -- sayfalanmış bir API endpoint'inin SQL'e ulaşmadan önce içeride yaptığı aynı aritmetik.

Büyük Tablolarda OFFSET'in Maliyeti

`OFFSET`, satırları bedavaya atlamaz -- PostgreSQL, gerçekten istenen satırları döndürmeye başlayabilmeden önce onların HER BİRİNİ hâlâ taramalı ve atmalıdır, bu yüzden `OFFSET 100000`, ikisi de aynı sayıda satır döndürse bile `OFFSET 10`dan anlamlı ölçüde daha fazla iş yapar. Bu projenin kendi tabloları farkın önemli olması için çok küçük, ama şimdi adlandırılmaya değer: bu kursta daha sonra "Indexes and Query Performance with EXPLAIN", buna özellikle geri döner, önemli olduğu durumlar için daha hızlı bir alternatifle (keyset pagination) birlikte.

Ham SQL'den Pageable/Page'e: Spring Data JPA Altta Ne Yapıyor

"Pagination, Sorting, and Projections" Java tarafında `Pageable` / `Sort` / `Page<T>` i zaten kapsamıştı -- bir repository metoduna geçirilen `PageRequest.of(1, 3, Sort.by("sortOrder"))`, yukarıdaki sayfa-2 sorgusunun tam olarak Java-seviyesi eşdeğeridir, ve Spring Data JPA onu tam olarak bu şekle çevirir: bir `ORDER BY sort_order`, bir `LIMIT 3`, ve bir `OFFSET 3`, artı `Page<T>` in toplam eleman/sayfa sayılarını hesaplamak için arka planda ikinci bir `SELECT count(*)` sorgusu. Burada Java API'sinin kendisi hakkında yeni bir şey açıklanmasına gerek yok -- önemli olan, `Pageable` in `LIMIT` / `OFFSET`den ayrı bir mekanizma olmadığını, yalnızca bu dersin elle az önce yazdığı tam SQL'i üreten tipli bir sarmalayıcı olduğunu tanımak.

NULLS FIRST / NULLS LAST

PostgreSQL, varsayılan olarak `NULL` değerlerini herhangi bir gerçek değerden daha büyük sıralar, bu da düz bir `ORDER BY estimated_minutes` in her `NULL` ı artan sırada *sona*, azalan sırada ise tam *başa* koyduğu anlamına gelir -- açıkça bilinmeye değer, çünkü bu projenin kendi `topic.estimated_minutes` i null-olabilir ("PostgreSQL Data Types" ve "SELECT and Filtering" ikisi de onu null-olabilir bir kolonun çalışan örneği olarak kullandı). `NULLS FIRST` / `NULLS LAST`, o varsayılanı doğrudan geçersiz kılar:

```
SELECT slug, estimated_minutes FROM topic
ORDER BY estimated_minutes ASC NULLS LAST;
```

Bu projenin kendi `topic` tablosundaki her gerçek satırın `estimated_minutes` i ayarlanmış durumda, bu yüzden bu spesifik sorgu şu anda pratikte hiçbir `NULL` yüzeye çıkarmaz -- ama cümle tam olarak bugün var olan belirli bir satır için değil, kolon türü için oradadır ("PostgreSQL Data Types" onun gerçekten null-olabilir olduğunu zaten kurdu).

Yaygın Yanlış Anlamalar

"Satırlar, aksi söylenmedikçe eklendikleri sırayla geri gelir." Hiç garanti edilmez -- PostgreSQL, kendi sorgu planının en ucuz bulduğu herhangi bir sırada satır döndürmekte

özgürdür, ki bu çoğu zaman küçük, basit bir tabloda ekleme sırasına benzer, bu yanlış anlamayı bir tablo büyüyene ya da bir sorgu planı değişene kadar doğru hissettiren tam olarak budur. **"`ORDER BY` siz `LIMIT`, güvenilir şekilde 'ilk N' satırı döndürür."** *Bazı* N satır döndürür, ama hangi N'in, ve hangi sırada olduğu belirsizdir -- "ilk" hiç kurulmamış bir sırayı ima eder. **"`Pageable` ve ham SQL sayfalama farklı özelliklerdir."** İki farklı katmandaki aynı özelliktir -- `Pageable`, bu dersin kapsadığı tam `ORDER BY` / `LIMIT` / `OFFSET` i üretmenin tipli bir yoludur, alternatif bir mekanizma değil.

Best Practices

- `LIMIT` / `OFFSET` i her zaman açık bir `ORDER BY` yle eşleştir -- bu projenin kendi örnekleri hiçbir zaman birini diğeri olmadan kullanmaz, çünkü "ilk 5" ve "sonraki 5", ilk ya da sonraki *olunacak* tanımlı bir sıra olmadan anlamsızdır.
- Birden fazla `ORDER BY` kolonunu, eşitlik kırma için gerçekte önemli olan sırayla listele, ve ilk kolonun tek başına daha sonrakileri zaten alakasız kılmadığından emin ol (örneğin önce bir `UNIQUE` kolona göre sıralamak, eşitlik kırılacak hiçbir şey bırakmaz).
- PostgreSQL'in varsayılanına (`NULL` lar artanda sonda, azalanda başta sıralanır) güvenmek yerine, null-olabilir herhangi bir sıralama kolonunda `NULLS FIRST` / `NULLS LAST` konusunda açık ol -- bir okuyucunun bir sorgunun çıktısını tahmin etmek için o kuralı hatırlamak zorunda kalmaması gerekir.
- `Pageable` in `PageRequest.of(page, size, sort)` ini doğrudan `LIMIT size OFFSET page * size` artı bir `ORDER BY` hesaplayan bir şey olarak tanı -- yavaş, sayfalanmış bir endpoint hakkında akıl yürütmek, ayrı bir Java-seviyesi maliyet modeli değil, bu SQL şekli hakkında akıl yürütmek demektir.

Yaygın Hatalar

- Offset'in yanlış sayfa boyutuyla hesaplandığı bir sayfa boyutu ve offset kombinasyonu geçmek (örneğin sayfa-boyutu-3 bir sorguya karşı sayfa-boyutu-10 bir offset formülünü karıştırmak) -- sorgu hatasız çalışır ve fark edilmesi bariz olacak boş bir sonuç değil, gerçekten yanlış bir sayfa satır döndürür.
- `ORDER BY` siz bir `LIMIT` in birebir aynı sorgunun tekrarlanan çalıştırmaları arasında deterministik olduğunu varsaymak -- özellikle altta yatan veri değiştikten sonra, aynı ifade bir sonraki çalıştırıldığında sessizce farklı bir satır kümesi döndürebilir.
- `OFFSET` in maliyetinin offset'in kendisiyle büyüdüğünü unutmak, sonra sayfalanmış bir endpoint'in "daha sonraki bir sayfasının" büyük bir tabloda sayfa birden ölçülebilir şekilde daha yavaş olduğunu görünce şaşırmak -- bir hata değil, bu dersin adlandırdığı ve "Indexes and Query Performance with EXPLAIN"in geri döneceği gerçek, iyi bilinen bir maliyet.

- `NULL`ların nereye düşmesi gerektiğine karar vermeden null-olabilir bir kolona göre sıralamak, sonra yalnızca `ASC` vs. `DESC`e bağlı olarak sonucun beklenmedik bir ucunda görünmelerini yaşamak -- varsayılana bırakılmaması, `NULLS FIRST` / `NULLS LAST`le açıkça verilmesi gereken bir karar.

Özet, Cheat Sheet ve Terimler Sözlüğü

Özet

- Satır sırası, açık bir `ORDER BY` olmadan hiçbir zaman garanti edilmez; birden fazla sıralama kolonu soldan sağa uygulanır, her biri kendinden öncekinin çözümlenmemiş bıraktığı eşitlikleri kırar.
- `LIMIT`, döndürülen satır sayısını sınırlar; `OFFSET`, satırları döndürmeden önce onları atlar -- ikisi de yalnızca bir `ORDER BY`yle birlikte anlamlıdır.
- Sonuçlar arasında sayfalamak, artan offset'lerde tekrarlanan `ORDER BY` artı `LIMIT <boyut> OFFSET <sayfa * boyut>`dur -- "Pagination, Sorting, and Projections"ta zaten kapsanan `Pageable`in tam olarak derlendiği şey.
- `OFFSET` bedava değildir -- PostgreSQL atlanan her satırı tarar ve atar, bu yüzden maliyet offset'in kendisiyle büyür; bu kursta daha sonra "Indexes and Query Performance with EXPLAIN", bunu daha hızlı bir alternatifle birlikte düzgünce kapsar.
- `NULL`, varsayılan olarak herhangi bir gerçek değerden daha büyük sıralanır (artanda son, azalanda ilk); `NULLS FIRST` / `NULLS LAST`, bu projenin kendi `topic.estimated_minutes`i gibi null-olabilir herhangi bir sıralama kolonunda yerleşimi açık yapar.

Cheat Sheet

```
SELECT a FROM t ORDER BY a;
SELECT a FROM t ORDER BY a DESC;
SELECT a, b FROM t ORDER BY a ASC, b DESC;
SELECT a FROM t ORDER BY a LIMIT 10;
SELECT a FROM t ORDER BY a LIMIT 10 OFFSET 20;
SELECT a FROM t ORDER BY a NULLS LAST;
```

```
sayfa n (sıfır-indeksli), boyut s  →  LIMIT s OFFSET (n * s)
```

Terimler Sözlüğü

- **ORDER BY**: garanti edilmiş bir satır sırası kuran cümle; olmadan, satır sırası belirsizdir.
- **LIMIT**: bir sorgunun döndürdüğü satır sayısını sınırlar.
- **OFFSET**: bir sorgu döndürmeye başlamadan önce belirli sayıda satırı atlar; atlanan satır sayısıyla orantılı iş yapar.
- **Keyset pagination**: büyük tablolar için `OFFSET`-tabanlı sayfalamaya daha hızlı bir alternatif, burada adlandırılan ve "Indexes and Query Performance with EXPLAIN"da düzgünce kapsanan.