JOINs

INNER/LEFT/RIGHT/FULL JOIN, bu projenin kendi gerçek topic→category→course zinciriyle ve gerçek TopicRepository.findBySlugWithCategoryAndCourse JPQL join fetch'iyle -- ve "henüz İngilizce'de yayınlanmamış topic'leri bulma" gibi gerçek bir LEFT JOIN örneğiyle. PostgreSQL Foundations kategorisinin 9.'u.

Orta 25 dk

Son güncelleme: 25 Ağustos 2026
Çevrimiçi oku: https://www.learnforgex.com/tr/topics/joins

Bu kursta şimdiye kadarki her sorgu, seferde bir tablodan okudu. Bu projenin kendi gerçek domain'i -- `course`, `category`, `topic`, `topic_translation` -- bilinçli olarak dört tabloya bölünmüş durumda, tam olarak "PostgreSQL ve İlişkisel Model"in ilk kurduğu şekilde, ve bir `JOIN`, tek bir sorgunun bu bölünme boyunca ilişkili satırları geri bir araya getirme yoludur. Spring Data JPA'nın kendi `@OneToMany` / `join fetch` JPQL'i, "Relationships, Fetching, and the N+1 Problem"den, bir katman yukarıda ilişkili bir problemi çözer -- bu ders altındaki ham SQL hakkında.

JOIN'lere Neden İhtiyaç Var: Tablolara Bölünmüş Veri

Bir `category` satırı yalnızca bir `course_id` saklar -- kursun `name`ini tekrarlamaz. Bir kategoriyi kendi kursunun adıyla birlikte tek bir sonuçta almak, bu yüzden iki tabloyu, satır satır, foreign key ilişkilerinin onları bağladığı her yerde birleştirmek anlamına gelir. O birleştirme işlemi bir `JOIN`dur -- neredeyse her zaman bir foreign key eşitliği olan bir koşula göre iki tablodan satırları eşleştirmekten daha egzotik hiçbir şey değil.

INNER JOIN: Yalnızca Eşleşen Satırlar

```
SELECT c.name AS course_name, cat.name AS category_name
FROM category cat
INNER JOIN course c ON cat.course_id = c.id;
```

`INNER JOIN` (çoğu zaman yalnızca `JOIN` olarak yazılır, `INNER` ima edilir), yalnızca her iki tarafta da bir eşleşmesi olan satırları döndürür -- eşleşen bir `course`u olmayan bir `category` satırı ("Constraints and Keys"in zaten kapsadığı `NOT NULL REFERENCES` sayesinde burada gerçekte hiç olamaz) basitçe görünmezdi. `ON`, join koşulunu belirtir, neredeyse her zaman

bir foreign key'i işaret ettiği primary key'e eşitleyerek; `AS`, her tabloya kısa bir takma ad (`c`, `cat`) verir, böylece her iki taraftan kolonlar da -- özellikle burada olduğu gibi iki tablo tam olarak `name` adlı bir kolona sahip olduğunda -- belirsizlik olmadan referans verilebilir.

Bu Projenin Kendi topic → category → course Zinciri, Ham SQL Olarak

Aynı deseni bu projenin gerçek içerik hiyerarşisinin üç seviyesine de yayarak:

```
SELECT t.slug, cat.name AS category_name, c.name AS course_name
FROM topic t
INNER JOIN category cat ON t.category_id = cat.id
INNER JOIN course c ON cat.course_id = c.id
WHERE t.slug = 'joins';
```

Bu projenin kendi gerçek verisine karşı çalıştırıldığında, bu tam olarak tek bir satır döndürür: `joins`, `PostgreSQL Foundations`, `PostgreSQL` -- bu dersin kurs hiyerarşisindeki kendi yerinin her parçası, tek bir sorguda üç ayrı tablodan çekilerek. Her `INNER JOIN`, kombinasyona bir tablo daha ekler; zincirlendikleri sıra sonucu değiştirmez, yalnızca (prensipte) PostgreSQL'in onu içeride nasıl çalıştırmayı seçebileceğini.

JPQL join fetch'ten Gerçek Bir SQL JOIN'e

Bu projenin kendi `TopicRepository`sinde, bir katman yukarıda, tam olarak bunu yapan gerçek bir metod zaten var:

```
@Query("select t from Topic t join fetch t.category c join fetch c.course where t.slug = :
slug")
Optional<Topic> findBySlugWithCategoryAndCourse(String slug);
```

"Relationships, Fetching, and the N+1 Problem", `join fetch`'i `t.category` ve `c.course` için ayrı takip sorguları vermeyi önleyen teknik olarak zaten kapsamıştı -- burada, SQL seviyesinde açıkça adlandırılmaya değer olan şey, bu JPQL'in esasen yukarıda elle yazılan aynı üç-tablolu `INNER JOIN` zincirine derlenmesi. `join fetch`, SQL'in `JOIN`ından farklı bir tür join değildir -- Hibernate'in, Java-seviyesi bir "bu ilişkili entity'yi de yükle" talimatını, ikinci bir gidiş-dönüş sorgusu yerine gerçek bir SQL `JOIN`u olarak ifade etmeyi seçmesidir.

LEFT JOIN: Eşleşmeyen Satırları da Tutmak

`LEFT JOIN` (`LEFT OUTER JOIN` olarak da yazılır), sağ tarafta bir eşleşme bulsun ya da bulmasın, sol taraftaki tablonun her satırını tutar -- eşleşme olmadığında, sağ tarafın kolonları basitçe `NULL` olarak geri gelir:

```
SELECT t.slug, tt.title
FROM topic t
LEFT JOIN topic_translation tt ON tt.topic_id = t.id AND tt.language = 'en';
```

Her `topic` satırı burada en az bir kez görünür, hiç İngilizce `topic_translation` satırı olmayan biri bile -- `tt.title`, bir `INNER JOIN`ın onu sonuçlardan sessizce kaybettireceği şekilde değil, onun için basitçe `NULL` olur. İkisi arasındaki gerçek fark bu: `INNER JOIN`, "eşleşmesi olan satırlar"ı cevaplar; `LEFT JOIN`, "soldaki her satır, artı varsa bir eşleşme"yi cevaplar.

Gerçek Bir LEFT JOIN: Henüz İngilizce'de Yayınlanmamış Topic'leri Bulmak

Bu projenin kendi iki-adımlı yayın iş akışı -- bir topic'in Türkçe çevirisi hemen yayına alınır, İngilizce çevirisi CLAUDE.md'nin belgelediği ve bu kursun kendi migration'larının (`connecting-to-postgresql`in `V402`si sonra `V403`ü) gösterdiği gibi daha sonra takip eder -- tam olarak bir `LEFT JOIN`ın inşa edildiği türden bir durum:

```
SELECT t.slug
FROM topic t
LEFT JOIN topic_translation en
    ON en.topic_id = t.id AND en.language = 'en' AND en.published = true
WHERE en.id IS NULL;
```

Bu, henüz yayınlanmış bir İngilizce çevirisi olmayan her `topic` slug'ını döndürür -- bu projenin kendi şemasına karşı gerçek, kullanışlı bir sorgu, varsayımsal bir tane değil. Sondaki `en.id IS NULL` kontrolü, "sağda eşleşmesi olmayan soldaki satırları bul" için standart deyimdir -- tam olarak çalışır çünkü gerçekten eşleşmeyen bir satırın sağ-taraf kolonları `NULL` olarak geri gelir, "SELECT and Filtering"in genel olarak `IS NULL` için zaten kapsadığı aynı `NULL`-eksik-anlamına-gelir davranışı.

RIGHT JOIN ve FULL JOIN

`RIGHT JOIN`, `LEFT JOIN`ın ayna görüntüsüdür -- sol yerine sağ tarafın her satırını tutar. Pratikte nadiren gerekir (ve bu projenin kendi kodu onu hiç kullanmaz) çünkü bir `LEFT JOIN` da iki tablonun sırasını değiştirmek özdeş sonucu üretir -- `A LEFT JOIN B` ile `B RIGHT JOIN A`, aynı satırları döndürür, yalnızca kolonlar farklı sırada. `FULL JOIN` (ya da `FULL OUTER JOIN`), eşleşme olsun olmasın *her iki* taraftan da her satırı tutar, karşılığı olmayan tarafa `NULL` doldurarak -- gerçekten simetrik karşılaştırmalar için kullanışlı (birbirini yansıtması gereken iki tablo arasındaki her eşleşmezliği bulmak gibi), bu projenin kendi domain'inin şu anda ihtiyaç duymadığı bir durum.

Yaygın Yanlış Anlamalar

"Bir JOIN, iki tabloyu kalıcı olarak tek bir tabloya birleştirir." Birleştirmez -- bir `JOIN`, yalnızca tek bir sorgunun sonuç kümesi süresince satırları birleştirir; altta yatan tablolar (`category`, `course`), "Databases, Schemas, Tables, and Basic SQL Syntax"in ilk gösterdiği kadar ayrı kalır. **"`LEFT JOIN`, her zaman `INNER JOIN`dan daha yavaştır."** Doğası gereği değil -- varsa performans farkı, join türünün kendisine değil, "Indexes and Query Performance with EXPLAIN"da düzgünce kapsanan index'lere ve satır sayılarına bağlıdır. **"JPQL'deki `join fetch`, bir SQL JOIN'inden tamamen farklı bir kavramdır."** Değildir -- birebir aynı ilişkisel işlemdir, yalnızca elle yazılmak yerine Java/JPQL tarafından tetiklenir.

Best Practices

- Çok-tablolu bir sorguda her tabloyu takma adlandır (bu ders boyunca kullanılan `t`, `cat`, `c`) -- bu, iki tablo `category` ve `course`daki `name` gibi bir kolon adını paylaştığı anda kolon referanslarını belirsizlik olmadan tutar.
- Gerçek soru "hangi satırların ilişkili bir satırı eksik" olduğunda `LEFT JOIN ... WHERE <sağ>.id IS NULL`e başvur -- bu projenin kendi "yayınlanmış İngilizce çevirisi olmayan topic'ler" sorgusu, desenin gerçek, yeniden kullanılabilir bir örneği.
- Eşleşmesi olmayan satırların sonuçta hayatta kalması için özel bir neden olmadıkça varsayılan olarak `INNER JOIN`e git -- hem daha yaygın durum hem de akıl yürütmesi daha kolay olanı.
- Bir JPQL `join fetch`ini (bu projenin kendi `findBySlugWithCategoryAndCourse`si gibi) tam olarak bir SQL `JOIN`una derlenen bir şey olarak tanı -- maliyeti ya da davranışı hakkında akıl yürütmek, ayrı bir Java-seviyesi kavram değil, bu derste kapsanan aynı `JOIN` mekaniği hakkında akıl yürütmek demektir.

Yaygın Hatalar

- `ON` koşulunu tamamen unutmak, ya da gerçekte paylaşılan bir key'e referans vermeyen bir tane yazmak -- PostgreSQL onu çalıştırmayı reddetmez, ama yanlış ya da eksik bir koşula sahip bir join, hata vermek yerine sessizce satır sayılarını çarpabilir (bir cross product).
- Gerçek niyet "eşleşme olmasa bile bu satırı tut" olduğunda `INNER JOIN` kullanmak -- bu projenin kendi İngilizce-çeviri sorgusu, bir `INNER JOIN` olarak yazılsaydı, bulmak için tasarlandığı tam topic'leri sessizce dışarıda bırakırdı, çünkü eşleşmeyen bir `topic` satırı basitçe hiç görünmezdi.
- Niyet eşleşmeyen satırları tutmak olduğunda, outer-join'lenmiş bir tablonun kolonunu `ON` cümlesinde değil `WHERE`de filtrelemek -- bir `LEFT JOIN`dan sonra `WHERE en.published =`

`true`, onu sessizce bir `INNER JOIN`in eşdeğerine geri döndürür, çünkü `WHERE` join'den sonra çalışır ve o koşulun `true` olmadığı her satırı düşürür, `LEFT JOIN`ın korumak istediği `NULL` satırlar dahil.

- Çok-tablolu bir `JOIN`dan sonra bir `SELECT`teki kolon sırasının, takma adı kontrol etmeden bir kolonun hangi tabloya "ait" olduğunu yansıttığını varsaymak -- belirsiz ya da yanlış atfedilen kolon referansları, çalışan ama yanlış değeri okuyan bir sorgunun yaygın bir kaynağıdır.

Özet, Cheat Sheet ve Terimler Sözlüğü

Özet

- Bir `JOIN`, bir koşula göre -- neredeyse her zaman bir foreign key eşitliği -- iki ya da daha fazla tablodan satırları, tek bir sorgu süresince birleştirir; altta yatan tablolar ayrı kalır.
- `INNER JOIN`, yalnızca her iki tarafta da eşleşmesi olan satırları döndürür; bu projenin kendi `topic` → `category` → `course` zinciri, üç tablo derinliğinde join'lenmiş, gerçek bir örnektir.
- Bu projenin kendi `TopicRepository.findBySlugWithCategoryAndCourse`sinin JPQL `join fetch` i, esasen aynı `INNER JOIN` zincirine derlenir -- "Relationships, Fetching, and the N+1 Problem"de kavramsal olarak zaten kapsandı, burada tekrarlanmadı.
- `LEFT JOIN`, eşleşme olmasa bile sol tablodan her satırı tutar, eşleşmeyen sağ-taraf kolonlarını `NULL` ile doldurur -- bu projenin kendi gerçek "henüz İngilizce'de yayınlanmamış topic'ler" sorgusuyla gösterilen, "ilişkili bir satırı eksik satırları" bulmanın standart yolu.
- `RIGHT JOIN`, `LEFT JOIN`ı yansıtır (pratikte nadiren gerekir); `FULL JOIN`, her iki taraftan da eşleşmeyen satırları tutar.

Cheat Sheet

```
SELECT a.x, b.y
FROM a
INNER JOIN b ON a.b_id = b.id;

SELECT a.x, b.y
FROM a
LEFT JOIN b ON a.b_id = b.id
WHERE b.id IS NULL;   -- b'de eşleşmesi olmayan a'daki satırlar
```

Terimler Sözlüğü

- **JOIN**: bir koşula, tipik olarak bir foreign key eşitliğine dayanarak iki ya da daha fazla tablodan satırları birleştiren bir sorgu işlemi.
- **INNER JOIN**: yalnızca birleştirilen her iki tabloda da eşleşmesi olan satırları döndürür.
- **LEFT JOIN**: sol tablodan her satırı döndürür, eşleşme olmadığında sağda `NULL` ile.

- **Alias**: bir sorguda bir tabloya verilen kısa bir ad (`t` , `cat` , `c`), join'lenmiş tablolar arasında paylaşılan kolonların belirsizliğini gidermek için kullanılır.