Rencontre R20

Performance

Bases de données et programmation Web

Aller chercher les bonnes infos

- Il est important d'aller chercher uniquement les informations nécessaires.
 - ◆ Ce sera d'autant plus important lorsque vous travaillerez en entreprises.
 Celles-ci utilisent généralement un cloud storage pour gérer les BD.
 - Ce type de service facture le client selon le volume des requêtes exécutées.
 - ♦ C'est aussi très important lorsque vous développez des applications mobiles.

Performance

❖ Au niveau de la BD:

♦ Utiliser les vues au lieu des requêtes Link complexes.

Lorsque vous créez une vue, la requête sous-jacente est optimisée par le serveur SQL. Son plan d'exécution est mis en cache dans le serveur pour une performance optimale.

Plus les requêtes Link sont complexes, moins elles sont performantes par rapport à une vue.

Performance

❖ Au niveau de la BD:

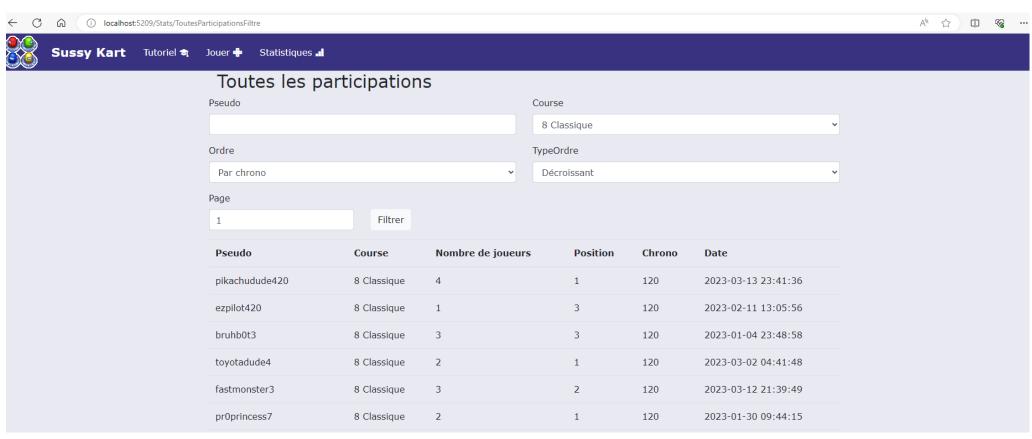
- ♦ Utiliser les index pour optimiser l'exécution des requêtes.
 - Si vous avez une requête pour voir les données sur les ventes les plus récentes, vous voudrez faire un index sur le champ DateVente DESC.
 - Les index sur les clés étrangères peuvent améliorer l'efficacité des jointures.

- ❖ Au niveau du code:
 - ◆ Utiliser des filtres avant d'aller chercher les données avec .ToListAsync()

```
List<Course> premieresCourses = await _context.Courses.Take(30).ToListAsync();
```

Au niveau du code:

Vous avez une vue Razor qui vous permet d'appliquer des filtres sur les données des participations.





Au niveau du code:

◆ Dans le controlleur STATS, dans l'action ToutesParticipationsFiltre(FiltreParticipationVM fpvm) vous allez cherchez les 25000 enregistrements de la vue des participations avec .ToListAsync().

```
// Obtenir les participations grâce à une vue SQL -- ÉTAPE 4
List<VwDetailsParticipation> participations = await _context.VwDetailsParticipations.ToListAsync();
```

◆ Puis vous appliquez les filtres désirés pour afficher finalement que 30 enregistrements.

CECI N'EST PAS PERFORMANT.

ALLER CHERCHER 25000 enregistrements pour n'en AFFICHER que 30!!!

Au niveau du code:

- ◆ Utiliser .AsQueryable() au lieu de .ToListAsync() pour obtenir un objet de type IQueryable<> puis appliquez les filtres sur cet objet.
- ♦ Utiliser ensuite .ToListAsync() sur cet objet à la fin pour aller chercher seulement les données qui nous intéresse.

```
//Construction d'une requête sur toutes les courses
IQueryable<Course> toutesLesCourses = _context.Courses.AsQueryable();
//Filtres pour garder les 30 courses les plus récentes
toutesLesCourses = toutesLesCourses.OrderByDescending(x => x.CourseId).Take(30);
//Effectivement aller chercher les données désirées dans la BD
List<Course> dernieresCourses = await toutesLesCourses.ToListAsync();
```

- **❖** IQueryable vs IEnumerable:
 - ◆ IEnumerable<> : le filtrage est fait côté client. Par conséquent, l'ensemble de la table est récupéré. Correct pour des petits jeux de données stockés localement.
 - ◆ IQueryable<> : le filtrage est fait côté serveur. Le filtrage est donc fait AVANT l'envoie sur la machine cliente. À privilégier pour de grands jeux de données stockés sur un autre serveur ou dans une base de donnée.

Performance

- Autres stratégies, au niveau du code:
 - ◆ Aller chercher uniquement les colonnes qui vous intéressent, avec .Select()
 - Surtout si certaines colonnes de la table, que vous ne voulez pas voir, sont 'lourdes' comme nvarchar(max) par exemple.

```
C# Copier

foreach (var blog in context.Blogs)
{
    Console.WriteLine("Blog: " + blog.Url);
}
```

ICI, on veut seulement voir le Url du blog. Mais l'entité entière du Blog est extraite et les colonnes inutiles sont transférées à partir de la base de données.

Vous pouvez optimiser cela à l'aide de Select pour indiquer à EF les colonnes à sélectionner :

```
C#

foreach (var blogName in context.Blogs.Select(b => b.Url))
{
    Console.WriteLine("Blog: " + blogName);
}
```