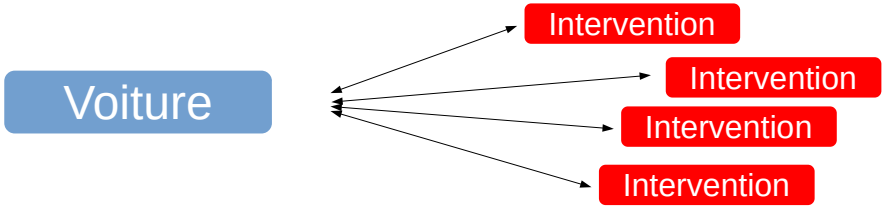
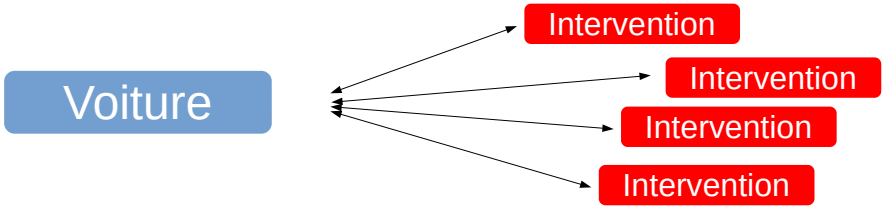
# Jpa – relations - FetchType

# Eager vs Lazy




- Les questions qui se posent  quand on veut lire des données c'est :
  - dois- je récupérer tout ?
  - Tout de suite ?
  - Ou plus tard quand j'en ferai la demande ?
- Selon le type de chargement, cela déterminera quand les données ont chargées depuis la bdd par hibernate.
- Et biensûr cela impactera les performance de votre application.

**@Eager : lorsqu'on récupère un objet qui possède une relation configurée en Eager avec d'autres objets, cela récupère tous les objets liés par cette relation dès le premier accès en base de données.**
**La jointure est réalisée et une grappe d'objets liés plus ou moins fournie, vous est retourné.**




**Ce fonctionnement peut vite devenir une affaire délicate, en fonction du volume des données et des resources en matériel disponibles.**

# Eager Loading : le cas de voiture & intervention

- Si nous cherchons une intervention  par mot clé
- Nous souhaitons que les résultats soient une liste composée des interventions qui contiennent ce mot clé et uniquement celles-ci.
- Mais un chargement eager va nous ramener toutes les interventions liées.
- La bonne pratique c'est de ne charger que les données nécessaires. On va préférer le Lazy Loading pour ces raisons.(il y a cependant des exceptions)

# Lazy Loading

- LE principe c'est qu'on charge d'abord l'entité principale qui est requise.
- Les entités qui sont liées seront éventuellement chargées si on en fait la demande programmatiquement.
- Par exemple on va charger une voiture , et quand on aura besoin de consulter une intervention réalisée sur ce véhicule on pourra y accéder à ce moment là.

# Spécifier le FetchType

- Nous n'avons pas configuré  ce paramètre pour l'instant , nous avons utilisé celui qui configuré par défaut.

| Mapping | Default Fetch Type |
|---|---|
| **@OneToOne** | FetchType.EAGER |
| **@OneToMany** | FetchType.LAZY |
| **@ManyToOne** | FetchType.EAGER |
| **@ManyToMany** | FetchType.LAZY |

# Override Default fetchType

```
@ManyToOne(fetch=FetchType.LAZY)
```

# Lazy Loading & session

L'accès différé çà la donnée nécessite une session hibernate, une connection à la base pour récuéré la donnée manquante.

Si la session a été fermée, hibernate va lever une exception .

Pour mieux visualiser cela on va s'employer à développer du mauvais code pour lever cette exception, puis résoudre ce problème et le contouner.