**Bonne pratique :**

Les colonnes faisant partie d'un index doivent être petites et stables.

Un facteur important est donc la profondeur de l'index : s'il vaut 3 alors nous atteignons les feuilles en 3 ios disque.

Un bon index est de taille raisonnable par rapport aux données et est constitué d'une ou plusieurs colonnes (dans ce cas on parle d'index composite).

Dans un **index cluster**, les niveaux intermédiaires de l'arbre de tri contiennent les valeurs de la clé (et un RowID), selon l'ordre physique de celle-ci, et seul le niveau feuille de l'index contient les pages de données de la table.

En ce sens :

- il ne peut y avoir qu'un seul index cluster par table,  
- un index cluster est plus sujet aux splits de page (mais on peut les éviter en spécifiant la colonne comme auto-incrémentée),  
- un index cluster est un doublon de la table,  
- une analogie en est l'index que vous trouvez au début d'un livre : il montre de quelle façon est organisé le livre dans l'ordre physique des pages de celui-ci.

Sous SQL Server, les indexes cluster sont créés :

- implicitement lors de la spécification de la clé primaire de la table,  
- lors de l'ajout d'une contrainte d'unicité spécifiant CLUSTERED (si la table ne comporte pas déjà de clé primaire).

Un **index non-cluster** est ordonné suivant l'ordre logique des valeurs de sa clé (et un keyID), et son niveau feuille ne contient pas les pages de données, mais les lignes de l'index cluster.

En ce sens :

- il peut y avoir plusieurs indexes non-cluster sur une table,  
- un index non-cluster est moins sujet aux splits de page,  
- une analogie en est l'index que vous trouvez à a fin d'un livre : il est ordonné suivant les titres des sujets qui sont traités, mais il référence la clé physique du livre : les numéros de page.

Sous SQL Server, tous les indexes sont implicitement créés non-cluster, à l'exception des indexes de clé primaire.

Dès lors, on comprend que le choix de la clé de l'index cluster d'une table est très important, puisqu'il influe sur la qualité des indexes non-cluster qui seront posés sur la même table.

Le choix des indexes reste complexe et à réaliser avec parcimonie, parce qu'il est difficile de trouver le parfait arbitrage entre les performances des requêtes et le coût de mise à jour de ceux-ci.  
Pour la préservation des performances générales d'une base de données, leur maintenance et un audit sur leur qualité et leur [absence](http://blog.developpez.com/elsuket/p6345/indexation/title-107/#more6345) doit être régulièrement effectué.

La diminution du nombre de lectures est bien sûr proportionnelle à la taille de la table : plus le volume est important et plus le rapport Nombre de lignes cherchées / nombre de lignes de la table est petit, et plus le gain est important.

Par habitude, je classe les tables en 4 catégories

 Petites tables : moins de 10000 lignes.

 Tables moyennes : de 10000 à 100000 lignes.

 Tables importantes : de 100000 à 1000000 lignes.

 Tables très importantes : plus de 1000000 de lignes.

**Bonne pratique :**

On évitera de créer des index sur des colonnes telles que Sexe (Homme, Femme) ou NomJourSemaine

En première approche, il y a deux types d’index :

• Les index clustered dont les feuilles sont en fait les lignes de la table : on a donc une fusion très intime de la table et de l’index.

• Les index non clustered plus indépendants par rapport aux tables.

Un index clustered implique que la table est rangée physiquement dans l’ordre de l’index : de ce fait , il ne peut y avoir au maximum qu’un index clustered par table.