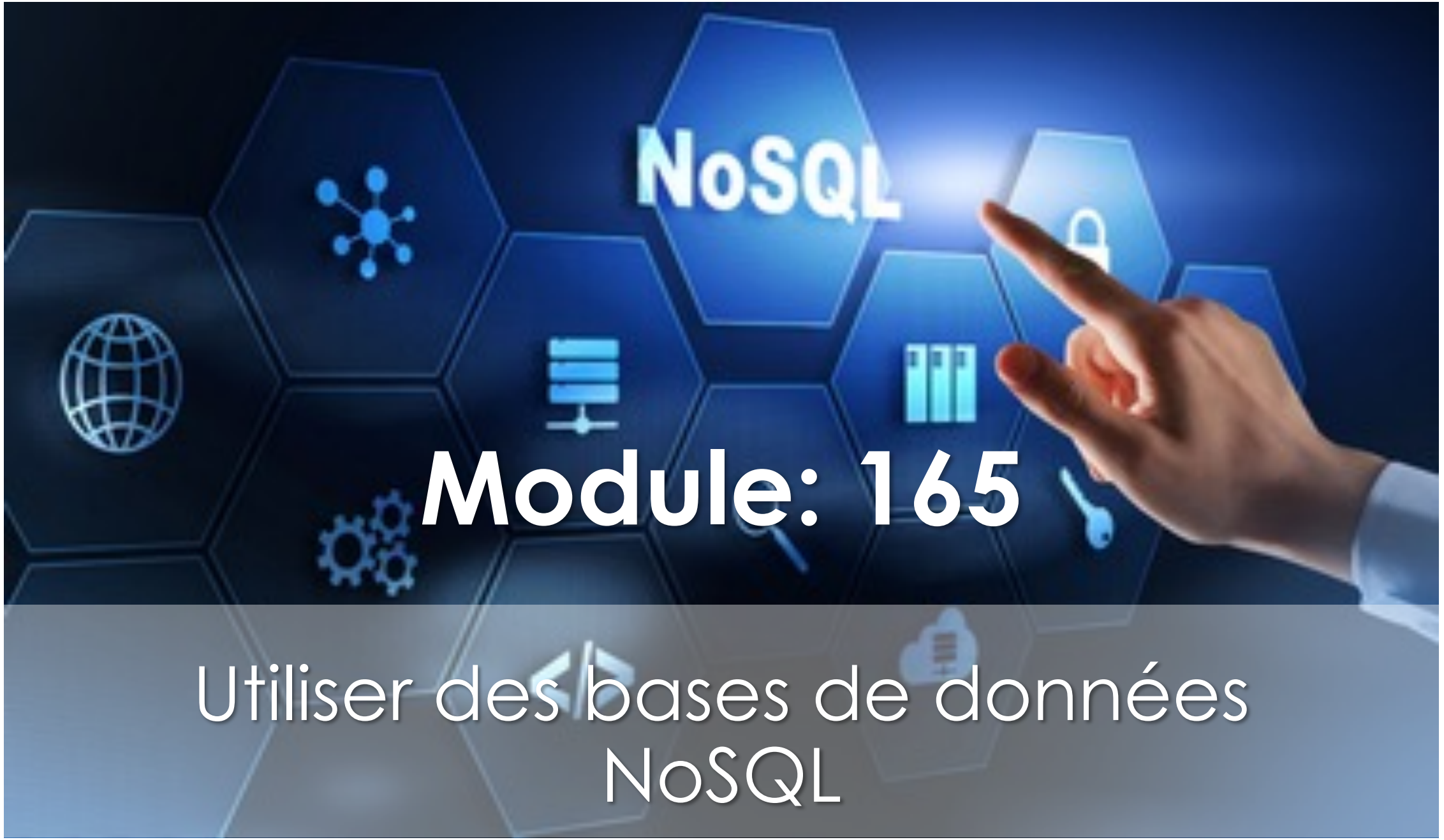

Module: 165

Utiliser des bases de données NoSQL

ETML Informatique

ETML Informatique

MongoDB

Autorisations et privilèges

MongoDB
Introduction

L'autorisation est un élément essentiel de la gestion des utilisateurs et du contrôle d'accès qui définit les politiques relatives à ce que chaque utilisateur est autorisé à faire sur le système. Décider quelles politiques sont appropriées et les mettre en œuvre dans vos bases de données garantit que les utilisateurs peuvent interagir avec les ressources dont ils ont besoin tout en se protégeant contre les comportements inappropriés.

MongoDB
Condition préalable

Pour ajuster la configuration de MongoDB et activer l'autorisation sur la base de données, vous avez besoin d'un accès de niveau « **root** » sur le serveur.

De plus, vous aurez besoin d'un compte disposant au moins du role « **userAdmin** ».

MongoDB
Comment fonctionne l'autorisation?

La gestion des autorisations et des privilèges dans MongoDB est implémentée à l'aide du contrôle d'accès basé sur les rôles (RBAC).

Dans ce système, différents niveaux d'accès sont associés à des rôles individuels. Pour accorder à un utilisateur l'autorisation d'effectuer une action, vous lui accordez l'appartenance à un rôle disposant des privilèges appropriés.

ETML Informatique

MongoDB

Les rôles

MongoDB
Rôles intégrés

Voici les rôles intégrés:

read	**readWrite**	
dbAdmin	**dbOwner**	**userAdmin**
backup	**restore**	

Les rôles dans MongoDB peuvent être imbriqués. Cela signifie que des rôles peuvent être attribués à d'autres rôles. Un rôle qui contient un autre rôle hérite de tous les privilèges du rôle enfant. Cela permet de créer de nouveaux rôles dotés des privilèges souhaités en combinant les rôles de manière appropriée.

MongoDB
Quelles actions sont disponibles?

Il existe un grand nombre d'actions disponibles dans MongoDB liées à l'utilisation générale, à la gestion des bases de données et à la gestion du système. En général, les actions correspondent à une ou plusieurs commandes ou méthodes au sein de MongoDB.

Voici certaines actions :

find	**insert**	**remove**
update	**createCollection**	**createIndex**
createRole	**dropCollection**	**dropIndex**

MongoDB
Créer un nouveau rôle

La méthode **db.createRole()** vous permet de définir un nouveau rôle auquel vous pouvez attribuer des privilèges et d'autres rôles.

À titre d'exemple, créons un rôle appelé salesMonitor qui fournit un accès en lecture seule à la base de données sale :

```
db.createRole({
    role: "salesMonitor",
    privileges: [],
    roles: [
        {
            role: "read",
            db: "sales"
        }
    ],
})
```

MongoDB
Créer un nouveau rôle plus limité

Nous pourrions exprimer un rôle similaire (mais plus limité) en utilisant le champ privileges au lieu du rôle read :

```
db.createRole({
    role: "salesMonitor",
    privileges: [
        {
            resource: { db: "sales", collection: "" },
            actions: [ "find" ]
        }
    ],
    roles: []
})
```

MongoDB
Afficher les rôles

Pour afficher tous les rôles disponibles sur le système, y compris tous les rôles intégrés et leurs privilèges associés, utilisez la méthode **db.getRoles()** avec les options showPrivileges et showBuiltinRoles définies sur **true** :

```
db.getRoles({
    rolesInfo: 1,
    showPrivileges: true,
    showBuiltinRoles: true
})
```

MongoDB
Afficher les informations d'un rôle

Pour obtenir des informations sur un rôle spécifique, utilisez plutôt la méthode **db.getRole()**. Vous devez être sur la base de données où l'utilisateur est défini avant d'exécuter la commande :

```
use admin
db.getRole(
    "root",
    {
        showPrivileges: true,
        showBuiltinRoles: true
    }
)
```

MongoDB
Afficher les rôles des utilisateurs

Pour vérifier quels rôles ont été accordés à chaque utilisateur, accédez à la base de données qui vous intéresse et utilisez la méthode **db.getUsers()** :

```
use admin
db.getUsers()
```

MongoDB
Afficher les rôles d'un utilisateur

Pour vérifier les rôles associés à un utilisateur spécifique, utilisez plutôt **db.getUser()** :

```
use admin
db.getUsers()
```

MongoDB
Ajouter des rôles à un utilisateur

La méthode **db.grantRolesToUser()** vous permet de spécifier des rôles supplémentaires que vous souhaitez ajouter à un utilisateur.

```
db.grantRolesToUser(
    "sally",
    [
        "read",
        "backup"
    ]
)
```

```
db.grantRolesToUser(
    "sally",
    [
        { role: "read", db: "sales"},
        { role: "readWrite", db: "callLogs"}
    ]
)
```

MongoDB
Supprimer des rôles à un utilisateur

Pour révoquer les rôles d'un utilisateur, vous pouvez utiliser la méthode compagnon appelée **db.revokeRolesFromUser()**.

```
db.revokeRolesFromUser(
    "sally",
    [
        "read",
        "backup"
    ]
)
```

```
db.revokeRolesFromUser(
    "sally",
    [
        { role: "read", db: "sales"},
        { role: "readWrite", db: "callLogs"}
    ]
)
```

MongoDB
Gérer les rôles personnalisés

db.grantPrivilegesToRole()	Pour accorder des privilèges supplémentaires à un rôle défini par l'utilisateur existant.
db.revokePrivilegesFromRole()	Pour supprimer des privilèges des rôles personnalisés.
db.grantRolesToRole()	Pour ajouter les privilèges définis par un rôle à un autre role.
db.revokeRolesFromRole()	Pour supprimer des rôles de rôles personnalisés
db.updateRole()	Pour redéfinir les caractéristiques d'un rôle défini par l'utilisateur.
db.dropRole()	Suppression de rôles définis par l'utilisateur

ETML Informatique

MongoDB
Les utilisateurs

MongoDB
Créer un utilisateur

L'opération suivante ajoute un utilisateur « myTester » à la base de données « test » qui a le rôle « readWrite » dans la base de données ainsi que le rôle « read » dans la base de données « reporting » :

```
use test
db.createUser(
  {
    user: "myTester",
    pwd:  passwordPrompt(),   // or cleartext password
    roles: [ { role: "readWrite", db: "test" },
             { role: "read", db: "reporting" } ]
  }
)
```

MongoDB
Afficher les utilisateurs existants

Pour renvoyer plusieurs utilisateurs, vous pouvez utiliser la méthode **db.getUsers()** pour afficher tous les utilisateurs de la base de données actuelle.

MongoDB
Afficher un utilisateur

Vous pouvez éventuellement inclure un objet **args** supplémentaire qui vous permet de spécifier les informations supplémentaires que vous souhaitez en définissant les clés suivantes à true :

```
use admin
db.getUser("tom",
{
    showCredentials: true,
    showPrivileges: true,
    showAuthenticationRestrictions: true
})
```

MongoDB
Changer le mot de passe

Pour changer le mot de passe d'un utilisateur, vous pouvez utiliser la méthode **db.changeUserPassword().**

```
use admin
db.changeUserPassword("tom", passwordPrompt())
...
Enter password:
```

MongoDB
Modifier les autres détails

Pour modifier d'autres informations associées à un compte utilisateur, vous pouvez utiliser la méthode **db.updateUser().**

```
use admin
db.updateUser("tom", {
    authenticationRestrictions: [ {
        clientSource: ["127.0.0.1", "::1"],
        serverAddress: ["127.0.0.1", "::1"]
    } ]
})
```

MongoDB
Supprimer un utilisateur

Pour supprimer les comptes d'utilisateurs MongoDB, vous pouvez utiliser la méthode **db.dropUser()**.

```
use test;
db.dropUser("tom");
```

Assurez-vous de vous connecter à la base de données d'authentification de l'utilisateur avant de les supprimer.