Le langage de requête SQL avec SQL Server

Module 5

DDL - La gestion des tables

Contenu du module

- Création des tables
- Les types de données SQL Server
- Mise en œuvre de l'intégrité des données
- Modification des tables
- Suppression des tables
- Indexation des données

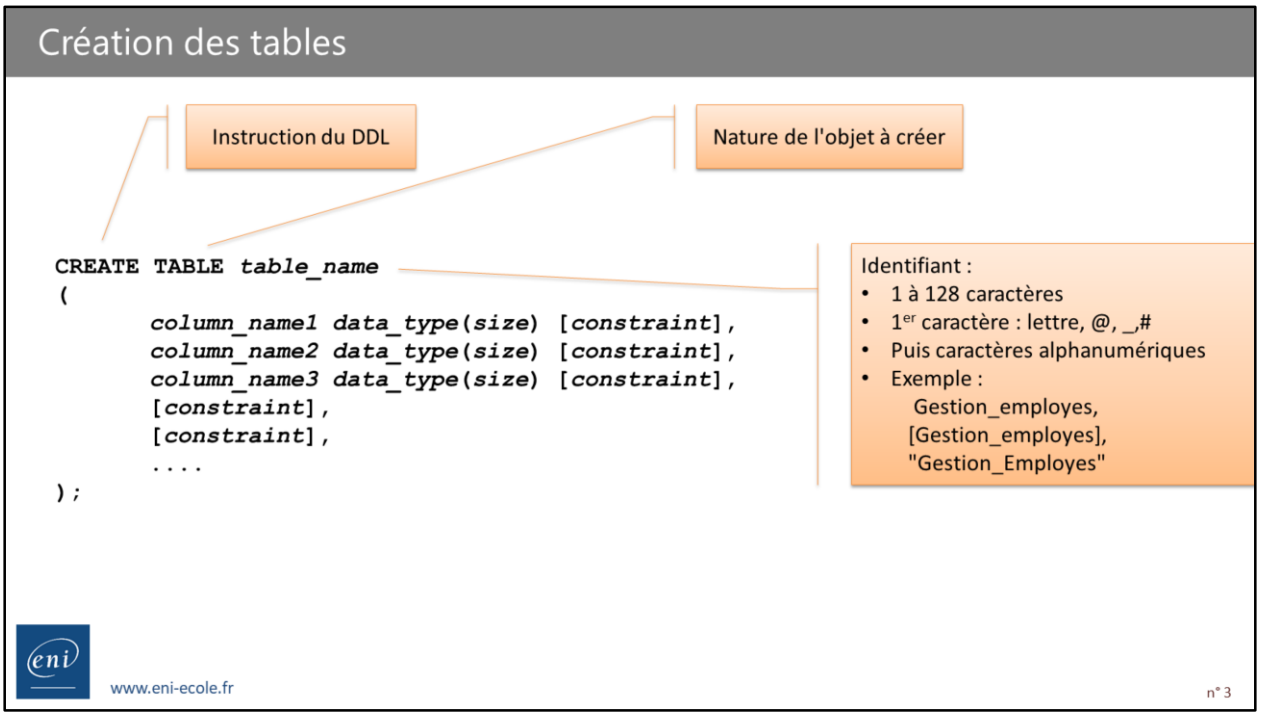
La base de données est maintenant créée, il est temps d'y structurer les données. Pour cela nous devons créer des tables, objets de la base de données définis au niveau logique de celle-ci. Elles permettent de stocker les données sous la forme de lignes et de colonnes.

SQL Server offre un assistant graphique sur lequel nous pourrions nous appuyer pour définir la structure des tables.
Simplement, la volonté de ce cours étant d'apprendre le SQL, nous réaliserons ce travail sous la forme de script SQL.
Disposer de ce script vous permettra également de récréer à volonté et rapidement ces tables.

Nous réaliserons cette gestion grâce au langage SQL et les instructions de la catégorie du DDL (Data Description Language).

Nous disposerons donc, en fin de module, des scripts SQL contenant les instructions de création, de modification et de suppression de tables.

Vous apprendrez également à traduire les règles de gestion concernant les données sous la forme de règles d'intégrités (ou contraintes).

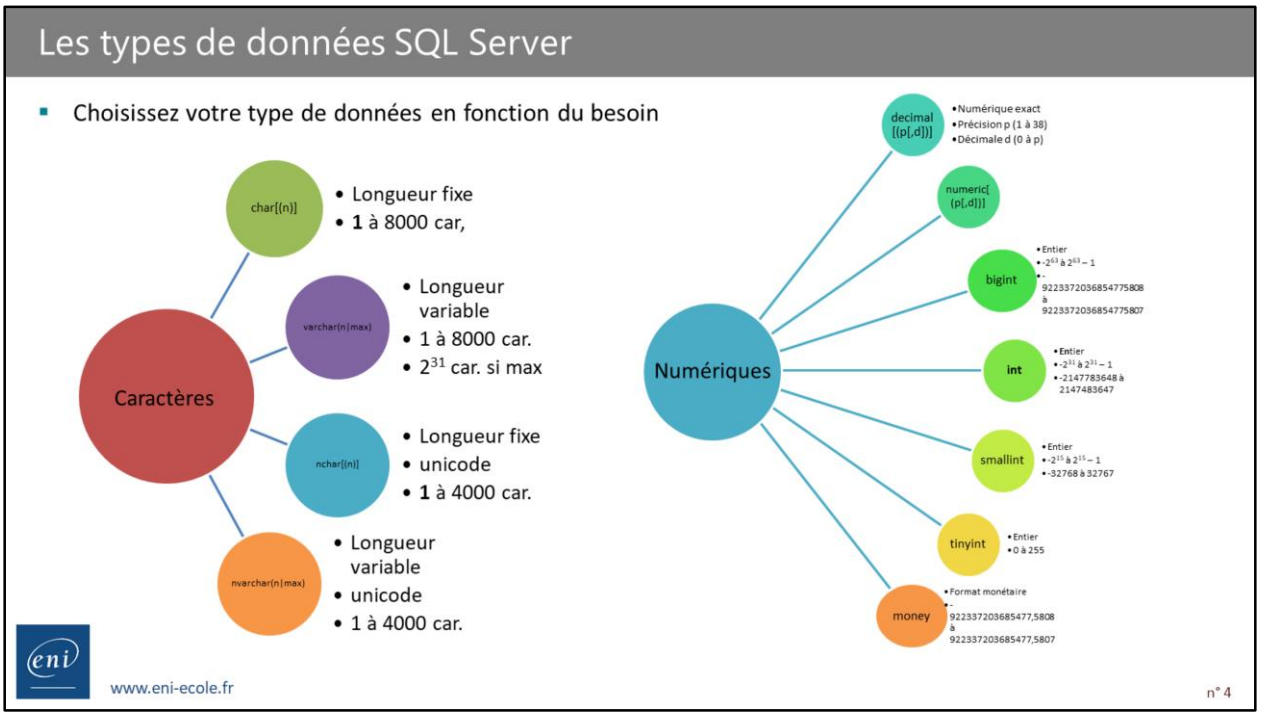


RI 55
C'est l'instruction du DDL CREATE TABLE qui permet la création d'une table.
Cette instruction permet de nommer la table, de structurer la table et d'associer des contraintes à la table.
2 objets de même type ne peuvent pas avoir le même identifiant.

Il existe 2 catégories d'identifiants d'objet :
Les identifiants réguliers : Gestion_Employes (la plus commune, **à privilégier**, la casse n'est pas importante).

Les identifiants délimités : [Gestion_Employes], "Gestion Employés" (utilisation possible des caractères spéciaux comme accentués, importance de la casse, => alourdit l'écriture des requêtes

Démarrer la manip sur la table Employés + expliquer le use en début de script



RI 56

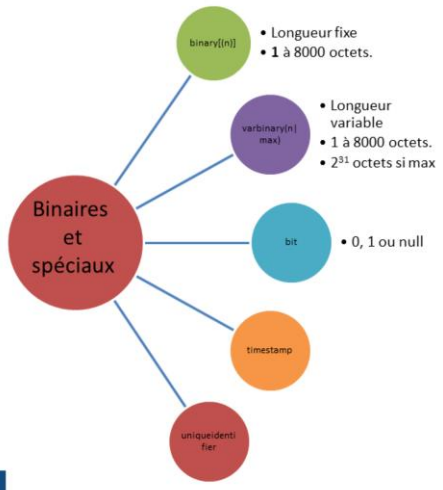
Lors de la définition d'une colonne, il est nécessaire de préciser le format d'utilisation de la donnée ainsi que son mode de stockage par le type de données.

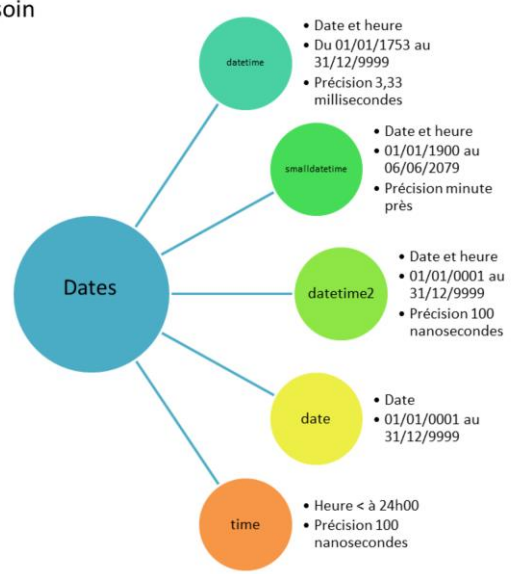
SQL Server propose un nombre de type de données important que vous pouvez consulter dans votre livre numérique. Cette diapositive et la suivante reprennent les types essentiels.

C'est au concepteur de la base de bien choisir le type de données à assigner à une colonne en maitrisant les conséquences de son choix sur l'espace disque et sur les performances.

Les types de données SQL Server

- Choisissez votre type de données en fonction du besoin





RI 56

Compatibilité avec le standard ISO

Synonyme	Type SQL Server
Caractères	
char varying	varchar
character	char(n)
character varying(n)	varchar(n)
national character(n)	nchar(n)
national char(n)	nchar(n)
national character varying(n)	nvarchar(n)
national char varying(n)	nvarchar(n)
national text	Ntext
Numériques	
dec	decimal
double precision	float
integer	int
Binaires	
binary varying	varbinary
Autres	
Rowversion	timestamp

RI 61

SQL Server propose également quelques synonymes associés à ses types de base. Ils sont souvent présents pour assurer la compatibilité avec le standard ISO. Il est donc possible de les utiliser en lieu et place des types sql server.

Les types de données

- Les colonnes typées

```
CREATE TABLE Employes(
    CodeEmp INT ,
    Nom VARCHAR(20),
    Prenom CHAR(20),
    DateNaissance DATE,
    DateEmbauche DATE,
    DateModif TIMESTAMP,
    Salaire DECIMAL(8,2),
    CodeService CHAR(5),
    CodeChef INT
);
```

Continuer la manip sur la table Employés

Les types de données SQL Server

- A partir du diagramme proposé dans le fichier en téléchargement *MLD – Gestion Employes.pdf*
  - construisez, sur le modèle de notre table Employes, les tables Services, Conges et Conges_Mens.

A vous de jouer :
A partir du diagramme proposé dans le fichier MLD – Gestion Employes.pdf, construisez, sur le modèle de notre table Employes, les tables Services, Conges et Conges_Mens.

Mise en œuvre de l'intégrité des données

- Assurer la cohérence des données
- Appliquer les règles de fonctionnement issues de l'analyse
- Traduction des règles du modèle relationnel

- Réalisable de manière procédurale par les déclencheurs (TRIGGER) ou de manière déclarative, au niveau de la structure de la table elle-même, par les contraintes (CONSTRAINT)

- **Ce cours ne traite que des contraintes.**

```
CREATE TABLE table_name
(
        column_name1 data_type(size) [constraint],
        column_name2 data_type(size) [constraint],
        column_name3 data_type(size) [constraint],
        [constraint],
        [constraint],
        ....
);
```

Contrainte niveau colonne

Contrainte niveau table

RI 72

Il est recommandé de passer, lorsque cela est possible, par des contraintes à la place des déclencheurs car les contraintes sont normalisées, déclaratives (elles limitent le codage et donc le risque d'erreur) et plus performantes.

Les contraintes peuvent être définie de niveau colonne ou de niveau table.
Une contrainte de niveau colonne permet d'exprimer une règle qui ne porte que sur une colonne.
Une contrainte de niveau table permet d'exprimer une règle à laquelle plusieurs colonnes participent.

Mise en œuvre de l'intégrité des données

- Contrainte de non nullité

```
CREATE TABLE Employes(
    CodeEmp INT             not null,
    Nom VARCHAR(20)         not null,
    Prenom CHAR(20)         null,
    DateNaissance DATE      null,
    DateEmbauche DATE       not null,
    DateModif TIMESTAMP     null,
    Salaire DECIMAL(8,2)    not null,
    Codeservice CHAR(5)     not null,
    CodeChef INT            null
);
```

RI 76

Cette règle s'applique à une colonne en particulier, elle se définit donc comme une contrainte de colonne.
Nous vous conseillons de préciser systématiquement NULL ou NOT NULL, car la valeur par défaut dépend du paramétrage mis en place au niveau de la base de données.

Mise en œuvre de l'intégrité des données

- Valeur par défaut

```
CREATE TABLE Employes(
    CodeEmp INT             not null,
    Nom VARCHAR(20)         not null,
    Prenom CHAR(20)         null,
    DateNaissance DATE      null,
    DateEmbauche DATE       not null CONSTRAINT DF_Employes_DateEmbauche DEFAULT getdate(),
    DateModif TIMESTAMP     null,
    Salaire DECIMAL(8,2)    not null CONSTRAINT DF_Employes_Salaire DEFAULT 0,
    Codeservice CHAR(5)     not null,
    CodeChef INT            null
);
```

RI 83

Permet de définir la valeur qui va être positionnée dans la colonne si aucune information n'est précisée lors de l'insertion de la ligne.
Utiles lorsque la colonne n'accepte pas les valeurs NULLES car elles garantissent l'existence d'une valeur.

Intérêt et nécessité de nommer ses contraintes en suivant une norme précise.

Faire la manip sur la table Employés

Mise en œuvre de l'intégrité des données

- Clé primaire

```
CREATE TABLE Employes(
    CodeEmp INT             not null CONSTRAINT PK_Employes PRIMARY KEY,
    Nom VARCHAR(20)         not null,
    Prenom CHAR(20)         null,
    DateNaissance DATE      null,
    DateEmbauche DATE       not null CONSTRAINT DF_Employes_DateEmbauche DEFAULT getdate(),
    DateModif TIMESTAMP     null,
    Salaire DECIMAL(8,2)    not null CONSTRAINT DF_Employes_Salaire DEFAULT 0,
    Codeservice CHAR(5)     not null,
    CodeChef INT            null
);
```

ou

```
CREATE TABLE Employes(
    CodeEmp INT             not null,
    Nom VARCHAR(20)         not null,
    Prenom CHAR(20)         null,
    DateNaissance DATE      null,
    DateEmbauche DATE       not null CONSTRAINT DF_Employes_DateEmbauche DEFAULT getdate(),
    DateModif TIMESTAMP     null,
    Salaire DECIMAL(8,2)    not null CONSTRAINT DF_Employes_Salaire DEFAULT 0,
    Codeservice CHAR(5)     not null,
    CodeChef INT            null,
    CONSTRAINT PK_Employes PRIMARY KEY(CodeEmp)
);
```


RI 77

Définir un identifiant clé primaire, c'est-à-dire une ou plusieurs colonnes n'acceptant que des valeurs uniques dans la table (règle d'unicité ou contraint d'entité).

2 syntaxes possibles. La seconde présentée sur la diapositive est obligatoire si la clé primaire se compose de plusieurs colonnes.

1 seule contrainte de ce type par table.
Les colonnes doivent être not null.
Jusqu'à 16 colonnes

Une colonne de type compteur

- La propriété IDENTITY
  - Valeurs générées à la création d'une ligne
  - 1 seule colonne IDENTITY par table
  - Peut produire des trous dans la numérotation

```
CodeEmp INT          not null IDENTITY(1,1),
```

RI 73

Cette propriété est une spécificité de SQL Server. Elle peut être affectée à une seule colonne dans la table. Cette colonne doit être de type numérique entier

Mise en œuvre de l'intégrité des données

- Clés secondaires

```
CREATE TABLE Services(
    CodeService char(5)      not null,
    Libelle varchar(30)      not null CONSTRAINT UN_Services_Libelle UNIQUE,
    CONSTRAINT PK_Services PRIMARY KEY(CodeService)
);
```

ou

```
CREATE TABLE Services(
    CodeService char(5)      not null,
    Libelle varchar(30)      not null,
    CONSTRAINT PK_Services PRIMARY KEY(CodeService),
    CONSTRAINT UN_Services_Libelle UNIQUE(Libelle)
);
```

RI 79

Permet de traduire la règle d'unicité pour les autres clés uniques de la table.

2 syntaxes possibles. La seconde présentée sur la diapositive est obligatoire si la clé unique se compose de plusieurs colonnes.

Plusieurs contraintes de ce type possibles par table.
Les colonnes peuvent être null (non recommandé)

Faire la manip sur les tables Employés et Services

Mise en œuvre de l'intégrité des données

- Contrainte de validation

```
CREATE TABLE Employes(
    CodeEmp INT             not null,
    Nom VARCHAR(20)         not null,
    Prenom CHAR(20)         null,
    DateNaissance DATE      null,
    DateEmbauche DATE       not null CONSTRAINT DF_Employes_DateEmbauche DEFAULT getdate(),
    DateModif TIMESTAMP     null,
    Salaire DECIMAL(8,2)    not null CONSTRAINT DF_Employes_Salaire DEFAULT 0
                                     CONSTRAINT CK_Employes_Salaire CHECK (salaire >= 0),
    CodeService CHAR(5)     not null,
    CodeChef INT            null,
    CONSTRAINT PK_Employes PRIMARY KEY(CodeEmp),
    CONSTRAINT CK_Employes_VerifDate CHECK (DateNaissance is null or DateEmbauche >= DateNaissance)
);
```

RI 85

Permet de définir des règles de validation mettant en rapport des valeurs issues de différentes colonnes de la même ligne.
Permet également de s'assurer que les données respectent un format précis lors de leur insertion ou mise à jour dans la table.
Permet de s'assurer également que la valeur proposée à la colonne est bien une valeur attendue

On peut définir la contrainte en utilisant cette phrase : "la ligne est acceptée si ….."

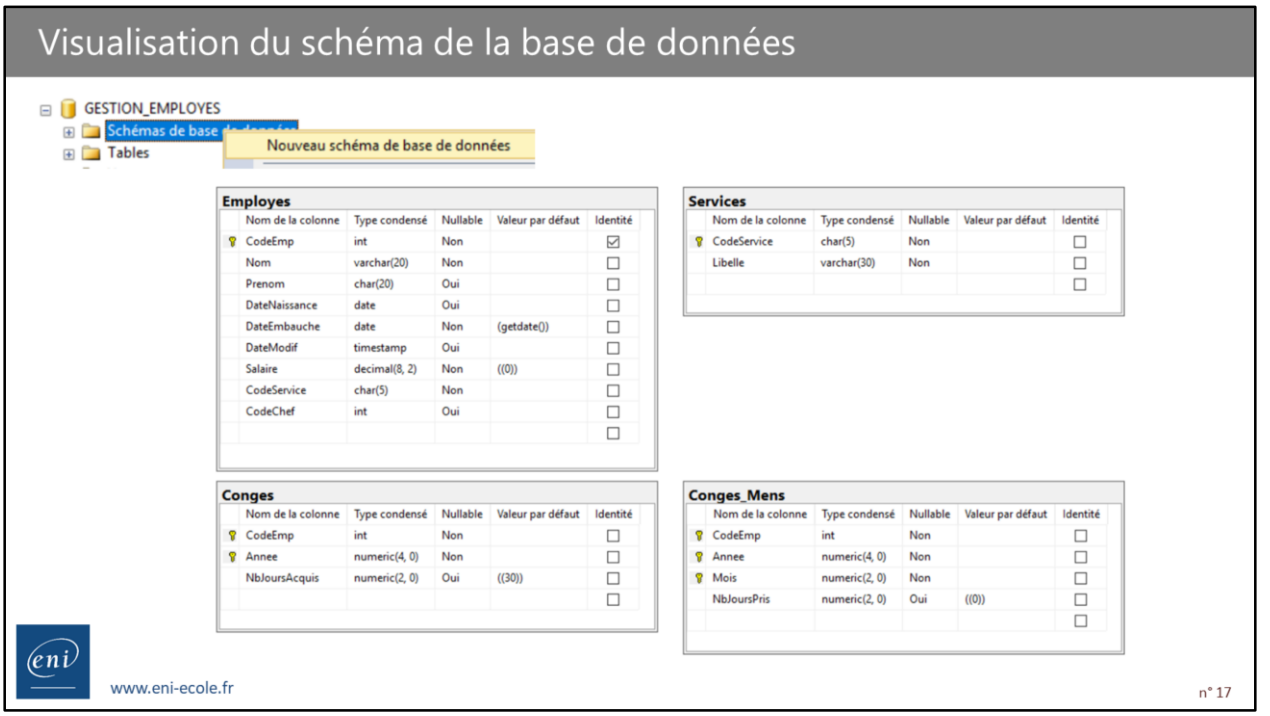
2 syntaxes possibles. La seconde présentée sur la diapositive est obligatoire si la règle fait intervenir plusieurs colonnes.

Faire la manip sur la table Employé et Service

Mise en œuvre de l'intégrité des données

- En vous appuyant sur le diagramme proposé dans le fichier en téléchargement *MLD – Gestion Employes.pdf,* appliquez les contraintes suivantes sur les tables :
  - Contraintes de non nullité
  - Valeurs par défaut :
    - 30 jours pour la colonne NbJoursAcquis
    - 0 jour pour la colonne NbJoursPris
  - Clés primaires
  - Contraintes de validation
    - La colonne Mois n'accepte qu'une valeur comprise entre 1 et 12

A vous de jouer :



Lancer une démonstration afin de présenter l'outil.
Sélection des tables.

- Menu contextuel sur le diagramme
    - Réorganiser les tables
- menu contextuel sur table sélectionnée
    - Vue de table
    - Visu des index/clés, contraintes …

Point noir de l'outil, le rafraichissement du schéma, necessite quelque fois de fermer et de rouvrir l'outil.
Dans le cadre de ce cours, nous n'utiliserons cet outil que pour visualiser mais pas pour modifier la structure des tables même s'il le permet.

Modification des tables – les colonnes

- Ajouter une colonne

```
ALTER TABLE table_name

        ADD column_name data_type(size) [NULL | NOT NULL]
;
```

- Modifier une colonne

```
ALTER TABLE table_name

        ALTER COLUMN column_name new_data_type(new_size) [NULL | NOT NULL]
;
```

- Supprimer une colonne

```
ALTER TABLE table_name

        DROP COLUMN column_name
;
```

RI 69

Une fois les tables créées, nous devons les maintenir, les faire évoluer. Hors de question de supprimer les tables puis de les récréer pour faire ces mises à jour, puisque nous perdrions également les données stockées dans ces tables.
Le SQL propose l'instruction ALTER TABLE pour réaliser des modifications structurelles et comportementales. Cette instruction appartient à la catégorie nommée DDL.
Elle permet d'ajouter, supprimer, modifier une colonne.
Elle permet d'ajouter, supprimer, d'activer/désactiver une contrainte.
Bien sur toutes ces modifications se font dans le respect des règles d'intégrité des données déjà posées sur la table.

Pour modifier la structure d'une table voici les syntaxes proposées.

Faire en // les modifications proposées dans le fichier
Alter_Tables_Gestion_Employes.sql

Modification des tables – les contraintes

- Ajouter une contrainte

```
ALTER TABLE table_name

        [WITH CHECK | WITH NOCHECK] ADD new_constraint_table;
```

- Supprimer une contrainte

```
ALTER TABLE table_name

        DROP CONSTRAINT constraint_name
;
```

- Activer/désactiver une contrainte

```
ALTER TABLE table_name

        {CHECK | NOCHECK} CONSTRAINT {ALL | constraint_name}
;
```

RI 69

Faire en // les modifications proposées dans le fichier
Alter_Tables_Gestion_Employes.sql

Mise en œuvre de l'intégrité référentielle

- Contrainte d'intégrité référentielle

```
ALTER TABLE Employes WITH CHECK ADD
    CONSTRAINT FK_Employes_CodeService FOREIGN KEY (CodeService)
        REFERENCES Services(CodeService),
    CONSTRAINT FK_Employes_CodeChef FOREIGN KEY (CodeChef)
        REFERENCES Employes(CodeEmp);
```

- ON DELETE | ON UPDATE
  - NO ACTION
  - CASCADE
  - SET NULL
  - SET DEFAULT

```
ALTER TABLE Conges WITH CHECK ADD
    CONSTRAINT FK_Conges_Employes FOREIGN KEY (CodeEmp)
        REFERENCES Employes(CodeEmp) ON DELETE CASCADE;
```

www.eni-ecole.fr

n° 20

RI 81

Cette contrainte traduit l'intégrité référentielle entre une clé étrangère d'une table et une clé primaire ou secondaire d'une autre table.

Les contraintes d'intégrité référentielle n'ont pas encore été mises en place dans notre base de données de gestion des employés. Les mettre en place lors de la création des tables oblige le concepteur à créer les tables dans un ordre bien précis (tables maitres avant les tables détails). Sur une base de données contenant plusieurs dizaines de tables, déterminer l'ordre de création des tables devient complexe. Nous allons donc mettre en place l'intégrité référentielle en utilisant la commande de modification d'une table : ALTER TABLE

Il est plus facile de définir cette contrainte une fois que toutes les clés primaires ont été créées. L'utilisation de la commande ALTER TABLE ADD CHECK CONSTRAINT est donc préconisée ici.

La clause FOREIGN KEY liste les colonnes participant à la définition de la clé étrangère alors que la clause REFERENCES liste les colonnes de clé primaires de la table cible. La structure de la clé étrangère doit être identique à la structure de la clé

primaire.

L'option de cascade permet de préciser le comportement que doit adopter le serveur de bases de données lorsque l'utilisateur met à jour ou tente de supprimer une colonne référencée.
NO ACTION : Valeur par défaut. La suppression/modification d'une clé primaire est impossible si elle est référencée,
CASCADE : La suppression/modification d'une clé primaire se répercute en cascade sur la clé étrangère soit par la suppression des lignes soit par la mise à jour des valeurs de ligne.
SET NULL : la suppression de la clé primaire valorise la clé étrangère à null.
SET DEFAULT : la suppression de la clé primaire valorise la clé étrangère à sa valeur par defaut.

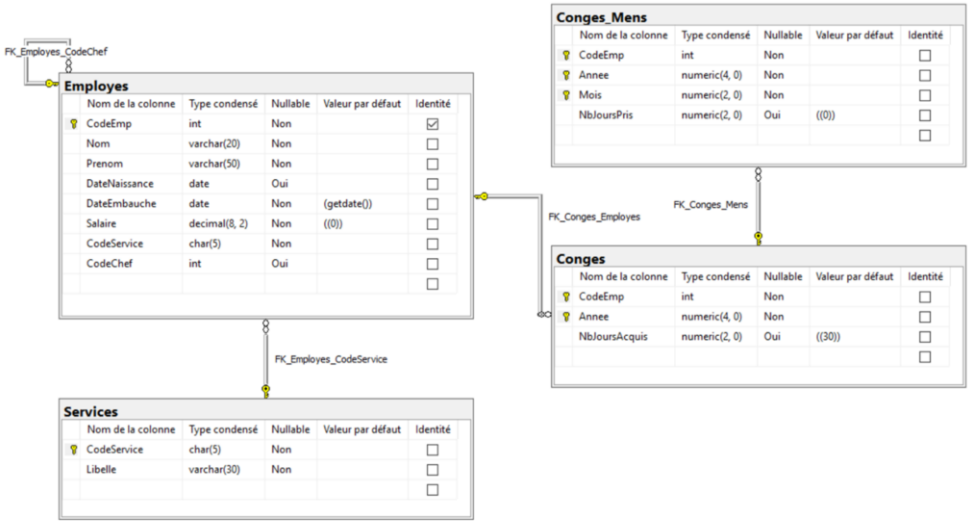
Faire évoluer la base de données

- En vous appuyant sur le diagramme proposé dans le fichier en téléchargement *MLD – Gestion Employes.pdf,* appliquez les contraintes suivantes sur les tables :
  - Contraintes d'intégrité référentielle entre les tables Conges et Employes, ainsi qu'entre les tables Conges_Mens et Conges
- Préparez un script de suppression de l'ensemble des contraintes d'intégrité référentielle.

A vous de jouer :

Schéma de la base de données

- Etat de la base Gestion_Employes après modifications



Suppression des tables

- Supprimer la structure et les données

```
DROP TABLE table_name [,table_name];
```

Tenir compte de l'intégrité référentielle

RI 70

La suppression d'une table est effectuée par la commande du DDL Drop Table.
Cela entraine la suppression de toutes les données présentes dans la table.
Il est possible de supprimer plusieurs tables à l'aide d'une seule instruction DROP TABLE. Il suffit de séparer chaque table par une ,

Attention à l'intégrité référentielle. Il n'est pas possible de supprimer une table qui est référencée par une autre.
Sur une base de données importante, il est certainement préférable de supprimer les contraintes de clé étrangère dans un premier temps, puis de supprimer les tables.

Essayez de supprimer la table Employes avant la table Services pour montrer l'erreur.

Suppression des tables

- Complétez votre script de suppression des contraintes d'intégrité référentielle afin qu'il se termine par la suppression physique des tables.

A vous de jouer :

Indexation des données

- Pourquoi indexer les données ?
  - Améliorer les performances d'accès aux informations en base dans le cadre d'une extraction ou d'une mise à jour.

www.eni-ecole.fr

n° 25

RI 86

L'objectif des index est de permettre un accès plus rapide à l'information que se soit dans le cadre d'une extraction ou d'une mise à jour.

Prenons l'exemple de notre livre Ressources Informatiques. Il est possible de parcourir ce livre en s'appuyant soit sur la table des matières, soit sur la liste des index.

La table des matières représente un index particulier car elle structure physiquement le livre et donc l'ordre des pages. SQL Server propose un index CLUSTERED qui organise physiquement les données. Il ne peut y en avoir qu'un seul par table. Il est généralement et automatiquement associé à la clé primaire. C'est pour cela que vos données sont classées dans l'ordre croissant de la valeur de la clé primaire et non pas dans l'ordre de leur enregistrement.

La liste des index en fin de livre permet de rechercher des informations par mots clés, par thèmes, …SQL Server vous propose également de créer différents index sur une table. Ces index référencent les lignes associées au mot clé ou au thème. Lors d'une recherche sur ce thème, SQL Server parcourt l'index correspondant pour

atteindre directement l'enregistrement stocké dans la table.

La gestion des index

- Une bonne stratégie :
  - Il est préférable d'avoir moins d'index que trop d'index,
  - Les index doivent être le plus large possible afin d'être utilisables par plusieurs requêtes,
  - Il faut s'assurer que les requêtes utilisent bien les index.
- Automatiquement créé pour :
  - Une contrainte primary key (index CLUSTERED),
  - Une contrainte unique.
- A définir généralement sur :
  - Une contrainte foreign key,
  - Sur les colonnes de recherche et de tri.

RI 86

L'indexation des données doit être réalisée judicieusement.
Avoir trop peu d'index nuit au performance d'extraction des données.
Mais avoir trop d'index peut les rendre coûteux dans le cas de mise à jour de la valeur contenue dans la colonne indexée. En effet, le SGBDR prendra le temps de revaloriser les index, de reclasser les données lors de l'ajout, modification ou suppression de données indexées.

L'ajout d'une page dans un livre nécessite de reconstruire la table des matières et la table des index.

La gestion des index

- Créer un index

```
CREATE [UNIQUE] [CLUSTERED | NONCLUSTERED] INDEX index_name
ON table_name(column_name [ASC | DESC] [,column_name]);
```

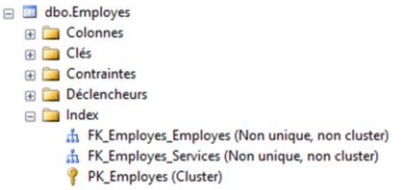
- Supprimer un index

```
DROP INDEX index_name ON table_name;
```

- Reconstruire les index

```
ALTER INDEX { index_name | ALL } ON table_name REBUILD;
```

```
CREATE NONCLUSTERED INDEX FK_Employes_Services
    ON Employes(CodeService ASC);
```



RI 91

Lancer les manipulations du fichier Create_Index_Gestion_Employes.sql

Montrer également les index créés dans l'explorateur d'objets.