



# Modéliser les données avec MySQL Workbench



Par François de Sainte Marie (fsmrel)

*Labor omnia vincit improbus*

Virgile (Géorgiques)

Date de publication : 30/01/2014

## Avant-propos

L'objet est ici de rassembler, harmoniser et compléter au mieux les réponses que j'ai faites aux questions des forumers de Developpez.com ayant déjà un minimum de connaissances en modélisation, soucieux de représenter correctement les structures de leurs bases de données sans devoir investir dans un AGL onéreux, alors que la version gratuite de MySQL Workbench<sup>1</sup> (MWB) permet de répondre de façon satisfaisante à leur besoin.

Du point de vue de l'abstraction, un diagramme MWB est en théorie un cran en dessous d'un MCD de Merise ou d'un diagramme de classes UML — il se situe au niveau MLD (modèle logique des données) de Merise —, il ne prend pas formellement en compte des concepts tels que la généralisation/spécialisation, l'agrégation, la composition, mais ça n'est pas vraiment un problème, l'outil offrant ce qui est nécessaire et suffisant pour leur mise en oeuvre. Par contre les diverses contraintes telles que l'exclusion, l'inclusion, la totalité ne sont pas considérées ici.

L'objet étant la modélisation des données, les possibilités offertes par l'outil qui ressortissent à l'administration de la base de données sont passées sous silence (structures physiques d'accueil des tables, partitioning), mais abordées quand c'est inévitable (index en relation avec les clés). Plus généralement pour tout ce qui relève du physique, à chacun (ou plutôt à son DBA préféré) de s'en préoccuper en fonction de son SGBD cible. La manipulation des données (requêtes SQL) n'est pas non plus prise en considération, on en reste à la structure et à l'intégrité des données.

L'article s'adresse donc à celui qui a besoin de disposer d'une représentation graphique de la base de données, à partir de quoi il pourra produire de façon automatique un script SQL conforme de création des tables, compte non tenu toutefois de l'ensemble des caractéristiques physiques de la base de données.

N.B. On utilise ici la version gratuite (open source) de MySQL Workbench 6.0 : l'outil ne vérifie pas la validité des modèles (tables en double, clés non définies ou incohérentes, etc.), mais cela ne doit pas gêner celui qui travaille avec sérieux. De toute façon, lors de l'exécution des instructions SQL CREATE/ALTER TABLE, le SGBD se chargera de rappeler à l'ordre l'étourdi.

A toutes fins utiles, signalons encore qu'on utilise ici MySQL Workbench pour Windows.

Un très grand merci à Fabien (f-leb) qui a bien voulu accepter de relire l'article et n'a pas ménagé sa peine, ainsi qu'à Kropernic dont l'œil vaut celui lynx.

Documentation officielle du produit : <http://dev.mysql.com/doc/index-gui.html> (PDF téléchargeable).  
(Si le lien n'est pas trouvé, rechercher : « MySQL Workbench Reference Manual »).

Pour l'ensemble des fonctionnalités (version open source / commerciale) :

<http://www.mysql.fr/products/workbench/features.html>

<sup>1</sup> © 2013, Oracle Corporation and/or its affiliates.

1.	Rappels généraux à propos de la modélisation des données .....	4
1.1.	Types d'entités .....	4
1.2.	L'entité-type forte .....	4
1.3.	L'entité-type faible .....	5
1.4.	L'entité-type associative.....	5
2.	Pour débiter avec MySQL Workbench .....	6
2.1.	Bienvenue.....	6
2.2.	Créer un modèle .....	7
2.2.1.	Accès à l'onglet « File » .....	7
2.2.2.	Nouveau diagramme.....	8
3.	Définir la structure des tables .....	10
3.1.	Icône « New Table » .....	10
3.2.	En-tête de table (et clé primaire) .....	12
3.3.	Notation « Workbench simplifiée » .....	13
3.4.	Clés alternatives .....	14
4.	Définir les associations entre les tables (plusieurs à plusieurs) .....	17
4.1.	En amont de MySQL Workbench .....	17
4.2.	Avec MySQL Workbench .....	17
4.3.	Cardinalité minimale 0 .....	18
4.4.	Clés étrangères.....	19
4.5.	Affichage du nom des associations .....	19
4.6.	Notation UML.....	20
4.7.	Notation « à la ACCESS ».....	21
4.8.	Ordre des colonnes dans les clés .....	21
4.9.	Ajout de colonnes .....	22
4.10.	Index redondants.....	23
5.	Associations de un à plusieurs .....	24
5.1.	Associations de un à plusieurs entre entités-types indépendantes.....	24
5.1.1.	Entité-type indépendante (rappel) .....	24
5.1.2.	Icône « 1:n Non-Identifying Relationship » .....	24
5.2.	Associations de un à plusieurs (entre entités-types fortes et dépendantes) .....	26
5.2.1.	Associations binaires.....	26
5.2.1.1.	A la vie à la mort .....	26
5.2.1.2.	Identification absolue / relative .....	27
5.3.	Retour sur les associations de plusieurs à plusieurs : associations ternaires .....	31
6.	Associations de un à un (généralisation/spécialisation).....	34
6.1.	Rappels .....	34
6.2.	Icône « 1:1 Identifying Relationship ».....	35
6.3.	Diagramme final .....	37
7.	Associations réflexives (nomenclatures, hiérarchies).....	38
7.1.	Nomenclatures .....	38
7.2.	Hiérarchies.....	40

8.	Génération du code SQL .....	42
8.1.	Commande « Export » .....	42
8.2.	Définition du nom du fichier contenant le code SQL .....	43
8.3.	Types d'objets à exporter .....	44
8.4.	Choix des tables à faire figurer dans le code SQL .....	45
8.5.	Affichage du résultat par MySQL Workbench .....	46
8.6.	Enregistrement du fichier .....	47
9.	Rétro-conception .....	48
9.1.	Objet de la rétro-conception .....	48
9.2.	A partir du code SQL .....	48
9.3.	A partir de DBDesigner 4 .....	53
10.	Annexes .....	56
10.1.	Afficher/cacher la grille .....	56
10.2.	Changer la couleur d'un objet .....	56
10.3.	Réduire la taille des objets à l'affichage .....	58
10.4.	Noms et types par défaut .....	59
10.4.1.	MySQL Workbench et les éléments par défaut .....	59
10.4.2.	Nom par défaut de la 1re colonne d'une table .....	60
10.4.3.	Nom et type par défaut d'une colonne banale .....	60
10.4.4.	Types par défaut .....	61
10.4.5.	Nom par défaut des clés étrangères .....	61
10.4.6.	Note à propos des actions de compensation (ON UPDATE, ON DELETE) .....	62
10.4.7.	Nom par défaut des tables associatives .....	67
10.5.	Changer l'ordre des colonnes d'une table .....	68
10.6.	Urbanisation des diagrammes .....	69
10.6.1.	Diagrammes comportant un grand nombre d'objets .....	69
10.6.2.	A la manoeuvre avec MySQL Workbench .....	70
10.6.3.	Ajouter un diagramme .....	71
10.6.4.	Renommer un diagramme .....	72
10.6.5.	Peupler un diagramme à partir du catalogue .....	75
10.6.6.	Masquer une table ou un lien dans un diagramme .....	79
10.6.7.	Les couches de peinture (layers) .....	82
10.7.	Définir une clé alternative .....	85
10.8.	Changer l'ordre des colonnes d'une clé .....	88
10.9.	Rendre un lien facultatif .....	88
10.10.	Supprimer un objet (sans s'énervier) .....	89
10.11.	Afficher le nom des associations entre tables .....	92
10.12.	MySQL Workbench 5.2 : Accueil .....	94
10.12.1.	Écran d'accueil .....	94
10.12.2.	Créer un nouveau modèle .....	94

## 1. Rappels généraux à propos de la modélisation des données

### 1.1. Types d'entités

Puisqu'on s'intéresse à la modélisation des données, il n'est pas inutile de rappeler quelques concepts du niveau sémantique, à savoir les différents types d'entités (**entités-types**).

Je fais ici référence à RM/T (Relational Model / Tasmania) de Ted Codd, qui développa ses idées sur la modélisation sémantique à partir du Modèle Relationnel de Données (cf. son article de 1979, *Extending the Database Relational Model to Capture More Meaning*). Le tenant de Merise ou d'UML saura faire le rapprochement entre les entités-types de Codd et les types d'entités ou classes qui lui sont familiers.

Codd classe les entités-types (types d'entités) de cette façon :

- **L'entité-type forte**, qu'il nomme *Kernel*,
- **L'entité-type faible** (*Characteristic*),
- **L'entité-type associative** (*Associative*).

Par ailleurs, chacune de ces entités-types peut jouer le rôle de surtype (*supertype*) ou de sous-type (*subtype*) par rapport à une entité-type de même nature : une entité-type forte peut être sous-type d'une autre entité-type forte, une entité-type faible peut être sous-type d'une autre entité-type faible et une entité-type associative peut être sous-type d'une autre entité-type associative. Signalons que MySQL Workbench ne propose pas formellement les concepts de surtype et de sous-type, mais on sait s'en arranger et ce sujet est abordé au paragraphe 6.

### 1.2. L'entité-type forte

Une entité (instance, occurrence) de ce type est autonome, insensible aux stimuli qui pourraient provoquer son altération, émis par des entités qui entretiennent des relations avec elle, en conséquence de quoi ces stimuli feraient l'objet d'un refus énergique de la part de cette entité s'ils conduisaient à sa destruction.

Imaginons que l'entreprise Dubicobit vende des savonnettes et que chacun de ses clients réside dans une commune. Le concepteur aura rédigé une règle de gestion des données ressemblant à ceci :

Un client réside dans exactement une commune (c'est-à-dire au moins une et au plus une) et dans une commune peuvent résider plusieurs clients (de 0 à une foultitude).

D'où la représentation simple mais parlante, bien connue des habitués du forum Schéma chez DVP :

[CLIENT]--1,1----(RÉSIDER)----0,N--[COMMUNE]

Ou dans le forum UML :

Résider

[CLIENT]--0..N-----1--[COMMUNE]

Si on supprime le client Raoul qui réside dans la commune de Morzy-les-Joyeuses, celle-ci n'en est pas avertie et reste donc parfaitement insensible à la disparition de ce malheureux Raoul.

Si c'est une commune qu'on cherche à supprimer, par exemple Yadupour, et si aucun client n'y réside, pas de problème. En revanche, si au moins un client y réside, changement de chanson, l'opération doit échouer : les clients résidant dans

cette commune sont sensibles au stimulus qui leur parvient et sont en droit de juger l'action parfaitement déplacée, ils réagissent donc vigoureusement et négativement, forçant le système à leur donner raison<sup>1</sup>.

### 1.3. L'entité-type faible

Une entité de ce type n'est jamais qu'une propriété multivaluée d'une entité plus forte qu'elle. Exemple, les lignes de facture constituent une propriété multivaluée d'une facture et, au nom de la normalisation (telle qu'énoncée par Codd dans son article fondateur de 1970, *A Relational Model of Data for Large Shared Data Banks*), ce genre de propriété fait l'objet d'une entité-type LIGNE\_FACTURE, mais il n'en demeure pas moins que si l'on supprime une facture, ses lignes ne peuvent que disparaître avec elle, car n'ayant pas d'existence propre.

Représentation à la façon de Merise :

```
[FACTURE]--1,N----(COMPOSER)----(1,1)--[LIGNE_FACTURE]
```

La cardinalité 1,1 côté LIGNE\_FACTURE figure entre parenthèses : par convention, LIGNE\_FACTURE est à considérer comme entité-type faible relativement à l'entité-type FACTURE. Ce procédé est emprunté à PowerAMC.

Avec WinDesign, la notation équivalente est la suivante :

```
[FACTURE]--1,N----(COMPOSER)----1,1(R)--[LIGNE_FACTURE]
```

Dans le cas d'un diagramme de classes :

```
[FACTURE]◆--1..N-----1--[LIGNE_FACTURE]
```

Une entité-type faible peut être considérée comme forte par rapport à une autre entité-type. Par exemple, si chaque ligne de facture est composée d'engagements sur ligne de facture, on aura la représentation suivante (Power AMC) :

```
[FACTURE]--1,N----(COMPOSER)----(1,1)--[LIGNE_FACTURE]--1,N----(GROUPE)----(1,1)--[ENGAGEMENT]
```

### 1.4. L'entité-type associative

Comme son qualificatif l'indique, ce type d'entité permet d'associer deux entités-types selon une relation de plusieurs à plusieurs : au niveau logique (tabulaire, disons SQL), une entité-type associative donne lieu à une table. Par exemple, une ligne de facture peut être perçue comme associant une facture à un article. Dans le style merisien :

```
[FACTURE]--1,N----(LIGNE_FACTURE)----0,N--[ARTICLE]
```

Au sens de Codd, LIGNE\_FACTURE est une entité-type associative, tandis que FACTURE et ARTICLE peuvent être des entités-types fortes, faibles ou associatives : pas de restriction (contrairement à ce qui se passe avec Merise qui traditionnellement n'admet pas les associations d'associations).

---

<sup>1</sup> Les habitués de SQL auront reconnu la mise en œuvre pour cela de l'option RESTRICT (ou NO ACTION) utilisée dans le contexte de l'intégrité référentielle :

```
CREATE TABLE CLIENT ...
  FOREIGN KEY (CommuneId) REFERENCES COMMUNE ON DELETE RESTRICT ...
```

## 2. Pour débuter avec MySQL Workbench

### 2.1. Bienvenue

Au démarrage, MySQL Workbench (version 6.0) nous souhaite la bienvenue.

L'endroit est plutôt lugubre, on dirait qu'il héberge un catafalque, aussi on ne s'y attardera pas...

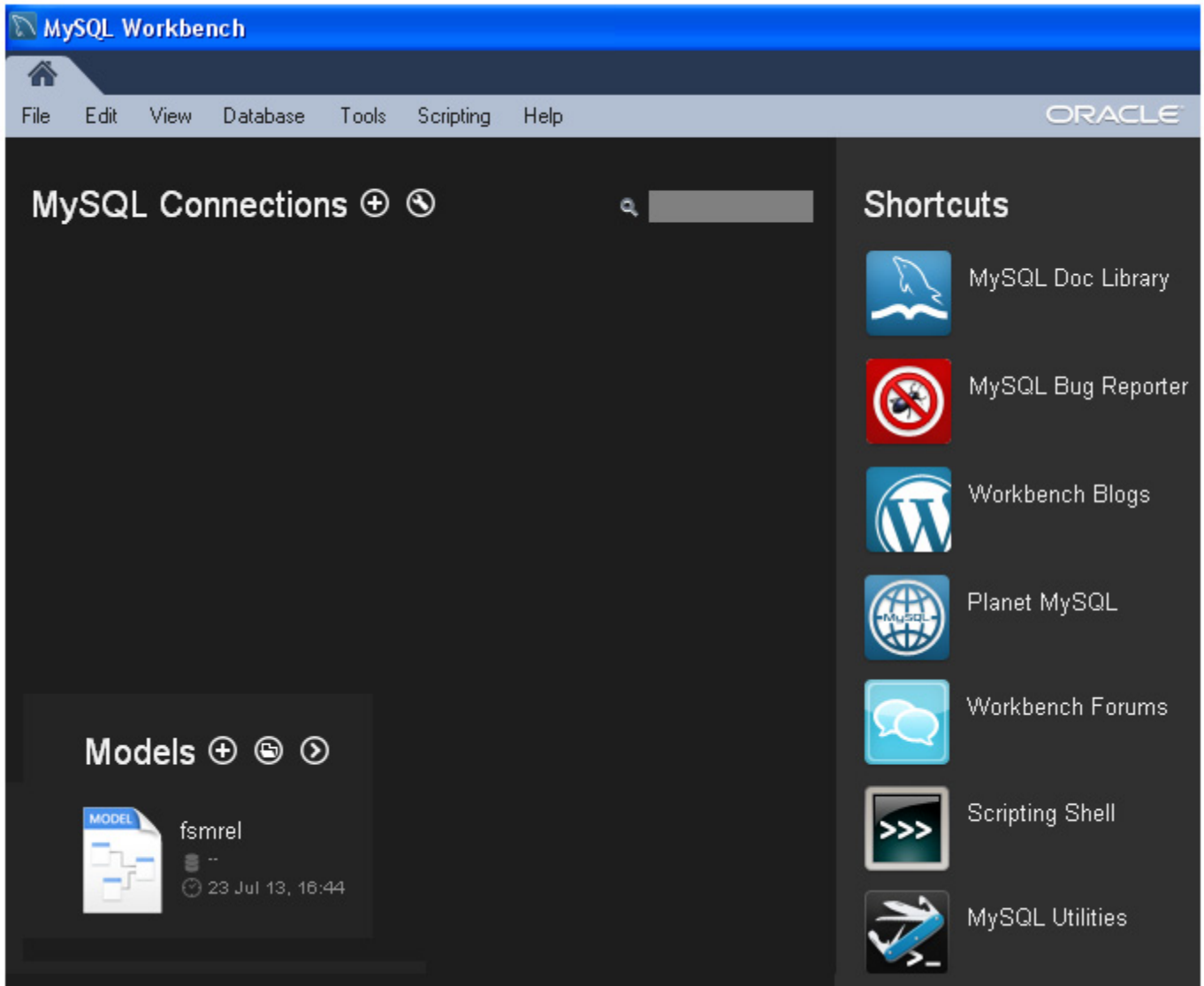


Figure 2.1 - Bienvenue de la part de MySQL Workbench

N.B. Les utilisateurs de MySQL Workbench qui en sont encore à la version 5.2 trouveront en annexe l'écran d'accueil qui lui correspond (cf. paragraphe 10.12).

## 2.2. Créer un modèle

### 2.2.1. Accès à l'onglet « File »

En dépit de l'aspect de l'écran d'accueil propre à rendre neurasthénique, il s'agit de créer un modèle, et pour cela on clique sur l'onglet « File » puis on choisit « New Model » :

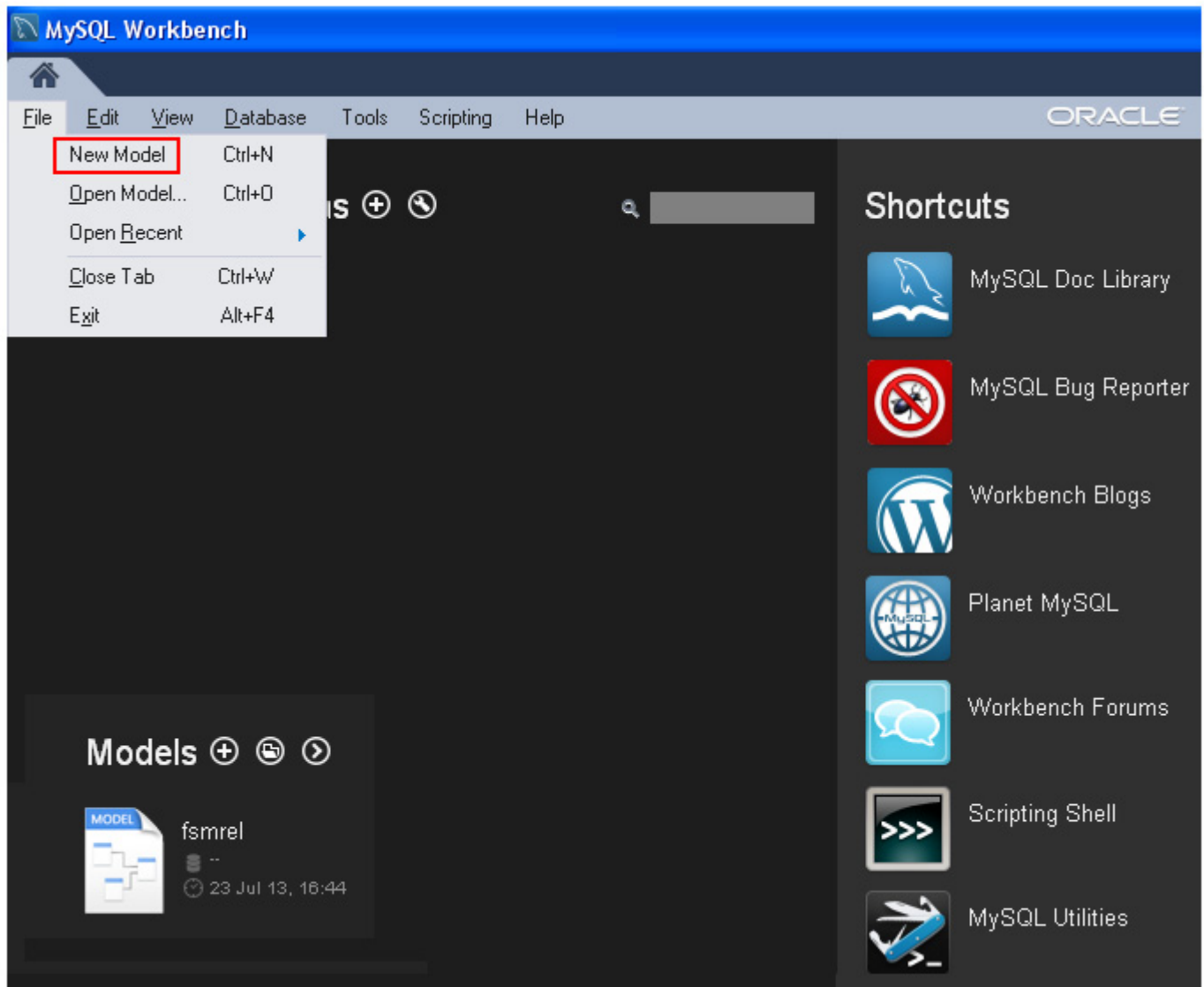


Figure 2.2 - Création d'un nouveau modèle

## 2.2.2. Nouveau diagramme

Une fois qu'on a fait le choix « New Model », MySQL Workbench propose la fenêtre suivante permettant de créer un MLD (modèle logique de données) sous la forme d'un diagramme :

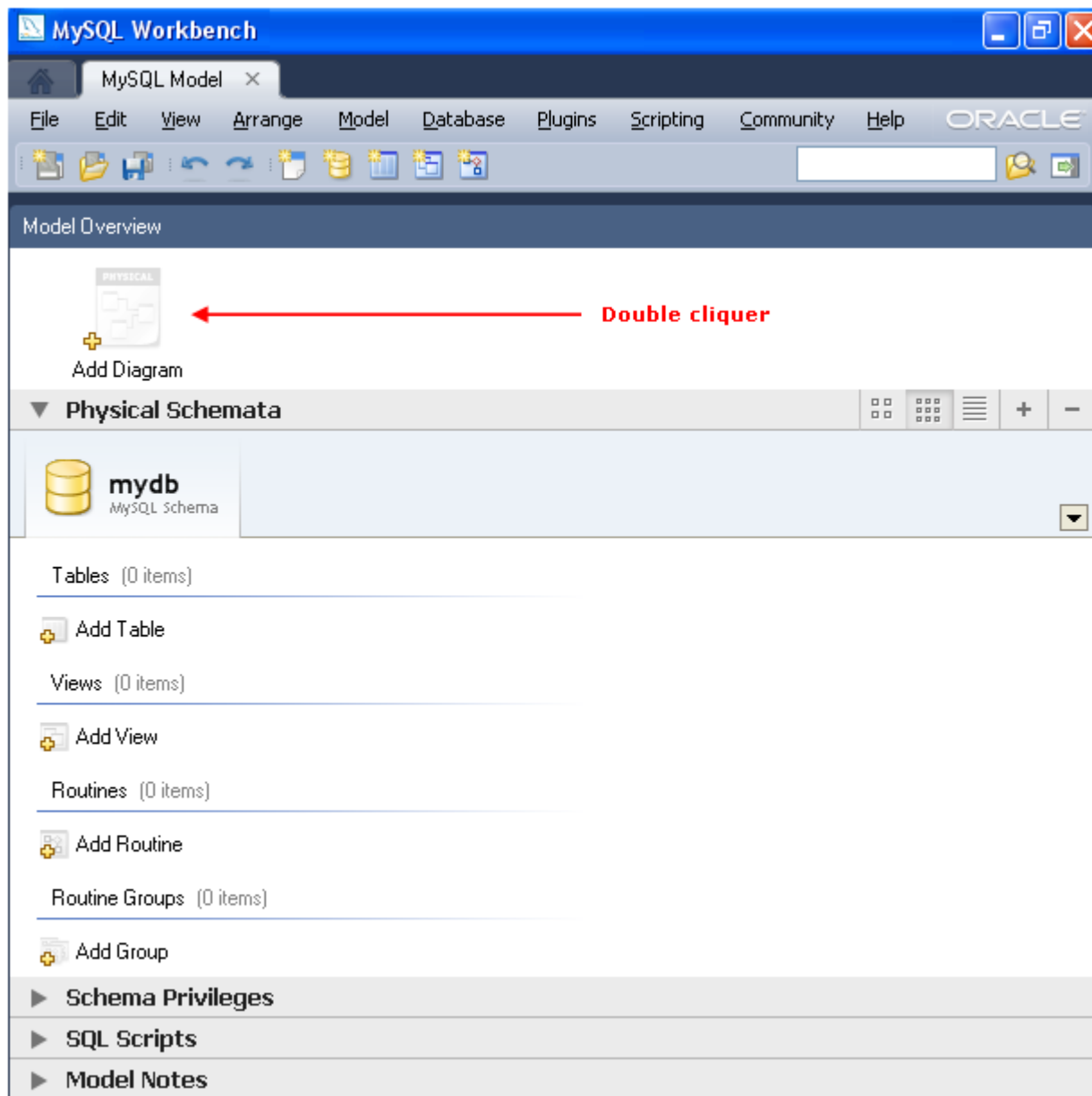


Figure 2.3 - Ajout d'un diagramme pour un MLD

Pour passer à la création effective d'un diagramme, on double-clique sur « Add Diagram ».



**A ce stade, on notera que les captures d'écran qui suivent correspondent à MySQL Workbench version 5.2, mais les éventuelles différences par rapport à la version 6.0 sont mineures.**



MySQL Workbench présente en retour une page vide de tout objet :

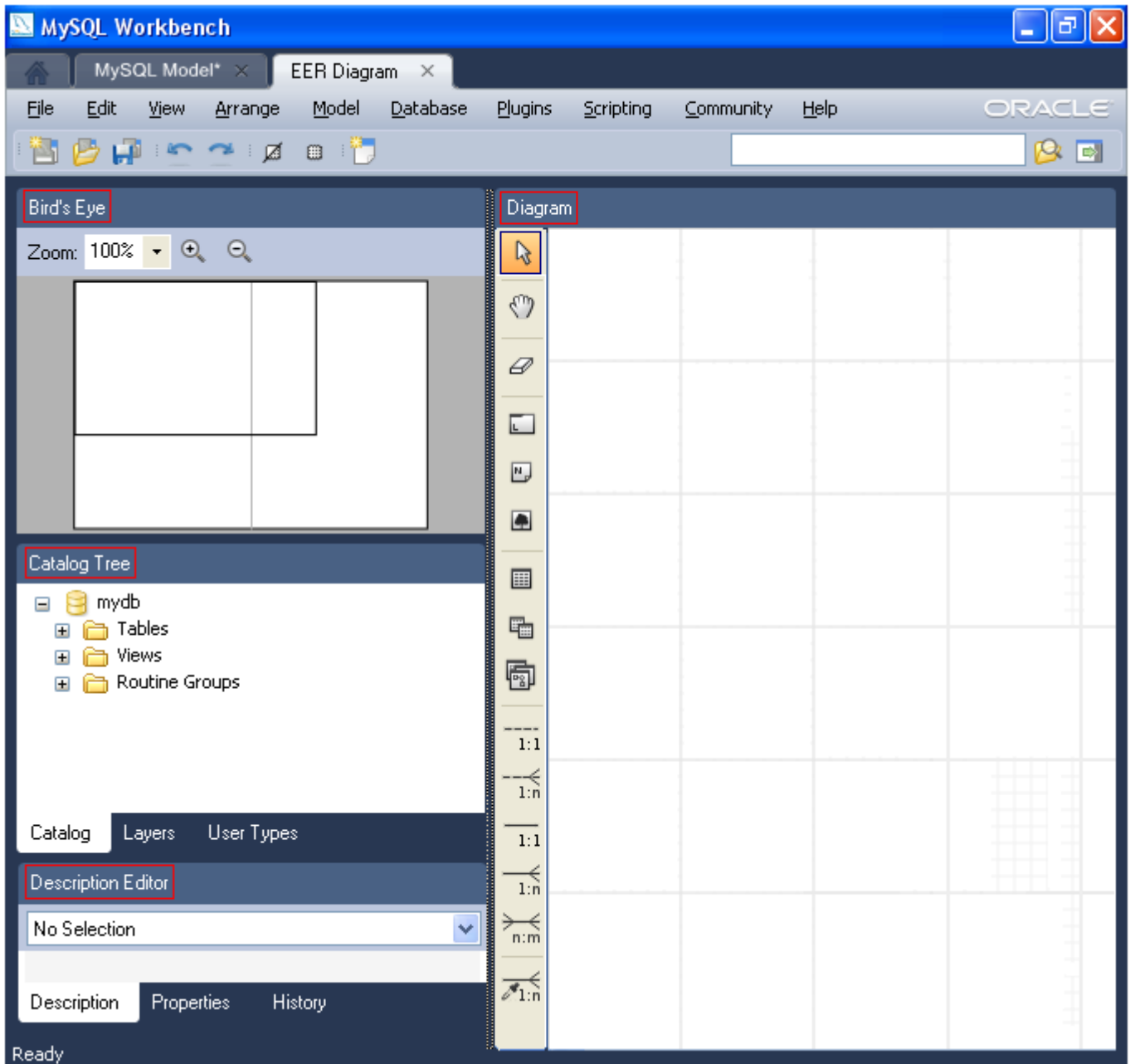


Figure 2.4 - Avant de créer les tables

L'espace proposé par MySQL Workbench est essentiellement composé des éléments suivants :

- **Diagram**, qui nous intéresse au premier chef.
- **Bird's Eye**, qui permet au besoin de réduire la taille des objets représentés dans le diagramme, quand par exemple ceux-ci commencent à être un peu nombreux.
- **Catalog Tree**, qui permet de représenter les objets sous forme arborescente (notamment les noms des tables).
- **Description Editor**, qui permet d'enjoliver les objets (changement de couleur, etc.)
- La version 6 de l'outil propose un gadget nommé « Modeling Additions » : structures de tables prédéfinies (**templates**) squelettiques, à chacun de voir l'intérêt qu'il peut y trouver et les enrichir au besoin.

On peut commencer par choisir le nom du schéma (au sens SQL du terme), en remplaçant celui qui est fourni par défaut (« mydb »). Ci-dessous, on remplace « mydb » par « fsmrel » (action : « Edit schema ») :

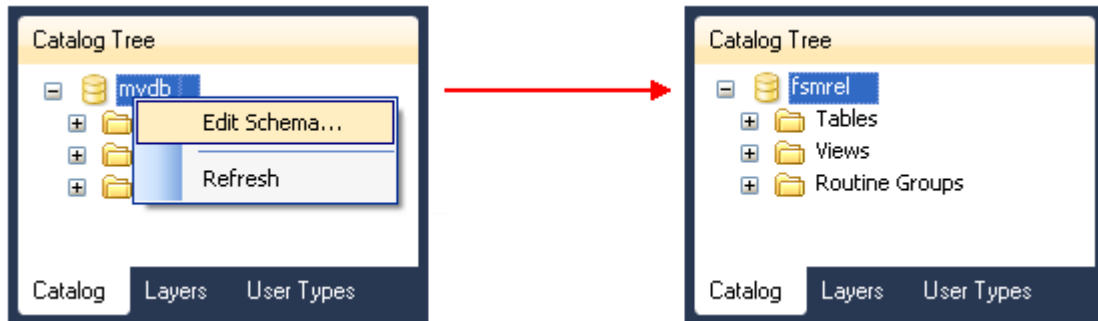


Figure 2.5 - Renommer un schéma (au sens SQL)

### 3. Définir la structure des tables

#### 3.1. Icône « New Table »

On en vient à ce qui nous intéresse au premier chef, définir la structure d'une table. Pour cela, on commence par cliquer sur l'icône ad-hoc :

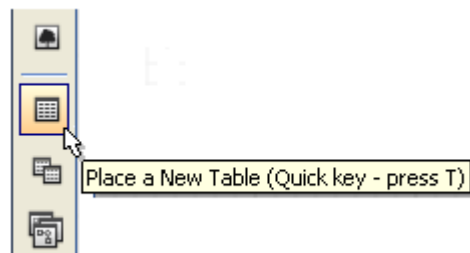


Figure 3.1 - Barre d'icônes, définir la structure d'une table

(La grille est ici cachée, pour la faire réapparaître, cf. paragraphe 10.1 : « Afficher/cacher la grille ».)

Une fois qu'on a cliqué sur l'icône, l'image d'un objet de type table apparaît :



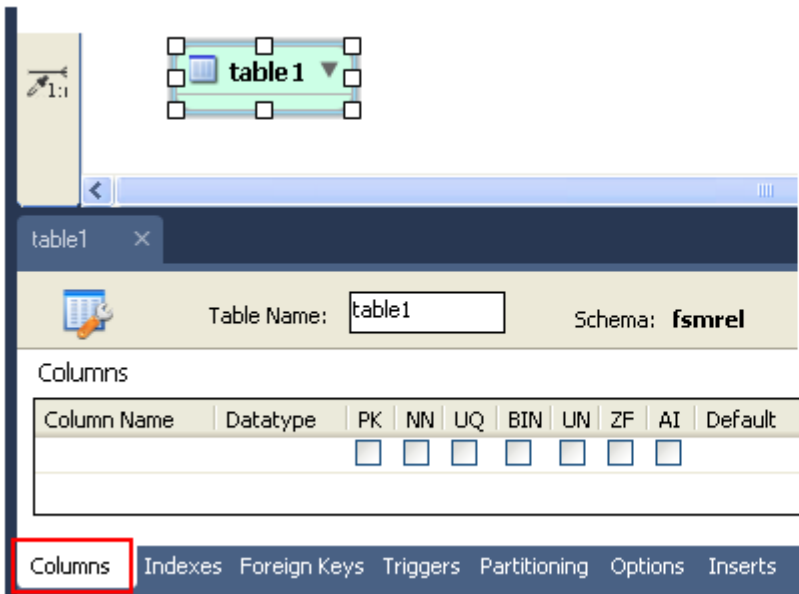
Figure 3.2 - Nouvelle table

Si la couleur bleu-pisseux fait loucher, il est possible d'en changer et passer au vert-pisseux ou autre (cf. paragraphe 10.2, « Changer la couleur d'un objet ») :



Figure 3.3 - Nouvelle table, coloriage

Pour renommer la table et passer à la construction de la structure de son en-tête (ensemble des colonnes de la table) : en cliquant sur l'objet « table1 » on provoque l'ouverture de la fenêtre affectée à la gestion des propriétés des tables, et on a accès à l'onglet « Columns » :



Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

**Datatype** : type utilisé pour la colonne

**PK** : la colonne appartient à la clé primaire

**NN** : NOT NULL

**UQ** : la colonne appartient à une clé alternative singleton

**Bin** : colonne binaire

**UN** : le type de la colonne est non signé

**ZF** : remplir la colonne (si numérique) avec des zéros

**AI** : auto-incrément

**Default** : valeur par défaut

Figure 3.4 - Propriétés globales d'une table

### 3.2. En-tête de table (et clé primaire)

Renommons donc la table et définissons-en l'en-tête, c'est-à-dire l'ensemble de ses colonnes (pour cela double-cliquer sur l'image de la table) :



Figure 3.5 - Onglet « Columns », ensemble des colonnes de la table

Il va sans dire que chaque table doit être dotée d'une clé primaire pour ne pas mériter le label d'infamie de « sac ». Dans cet exemple, la clé primaire a pour (seul) élément la colonne ClientId. Rappelons à cette occasion que la clé primaire est un cas particulier de la [clé candidate](#).

Rappelons aussi à cette occasion que le bonhomme Null est hors-la-loi dans les bases de données authentiquement relationnelles.

Le type des données (INT, VARCHAR(45), etc.) est celui qui est proposé par défaut. Si l'on estime qu'un type par défaut (par exemple VARCHAR(45)) d'une façon générale ne convient pas, on a le loisir de le changer (cf. paragraphe 10.4.4).

N.B. Il est possible de réduire la taille des objets à l'affichage, ce qui est intéressant quand l'espace est encombré de tables (cf. paragraphe 10.3).

### 3.3. Notation « Workbench simplifiée »

Pour alléger l’affichage des tables, on peut éviter d’afficher le cartouche « Indexes » (qui du reste n’a rien à faire au niveau conceptuel ou logique) : à cet effet on utilise la notation « Workbench simplifiée » :

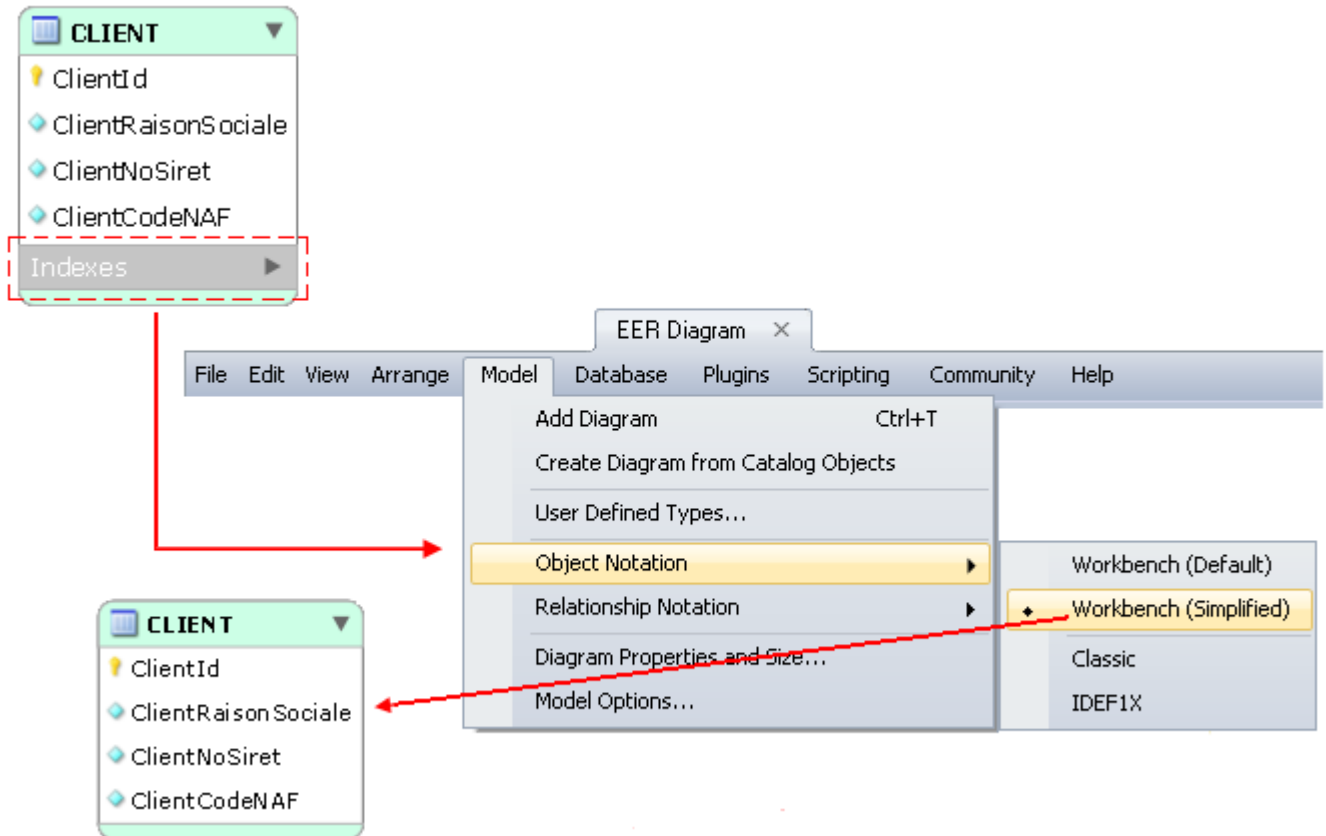


Figure 3.6 - Notation « Workbench simplifiée »

A noter qu’en cliquant sur le symbole « ▼ » figurant à la droite du nom de la table, on peut aussi se contenter d’un affichage le plus succinct qui soit (et revenir à l’affichage précédent en cliquant sur le symbole « ► ») :



Figure 3.7 - Réduction au seul nom de la table

### 3.4. Clés alternatives

En revenant à la notation « Workbench par défaut », on peut afficher la liste des index<sup>1</sup>, bien qu'il n'y ait en l'occurrence pas de valeur ajoutée, puisque seul le nom de ceux-ci est affiché :

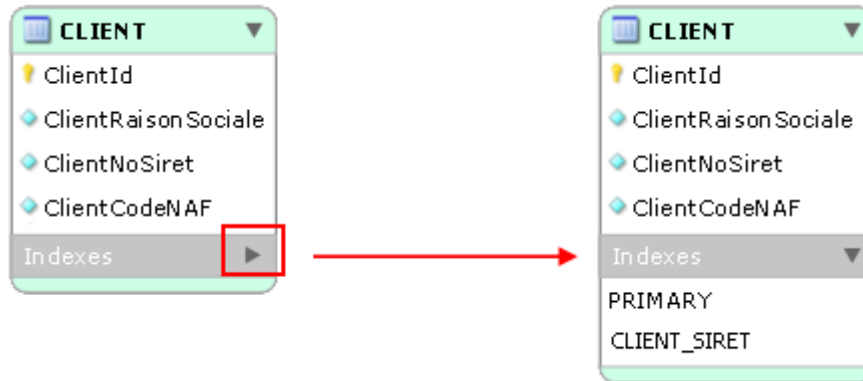


Figure 3.8 - Affichage de la liste des index d'une table

Par le truchement d'une infobulle, on peut toutefois savoir si on a défini des [clés alternatives](#) et quelle en est la composition :

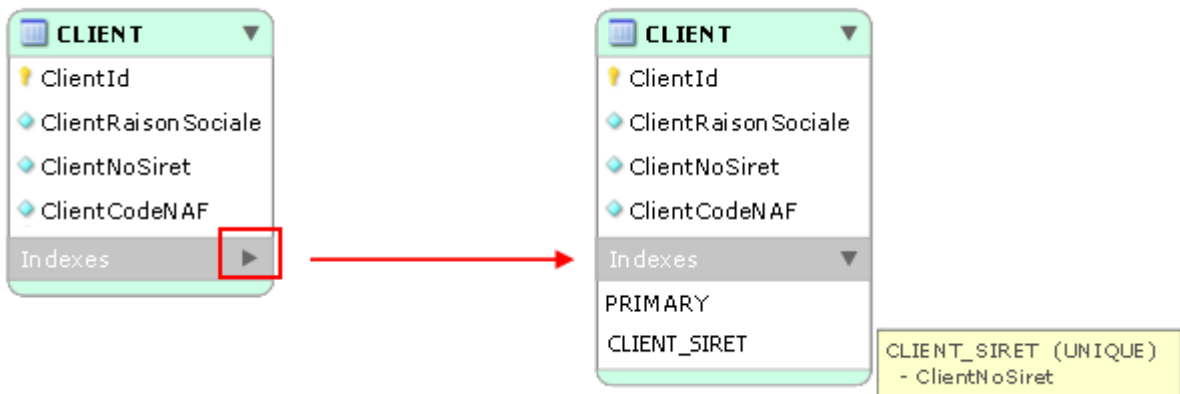


Figure 3.9 - Affichage de la liste des index d'une table, composition des clés

<sup>1</sup> Il est fâcheux que MySQL Workbench fasse un amalgame, ne propose pas la mise en œuvre des [clés alternatives](#) pour ce qu'elles sont (d'autant que ce sont des **ensembles** au sens de la théorie des ensembles), et que pour garantir le rôle de ces clés il faille en passer par le niveau physique au moyen d'index « UNIQUE ». Les index jouent évidemment un rôle crucial pour la rapidité de l'accès aux données, mais tout cela doit quand même rester sous le capot...

Pour avoir une connaissance objective de la situation des clés, on peut de préférence passer par l'onglet « Indexes ». Ainsi voit-on que la clé primaire fait l'objet de l'« index primaire » :

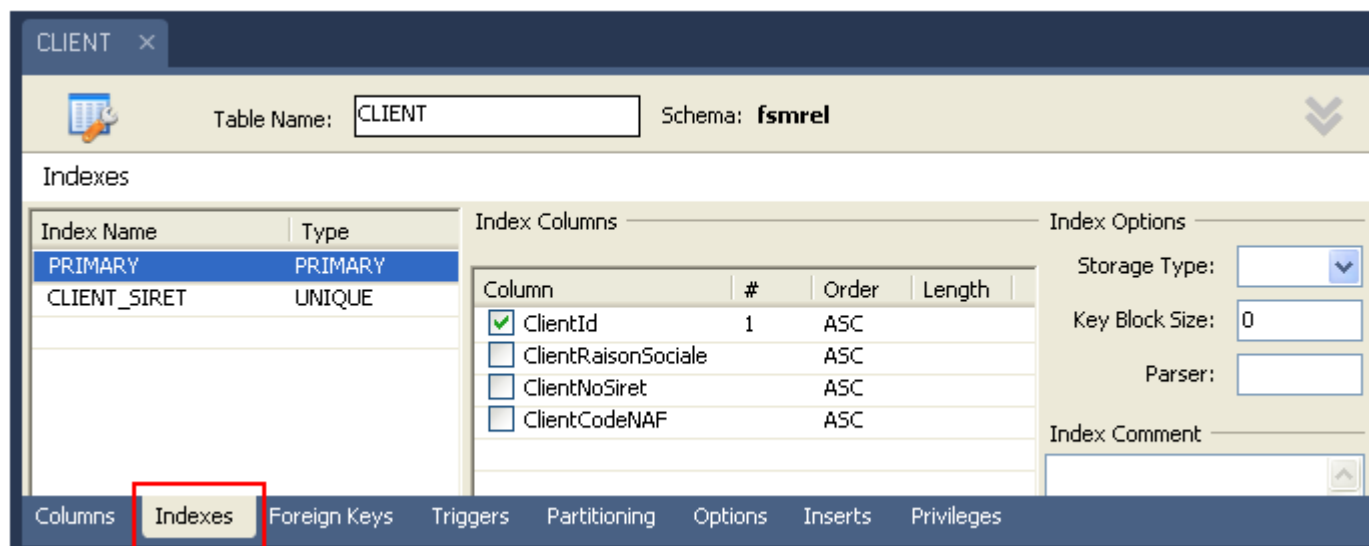


Figure 3.10 - Affichage de la liste des index d'une table, onglet « Indexes »

Onglet « Indexes » : une clé alternative {ClientNoSiret} garantie par un index de type « UNIQUE » :

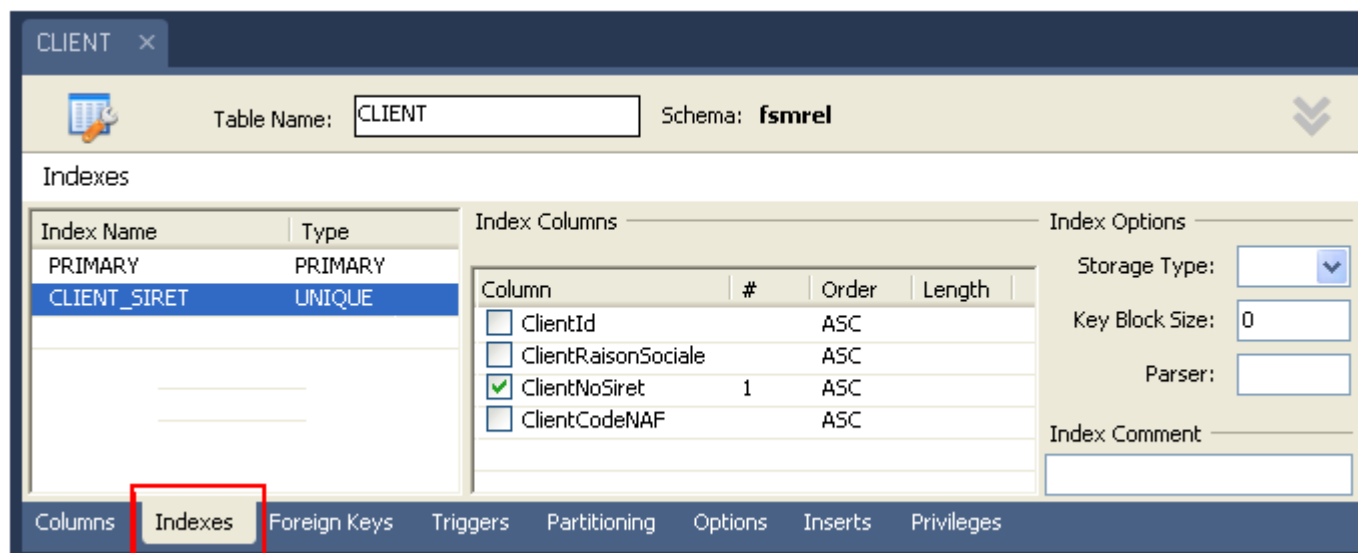


Figure 3.11 - Index « UNIQUE » au service d'une clé alternative

Les clés alternatives singletons (c'est-à-dire composées d'une seule colonne) sont aussi cochées « UQ » (onglet « Columns ») :

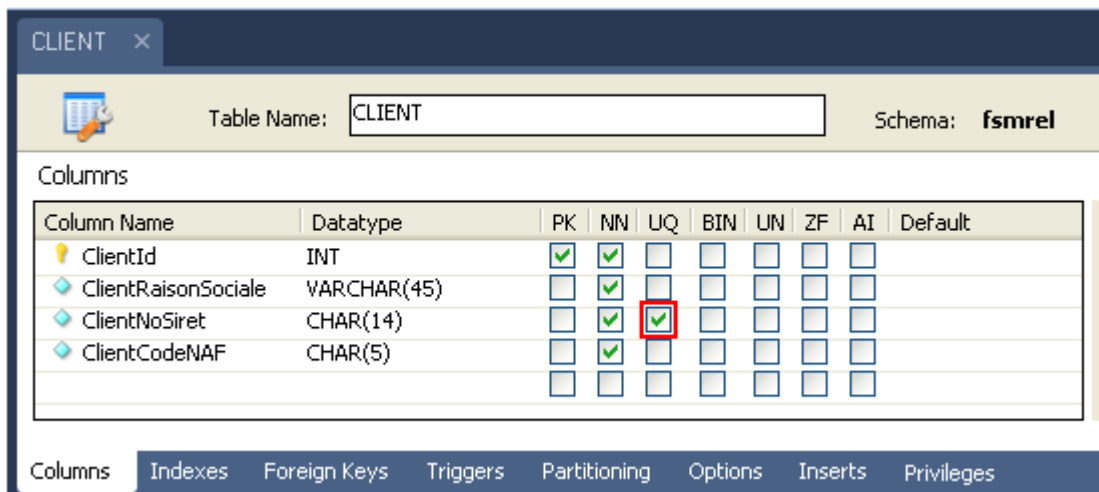


Figure 3.12 - Clé alternative singleton (onglet « Columns »)



## 4. Définir les associations entre les tables (plusieurs à plusieurs)

### 4.1. En amont de MySQL Workbench

Supposons qu'au niveau conceptuel on ait les règles de gestion des données suivantes :

RG314 : Un article peut être stocké dans un ou plusieurs dépôts.

RG315 : Un dépôt est un lieu de stockage pour au moins un article.

RG316 : La quantité d'articles par dépôt doit être connue.

N.B. Par « article », on sous-entend plutôt « type d'article ».

Structure des données, version MCD merisien :

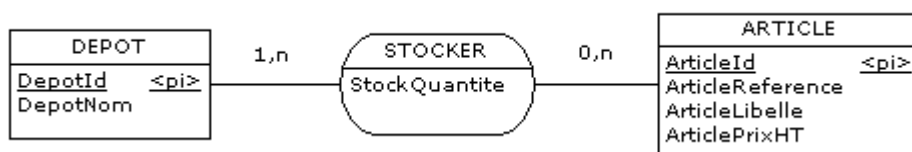


Figure 4.1 - Association N-M (Merise)

Structure des données, diagramme de classes correspondant :

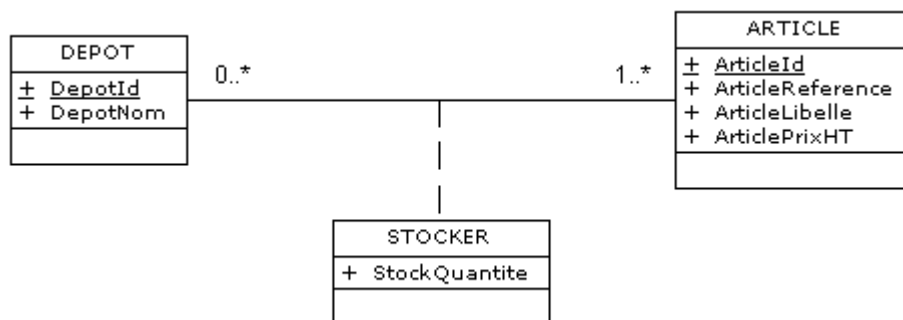


Figure 4.2 - Association N-M (UML)

### 4.2. Avec MySQL Workbench

Au niveau MLD, avant qu'on établisse une association entre DEPOT et ARTICLE, la situation est la suivante :

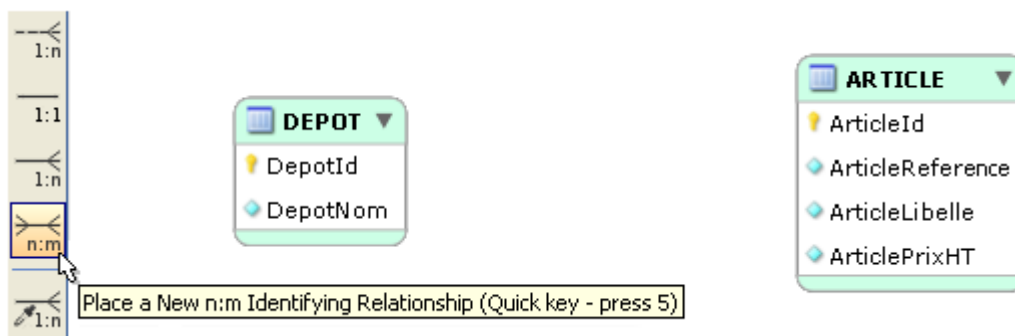


Figure 4.3 - Association identifiante, avant

Pour établir l'association, on clique sur l'icône « n:m Identifying Relationship », puis successivement sur les objets DEPOT et ARTICLE ; au résultat :

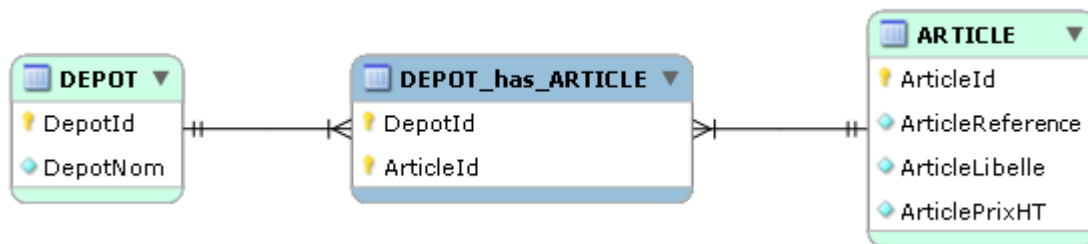


Figure 4.4 - Association identifiante, après



La notation utilisée pour les cardinalités portées par les liens connectant les tables est dite « patte de corbeau » (*Crow's foot*). Le symbole «  $\Leftarrow$  » est synonyme de « N » et le symbole « | » synonyme de « 1 » : un dépôt a au moins un article et un article se trouve dans au moins un dépôt.

### 4.3. Cardinalité minimale 0

L'outil a créé une table d'association DEPOT\_has\_ARTICLE (nommage par défaut, mais on peut changer la règle, cf. paragraphe 10.4.7), dotée d'une clé primaire {DepotId, ArticleId}. On peut renommer cette table en STOCKER ou (STOCK, employer un verbe ou un substantif ou autre chose, après tout peu importe, à chacun ses goûts et ses habitudes). En tout cas, on ajustera les cardinalités minimales pour respecter les règles de gestion et rendre conforme le diagramme à la règle selon laquelle un article n'est pas toujours forcément stocké dans un dépôt : pour cela on double-clique sur la patte connectant STOCK et ARTICLE, puis on décoche la case « Mandatory » associée à STOCK (*Referencing Table*) :

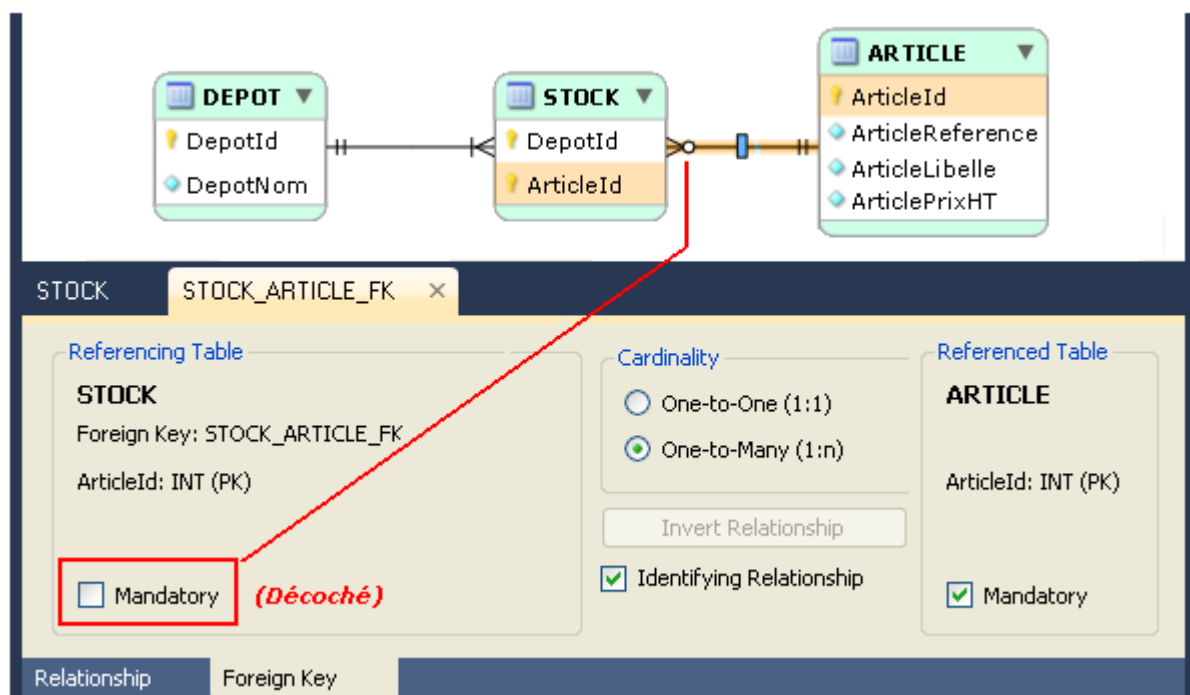


Figure 4.5 - Cardinalité minimale 0

#### 4.4. Clés étrangères

On observera que l'outil a défini automatiquement les clés étrangères nécessaires :

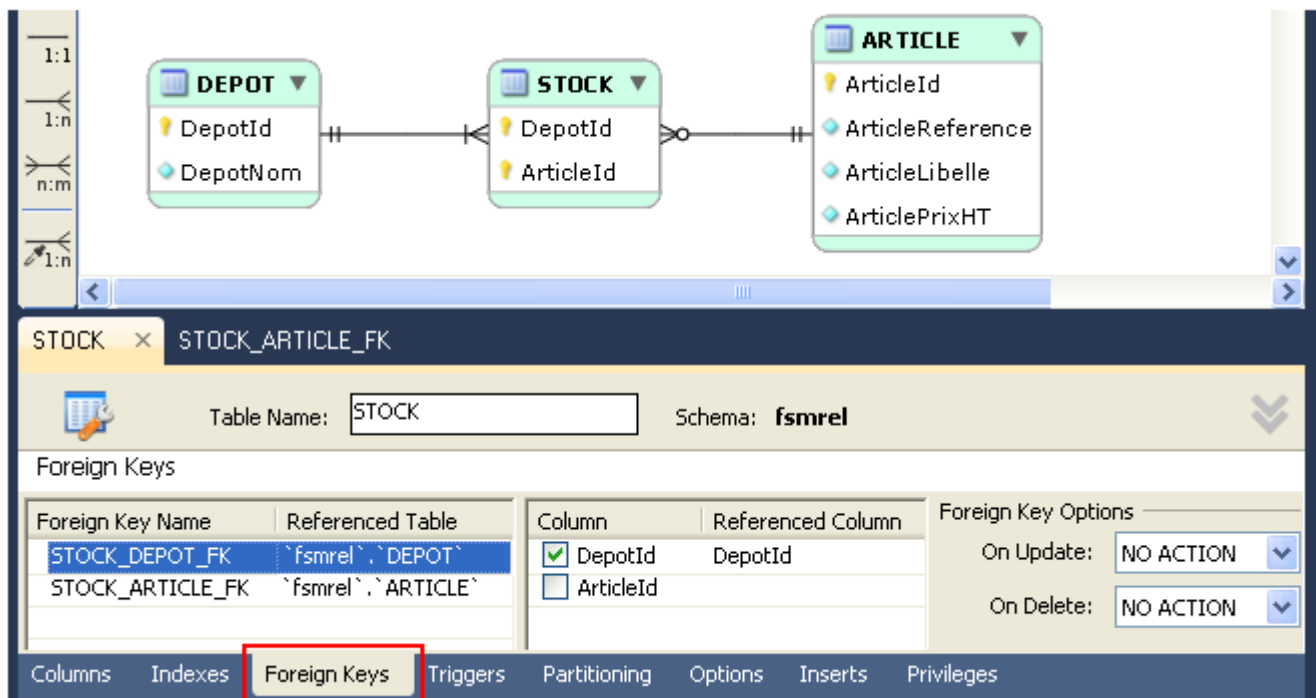


Figure 4.6 - Création des clés étrangères par MySQL Workbench

Au sujet des clés étrangères, de l'intégrité référentielle et du métabolisme des données (option ON DELETE), se reporter au paragraphe 10.4.6.

#### 4.5. Affichage du nom des associations

Le nom des associations devrait être affiché automatiquement. Si ça n'est pas le cas, vérifier si la case « Hide Captions » est bien décochée (cf. paragraphe 10.11). Au besoin, on peut pallier avec l'outil « Texte » :

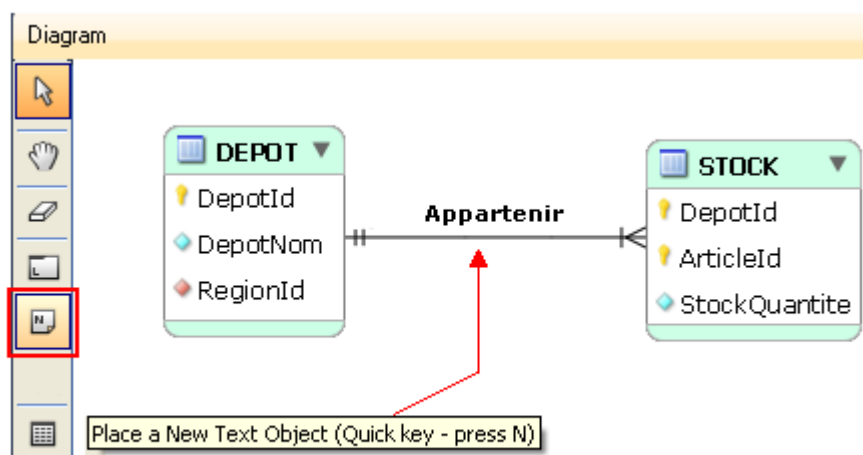


Figure 4.7 - Affichage du nom de l'association : « Appartenir »

## 4.6. Notation UML

La notation utilisée jusqu'ici pour les cardinalités est celle d'IE (Information Engineering), dite encore « patte d'oie » ou « patte de corbeau » (*Crow's foot*). On peut toutefois préférer la notation UML :

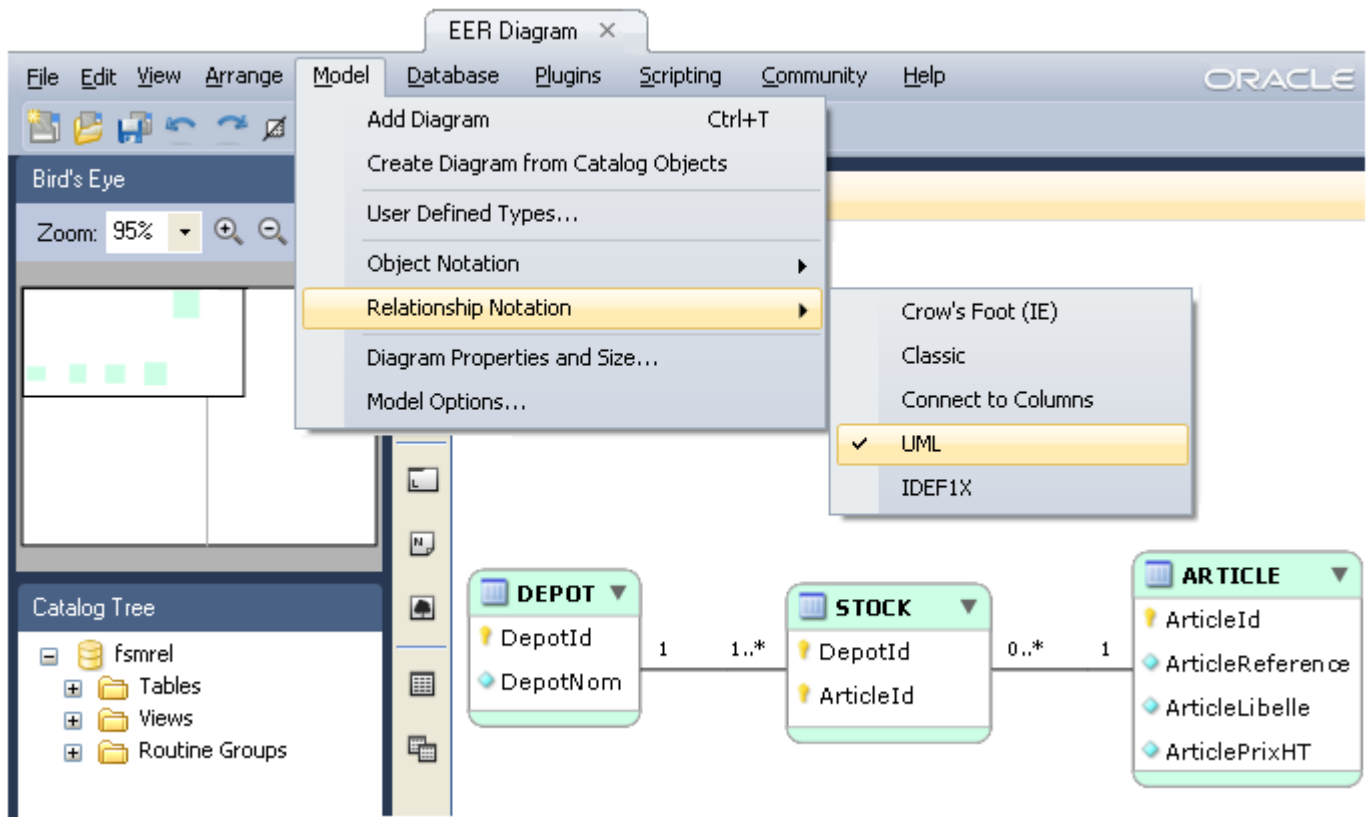


Figure 4.8 - Notation UML

Les Merisiens ne trouveront pas de notation dans le style de Merise, puisqu'on se situe ici au niveau logique (MLD) où les ovales des MCD sont sans objet.

#### 4.7. Notation « à la ACCESS »

Pour les habitués de Microsoft ACCESS, leur notation habituelle est disponible sous l'appellation « Connect to columns », se reporter à [The MySQL Workbench Community Blog](#), Section « About », paragraphe « Notation Styles ». (Voir aussi la représentation d'une hiérarchie, cf. Figure 7.10) :

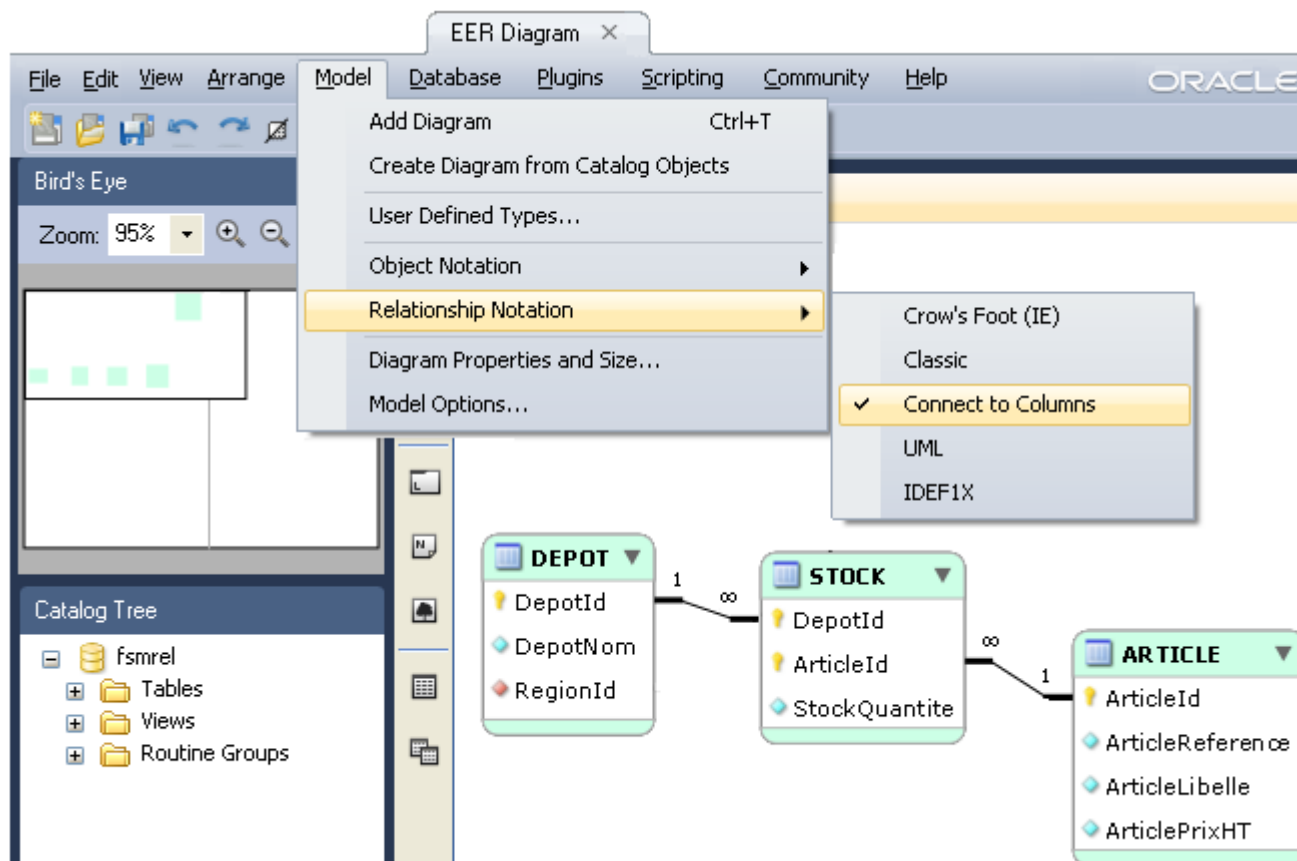


Figure 4.9 - Notation « à la ACCESS »

#### 4.8. Ordre des colonnes dans les clés

En passant, même si la performance des requêtes n'est pas ici l'objet (cf. paragraphe 10.8), on peut vérifier que l'ordre des colonnes dans les clés est bien celui qui convient à ce sujet (au besoin on change cet ordre en agissant sur la colonne « # ») :

STOCK




Table Name:

STOCK

Schema: fsmrel

Indexes

Index Name	Type
PRIMARY	PRIMARY
STOCK_ARTICLE_FK	INDEX
STOCK_DEPOT_FK	INDEX

Index Columns			
Column	#	Order	Length
<input checked="" type="checkbox"/> DepotId		ASC	
<input checked="" type="checkbox"/> ArticleId	1	ASC	
	2		

Figure 4.10 - Ordre des colonnes composant les clés

#### 4.9. Ajout de colonnes

Reste à compléter l'en-tête de la table STOCK, auquel manque la colonne StockQuantite, ce que l'on fait manuellement :

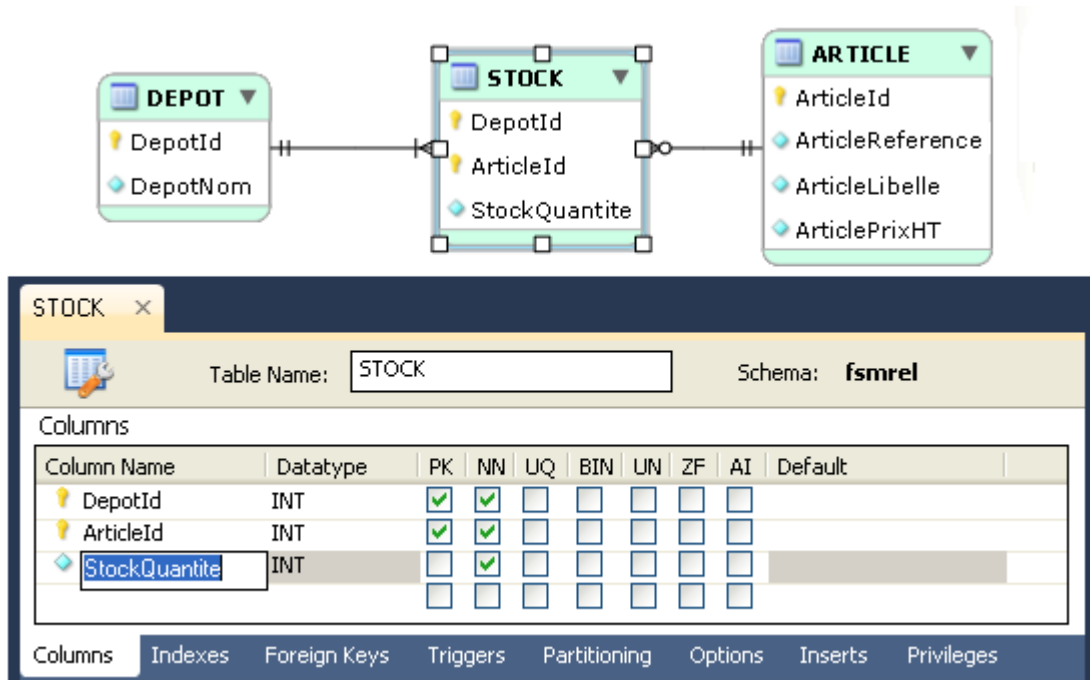


Figure 4.11 - Ajout de colonne

Le Merisien aura noté que la colonne StockQuantite correspond à une propriété appartenant à une association dite « porteuse de propriétés ».

#### 4.10. Index redondants

Une remarque concernant MySQL et le niveau physique :

Quand on définit la clé primaire d'une table, pour être en phase MySQL, dans la foulée MySQL Workbench crée automatiquement un index « primaire » (PRIMARY) branché sur cette table et dont la clé est composée des colonnes de la clé primaire. Ainsi, la clé primaire de la table STOCK étant la paire {DepotId, ArticleId}, le couple <DepotId, ArticleId> constitue la clé de l'index primaire de la table STOCK : jusque-là pas de problème. Mais, comme MySQL exige dans un 1<sup>er</sup> temps que chaque clé étrangère soit dotée de son propre index (toutefois dans un 2<sup>e</sup> temps « *This index might be silently dropped...* »), à son tour MySQL Workbench crée automatiquement un index STOCK\_DEPOT\_FK pour la clé étrangère STOCK\_DEPOT\_FK représentée par le singleton {DepotId}.

Cas des autres SGBD, par exemple DB2 ou MS SQL :

Le raisonnement tenu par MySQL est trop simple (voir la documentation officielle de la version 5.7, paragraphe 13.1.14.2 « [Using FOREIGN KEY Constraints](#) »), puisque pour des SGBD comme DB2 ou MS SQL Server, en l'absence de l'index STOCK\_DEPOT\_FK, l'index primaire pallie. L'index STOCK\_DEPOT\_FK n'apporte rien, c'est un poids mort, voire nuisible (si l'on réfléchit par exemple à l'[effet cluster](#)), et au nom du rasoir d'Ockham il est à supprimer (à faire en dehors de MySQL Workbench s'il s'y oppose !) Quand les index fleurissent, qu'ils soient utiles ou non, c'est toujours au détriment de la performance des opérations de mise à jour qui dans ces conditions peuvent être particulièrement pénalisées (cf. [l'expérience méridionale](#)).

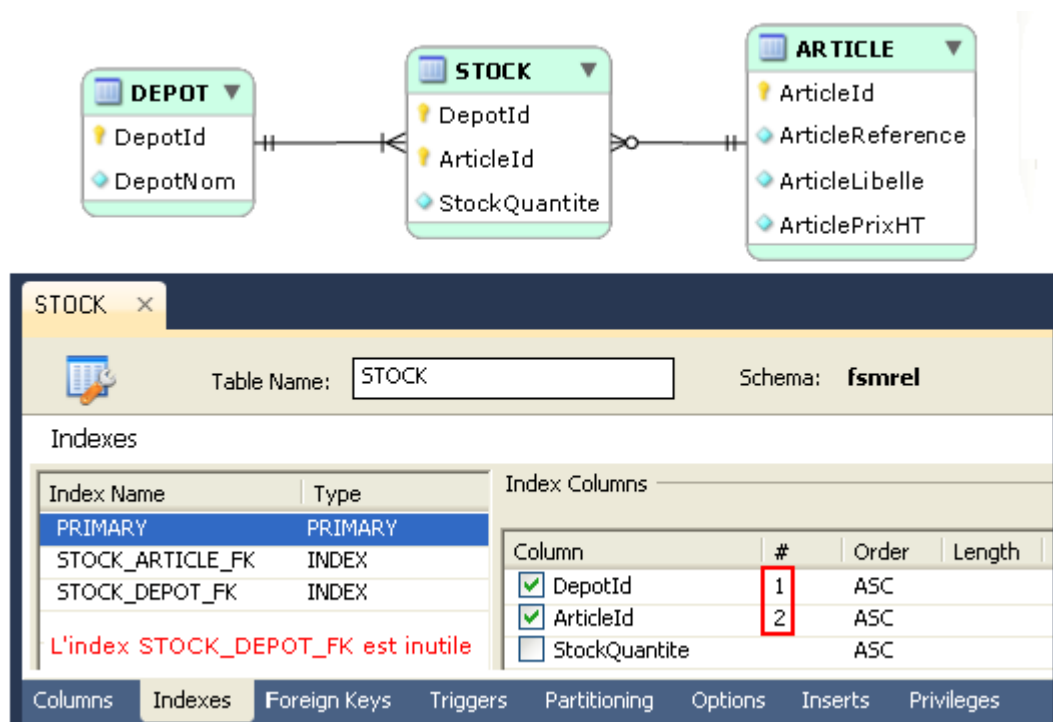


Figure 4.12 - Index redondant, inutile (et pénalisant en mise à jour)



Bonne nouvelle toutefois : à partir de la version 6.0.9, MySQL Workbench autorise la suppression de ce type d'index, merci !

A noter que si le couple <ArticleId, DepotId> constituait la clé de l'index primaire (autre possibilité, choix du DBA), alors ça serait à l'index STOCK\_ARTICLE\_FK de disparaître.

## 5. Associations de un à plusieurs

### 5.1. Associations de un à plusieurs entre entités-types indépendantes

#### 5.1.1. Entité-type indépendante (rappel)

MySQL Workbench permet d'exprimer au niveau logique un concept important, celui d'entité forte, indépendante (cf. paragraphe 1.2). Rappelons qu'une entité (instance) est indépendante (forte, autonome) si son existence ne dépend pas de façon permanente et définitive de celle d'une autre entité. Par exemple, chez Dubicobit, un dépôt n'est pas forcément lié sa vie durant à une même région commerciale, il peut changer d'affectation dans le temps, tandis que la suppression d'une région n'entraînera pas celle des dépôts qui la desservent (pour arriver à supprimer une région il aura donc fallu préalablement rattacher ses dépôts à une autre région).

Structure des données en amont, version MCD merisien :

Une région peut être desservie par plusieurs dépôts et un dépôt dessert exactement une région.



Figure 5.1 - Association 0,N-1,1 (Merise)

Version diagramme de classes :



Figure 5.2 - Association 0,N-1,1 (UML)

#### 5.1.2. Icône « 1:n Non-Identifying Relationship »

Au niveau MLD, avant qu'on associe DEPOT et REGION, la situation est la suivante :

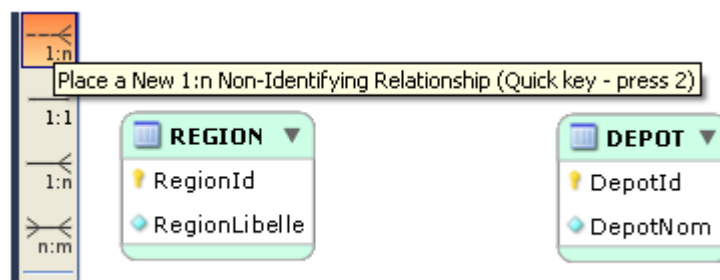


Figure 5.3 - Association 1 à N non identifiante



On sélectionne l'icône « 1:n Non-Identifying Relationship », puis on clique successivement sur les objets DEPOT et REGION (dans cet ordre, c'est-à-dire le référençant d'abord, puis le référencé). Au résultat :

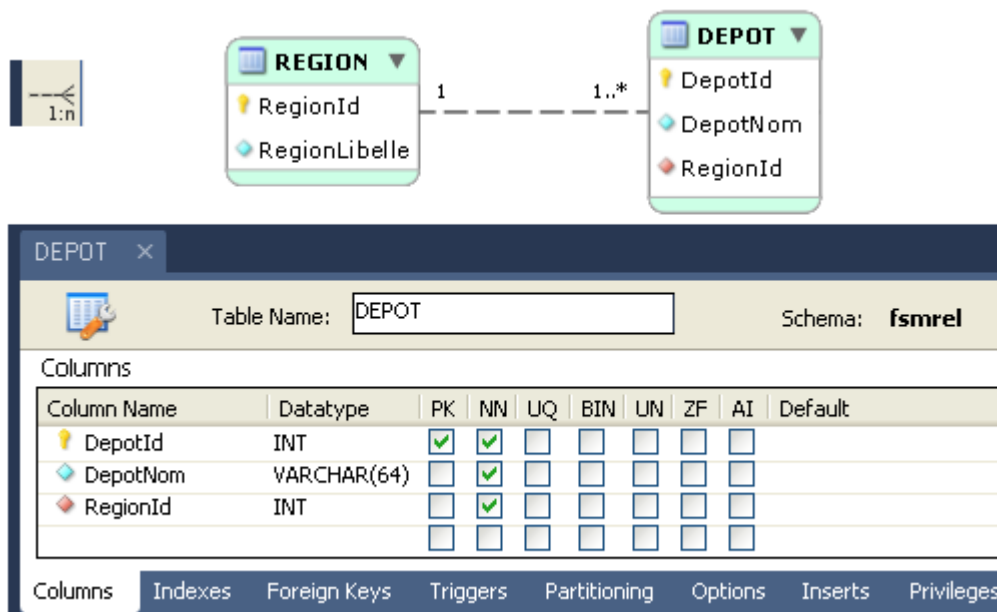


Figure 5.4 - Association 1 à N non identifiante, suite

La patte entre DEPOT et REGION est en pointillés : c'est la façon qu'a l'outil de signaler que DEPOT n'est pas une propriété (multivaluée) de REGION. DEPOT et REGION sont bien des types d'entités fortes, indépendantes (*kernels*) au sens de Codd, et DEPOT se contente de désigner REGION.

Comme une région n'est pas nécessairement desservie par un dépôt, la cardinalité 1,N est à remplacer remplacée par 0,N (décocher la case « Mandatory » du côté DEPOT (*Referencing Table*)) :

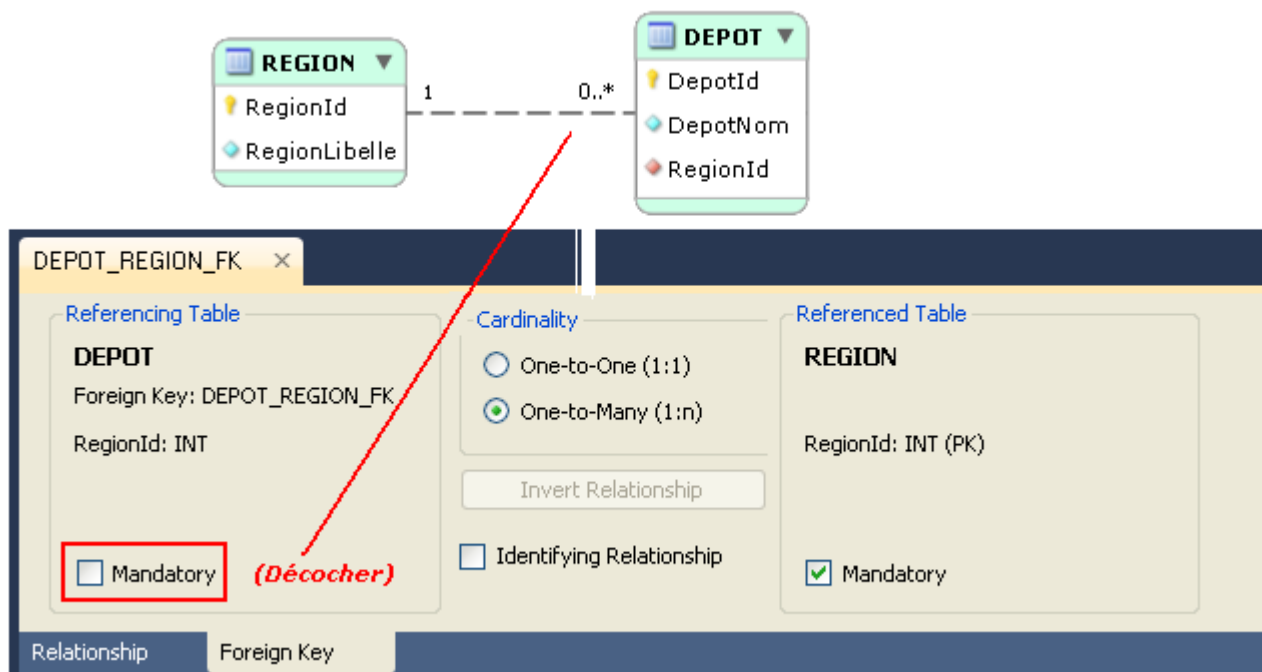


Figure 5.5 - Cardinalité minimale 0

## 5.2. Associations de un à plusieurs (entre entités-types fortes et dépendantes)

### 5.2.1. Associations binaires

#### 5.2.1.1. A la vie à la mort

Supposons qu'en amont du MLD, au niveau conceptuel, on ait les règles de gestion des données suivantes relatives aux commandes des clients :

RG330 : Un client a pu passer plusieurs commandes d'articles.

RG331 : Une commande est passée par exactement un client (c'est-à-dire un et un seul).

RG332 : Une ligne de commande fait référence à exactement un article.

Quand une commande est passée par un client, disons Fernand, elle lui est attachée de manière exclusive, c'est « à la vie, à la mort », elle n'est pas transférable au client Raoul, et la suppression de Fernand devrait en théorie entraîner celle de ses commandes. Une commande devrait donc être considérée comme une entité faible (cf. paragraphe 1.3) relativement au client auquel elle est attachée, mais, à défaut de procéder par historisation ou tactique de ce genre, on joue l'exception qui confirme la règle, à savoir qu'on ne supprime pas un client qui a des commandes en cours, il ne faudrait surtout pas jouer un vilain tour à l'utilisateur (« Où sont passées mes commandes ? »)

Pour mettre en évidence cette situation de rattachement exclusif de ses commandes à Fernand, on pourra utiliser l'identification relative, tout en notant bien qu'il ne sera pas possible de supprimer ce client tant qu'il aura des commandes (sous-entendu : en cours).

La situation initiale :

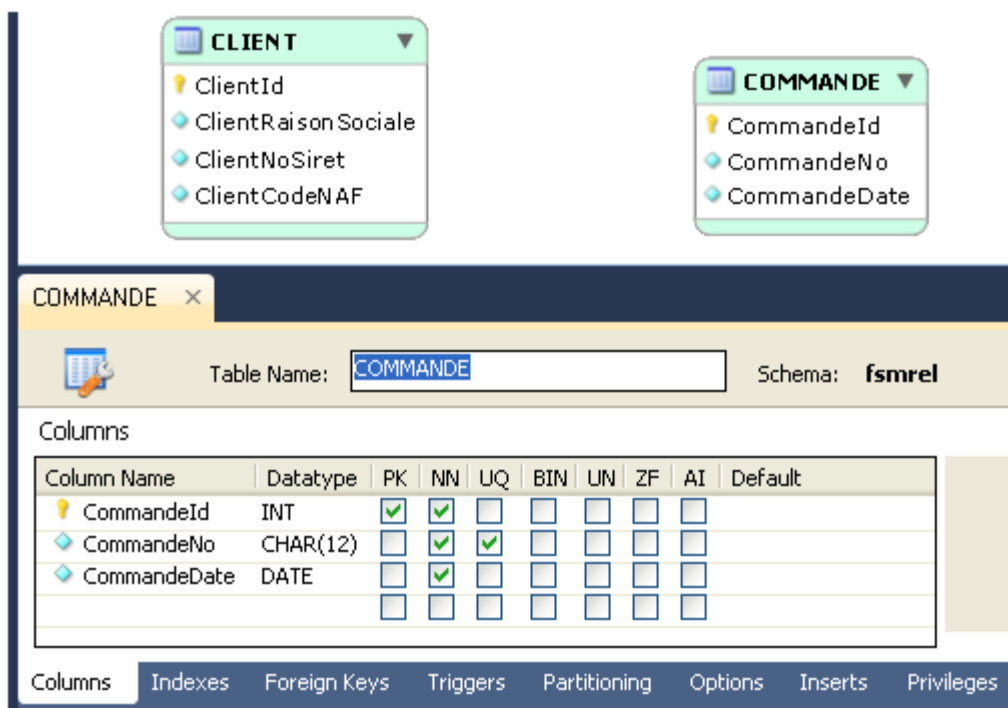


Figure 5.6 - Association 1 à N, à rendre au besoin identifiante

### 5.2.1.2. Identification absolue / relative

Ci-dessus, la table COMMANDE a pour clé primaire {CommandeId} et pour clé alternative {CommandeNo}. Par hypothèse, la colonne CommandeNo représente le numéro de commande, lequel doit être unique, mais c'est aussi une propriété naturelle de la commande et sa structure est du ressort de l'utilisateur. Dans ces conditions, il est nécessaire de disposer d'une clé **invariante** et **non significative**, c'est-à-dire dont le contrôle échappe à l'utilisateur, d'où la mise en œuvre pour cela de la colonne CommandeId, correspondant à une propriété artificielle. On peut en rester là si on se cantonne à l'identification « absolue » de COMMANDE, mais on peut aussi identifier COMMANDE **relativement** à CLIENT : il en résulte un glissement sémantique, mais on résout ainsi une très grande majorité des problèmes liés aux contraintes de chemin comme dans l'exemple des [devis et des factures](#). Au-delà de la modélisation pure, cela peut aussi aider le DBA à résoudre des problèmes de performance observés au cours des travaux de prototypage (récupération par exemple de toutes les commandes de tel client), de stratégie de sauvegarde, toutes choses pour lesquelles le choix de l'index dit [cluster](#) sera déterminant (choix crucial, car il est en général très difficile de changer une fois les tables en production).

Nous ne prendrons pas parti ici. Si on préfère en rester à l'identification absolue, on sera amené à procéder comme dans le cas des entités-types indépendantes (cf. paragraphe 5.1.2), pour obtenir :

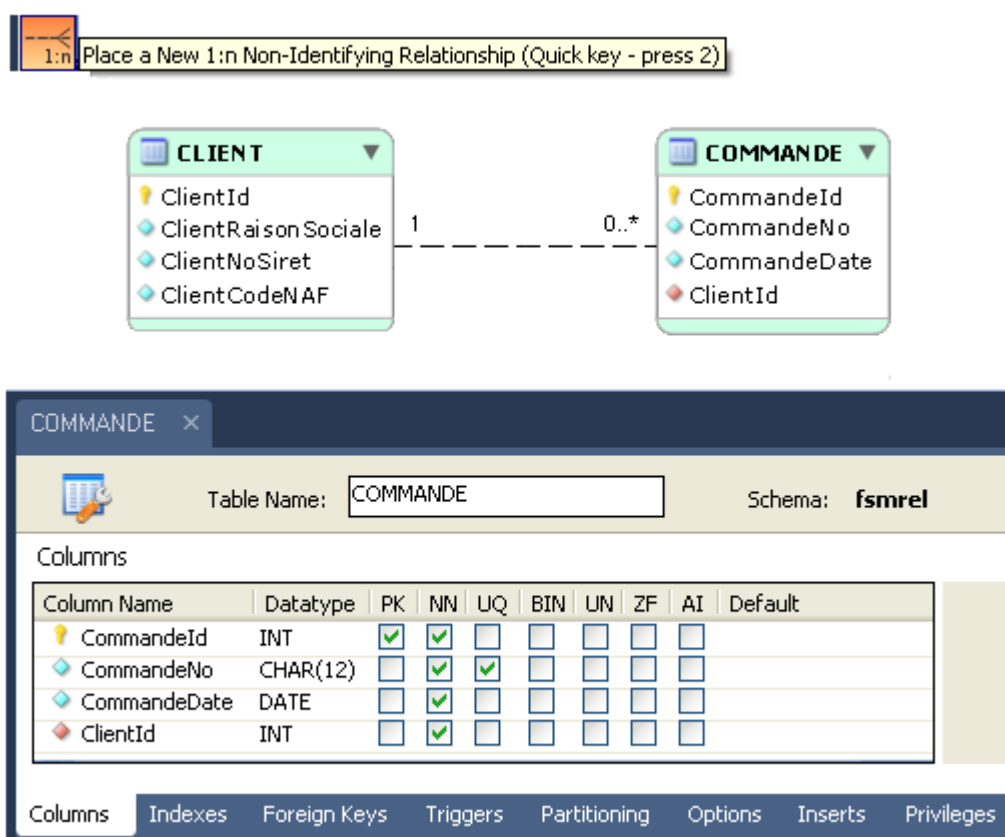


Figure 5.7 - Association 1 à N (identification absolue)

N.B. Dans le cas de l'identification absolue, si on utilise par exemple DB2 ou SQL Server, la clé de l'index cluster sera de préférence composée de la colonne ClientId.

Comme on l'a évoqué, on peut donc aussi choisir d'utiliser l'identification relative :

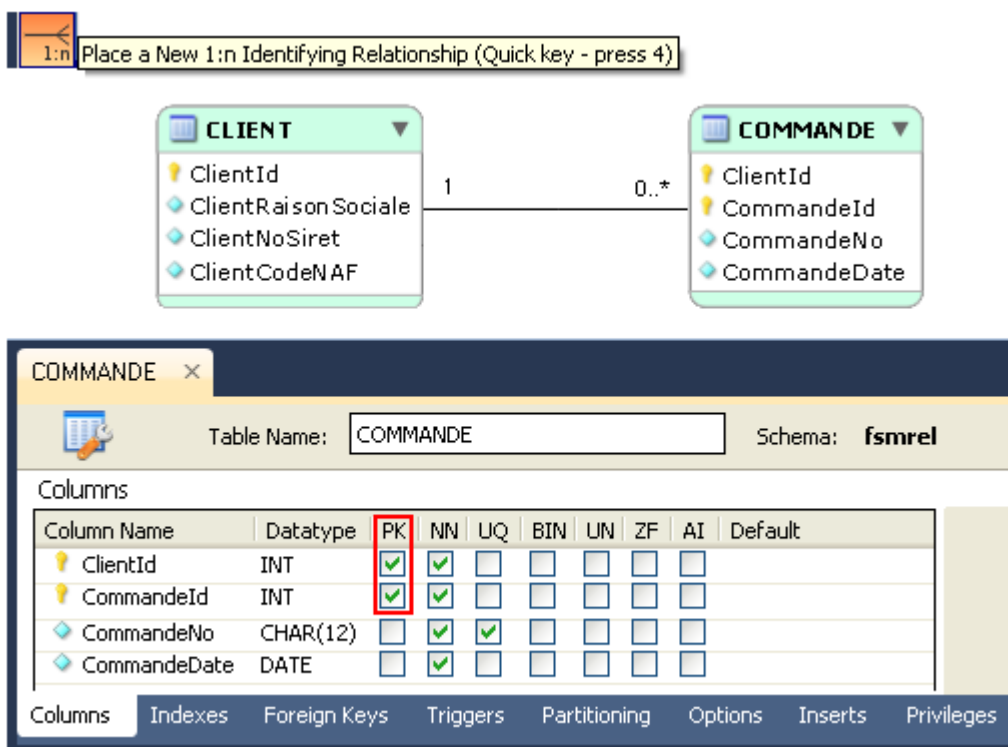


Figure 5.8 - Association 1 à N, identification relative

N.B. Le DBA pourra préférer cette solution permettant de diminuer d'une unité le nombre d'index ; en effet l'index « primaire » (PRIMARY) a pour clé le couple <ClientId, CommandeId> tandis que l'index COMMANDE\_CLIENT\_FK a pour clé le singleton <ClientId> : on a de nouveau affaire aux index redondants (cf. paragraphe 4.10) :

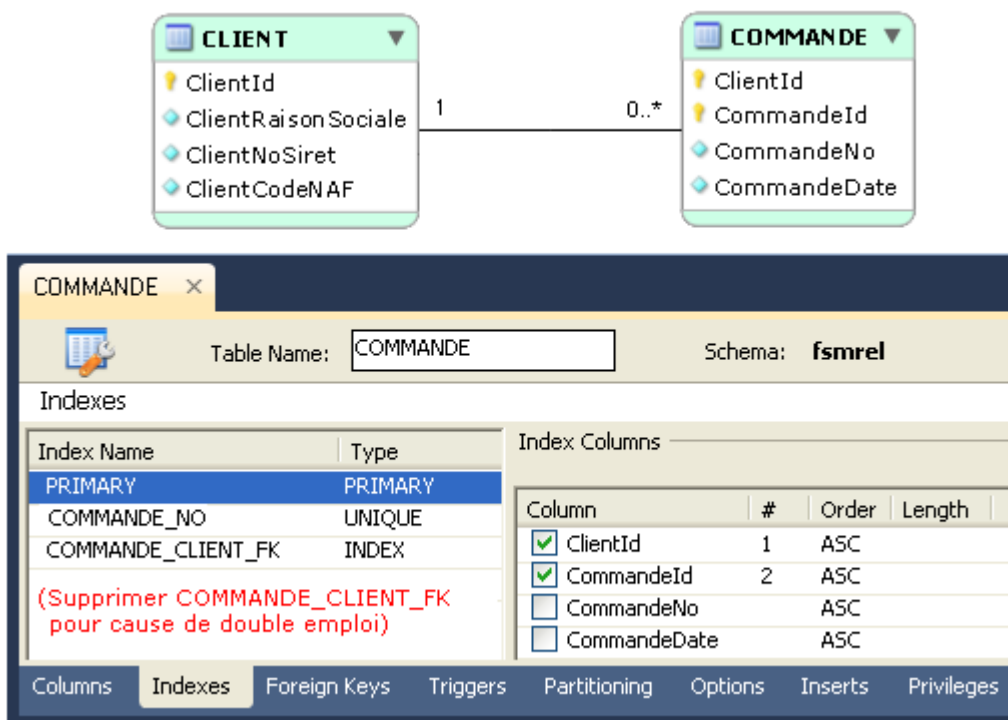


Figure 5.9 - Suppression des index inutiles

Rappelons à cette occasion que les valeurs prises par la colonne CommandeId sont relatives, c'est-à-dire qu'elles sont obtenues par incrémentation, mais avec une numérotation de CommandeId qui commence à 1 pour chaque client. Voir le billet de [CinePhil](#) à ce sujet : « [Trigger pour incrémentation relative](#) ». Rappelons aussi que la suppression d'un client doit être refusée si celui-ci a au moins une commande, se reporter au paragraphe 10.4.6 pour ce qui a trait au métabolisme des données (option ON DELETE) :

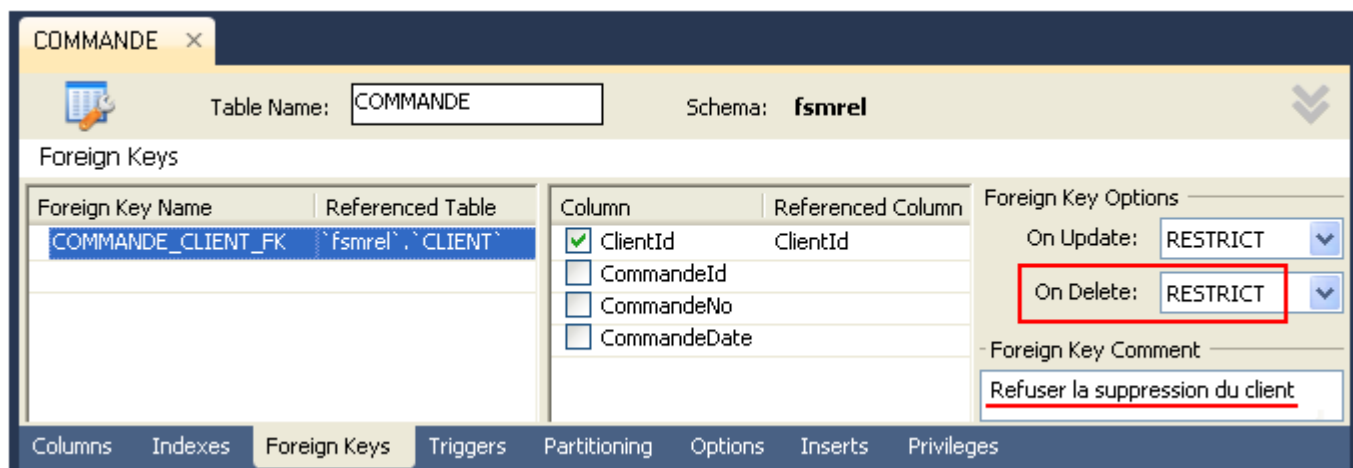


Figure 5.10 - Option RESTRICT

En tout cas, que l'on ait identifié COMMANDE de façon absolue ou relative, il n'y a pas à hésiter en ce qui concerne les lignes de commande, car elles correspondent véritablement à une propriété (multivaluée) de la commande (on peut aussi voir une ligne de commande comme une association entre une commande et un produit).

Situation avant établissement de la relation entre COMMANDE et LIGNE\_COMMANDE :

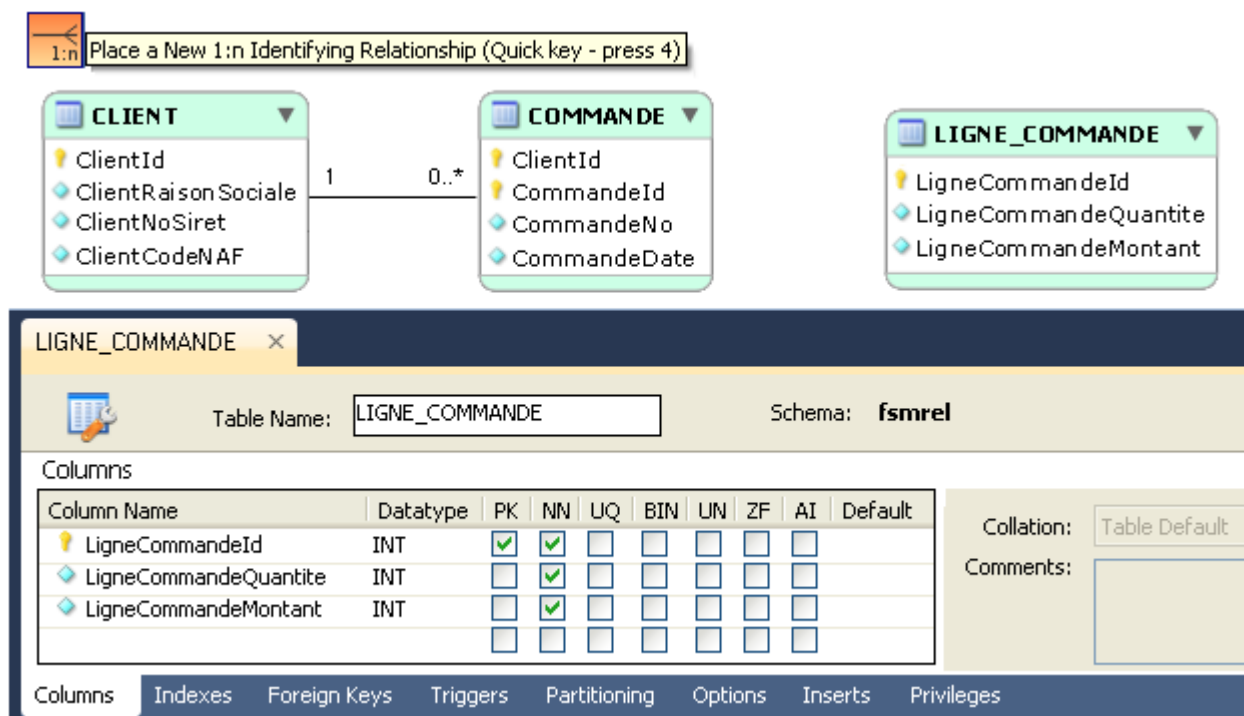


Figure 5.11 - LIGNE\_COMMANDE (avant)

Situation après :

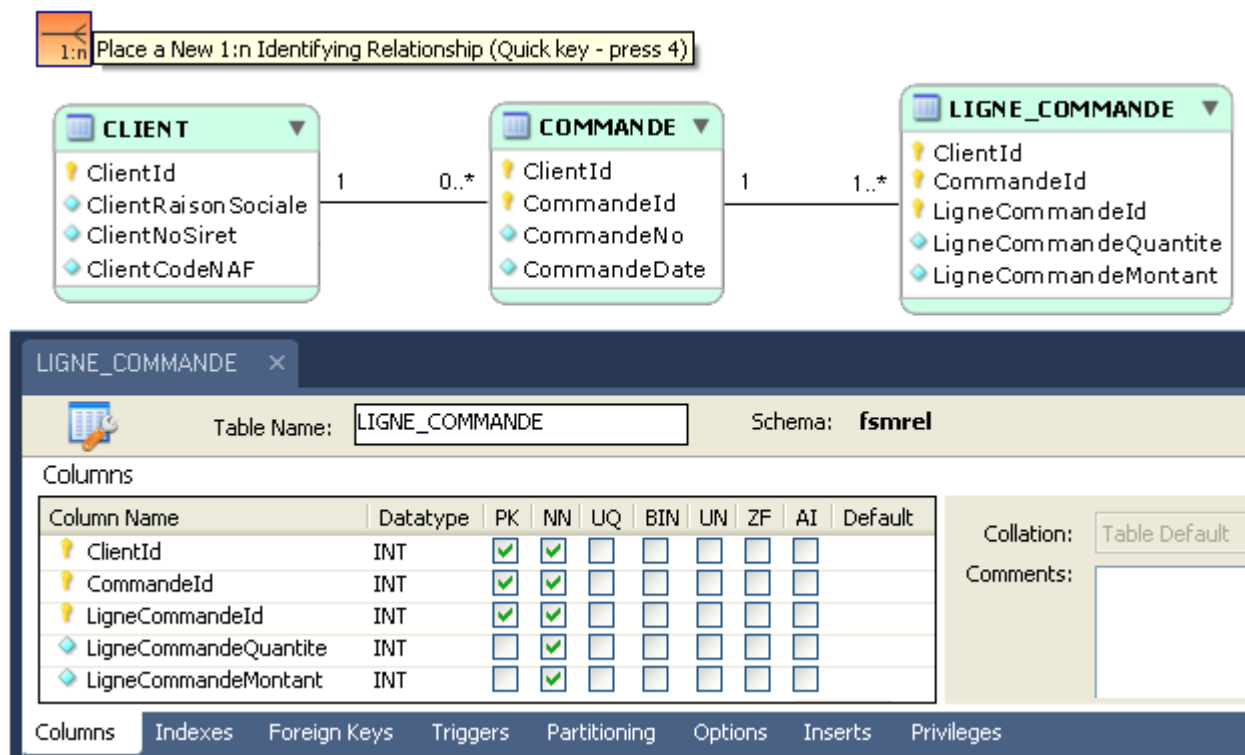


Figure 5.12 - LIGNE\_COMMANDE (après)

Un index, à savoir LIGNE\_COMMANDE\_COMMANDE\_FK, est à éliminer (cf. paragraphe 4.10), car il a pour clé le couple <ClientId, CommandeId>, inclus dans le triplet <ClientId, CommandeId, LigneCommandeId>, lui-même utilisé pour la clé de l'index primaire de la table LIGNE\_COMMANDE (où — identification relative oblige — LigneCommandeId est à numéroté relativement à <ClientId, CommandeId>) :

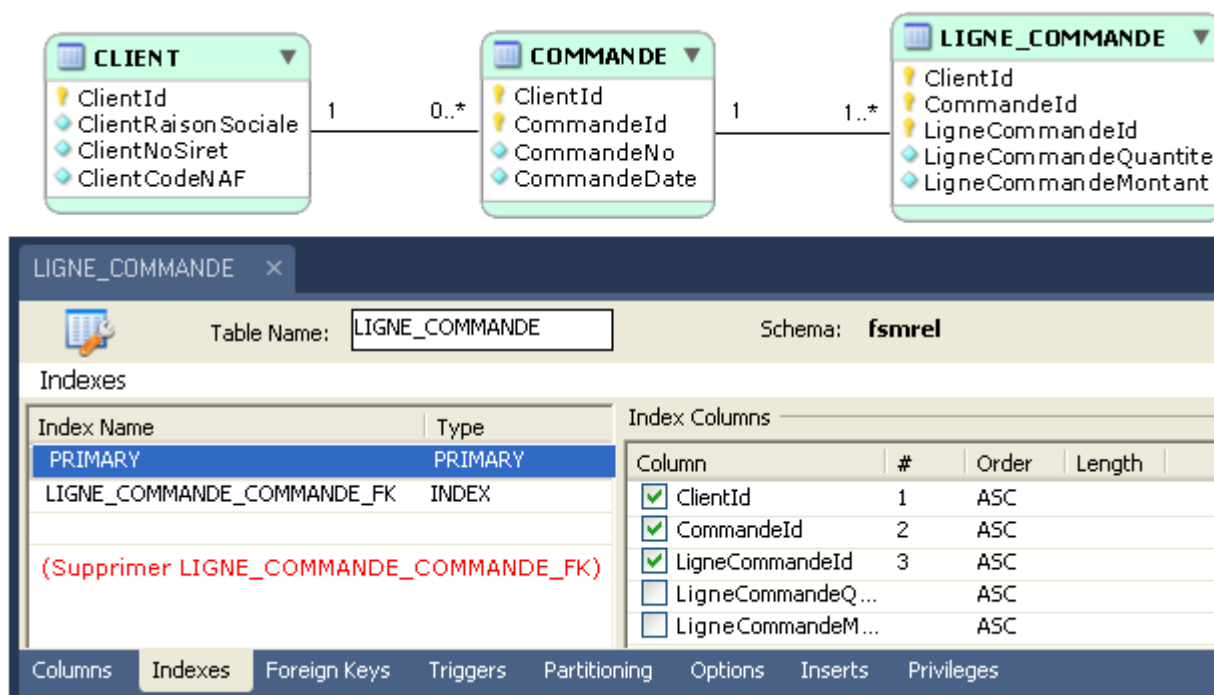


Figure 5.13 - Index superflu à éliminer

En ce qui concerne l'action compensatoire ON DELETE, elle est évidemment CASCADE puisque les lignes de facture sont réellement des entités faibles (cf. paragraphe 1.3), ce qui oriente les choix quant aux conséquences de la suppression d'une commande cf. paragraphe 10.4.6) :

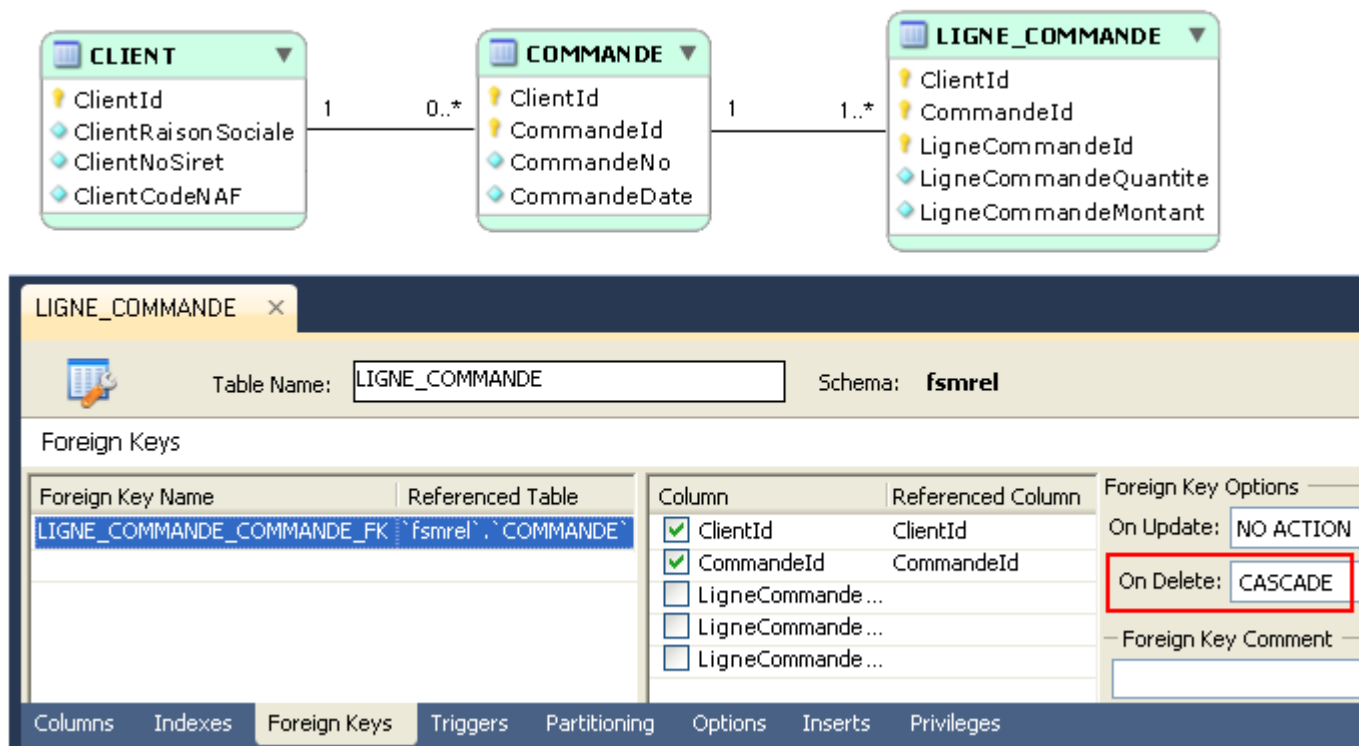


Figure 5.14 - ON DELETE CASCADE

### 5.3. Retour sur les associations de plusieurs à plusieurs : associations ternaires

Le procédé utilisé pour les associations binaires vaut bien entendu pour les associations ternaires et au-delà. Supposons qu'on veuille établir une telle association pour savoir quels langages de programmation les développeurs de l'entreprise Dubicobit utilisent dans le développement des projets de la maison.

Situation initiale :

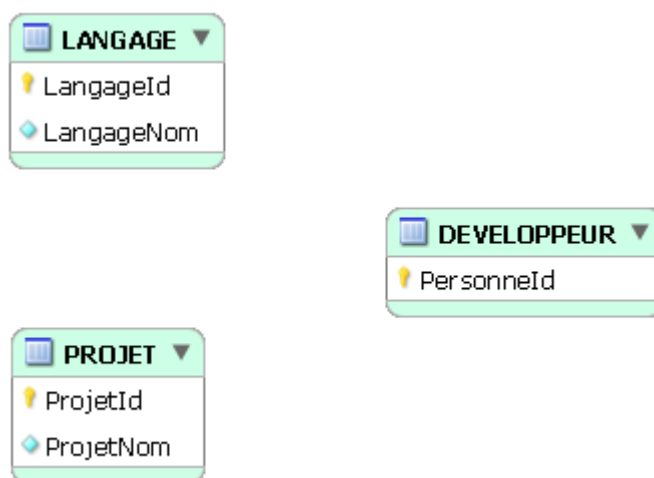


Figure 5.15 - Association ternaire, situation initiale

On définit la table servant à implémenter l'association ternaire, appelons-la UTILISATION :

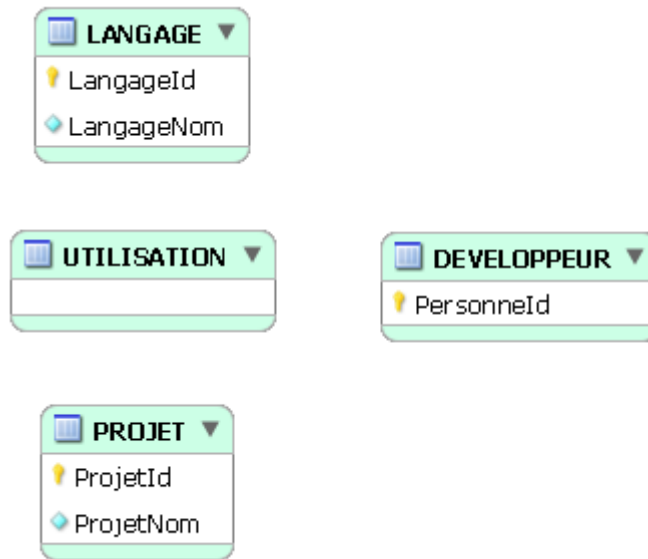


Figure 5.16 - Association ternaire, table UTILISATION vide

On établit un 1er lien identifiant :

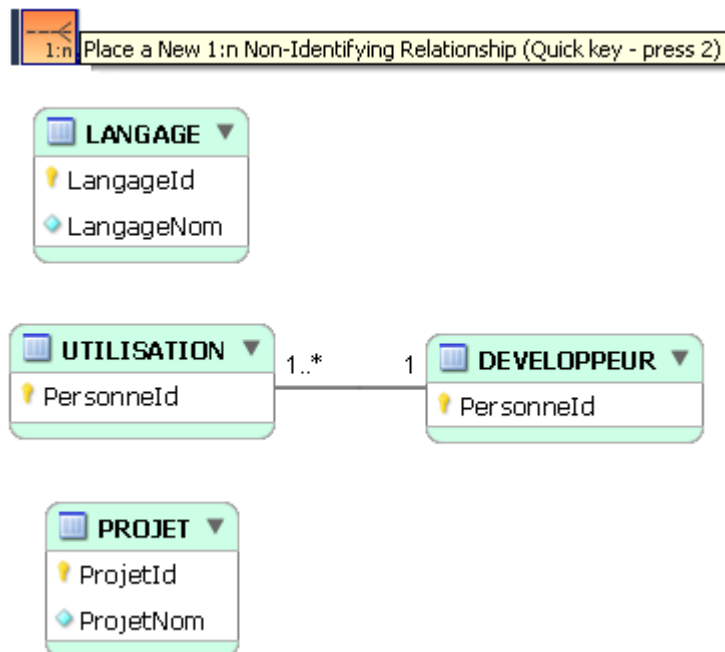


Figure 5.17 - Association ternaire, un 1er lien



Puis on établit un 2e lien :

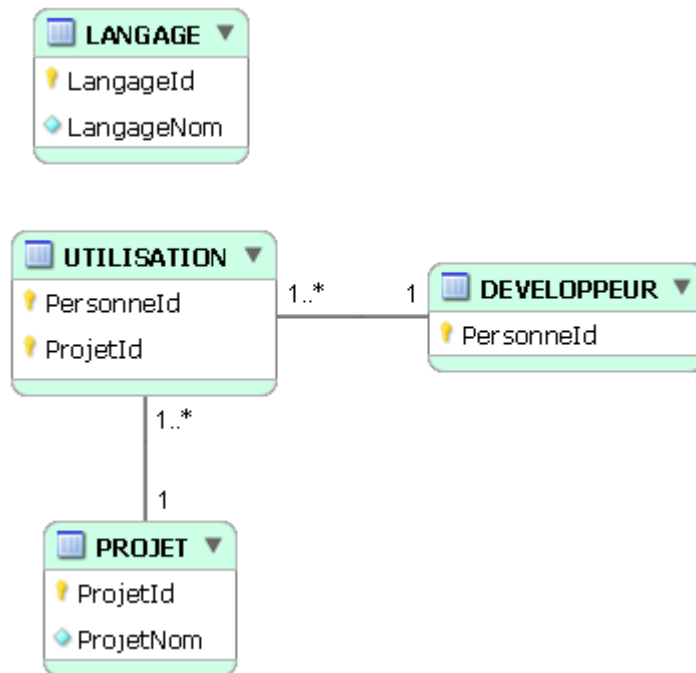


Figure 5.18 - Association ternaire, un 2e lien

Et enfin le 3e lien :

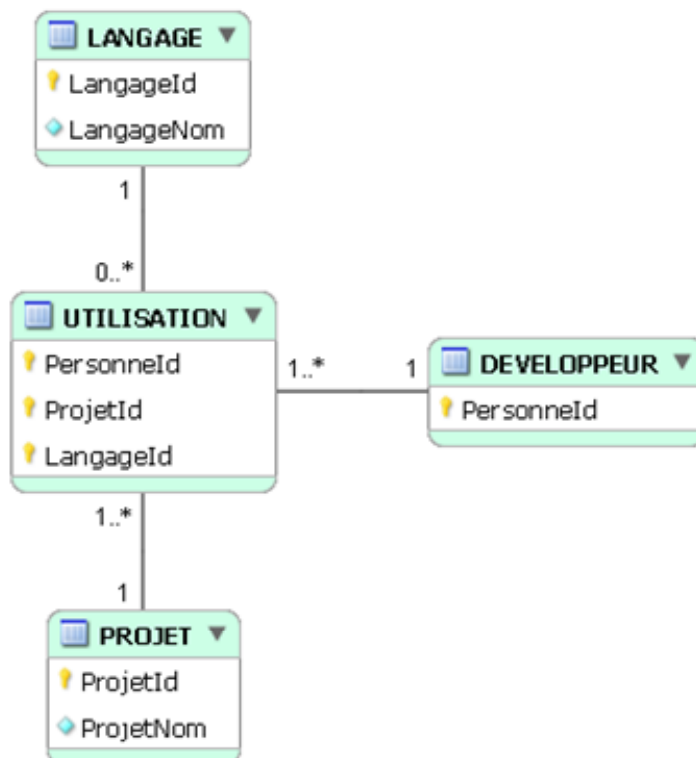


Figure 5.19 - Association ternaire complétée

## 6. Associations de un à un (généralisation/spécialisation)

### 6.1. Rappels

Partons de la généralisation/spécialisation des types d'entités merisiennes ou des classes umliennes. Le but est de traduire en diagramme MWB le diagramme de classes ci-dessous, relatif au référentiel PERSONNES de l'entreprise Dubicobit, selon lequel :

- Une personne peut avoir plusieurs adresses ;
- Une personne peut être un collaborateur Dubicobit ou un tiers ;
- Un collaborateur peut être un directeur ou un employé ;
- Un tiers peut être un client ou un fournisseur de Dubicobit.

Diagramme de classes UML :

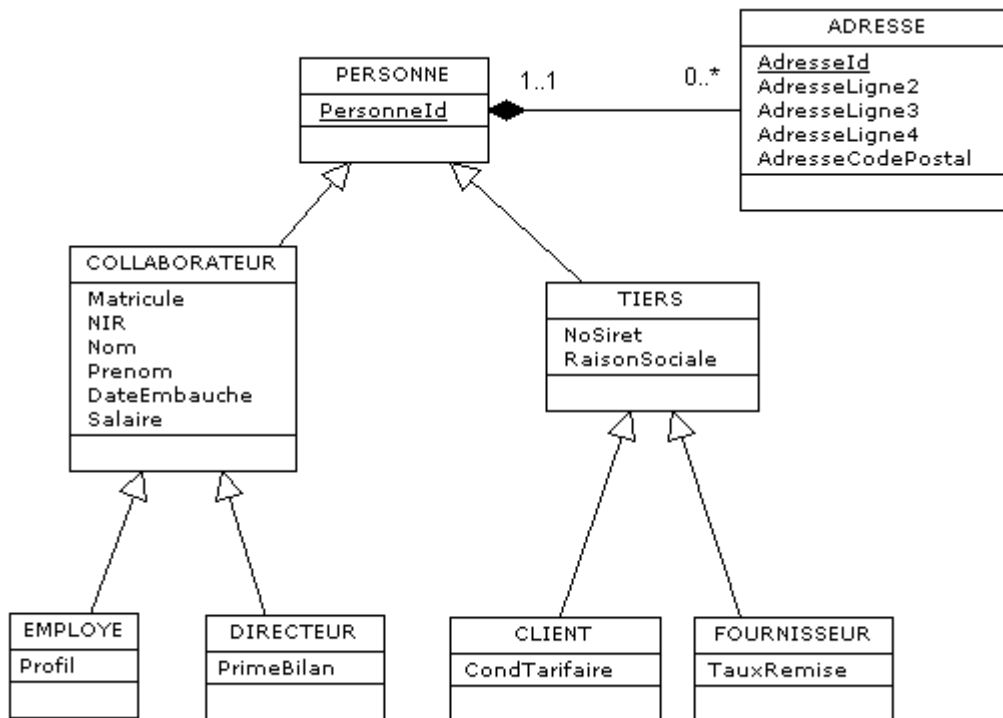


Figure 6.1 - Généralisation/spécialisation, diagramme de classes

Rappelons au passage que le losange accolé à PERSONNE symbolise une relation de composition, c'est-à-dire qu'une adresse est la propriété d'une personne, ADRESSE est un type d'entité faible au sens Entité/Relation (cf. paragraphe 1.3) : l'association est identifiante, du type un à plusieurs (cf. paragraphe 5.2). En revanche une relation de spécialisation est à traduire par une association identifiante de type 1:1 (1,1 : 0,1). Dans cette affaire sont donc parties prenantes les relations de spécialisation qu'entretiennent PERSONNE et COLLABORATEUR, PERSONNE et TIERS, COLLABORATEUR et EMPLOYE, COLLABORATEUR et DIRECTEUR, TIERS et CLIENT, TIERS et FOURNISSEUR.

## 6.2. Icône « 1:1 Identifying Relationship »

Mettons en œuvre une table PERSONNE (de clé primaire {PersonneId}) et une table COLLABORATEUR (ici d'en-tête vide) et « tirons » un lien identifiant 1:1 entre les deux tables (*Identifying Relationship*) :

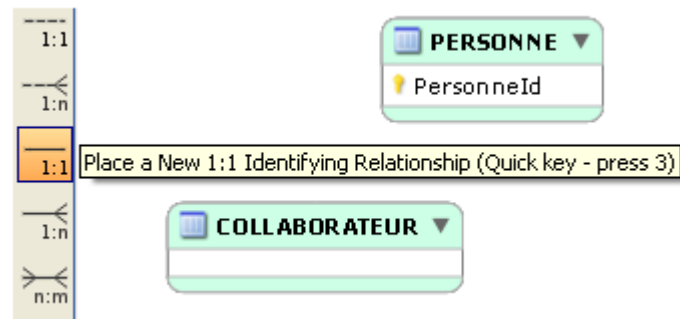


Figure 6.2 - Lien identifiant 1:1 à établir

Une fois les deux tables connectées, la situation est la suivante :

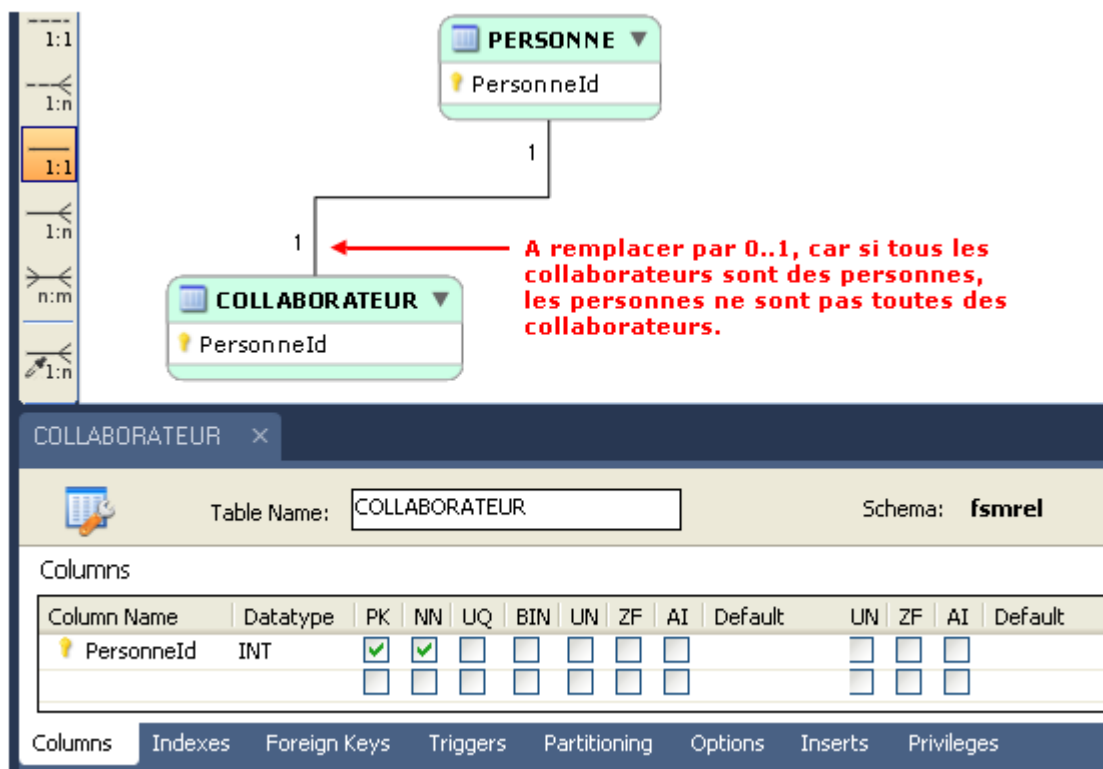


Figure 6.3 - Lien identifiant (bijection)

L'outil a ajouté une colonne PersonneId à l'en-tête de la table COLLABORATEUR et établi une bijection entre celle-ci et la table PERSONNE. Le singleton {PersonneId} est à la fois clé primaire de COLLABORATEUR et clé étrangère vis-à-vis de PERSONNE.

Pour remplacer la cardinalité minimale (multiplicité) 1 par 0, il faut cliquer sur le lien, puis sur l'onglet « Foreign Key » et décocher la case « Mandatory » côté COLLABORATEUR (Referencing Table) :

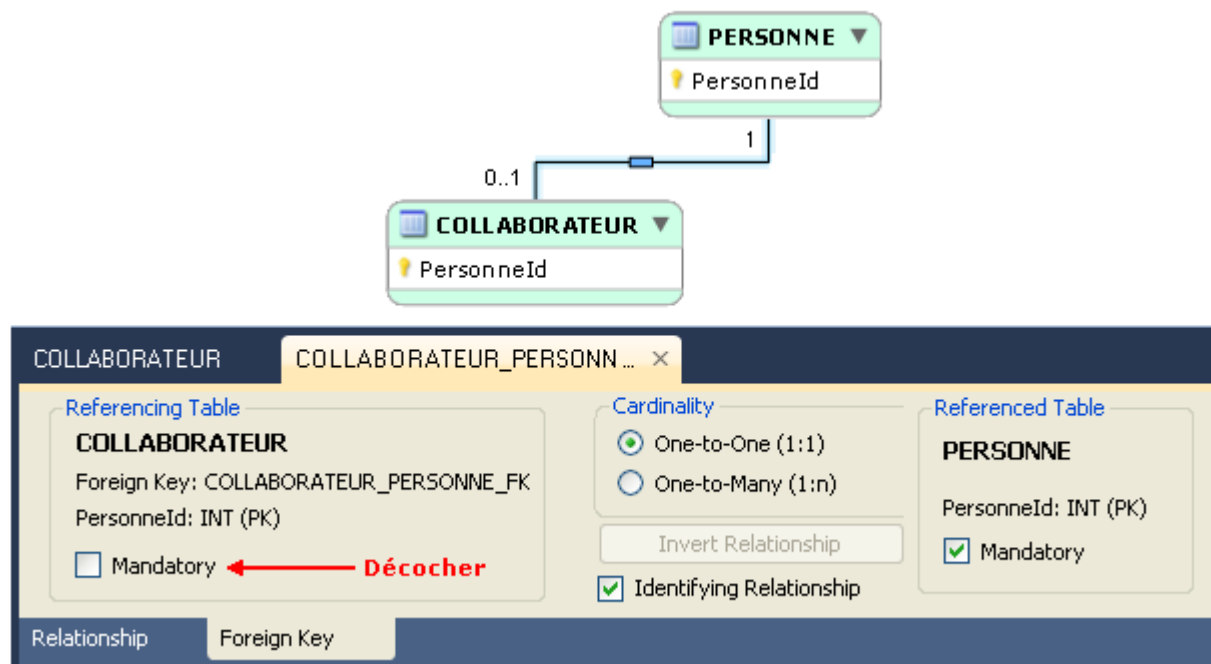


Figure 6.4 - Lien identifiant (injection)

Au sujet du métabolisme (cf. paragraphe 10.4.6) : ci-dessous, on retient l'option ON DELETE CASCADE pour la clé étrangère, car un collaborateur étant une personne, sémantiquement parlant on supprime un collaborateur plutôt qu'une personne au sens large, même si (à moins d'en passer par une vue qui soit la jointure naturelle de PERSONNE et COLLABORATEUR) les opérations de suppression portent en fait sur la table PERSONNE (DELETE FROM PERSONNE WHERE ...) :

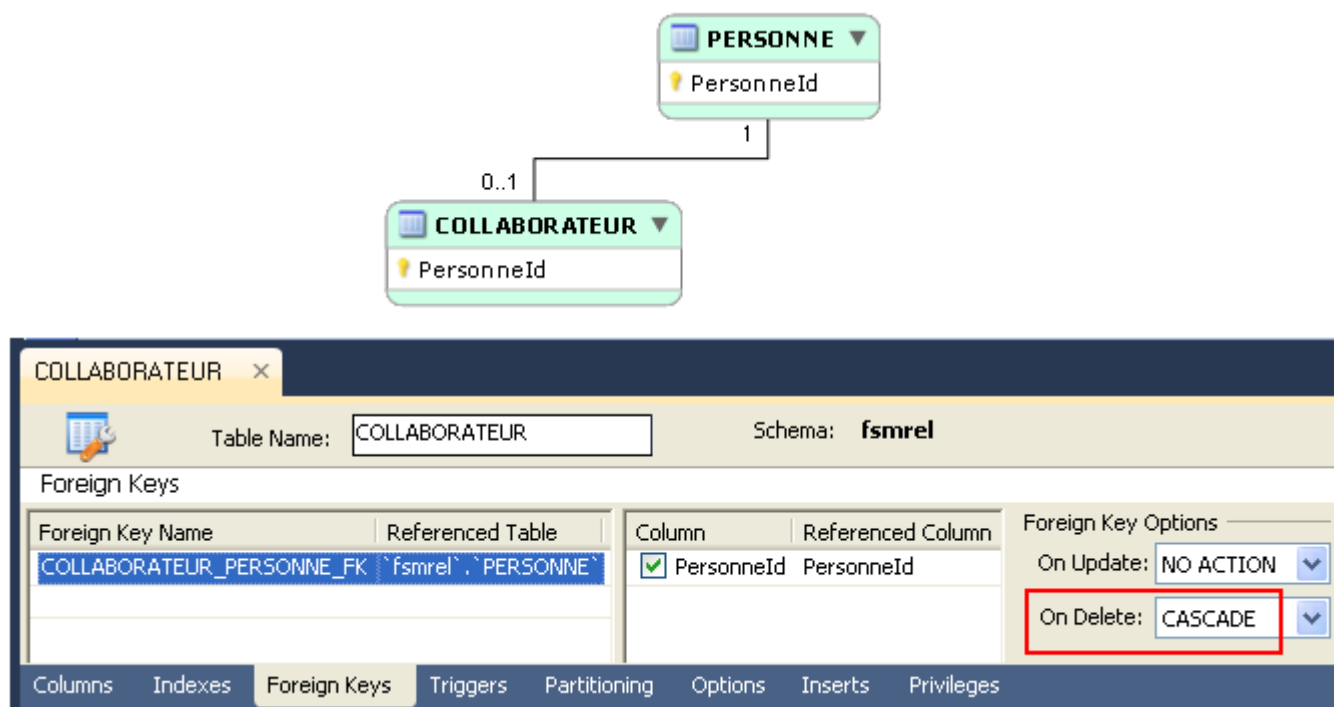


Figure 6.5 - Lien identifiant (ON DELETE CASCADE)

### 6.3. Diagramme final

On procède de la même façon avec les autres tables, sans oublier de compléter l'en-tête des tables, pour obtenir au bout du compte le diagramme :

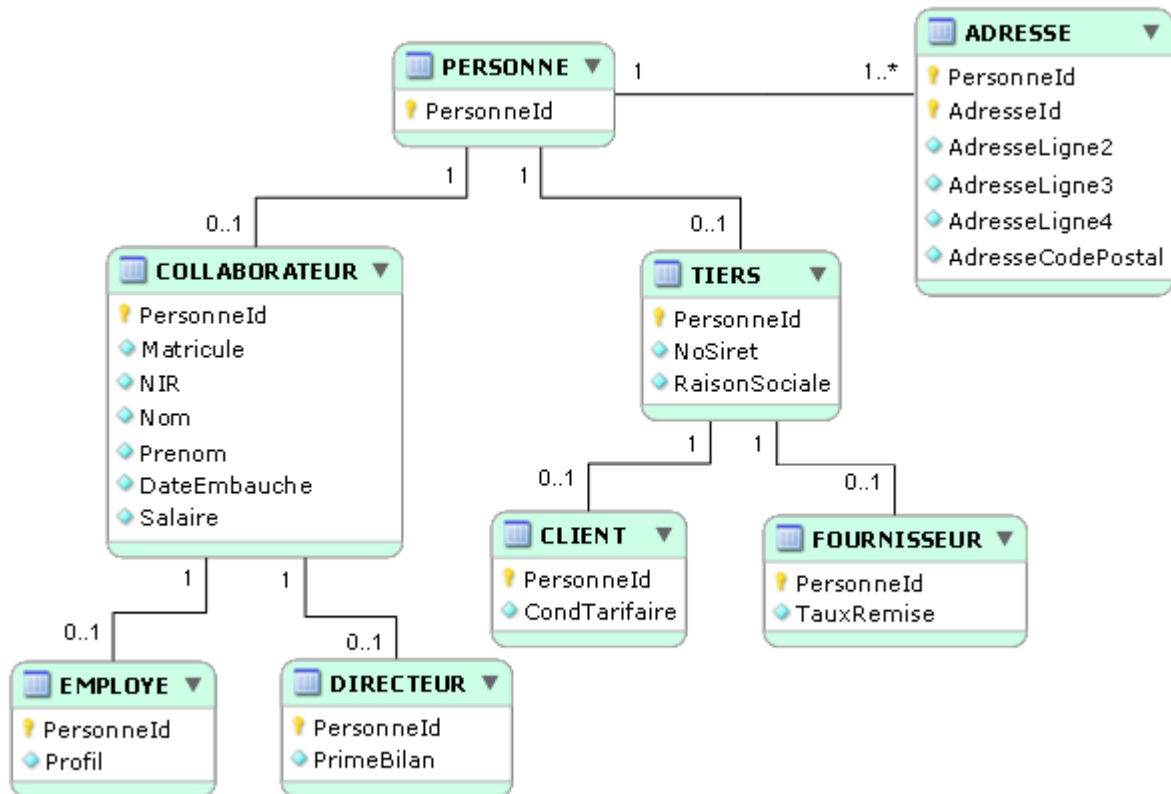


Figure 6.6 - Généralisation/spécialisation, au final

N.B. Pour afficher le nom des rôles, se reporter au paragraphe 10.11 :

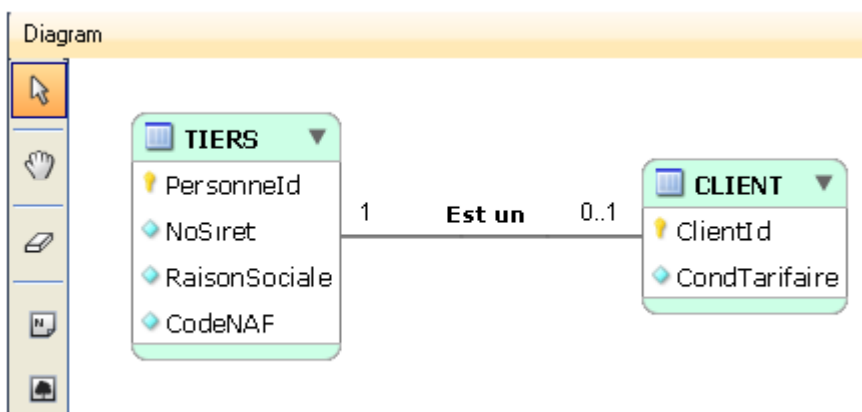


Figure 6.7 - Nom de rôle pour mettre en évidence une spécialisation

## 7. Associations réflexives (nomenclatures, hiérarchies)

### 7.1. Nomenclatures

Reprenons un exemple proposé par Donald Chamberlin (co-inventeur de SQL avec Raymond Boyce) et qui figure dans son article « Recursion in SQL: Tips and Techniques », paru en mai 1996 dans *Database Programming and Design*. Cet exemple concerne plus précisément une nomenclature des pièces qui entrent dans la composition des ailes d'avion.

Les tables sont les suivantes :

PIECE		
	Pieceld	PieceNom
+	aile	Aile d'avion
+	aileron	Aileron d'aile
+	charnière	Charnière à tout faire
+	longeron	Longeron d'aile
+	rivet	rivets à la pelle
+	train	Train de machin

COMPOSITION			
	ComposantId	Composeld	Quantite
	aile	aileron	1
	aile	longeron	5
	aile	rivet	100
	aile	train	1
	aileron	charnière	2
	aileron	rivet	5
	charnière	rivet	4
	longeron	rivet	10
	train	charnière	3
	train	rivet	8

Figure 7.1 - Nomenclature, contenu des tables

Graphe correspondant (une aile est composée de 5 longerons, 1 aileron, 1 train ; etc.) :

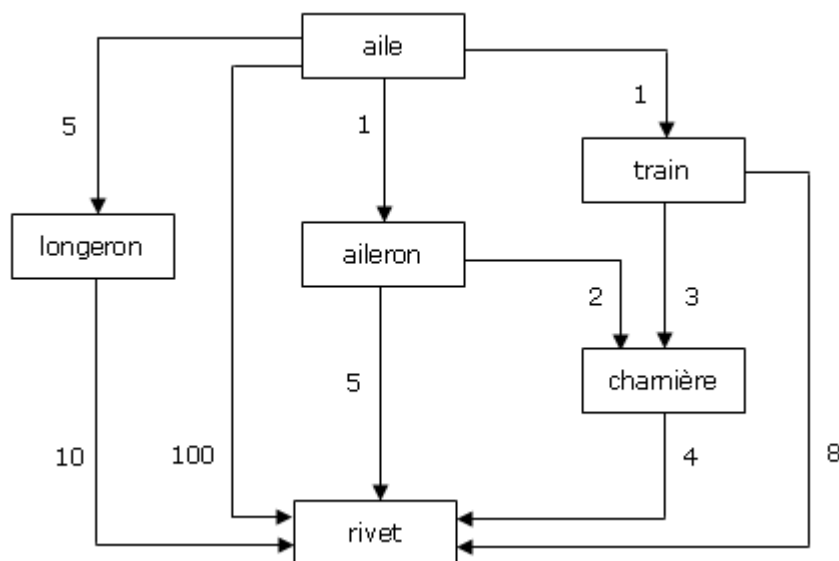


Figure 7.2 - Nomenclature, graphe

Une pièce peut entrer dans la composition de plusieurs pièces (sauf d'elle-même...) et une pièce peut être composée de plusieurs autres pièces. La modélisation avec MySQL Workbench ne pose pas de problème particulier, on met en œuvre deux liens identifiants (icône « 1:n Identifying Relationship ») et on remplace la cardinalité 1,N par 0,N tant pour les composants (une aile n'entre dans la composition d'aucune pièce) que pour les composés (un rivet n'est composé d'aucune pièce) :

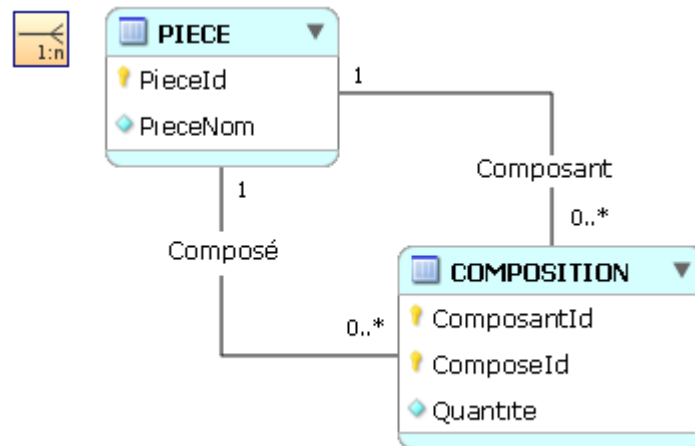


Figure 7.3 - Nomenclature (MLD)

Pour afficher le nom des rôles, se reporter au paragraphe 10.1110.11. Pour la petite histoire, la représentation ci-dessus correspond à la dérivation du MCD merisien suivant :

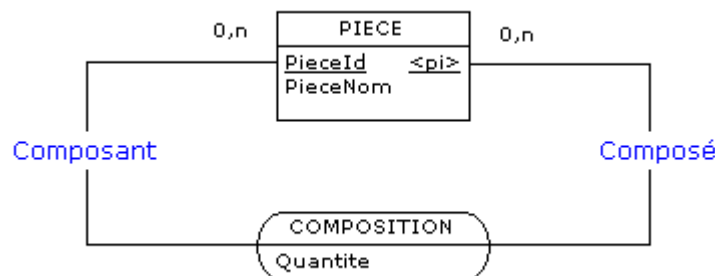


Figure 7.4 - Nomenclature (MCD Merise)

Ou à la dérivation du diagramme de classes UML :

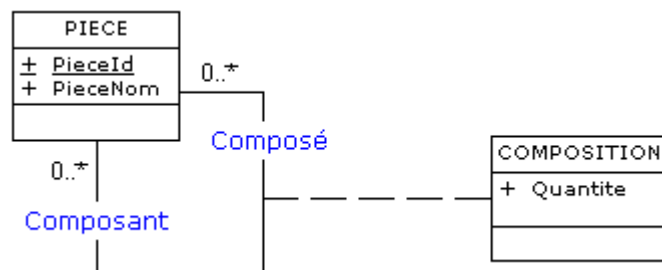


Figure 7.5 - Nomenclature (diagramme de classes)

## 7.2. Hiérarchies

Prenons le cas de la hiérarchie des personnes de l'entreprise Dubicobit : une personne peut être encadrée par une personne au plus (le président n'est encadré par personne, *sic* !) et une personne peut encadrer plusieurs personnes. La modélisation conceptuelle est la suivante :

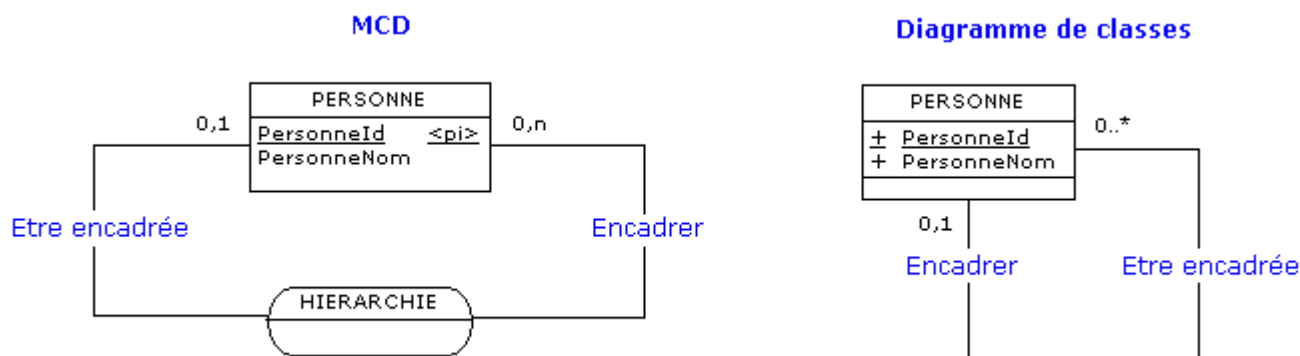


Figure 7.6 - Hiérarchie - MCD, DC

Cette modélisation conceptuelle donne habituellement lieu au diagramme MWB suivant, caractéristique de l'auto-référence :

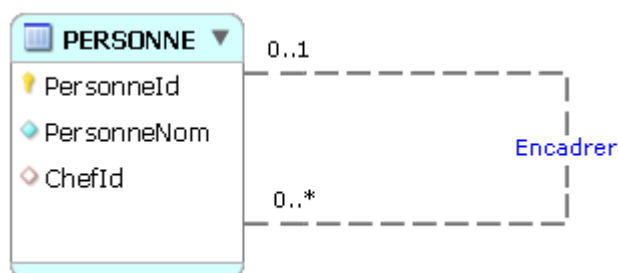


Figure 7.7 - Auto-référence avec MySQL Workbench

Mais comme cette façon de procéder permet au bonhomme Null d'infecter la base de données, il est préférable de représenter ainsi la hiérarchie :

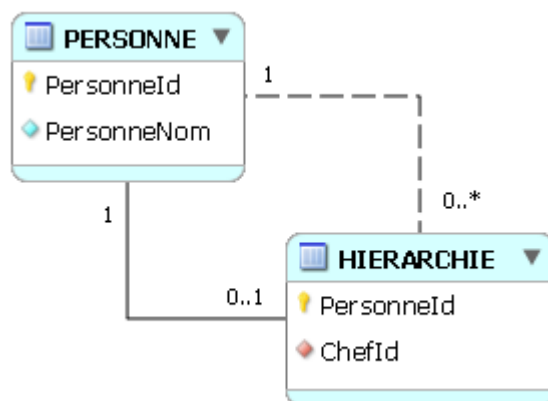


Figure 7.8 - Nomenclature sans bonhomme Null



Pour ceux qui utilisent la notation « Connect to columns », c'est-à-dire « à la ACCESS », il est intéressant de comparer la représentation d'une hiérarchie selon ACCESS et MySQL Workbench.

Représentation selon MySQL Workbench :



Figure 7.9 - Hiérarchie selon MySQL Workbench (dans le style ACCESS)

Représentation selon ACCESS :

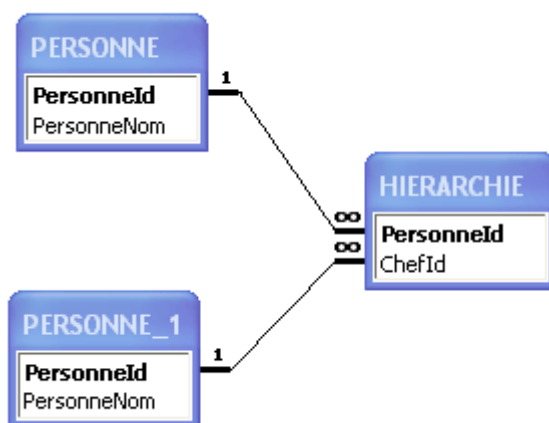


Figure 7.10 - Hiérarchie selon ACCESS (natif)

## 8. Génération du code SQL

### 8.1. Commande « Export »

Produire le code SQL de création des tables SQL (CREATE TABLE) ne pose pas de problème particulier. Dans un 1er temps on utilise à cet effet la commande « Export » et l'on choisit l'option « Forward Engineer SQL CREATE SCRIPT » :

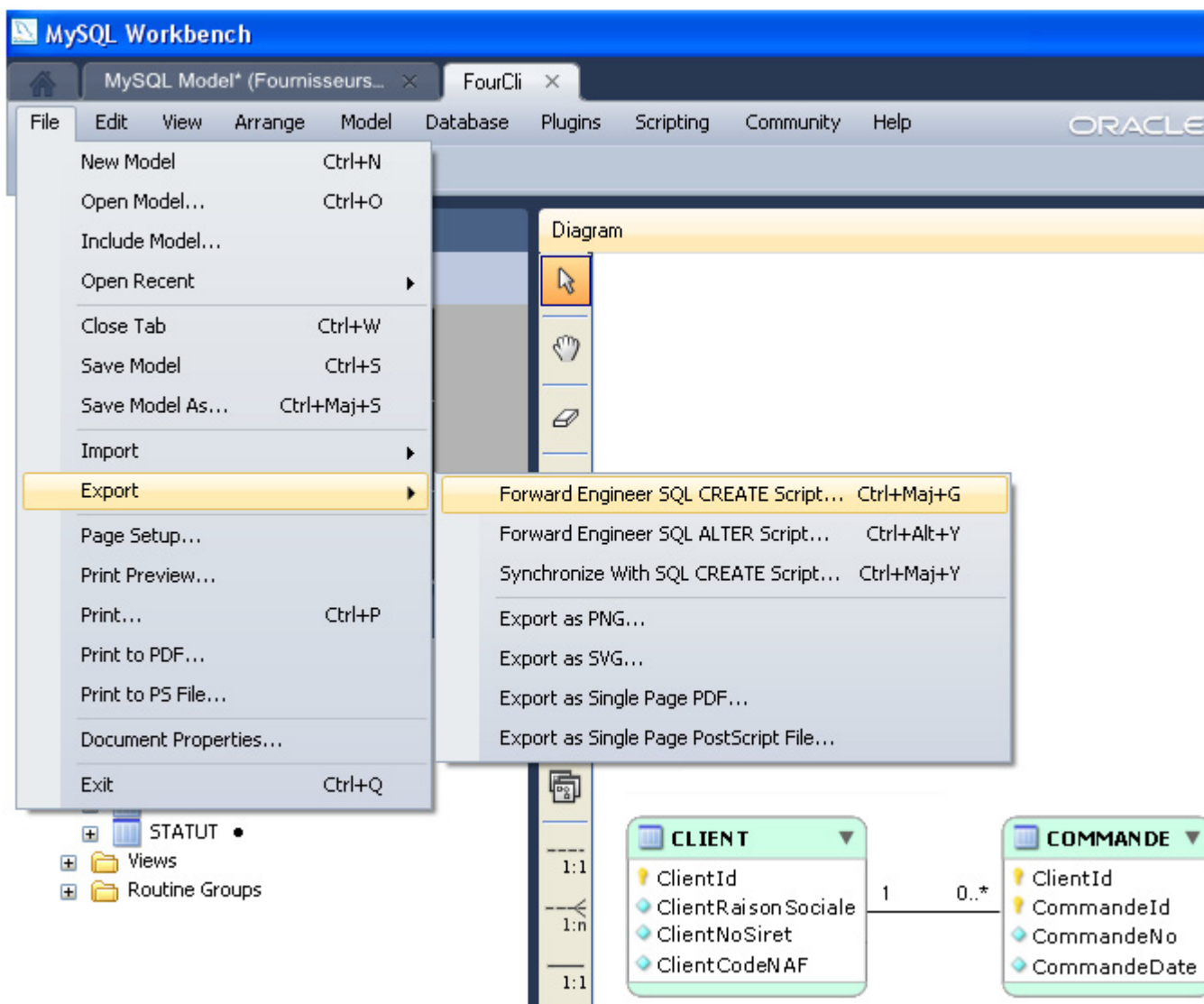


Figure 8.1 - Production du code SQL de création des tables, commande « Export »

## 8.2. Définition du nom du fichier contenant le code SQL

Dans un 2e temps, on définit le nom du fichier qui contiendra le code SQL :

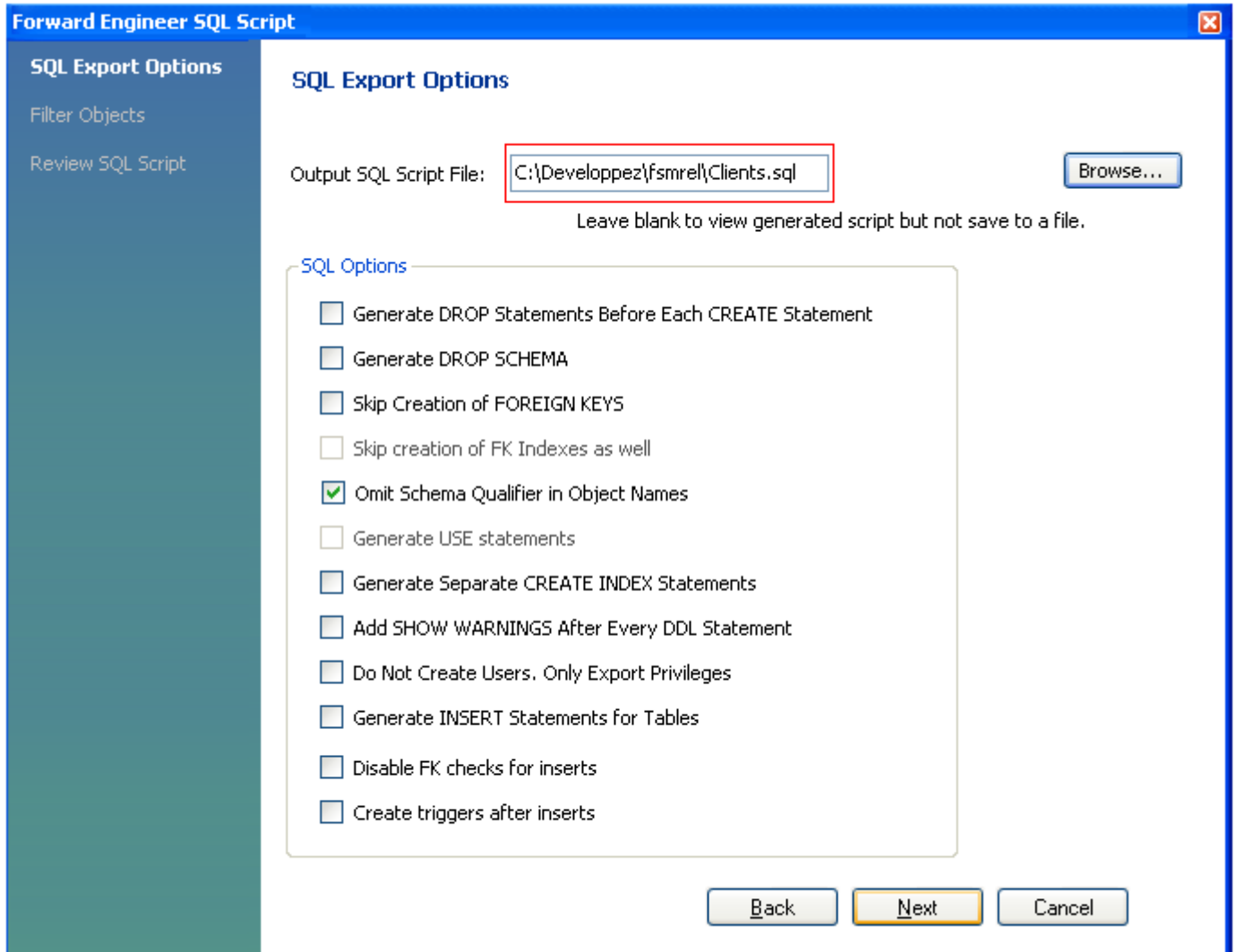


Figure 8.2 - Définition du fichier destiné à contenir le code SQL

### 8.3. Types d'objets à exporter

On ne demande ici que la génération du code de création des tables (CREATE TABLE) :

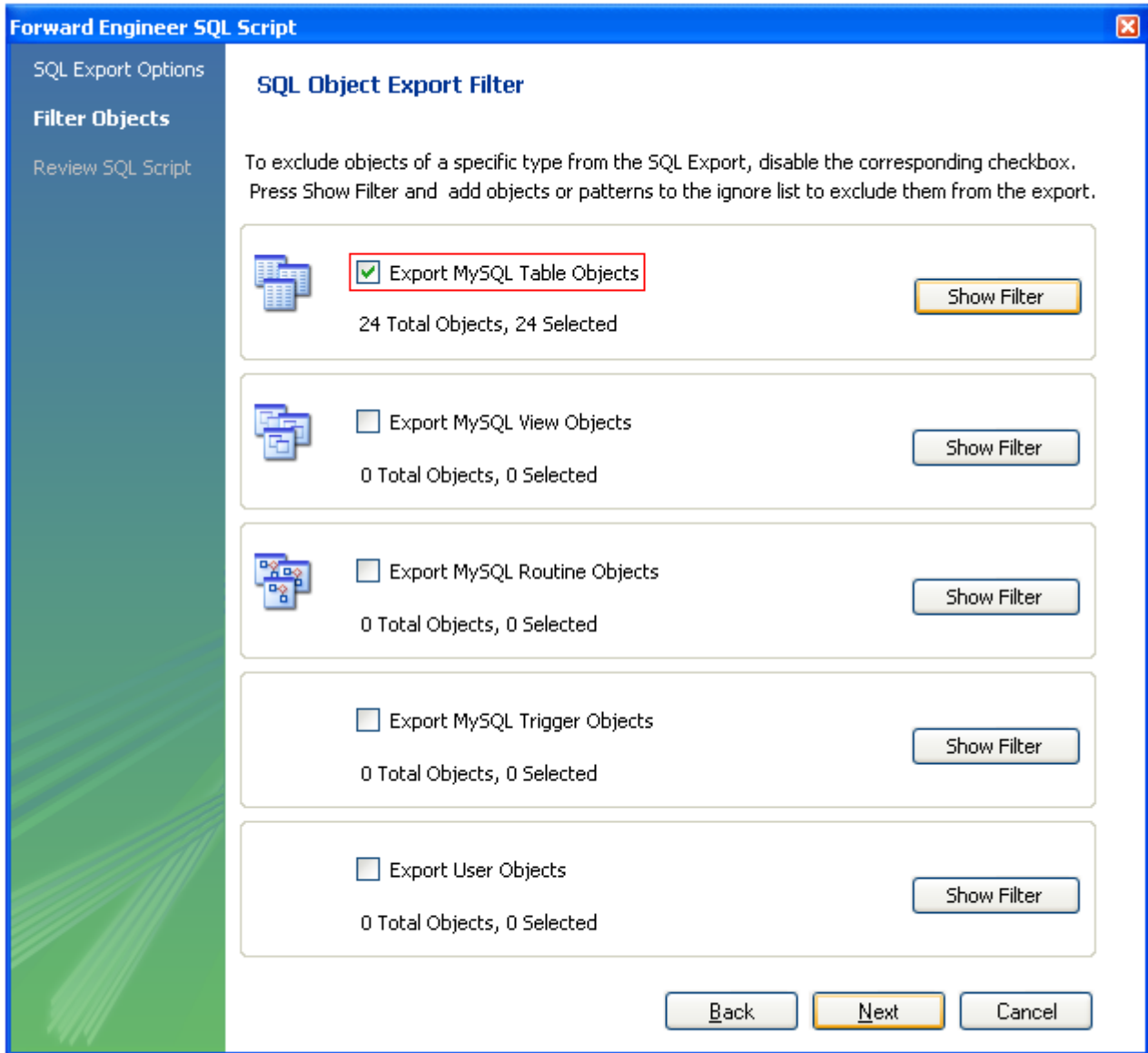


Figure 8.3 - On ne s'intéresse ici qu'aux tables

## 8.4. Choix des tables à faire figurer dans le code SQL

Pour produire un script SQL ne faisant mention que d'un sous-ensemble de noms de tables, par exemple ceux des tables CLIENT et COMMANDE, au moyen du bouton « >> » on transfère l'ensemble des noms des tables depuis le cartouche de gauche à destination de celui de droite, puis on rapatrie les noms des deux tables CLIENT et COMMANDE dans le cartouche de gauche (bouton « << ») :

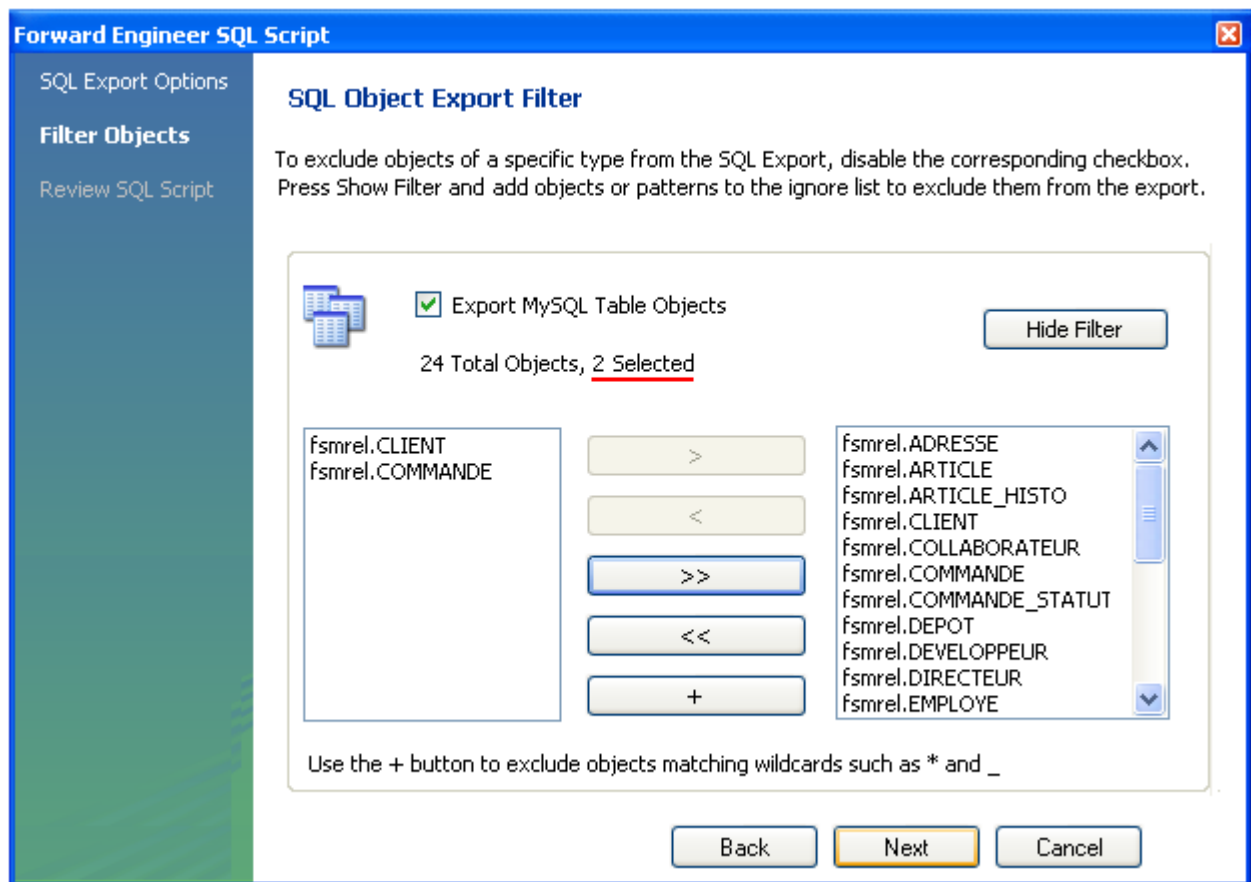


Figure 8.4 - Sélection d'un sous-ensemble de tables

## 8.5. Affichage du résultat par MySQL Workbench

Au résultat :

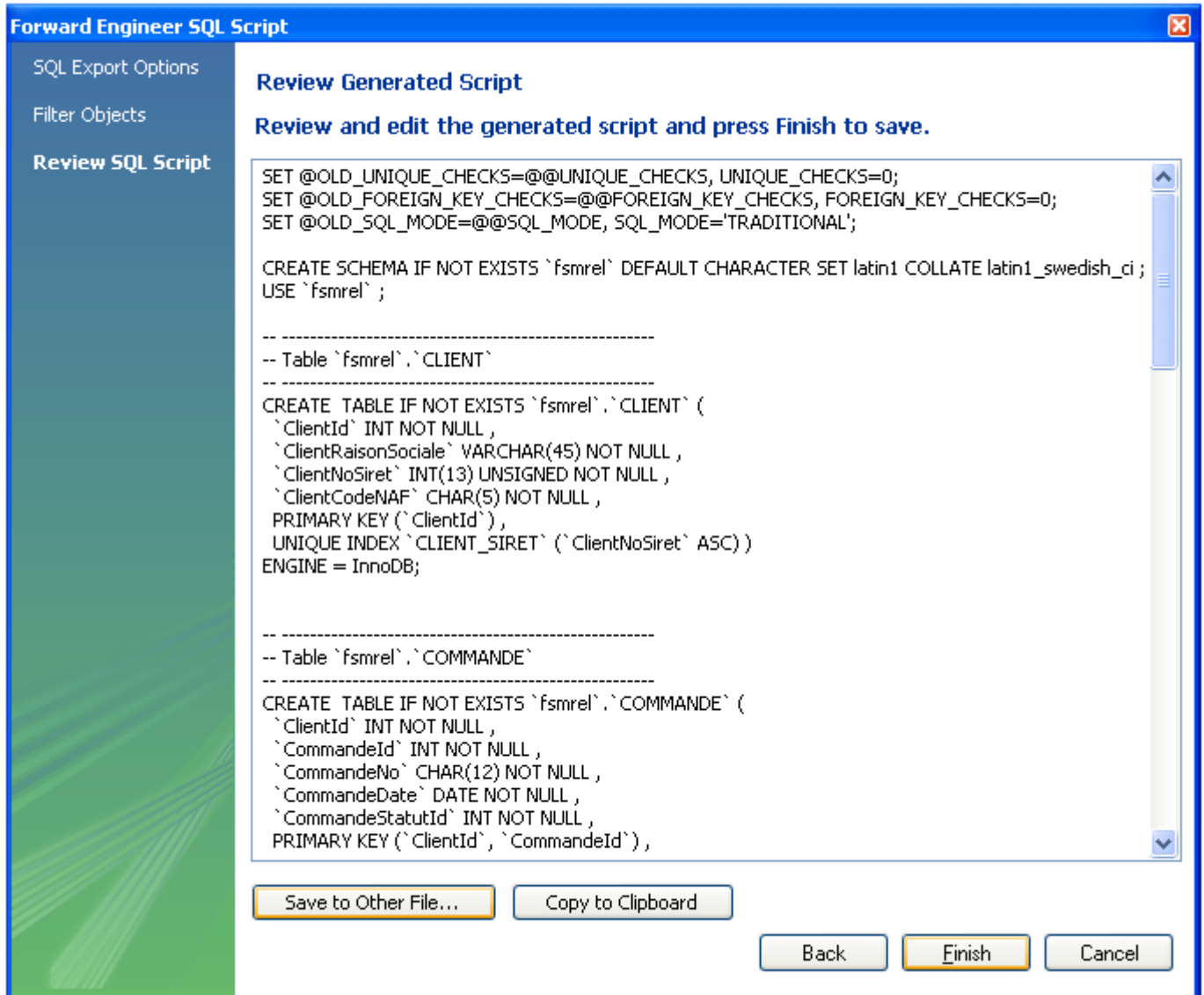


Figure 8.5 - Affichage du résultat

## 8.6. Enregistrement du fichier

Enregistrer le fichier en l'état :

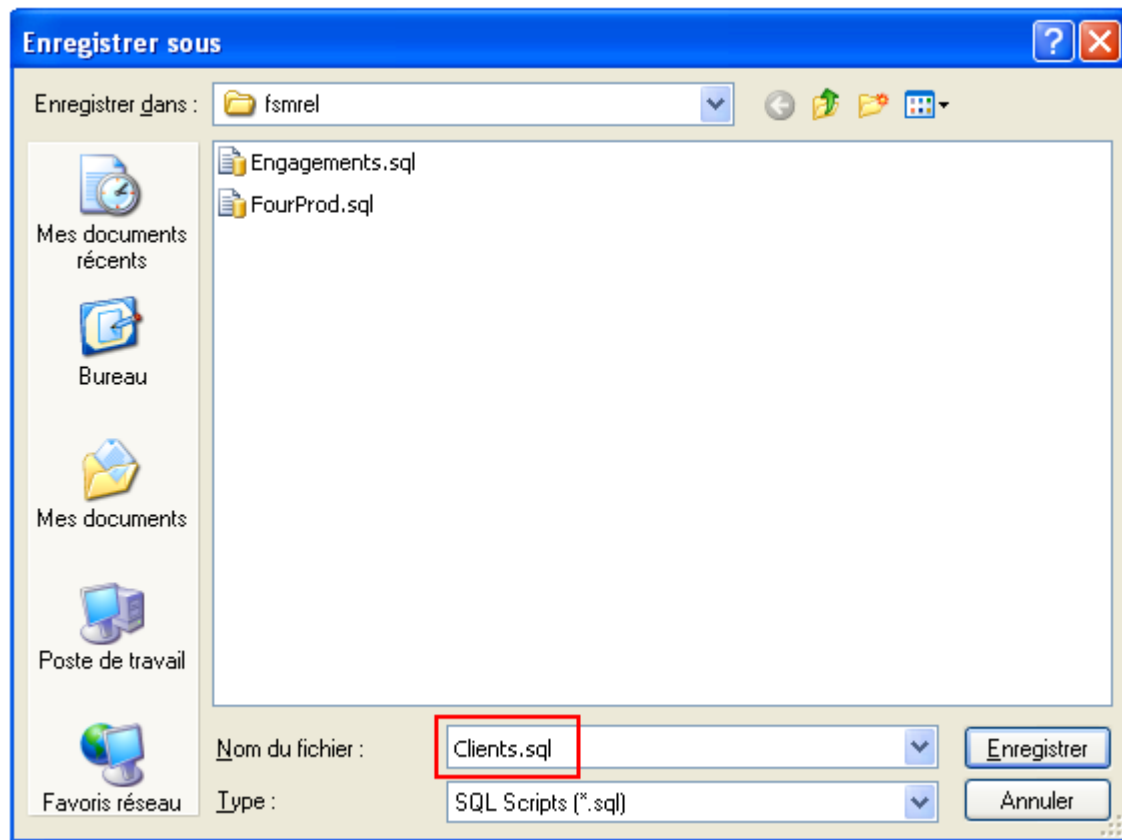


Figure 8.6 - Enregistrement du résultat

Si on utilise un autre SGBD que MySQL, il ne reste plus qu'à nettoyer le fichier qui vient d'être créé, par exemple supprimer toutes les occurrences du caractère « ` », supprimer les instructions propres à MySQL et celles auxquelles on ne prête pas d'intérêt, remplacer les clauses UNIQUE INDEX par UNIQUE (conformément à la norme SQL), supprimer toutes les clauses index restantes (la mise en oeuvre des index est du ressort de l'étape MPD (modèle physique des données)), regarder si les colonnes sont dans le bon ordre dans les clés (primaires alternatives, étrangères), supprimer les « IF NOT EXISTS », les « ENGINE = InnoDB » et autres éléments propres à MySQL, etc.

Le nettoyage du fichier est quelque chose de rapide, pour celui qui vient d'être créé une paire de minutes suffit (à condition d'avoir déjà codé des instructions CREATE TABLE quand même...)

## 9. Rétro-conception

### 9.1. Objet de la rétro-conception

La rétro-conception permet de produire un diagramme MWB à partir du code SQL (CREATE TABLE) ou encore — ce qui est propre à MySQL Workbench — produire un tel diagramme à partir d'un diagramme DBDesigner 4.

### 9.2. A partir du code SQL

Après avoir créé un modèle vierge (commande « File > New Model », cf. paragraphe 2.2.1), on importe le fichier contenant le code SQL utilisé pour la rétro-conception (le format .txt ou équivalent est parfait pour cela) :

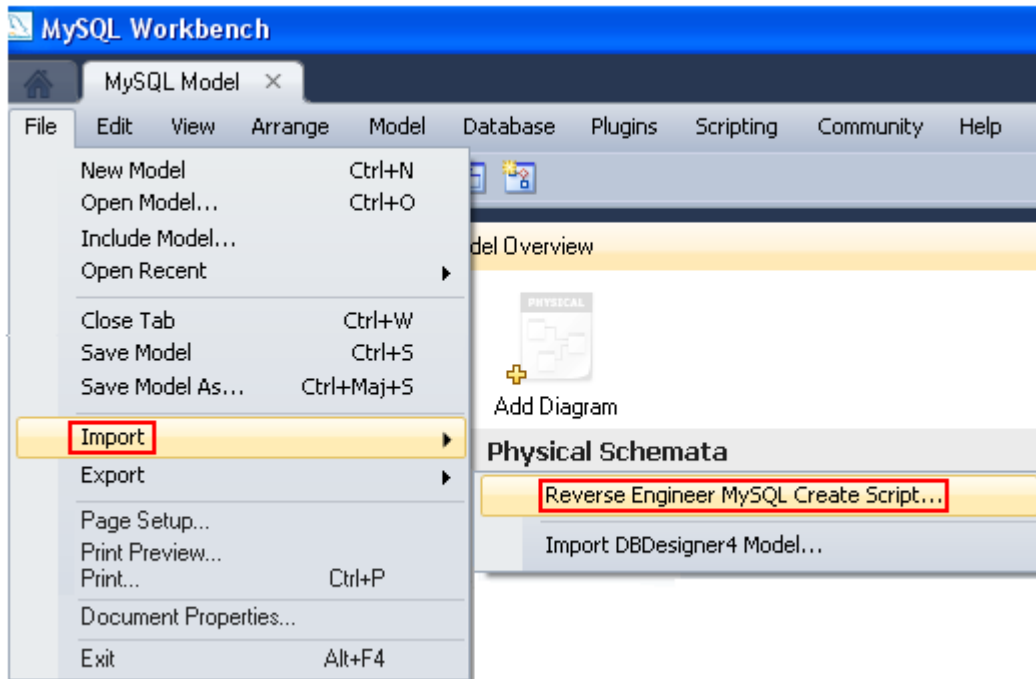


Figure 9.1 - Rétro-conception à partir du code SQL (CREATE TABLE)

Suite à l'ouverture de la fenêtre « Reverse Engineer SQL Script », en y cochant la case « Place imported objects on a diagram », on demande que les objets soient directement placés dans le diagramme. (On peut aussi se contenter de leur inscription au catalogue, et plus tard utiliser la commande « Model > Create Diagram from Catalog Objects » pour qu'ils apparaissent dans le diagramme) :

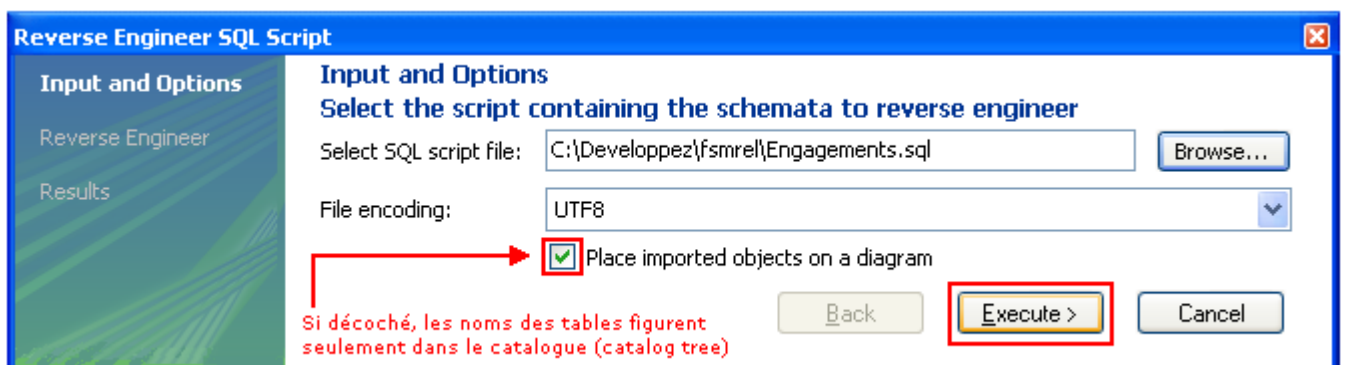


Figure 9.2 - Rétro-conception, choix du fichier source



On peut avoir droit à quelques remarques :

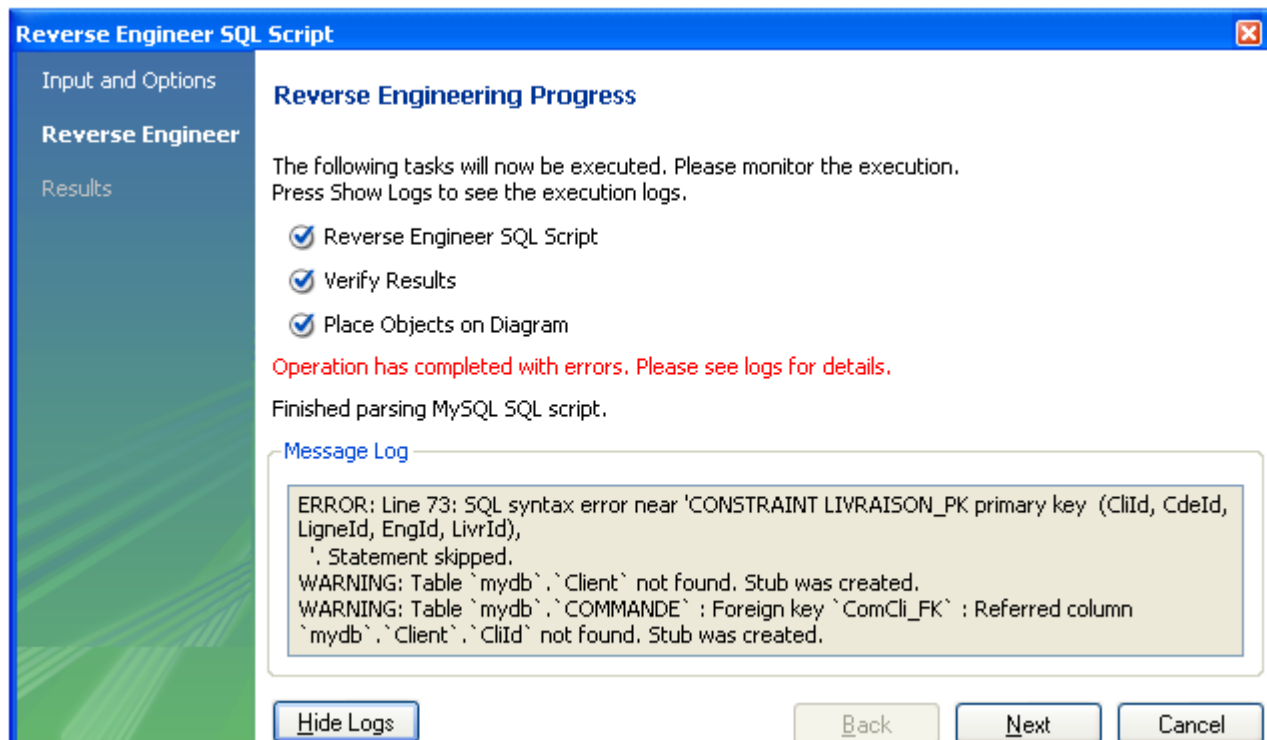


Figure 9.3 - Rétro-conception, détection des erreurs par MySQL Workbench

Une fois corrigées les erreurs, MySQL Workbench prend acte :

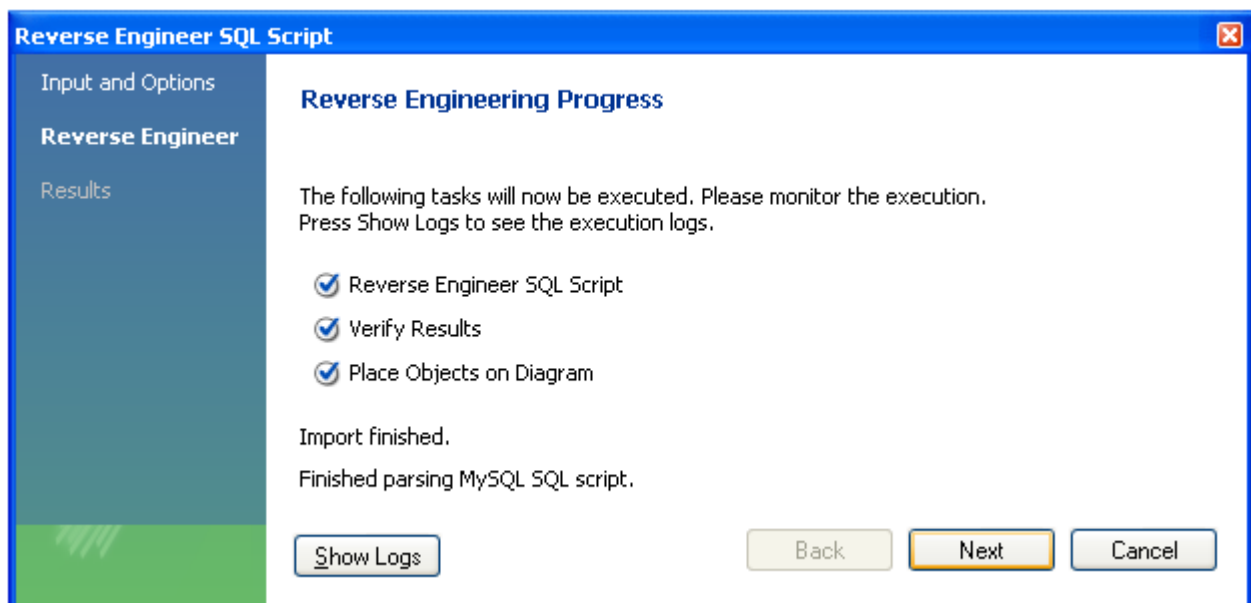


Figure 9.4 - Rétro-conception, aucune erreur détectée

Et comme dans la chanson, tout finit par s'arranger...

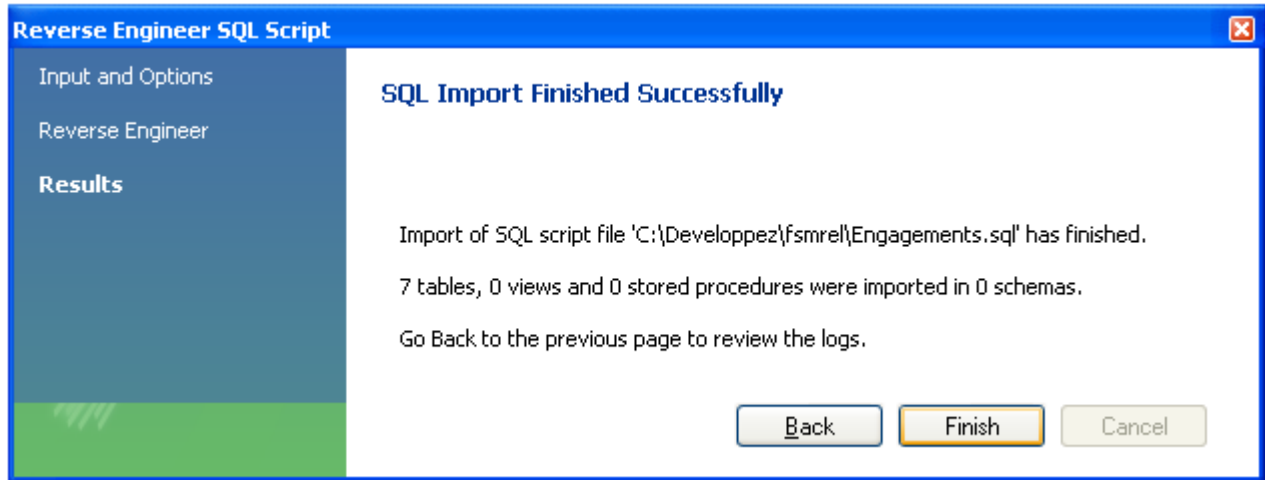


Figure 9.5 - Rétro-conception, c'est tout bon

On a droit à un gros pâté qu'il restera à mettre en forme en déplaçant judicieusement les objets affichés :

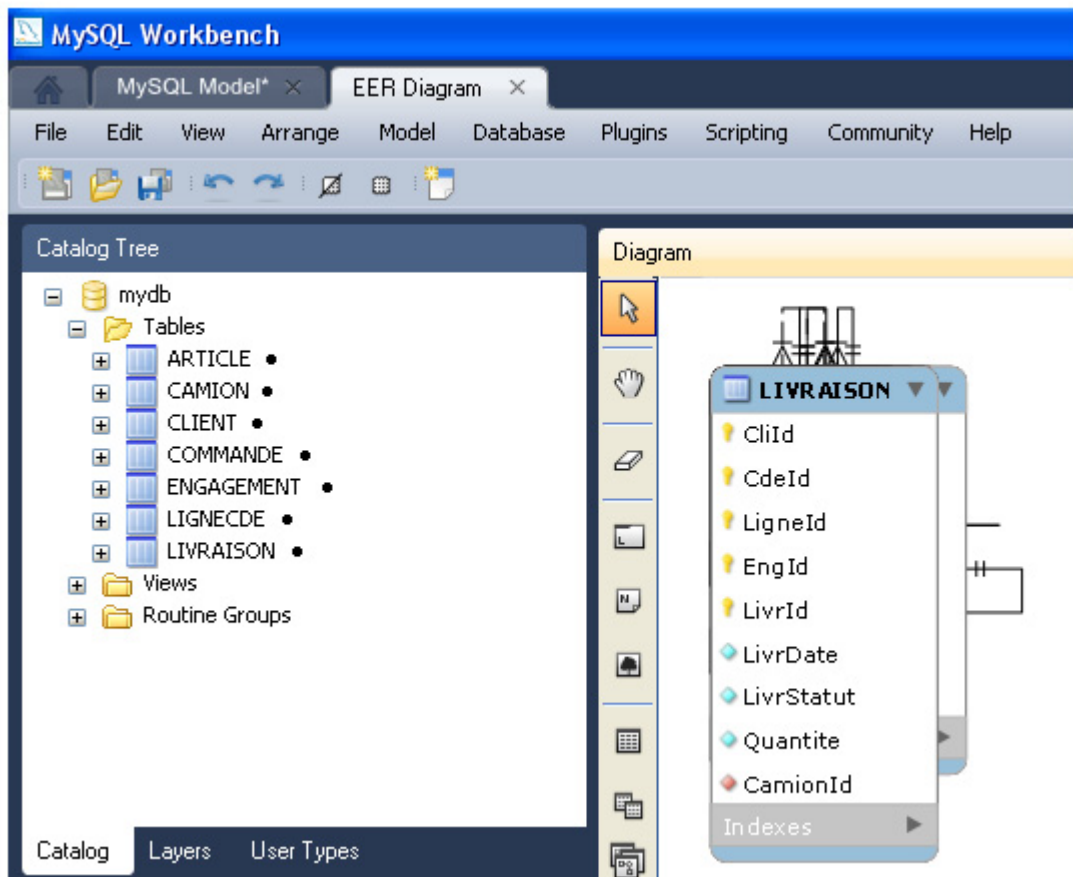


Figure 9.6 - Rétro-conception, les tables importées

Suite aux déplacements judicieux, le diagramme prend forme :

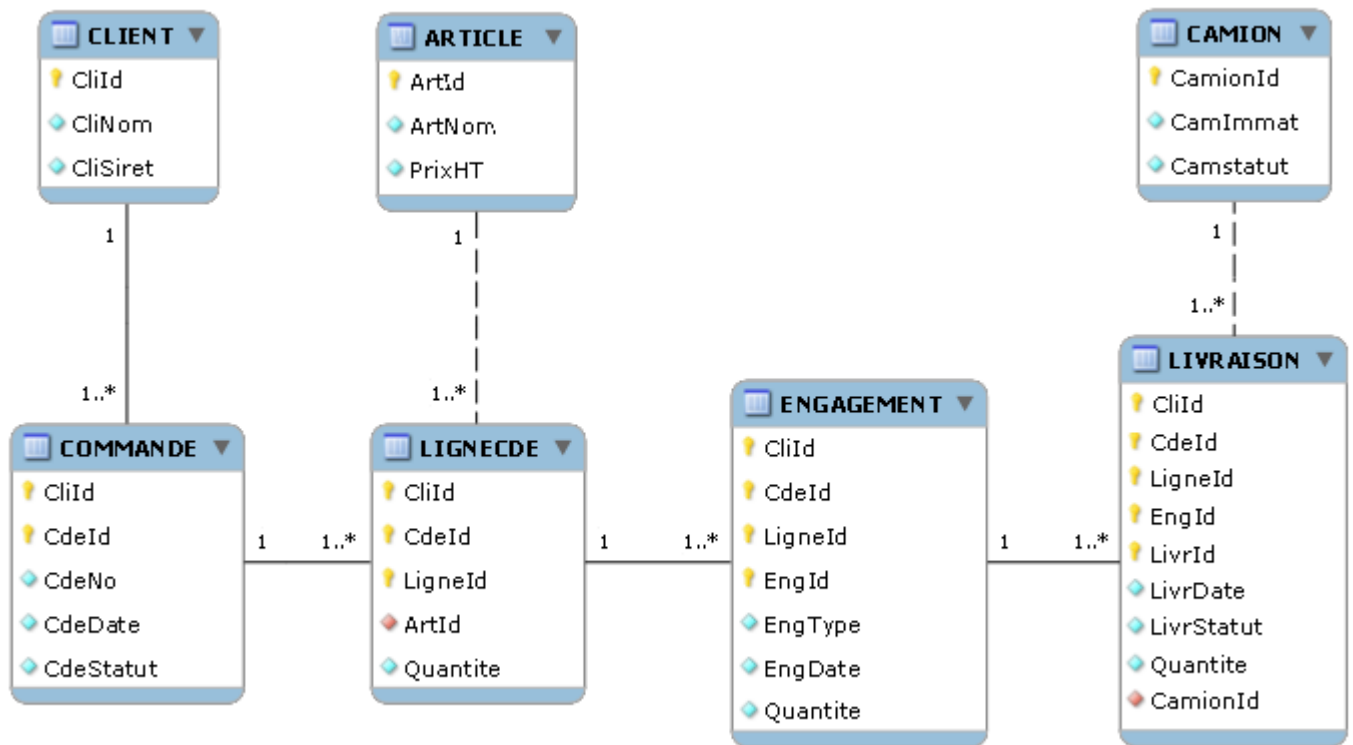


Figure 9.7 - Rétro-conception, fin

Il reste à remplacer certaines cardinalités minimales 1 par 0 si cela est nécessaire (par exemple, certains articles peuvent ne pas avoir été commandés).

Le code SQL source ayant servi pour la rétro-conception :

```

CREATE TABLE CLIENT
(
    CliId          INT          NOT NULL
    , CliNom       VARCHAR(48)  NOT NULL
    , CliSiret     CHAR(14)     NOT NULL
    , CONSTRAINT CLIENT_PK PRIMARY KEY (CliId)
    , CONSTRAINT CLIENT_AK1 UNIQUE (CliSiret)
) ;
CREATE TABLE ARTICLE
(
    ArtId          INT          NOT NULL
    , ArtNom       VARCHAR(48)  NOT NULL
    , PrixHT       DECIMAL(16)  NOT NULL
    , CONSTRAINT ARTICLE_PK PRIMARY KEY (ArtId)
) ;
CREATE TABLE CAMION
(
    CamionId       INT          NOT NULL
    , CamImmat     VARCHAR(14)  NOT NULL
    , Camstatut    CHAR(2)     NOT NULL
    , CONSTRAINT CAMION_PK PRIMARY KEY (CamionId)
    , CONSTRAINT CAMION_AK1 UNIQUE (CamImmat)
) ;
    
```

```

CREATE TABLE COMMANDE
(
  CliId          INT          NOT NULL
, CdeId          SMALLINT     NOT NULL
, CdeNo          VARCHAR(12)   NOT NULL
, CdeDate        CHAR(10)      NOT NULL
, CdeStatut      CHAR(2)       NOT NULL
, CONSTRAINT COMMANDE_PK PRIMARY KEY (CliId, CdeId)
, CONSTRAINT COMMANDE_AK1 UNIQUE (CdeNo)
, CONSTRAINT COMMANDE_CLIENT_FK FOREIGN KEY (CliId)
  REFERENCES CLIENT (CliId) ON DELETE CASCADE
) ;

CREATE TABLE LIGNECDE
(
  CliId          INT          NOT NULL
, CdeId          SMALLINT     NOT NULL
, LigneId        SMALLINT     NOT NULL
, ArtId          INT          NOT NULL
, Quantite       INT          NOT NULL
, CONSTRAINT LIGNECDE_PK PRIMARY KEY (CliId, CdeId, LigneId)
, CONSTRAINT LIGNECDE_CDE_FK FOREIGN KEY (CliId, CdeId)
  REFERENCES COMMANDE (CliId, CdeId) ON DELETE CASCADE
, CONSTRAINT LIGNECDE_ART_FK FOREIGN KEY (ArtId)
  REFERENCES ARTICLE (ArtId)
) ;

CREATE TABLE ENGAGEMENT
(
  CliId          INT          NOT NULL
, CdeId          SMALLINT     NOT NULL
, LigneId        SMALLINT     NOT NULL
, EngId          SMALLINT     NOT NULL
, EngType        CHAR(2)       NOT NULL
, EngDate        CHAR(10)      NOT NULL
, Quantite       INT          NOT NULL
, CONSTRAINT ENGAGEMENT_PK PRIMARY KEY (CliId, CdeId, LigneId, EngId)
, CONSTRAINT ENGAGEMENT_LIGNE_FK FOREIGN KEY (CliId, CdeId, LigneId)
  REFERENCES LIGNECDE (CliId, CdeId, LigneId) ON DELETE CASCADE
) ;

CREATE TABLE LIVRAISON
(
  CliId          INT          NOT NULL
, CdeId          SMALLINT     NOT NULL
, LigneId        SMALLINT     NOT NULL
, EngId          SMALLINT     NOT NULL
, LivrId         SMALLINT     NOT NULL
, LivrDate       CHAR(10)      NOT NULL
, LivrStatut     CHAR(2)       NOT NULL
, Quantite       INT          NOT NULL
, CamionId       INT          NOT NULL
, CONSTRAINT LIVRAISON_PK PRIMARY KEY (CliId, CdeId, LigneId, EngId, LivrId)
, CONSTRAINT LIVRAISON_ENG_FK FOREIGN KEY (CliId, CdeId, LigneId, EngId)
  REFERENCES ENGAGEMENT (CliId, CdeId, LigneId, EngId) ON DELETE CASCADE
, CONSTRAINT LIVRAISON_CAMION_FK FOREIGN KEY (CamionId)
  REFERENCES CAMION (CamionId)
) ;
  
```

L'outil a évidemment correctement interprété les clés étrangères, et il sait repérer les clés alternatives, par exemple la clé {CdeNo} figurant dans la contrainte COMMANDE\_AK1 :

```
CONSTRAINT COMMANDE_AK1 UNIQUE (CdeNo)
```

La clé alternative {CdeNo} est bien prise en compte (case « UQ » cochée) :




Table Name:

COMMANDE

Columns






Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
 CliId	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 CdeId	SMALLINT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 CdeNo	VARCHAR(12)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 CdeDate	CHAR(10)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 CdeStatut	CHAR(2)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Figure 9.8 - Rétro-conception, clé alternative en place

### 9.3. A partir de DBDesigner 4

Modèle DBDesigner 4 à rétro-concevoir (la hiérarchie des fournisseurs est discutable, mais bon...) :

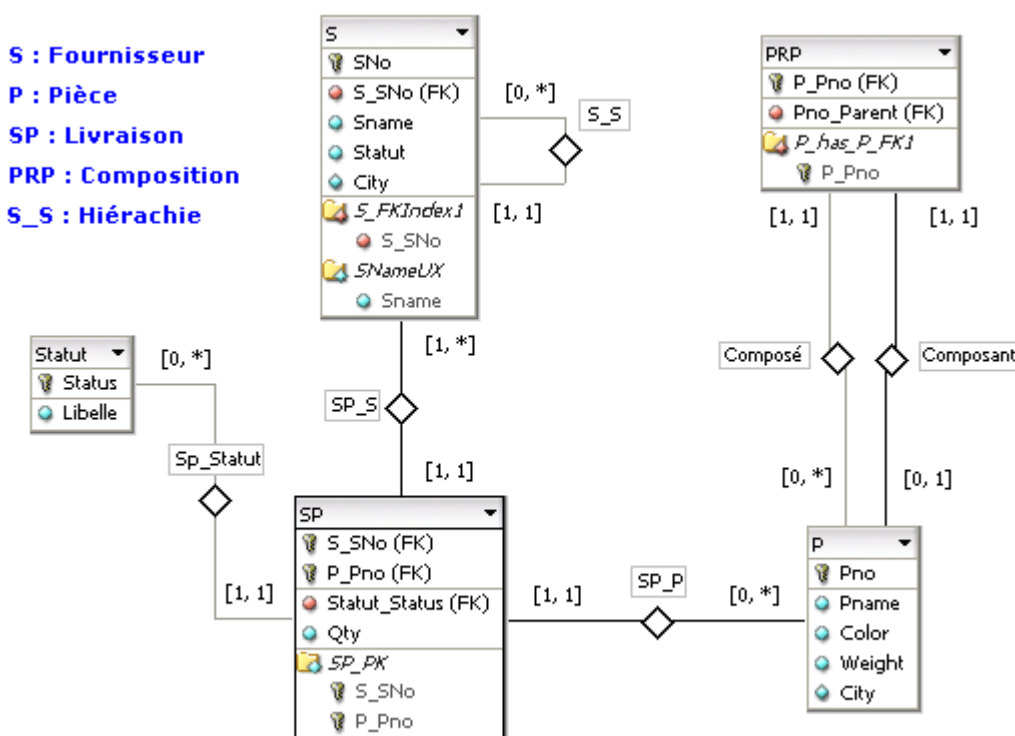


Figure 9.9 - Rétro-conception, modèle DBDesigner 4 source

Faire : « File > New Model », puis « Import » et « Import DBDesigner4 Model » :

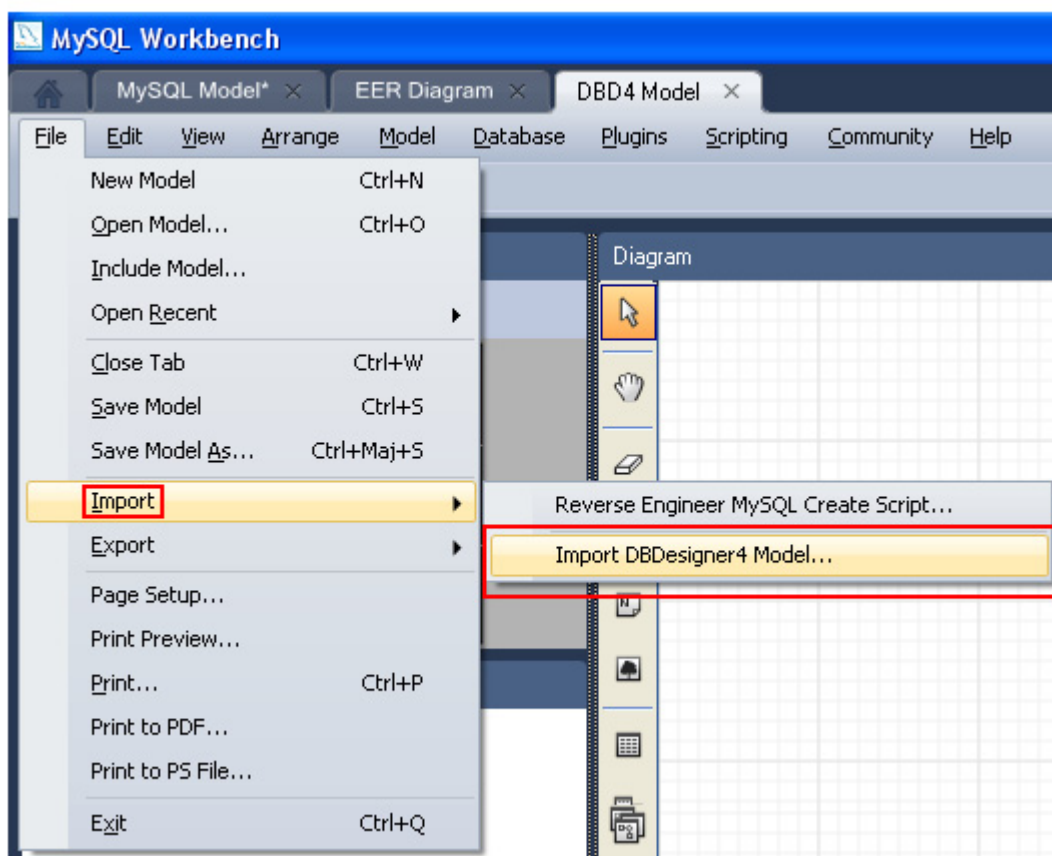


Figure 9.10 - Rétro-conception, import d'un modèle DBDesigner 4

Saisie du nom du fichier au format XML :

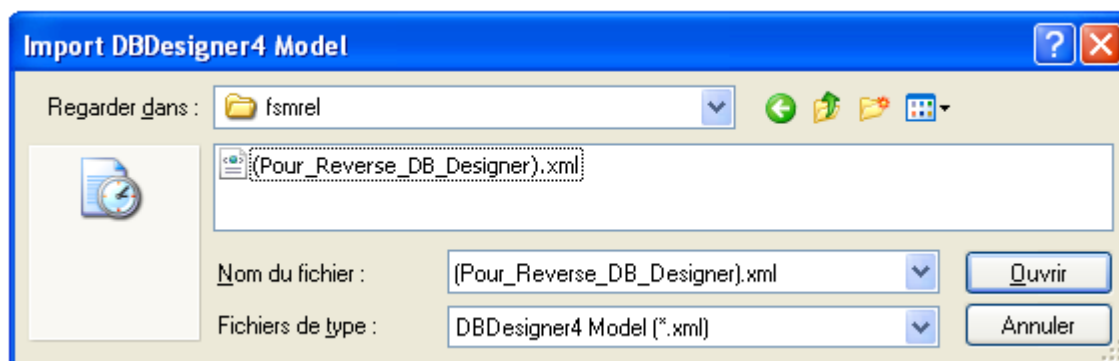


Figure 9.11 - Rétro-conception, import d'un modèle DBDesigner 4, nom du fichier XML

Au résultat :

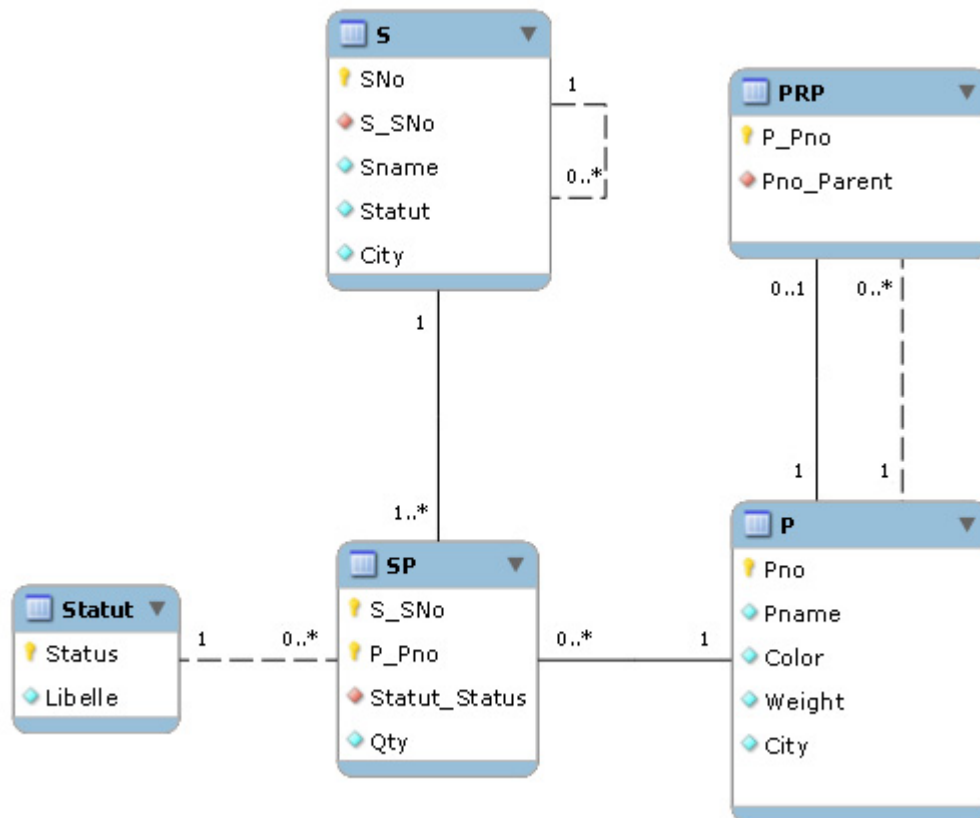


Figure 9.12 - Rétro-conception, modèle DBDesigner 4 converti

N.B. L'association réflexive portée par la table S peut être aménagée, comme dans le cas de la hiérarchie figurant au paragraphe 7.2.

## 10. Annexes

### 10.1. Afficher/cacher la grille

Menu : View > Toggle grid :

Un clic sur « Toggle Grid » et la grille est cachée. Même principe pour cacher les guides de page (Toggle Page Guides).

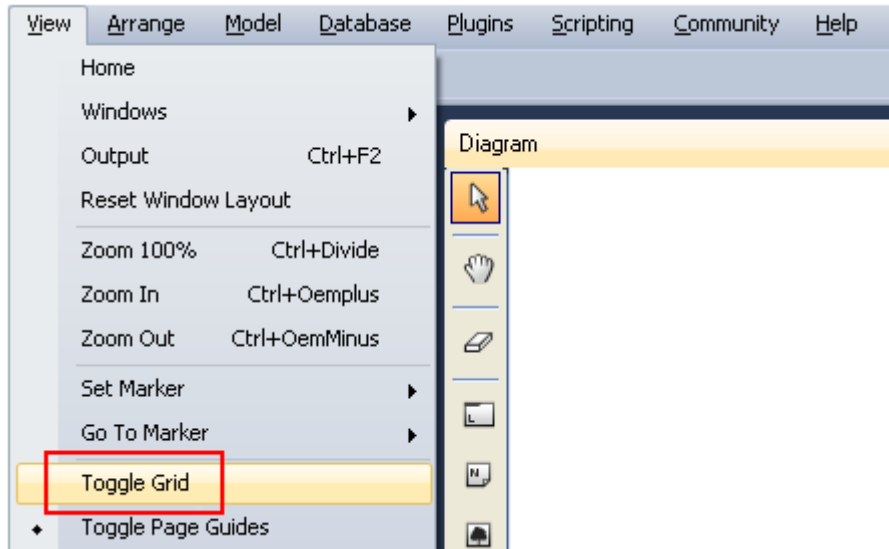


Figure 10.1 - Grille cachée

### 10.2. Changer la couleur d'un objet

On peut changer la couleur d'un objet, pour cela on commence par cliquer sur cet objet, ce qui provoque l'ouverture de la fenêtre « Description Editor » :

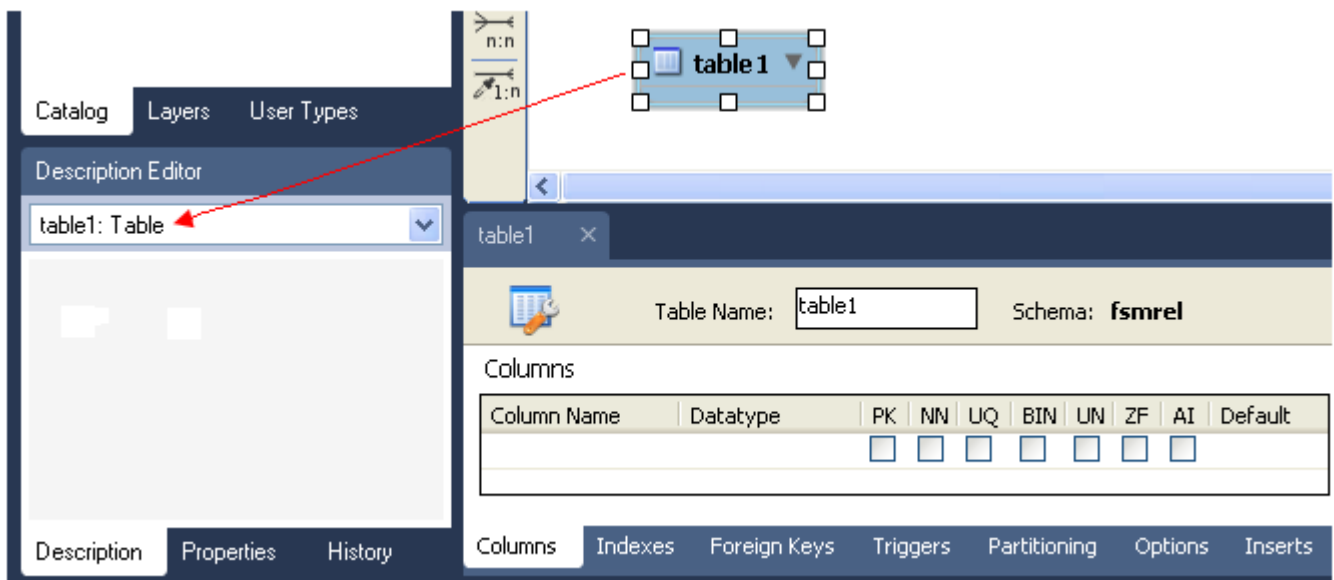


Figure 10.2 - Changer la couleur d'un objet



On clique ensuite sur l'onglet « Properties » de la fenêtre « Description Editor » :

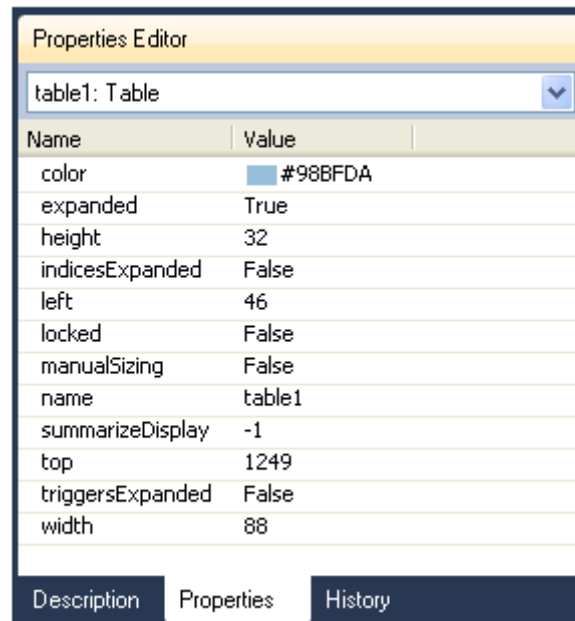
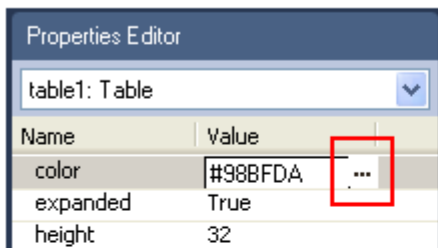


Figure 10.3 - Couleur actuelle

Puis on choisit la couleur qui va bien :

**Avant :**



**Après :**

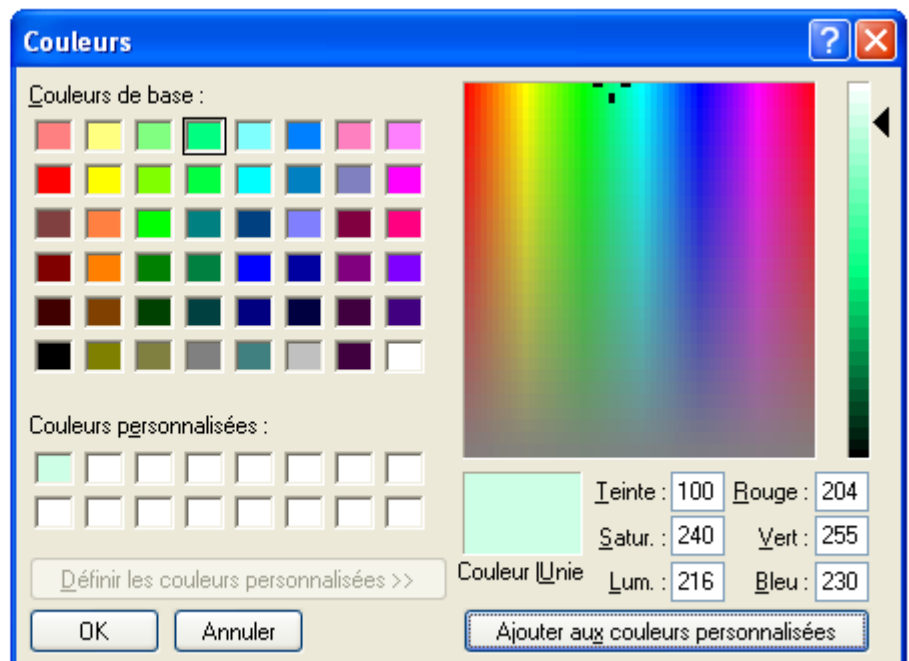
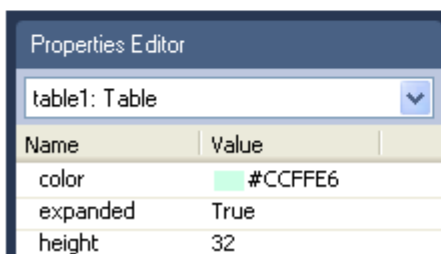


Figure 10.4 - Palette de couleurs

Au résultat :



Figure 10.5 - Nouvelle couleur

### 10.3. Réduire la taille des objets à l'affichage

Si l'on manque de place pour afficher les objets, on peut en réduire la taille avec les possibilités de zoom de Bird's Eye :

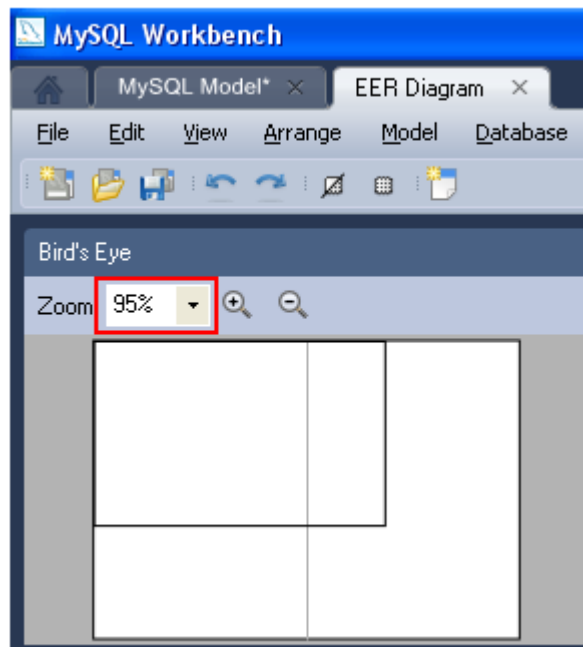


Figure 10.6 - Bird's eye

## 10.4. Noms et types par défaut

### 10.4.1. MySQL Workbench et les éléments par défaut

MySQL Workbench utilise des noms par défaut pour les objets suivants : colonnes des tables, type des colonnes, tables associatives (cf. par exemple le paragraphe 4.3), clés étrangères. Pour modifier ces noms : menu « Edit » puis « Preferences » :

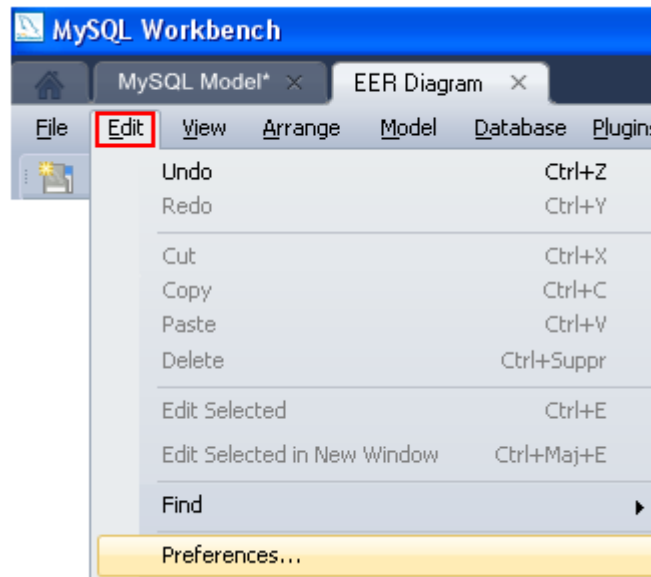


Figure 10.7 - Noms par défaut, préférences

On ouvre ensuite l'onglet « Model » de la fenêtre « Workbench Preferences » (les valeurs par défaut ci-dessous ne sont pas forcément celles qui étaient en vigueur lors de l'installation du produit) :

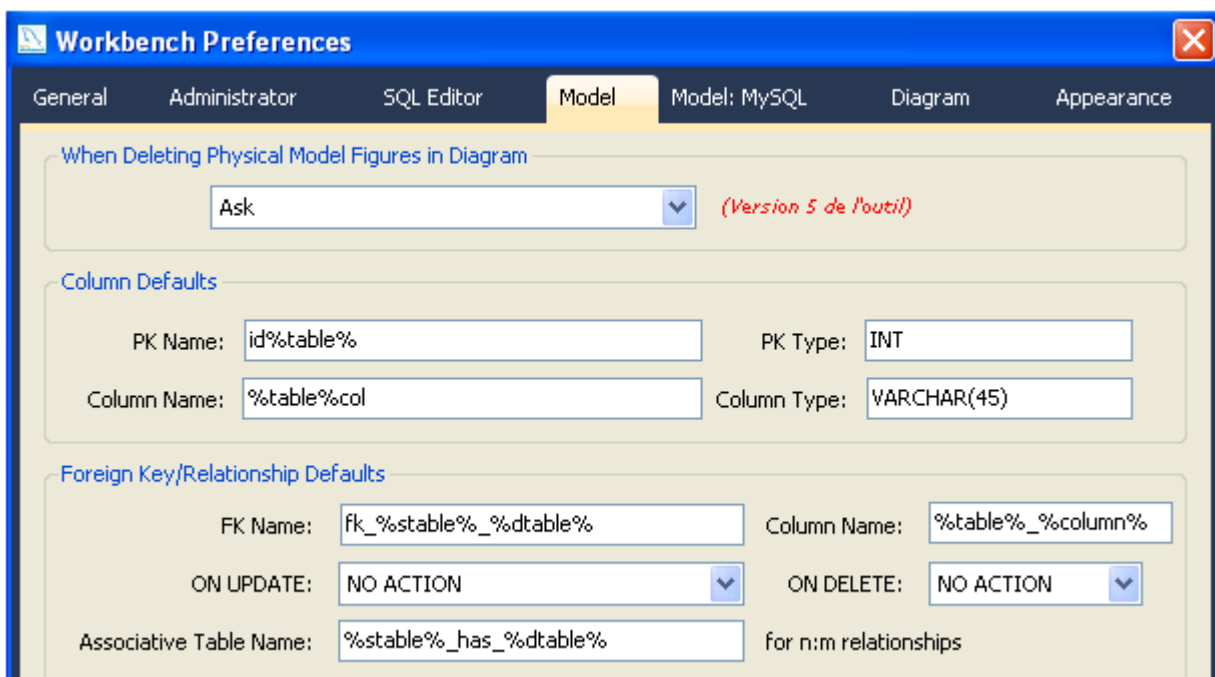


Figure 10.8 - Fenêtre Preferences, onglet « Model »

### 10.4.2. Nom par défaut de la 1re colonne d'une table

Le nom par défaut de la 1re colonne d'une table est constitué du nom de la table en minuscules, préfixé par « id ». Le type de cette colonne est INT. Par défaut, cette colonne est le seul élément de la clé primaire de la table. Tout ceci ne vaut que si la table n'est pas identifiée relativement une autre table (auquel cas MySQL Workbench recopie le nom de la (des) colonne(s) composant la clé primaire de cette dernière) :

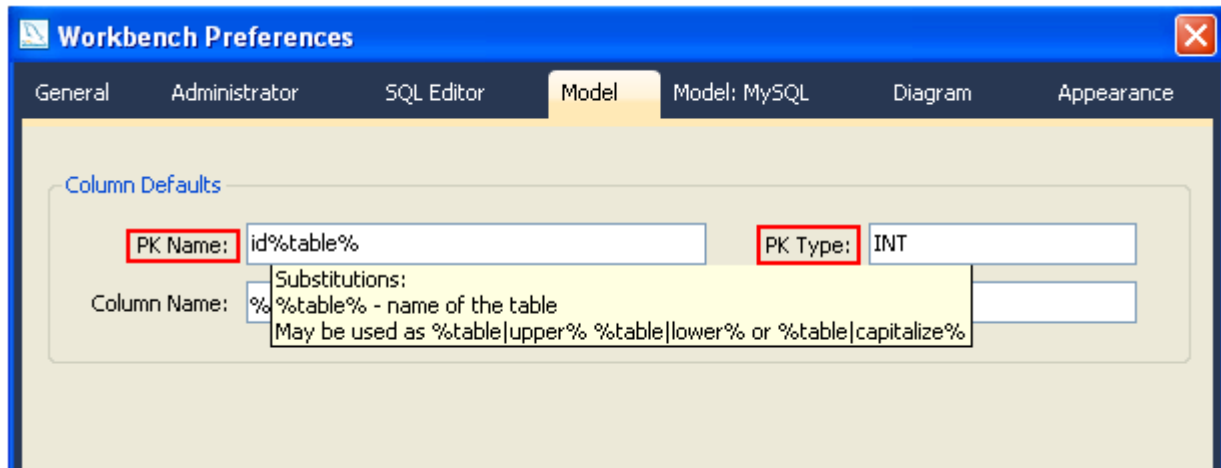


Figure 10.9 - Nom par défaut de la 1re colonne d'une table

Une infobulle permet de voir que MySQL Workbench propose de mettre le nom par défaut de la table en lettres majuscules, minuscules, capitales.

### 10.4.3. Nom et type par défaut d'une colonne banale

Pour les colonnes qui ne participent pas à la clé primaire, on voit ci-dessous que le nom par défaut est celui de la table, suffixé par « col ». Afin d'éviter les doublons, l'outil remplace en fait « col » par « col1 », « col2 », etc. Comme autre nom possible par défaut, MySQL Workbench propose le minimum syndical, à savoir seulement le nom de la table, ce qui du reste n'est pas plus mal :

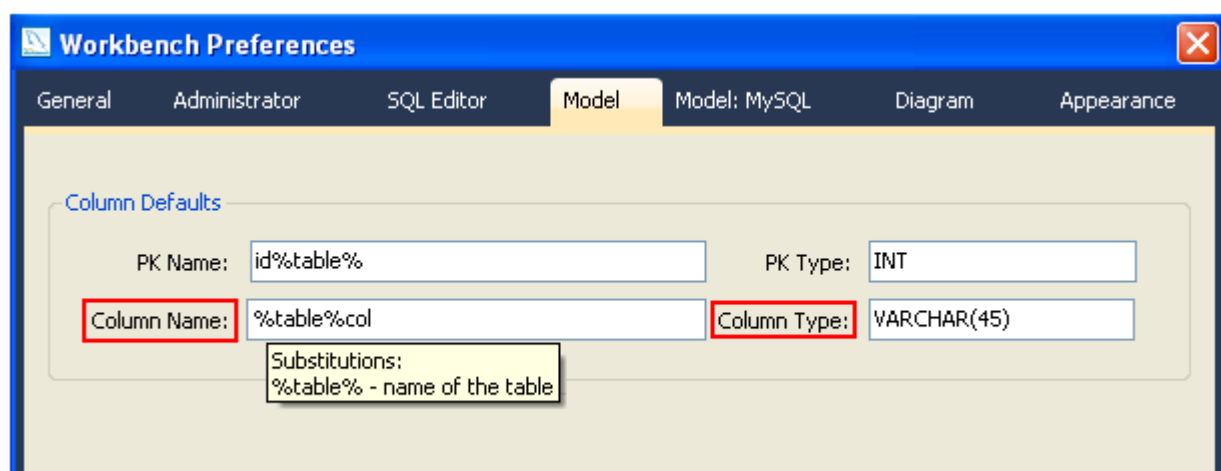


Figure 10.10 - Nom des autres colonnes

#### 10.4.4. Types par défaut

Outre le nom par défaut des colonnes, on peut en définir le type par défaut, par exemple CHAR(8) au lieu de VARCHAR(45) :

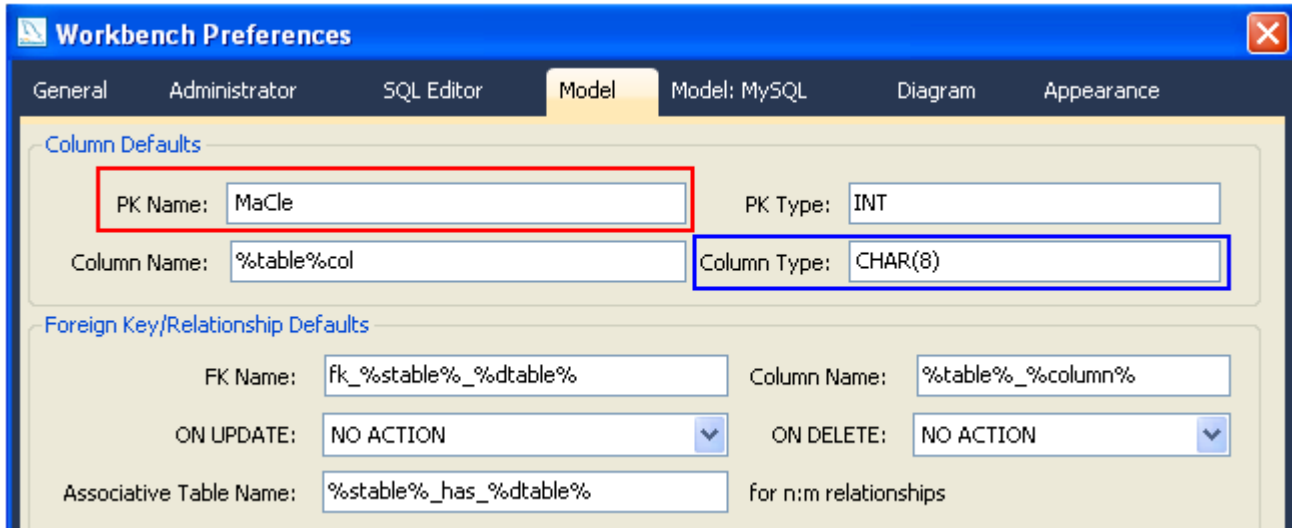


Figure 10.11 - Nouveaux noms et types par défaut pour les colonnes

#### 10.4.5. Nom par défaut des clés étrangères

Par défaut, le nom d'une étrangère est la concaténation de la constante « fk\_ », du nom de la table référençante, de la constante « \_ » et du nom de la table référencée, avec des possibilités d'autres noms par défaut (substitutions) :

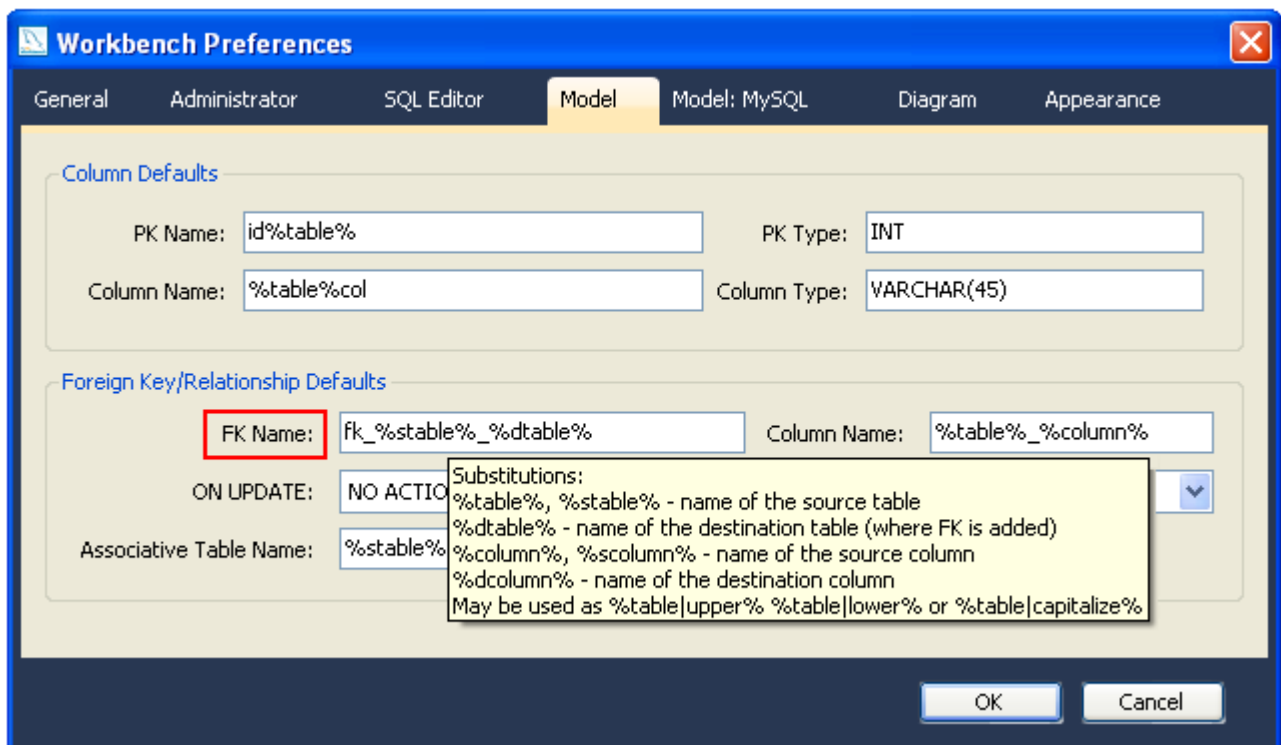


Figure 10.12 - Clé étrangère, nom par défaut

Les colonnes d'une clé étrangère ont aussi un nom par défaut : nom de la table référençante concaténé avec le nom de la colonne correspondante de la clé primaire de la table référencée :

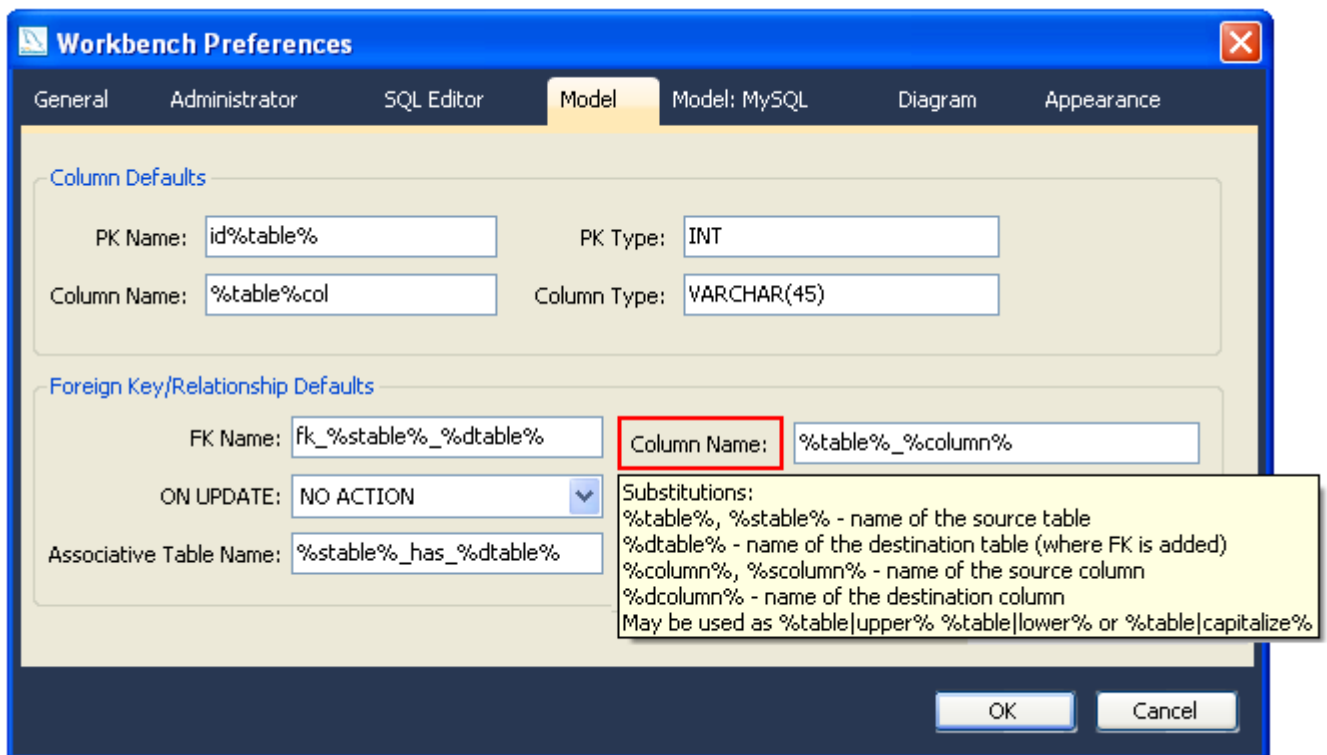


Figure 10.13 - Nom par défaut de colonnes d'une clé étrangère

Si on ne souhaite pas que le nom de la table référençante fasse partie du nom par défaut des colonnes composant la clé étrangère, il suffit de remplacer « %table%\_%column% » par « %column% ».

#### 10.4.6. Note à propos des actions de compensation (ON UPDATE, ON DELETE)

La fenêtre « Model » utilisée jusqu'ici pour définir les valeurs par défaut des noms des colonnes et leur type permet aussi de définir les actions de compensation par défaut. Une action de compensation permet de décider du comportement du SGBD au cas où :

- On souhaite remplacer par une valeur k2 la valeur k1 ( $k1 \neq k2$ ) de la clé (primaire en général) d'une table T1 qui fait l'objet de contraintes référentielles de la part d'autres tables T2, ..., Tn (voire T1 elle-même en cas d'auto-référence), T1 jouant le rôle de table référencée ;
- On souhaite supprimer au moins une ligne de T1 : les tables énumérées ci-dessus touchées seront en droit de donner leur accord pour en subir les conséquences, ou bien refuser énergiquement et donc faire échouer l'opération.

On retrouve ici les options possibles :

RESTRICT (et/ou NO ACTION selon les SGBD) pour refuser énergiquement les opérations de mise à jour portant sur la table référencée, et CASCADE pour les accepter et les propager dans les tables référençantes. Il va de soi que SET NULL est une incongruité laissant le champ libre au bonhomme Null (*Horresco referens !*), lequel est hors-la loi au pays des relations.

A cette occasion, faisons un retour rapide sur l'intégrité référentielle et le métabolisme des données.

Prenons l'exemple des commandes passées par les clients. On a vu à l'occasion de la déclaration des tables CLIENT et COMMANDE que dans un 1er temps ces tables n'entretenaient aucune relation (cf. paragraphe 5.2.1) :

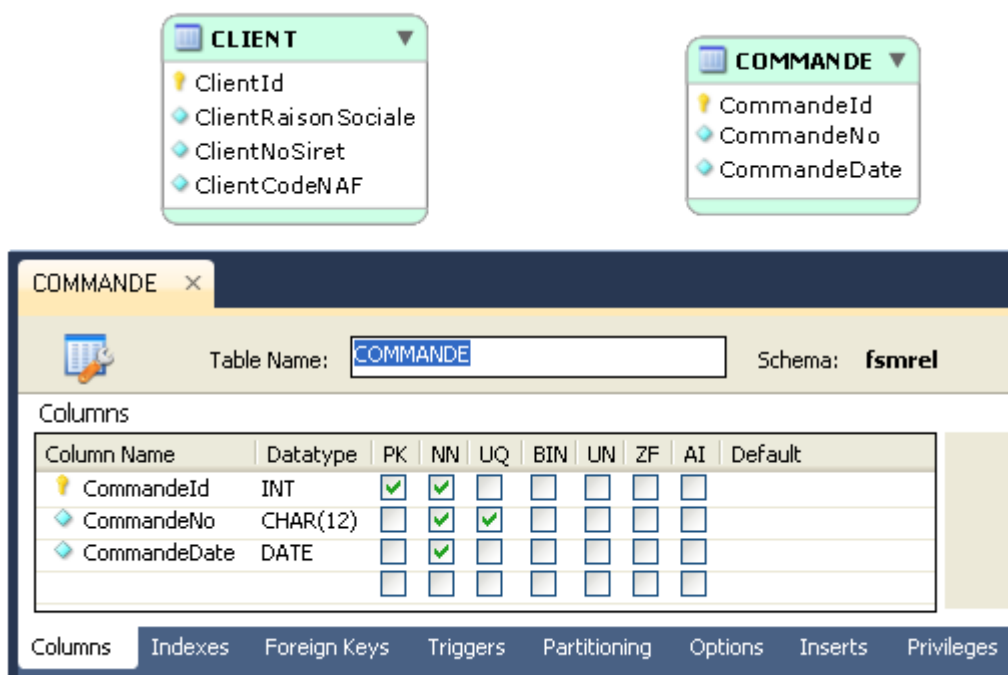


Figure 10.14 - Absence de relation entre tables

Une fois qu'on a connecté les deux tables, MySQL Workbench a recopié la colonne ClientId de l'en-tête de la table Client dans l'en-tête de la table COMMANDE :

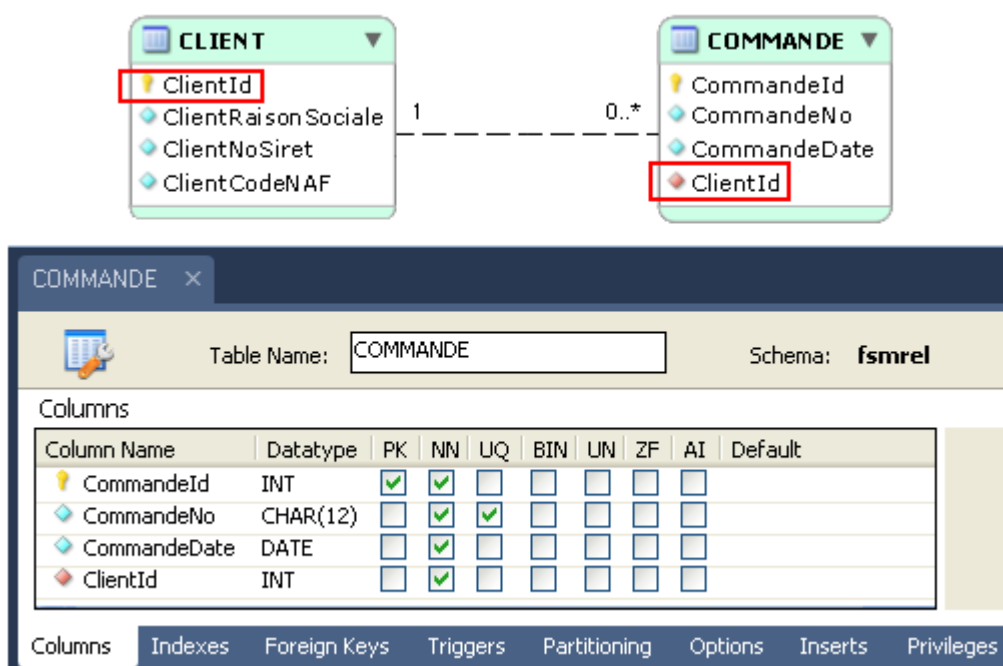


Figure 10.15 - Relation effective entre tables

Un lien a été établi entre les deux tables, ce qui signifie fondamentalement que chaque valeur prise par la colonne ClientId de l'en-tête de la table COMMANDE doit être une valeur prise en premier lieu par la colonne ClientId de l'en-tête de la table CLIENT : comme {ClientId} est clé de la table CLIENT (clé primaire dans l'exemple ou [clé candidate](#) dans le cas général), on conclut qu'une commande doit nécessairement faire référence à un client existant, ce qui du reste est la moindre des choses si l'on tient à ce que la base de données soit valide et que les comptables de chez Dubicobit ne viennent pas exiger, à juste titre, que les choses rentrent dans l'ordre...

Par définition la colonne ClientId appartient à une **clé étrangère** {ClientId} de la table COMMANDE, dont les valeurs doivent d'abord être des valeurs de la clé {ClientId} de la table CLIENT. Plus précisément on est en présence d'une contrainte d'**intégrité référentielle**. Si la colonne ClientId de l'en-tête de la table COMMANDE prenait une valeur qui ne fût pas une valeur prise par la colonne ClientId de l'en-tête de la table CLIENT, alors la contrainte d'intégrité référentielle serait violée : à charge du SGBD de veiller au respect de la contrainte, ce qu'on lui signifie ainsi en SQL :

```
CREATE TABLE COMMANDE
... ClientId          INT          NOT NULL ...
  CONSTRAINT COMMANDE_CLIENT_FK FOREIGN KEY (ClientId)
    REFERENCES CLIENT (ClientId)...
```

Où COMMANDE\_CLIENT\_FK est le nom de la contrainte à faire respecter.

### Du métabolisme des données

La contrainte COMMANDE\_CLIENT\_FK ci-dessus est manifestement statique, elle dit seulement qu'une commande doit faire référence à exactement un client, ni plus, ni moins. Comme disait Ted Codd, au-delà de l'aspect statique, anatomique de la base de données, il faut aussi se préoccuper du métabolisme des données pour mieux comprendre la nature de celles-ci, de leurs relations, et à un niveau plus terre à terre, ne pas avoir la sensation d'avoir les pieds pris dans le béton lors des opérations de mise à jour, et cela grâce à des actions de compensation illustrées par les clauses ON UPDATE et ON DELETE :

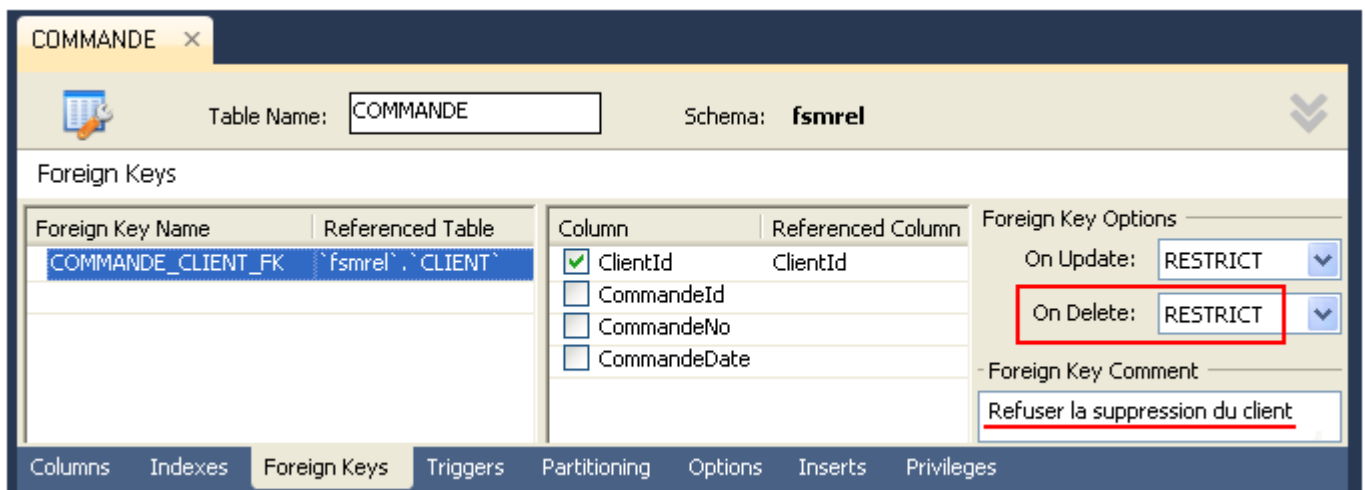


Figure 10.16 - Du métabolisme des données



Si, comme dans la figure ci-dessus, c'est l'option RESTRICT qui a été retenue pour la table COMMANDE :

```
CREATE TABLE COMMANDE
... ClientId          INT          NOT NULL ...
  CONSTRAINT COMMANDE_CLIENT_FK FOREIGN KEY (ClientId)
    REFERENCES CLIENT (ClientId) ON DELETE RESTRICT ...
```

Alors, si l'on exécute une instruction DELETE :

```
DELETE FROM CLIENT
WHERE ClientId = 123 ;
```

Comme on l'a évoqué dans le paragraphe 1.2, de deux choses l'une, ou bien il y a au moins une commande pour le client pour lequel on vérifie la condition « ClientId = 123 », auquel cas la demande de suppression du client sera rejetée, ou bien il n'y a aucune commande pour ce client, auquel cas la demande de suppression sera satisfaite (à condition, bien sûr, qu'il n'y ait pas d'autres tables référençant la table CLIENT et pouvant provoquer, comme dans le cas de la table COMMANDE, un rejet de la demande de suppression).

En passant : avec certains SGBD (par exemple MS SQL Server) on n'utilise pas RESTRICT mais NO ACTION. Avec d'autres (par exemple DB2 for z/OS ou MySQL), on peut utiliser RESTRICT et NO ACTION.

Rappelons que l'option CASCADE n'a guère de sens que lorsque les tables atteintes par les stimuli engendrés par un DELETE correspondent à des entités-types spécialisées (cf. paragraphe 6.1) ou à des entités-types faibles (cf. paragraphe 1.3), cas en l'occurrence des lignes de commande, lesquelles sont des propriétés multivaluées des commandes : une ligne de commande n'a pas plus de raison que la date de commande de s'opposer à la suppression légitime d'une commande...

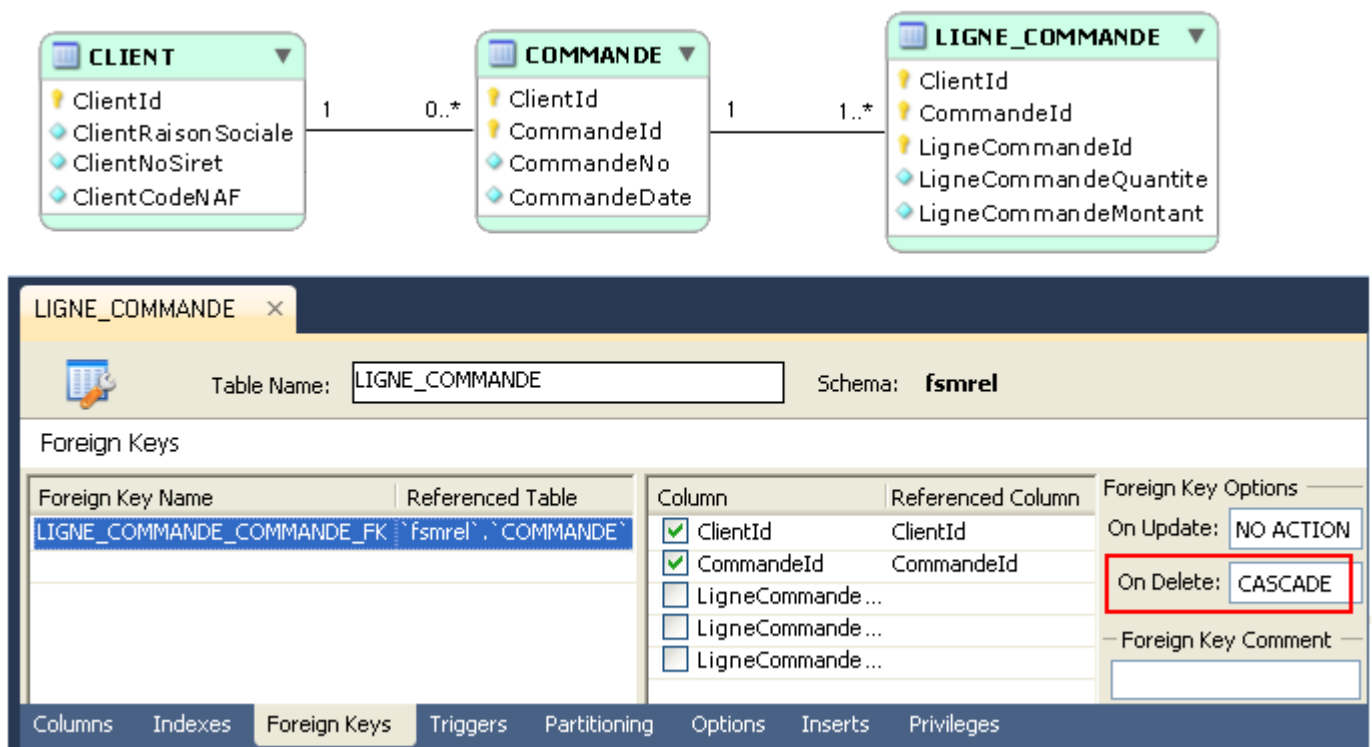


Figure 10.17 - La suppression d'une commande entraîne celle de ses lignes

Partant de là, on retient l'option CASCADE pour la table LIGNE\_COMMANDE :

```
CREATE TABLE LIGNE_COMMANDE
...
CONSTRAINT LIGNE_COMMANDE_COMMANDE_FK FOREIGN KEY (ClientId, CommandeId)
REFERENCES COMMANDE ON DELETE CASCADE ...
```

En conséquence de quoi, si aucune autre contrainte ne s'y oppose, l'exécution de l'instruction suivante provoque, dans la table LIGNE\_COMMANDE, la suppression des lignes de la commande 2 du client 123 :

```
DELETE FROM COMMANDE
WHERE ClientId = 123 AND CommandeId = 2 ;
```

#### 10.4.7. Nom par défaut des tables associatives

Rappelons (cf. paragraphe 4) qu'une table associative résulte d'une association de plusieurs à plusieurs entre deux tables (ou plusieurs, cf. paragraphe 5.3), par exemple DEPOT et ARTICLE : un type d'article donné peut être stocké dans plusieurs dépôts et dans un dépôt donné peuvent être stockés plusieurs types d'articles.

Situation avant connexion des tables :

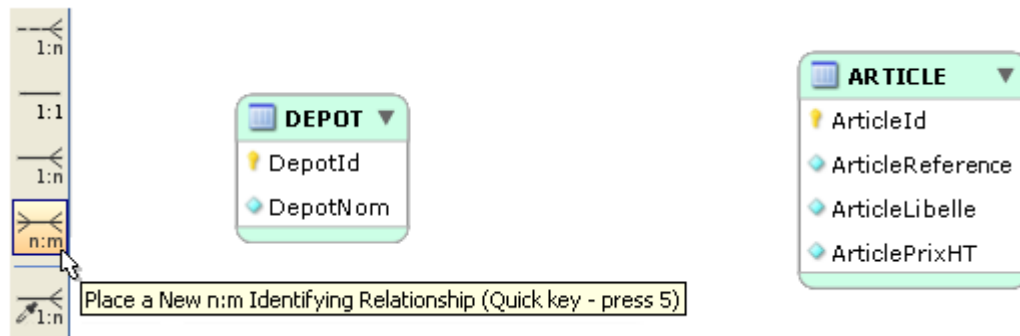


Figure 10.18 - Tables à associer

Du fait du paramétrage ci-dessous, par défaut, la table associative connectant DEPOT (considérée ci-dessous comme source) et ARTICLE (considérée comme cible) sera nommée DEPOT\_has\_ARTICLE :

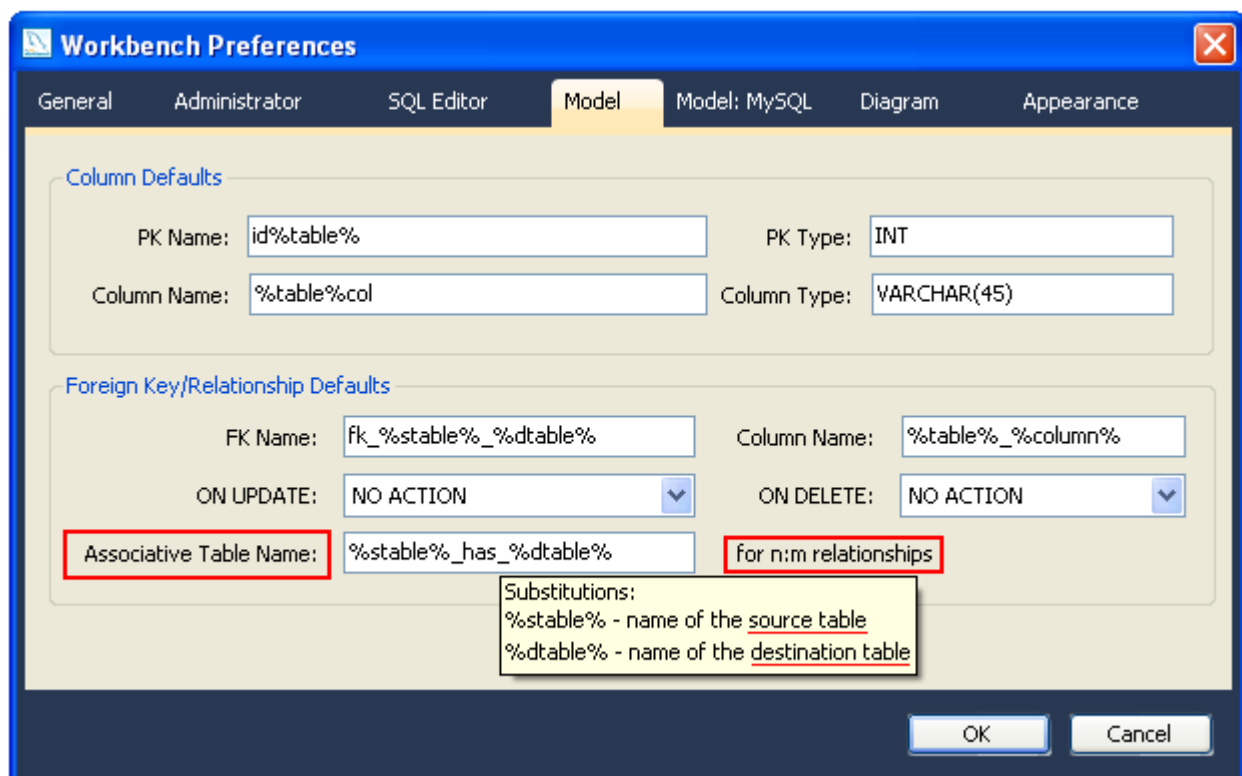


Figure 10.19 - Nom par défaut d'une table associative

Mais si, par exemple, on remplace « %stable%\_has\_%dtable » par « %stable%\_%dtable », le nom par défaut de la table associative sera DEPOT\_ARTICLE.

## 10.5. Changer l'ordre des colonnes d'une table

Pour changer l'ordre des colonnes dans l'en-tête d'une table, il suffit de sélectionner la colonne à changer de place (colonne CodeNAF ci-dessous) et effectuer un glisser-déposer vers la position voulue ; l'outil permet de visualiser le déplacement de la colonne en affichant une barre ad-hoc.

Avant et pendant :

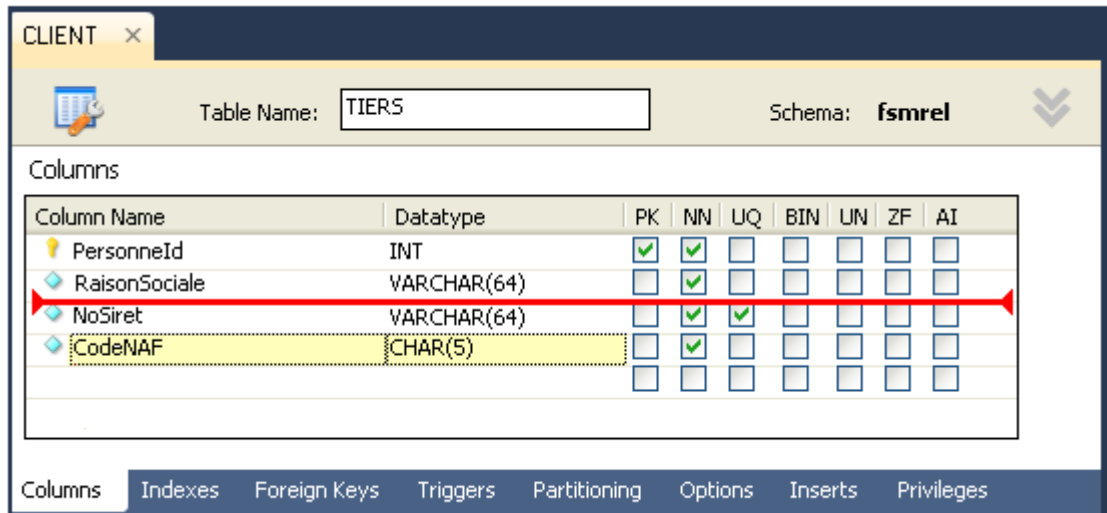


Figure 10.20 - Changer l'ordre des colonnes d'une table : situation avant

Après :

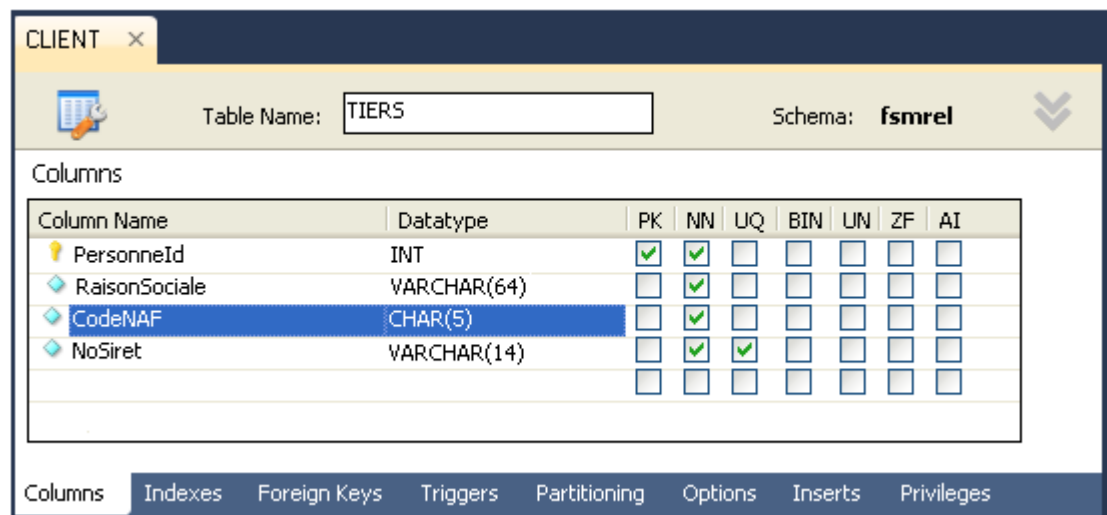


Figure 10.21 - Changer l'ordre des colonnes d'une table : après

## 10.6. Urbanisation des diagrammes

### 10.6.1. Diagrammes comportant un grand nombre d'objets

Un défi ergonomique est de faire tenir les objets d'un diagramme sur une seule page au format A4 ou A3. A force d'être chargé, un diagramme devient illisible, avec des ficelles qui se croisent dans tous les sens, ça tient de l'art new-yorkais, et le concepteur tente en vain d'y mettre de l'ordre. Comme disait un des champions de Merise, Arnold Rochfeld, quand on voit un diagramme comportant plus de sept objets, on peut préparer l'aspirine...

Qui plus est, si le grand projet de la société Volfoni Frères (concurrente de Dubicobit) donne lieu à un diagramme comportant quelque chose comme mille types d'entités (ou schémas de tables dans le cas de MySQL Workbench), l'administration des données prendra la précaution d'urbaniser tout cet ensemble en grands domaines (référentiels), sous-domaines et sous-sous-domaines cohérents. Ainsi, ce projet aura été décomposé en référentiels : Personnes, Contrats, Cotisations, Catalogue produits, Prospection de masse, Éditique, Habilitations, etc., peut-être une vingtaine, voire plus. Au niveau le plus fin, la représentation graphique doit tenir — de façon claire et lisible — sur l'équivalent d'un ensemble de pages au format (disons) A3, chacune d'elles constituant une vue suffisante pour qu'un développeur puisse s'y retrouver.

Dans le même sens, supposons que l'on effectue la rétro-conception (cf. paragraphe 9) d'un modèle de taille raisonnable et que le résultat (pour le moins glauque !) de l'opération soit le suivant :

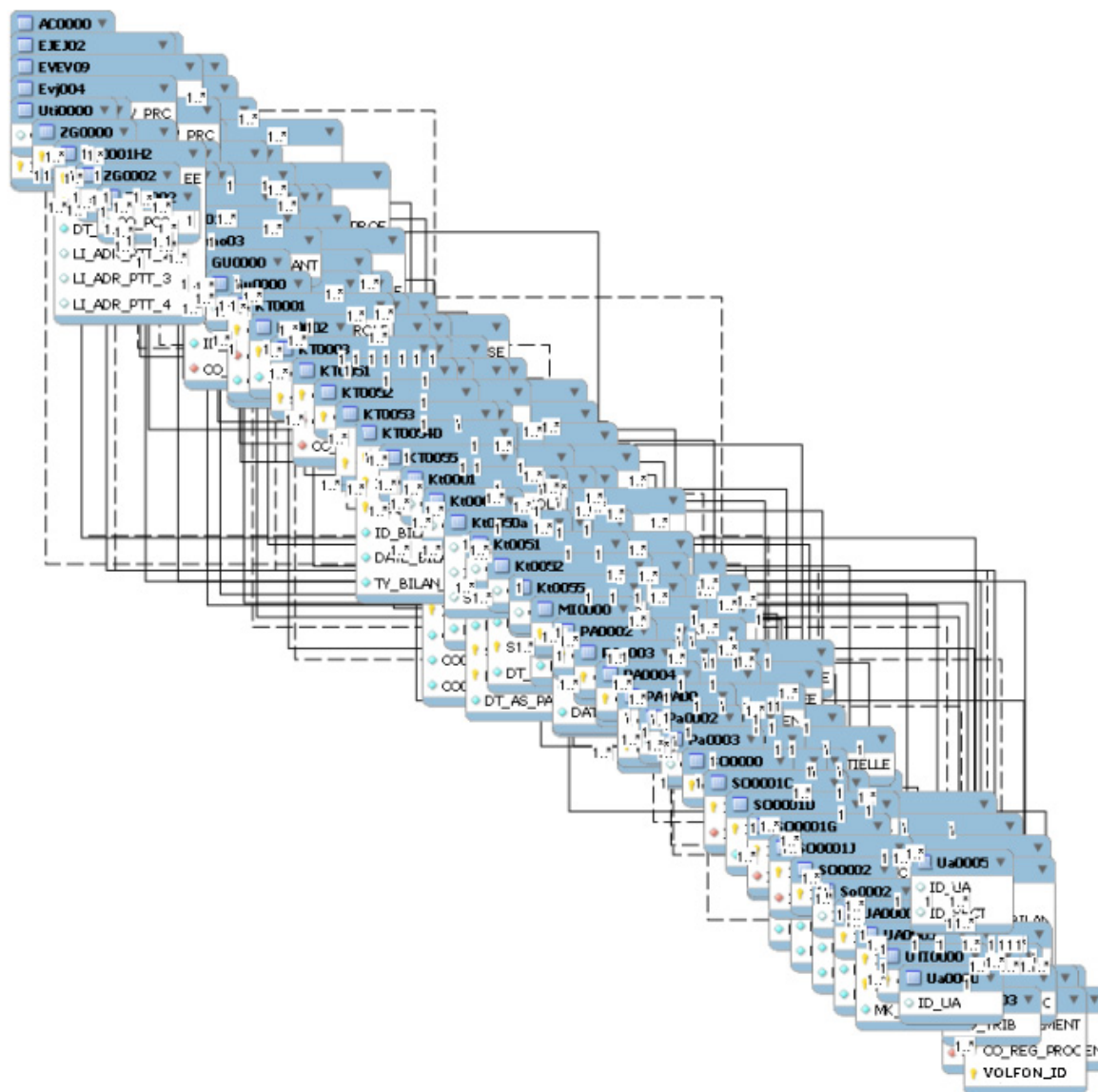


Figure 10.22 - Un magma à urbaniser

De cette bouillie indigeste de 180 schémas de tables, faire quelque chose de lisible tenant sur un seul diagramme est évidemment impossible : il n'y a plus qu'à urbaniser, faire figurer dans un même diagramme les tables dont les liens sémantiques sont évidents, sans pour autant supprimer le diagramme bouillie, valide par hypothèse, mais qui n'aura plus guère l'occasion de sortir de la cave.

Du strict point de vue technique, MySQL Workbench permet d'urbaniser sans difficulté particulière. A titre indicatif, on peut procéder ainsi : en plus du diagramme bouillie, créer un diagramme par sous-ensemble (domaine, sous-domaine, etc.) dont on veut une vue intelligible et cohérente, et sur lequel on pourra travailler confortablement.

### 10.6.2. A la manoeuvre avec MySQL Workbench

Exemple : Supposons qu'on veuille créer un diagramme « Commandes » dédié aux commandes, à partir du diagramme « Général », lequel contient l'ensemble des tables, dont celles à faire figurer dans le nouveau diagramme (notamment les tables TIERS et CLIENT visibles ci-dessous) :

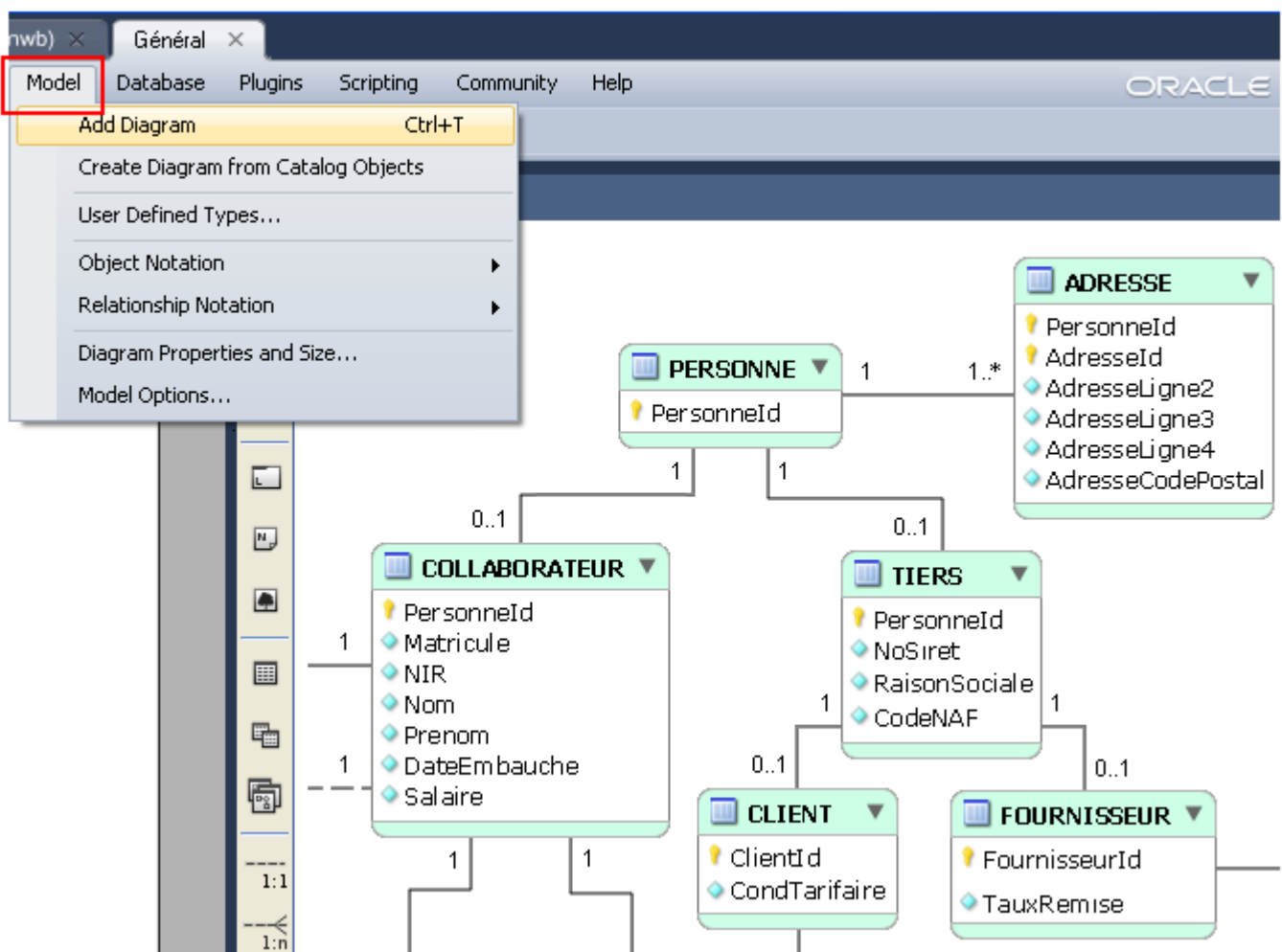



Figure 10.23 - Diagramme « Général » source (vue partielle)

Opérations pour construire le diagramme « Commandes » :

- Créer un diagramme à l'aide de la commande ad-hoc (cf. paragraphe 10.6.3).
- Lui donner un nom (cf. paragraphe 10.6.4).
- Le peupler avec les tables voulues (cf. paragraphe 10.6.5).
- En retirer (en les masquant) les objets (tables, liens) qu'on aura fait figurer inutilement (cf. paragraphe 10.6.6).

### 10.6.3. Ajouter un diagramme

Pour créer le diagramme « Commandes » (vide), utiliser CTRL-T ou la commande « Model > Add diagram » ou encore cliquer sur l'icône  de la barre de menus. Par défaut, le nouveau diagramme porte le nom de « EER Diagram » :

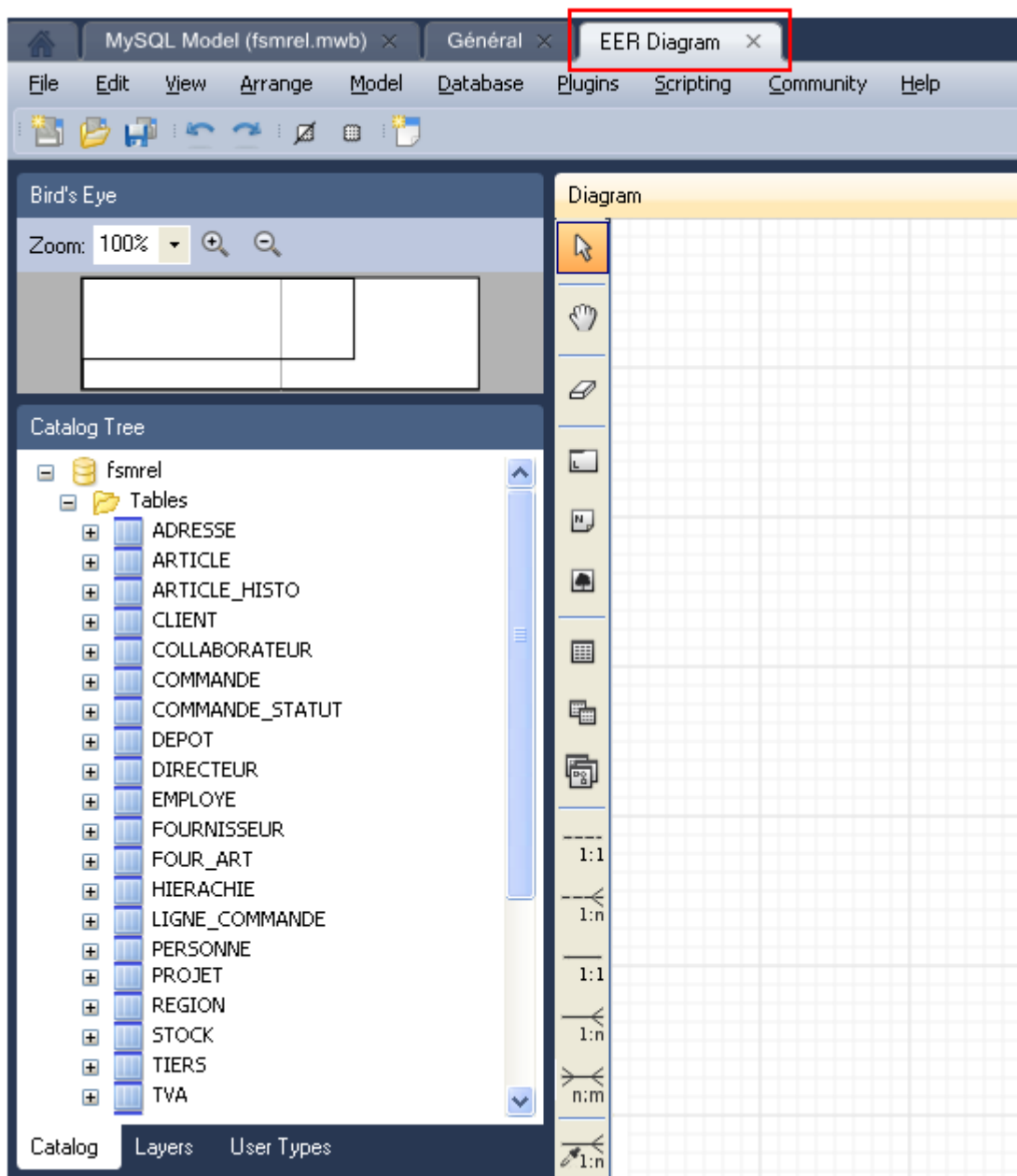


Figure 10.24 - Nouveau diagramme « EER Diagram »

#### 10.6.4. Renommer un diagramme

Le diagramme à renommer porte actuellement le nom de « EER Diagram » :



Figure 10.25 - Diagramme à renommer

Passer par l'onglet « Model » et choisir « Diagram properties and Size... » :

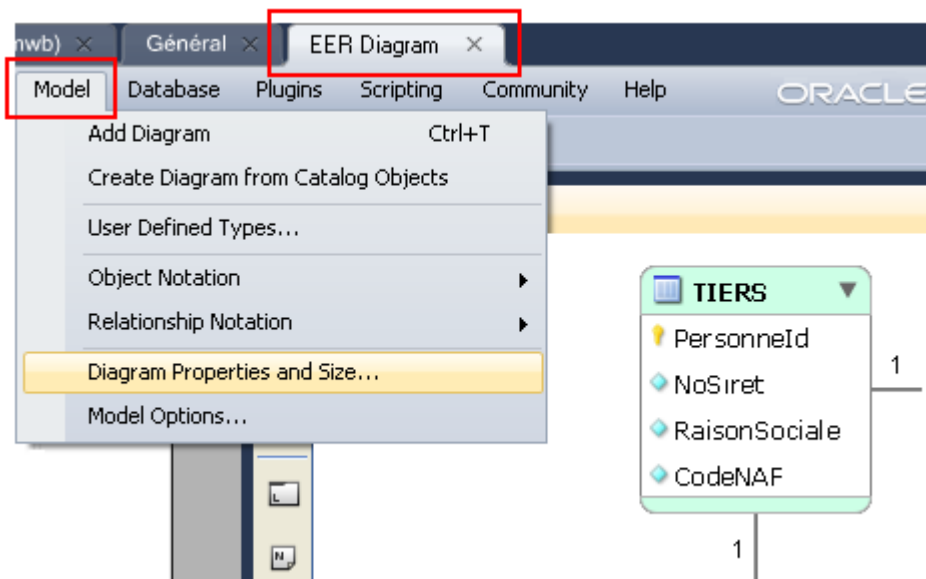


Figure 10.26 - « Diagram properties and Size »



Apparaît la fenêtre « Diagram properties » : on pourra alors renommer « EER Diagram » comme on l'entend.

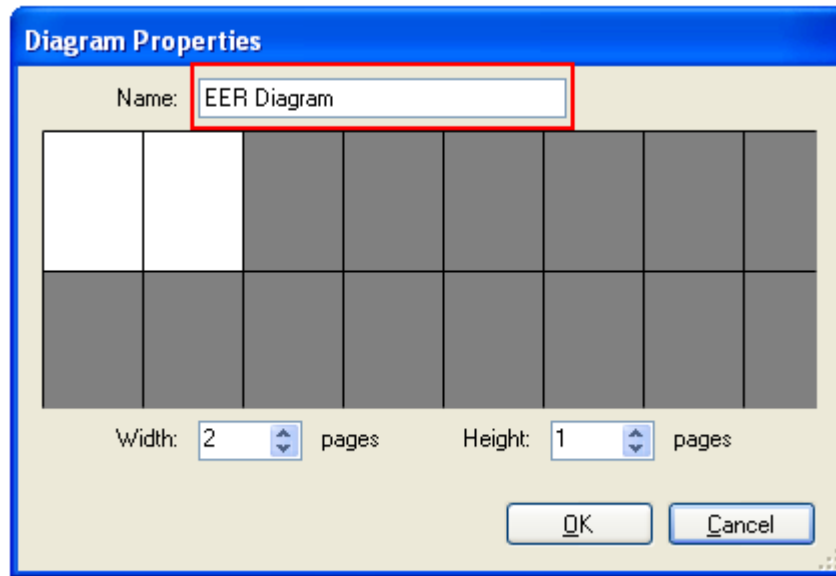


Figure 10.27 - Fenêtre « Diagram Properties »

Une autre méthode pour renommer un diagramme : cliquer sur l'onglet « MySQL Model », l'outil affiche alors la fenêtre « Model Overview » et un nom de diagramme : « EER Diagram » :

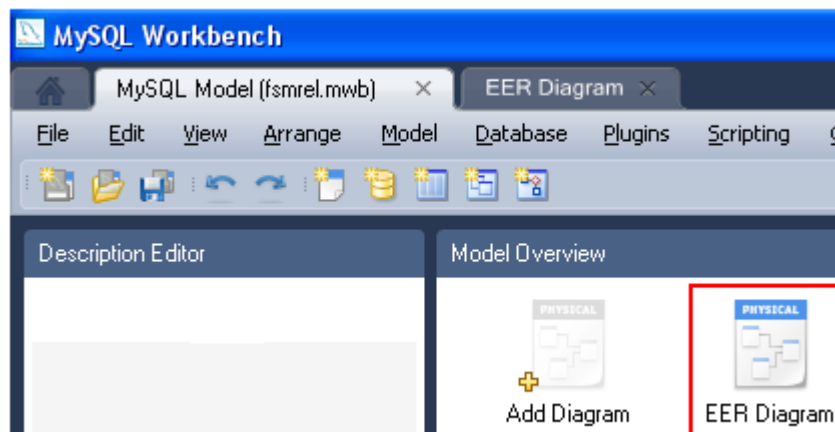


Figure 10.28 - Renommer un diagramme, autre méthode (a)

Pour changer le nom du diagramme, on clique délicatement sur le symbole du diagramme, ce qui suffit normalement pour sélectionner le nom et le changer, ou bien on utilise la touche F2 :

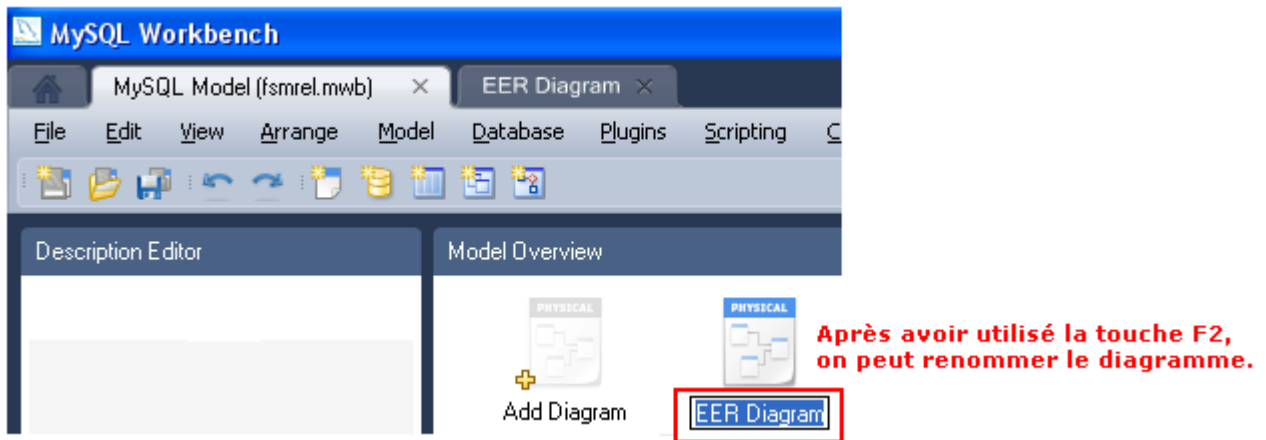


Figure 10.29 - Renommer un diagramme, autre méthode (b)

On peut alors renommer « EER Diagram », par exemple en « Général » :

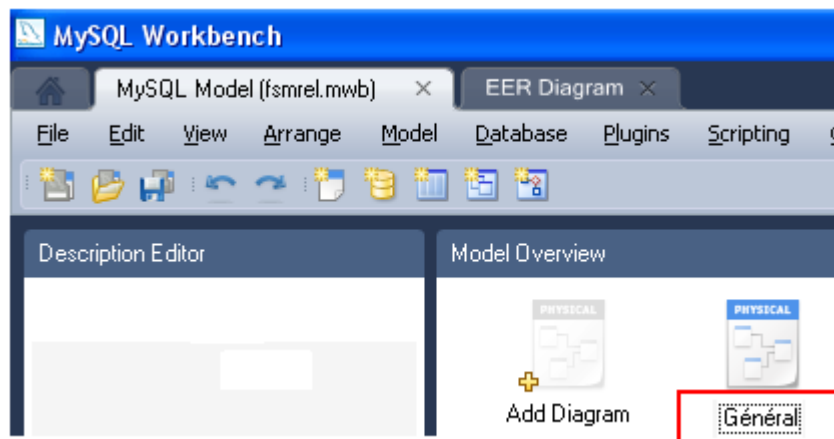


Figure 10.30 - Renommer un diagramme, autre méthode (c)

En double-cliquant sur « Général », on ouvre le diagramme correspondant.

### 10.6.5. Peupler un diagramme à partir du catalogue

Les noms des tables à faire figurer dans le diagramme « Commandes » apparaissent dans le catalogue (*Catalog Tree*) :

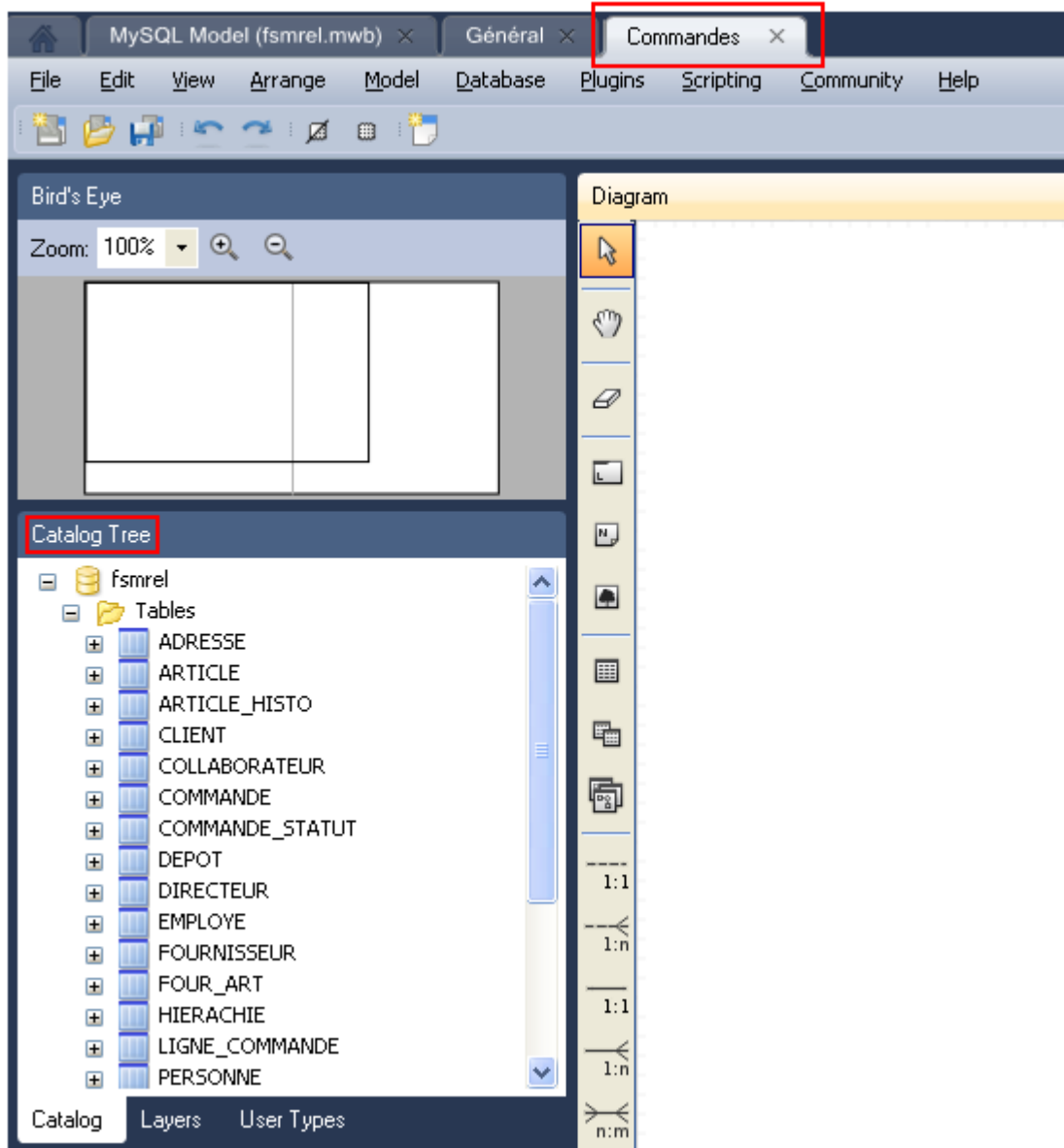


Figure 10.31 - Diagramme « Commandes » (a)

Le diagramme étant vide de tout objet, pour y faire figurer par exemple les tables COMMANDE et LIGNE\_COMMANDE, on procède par glisser-déposer de leur nom à partir du catalogue, un par un ou ensemble :

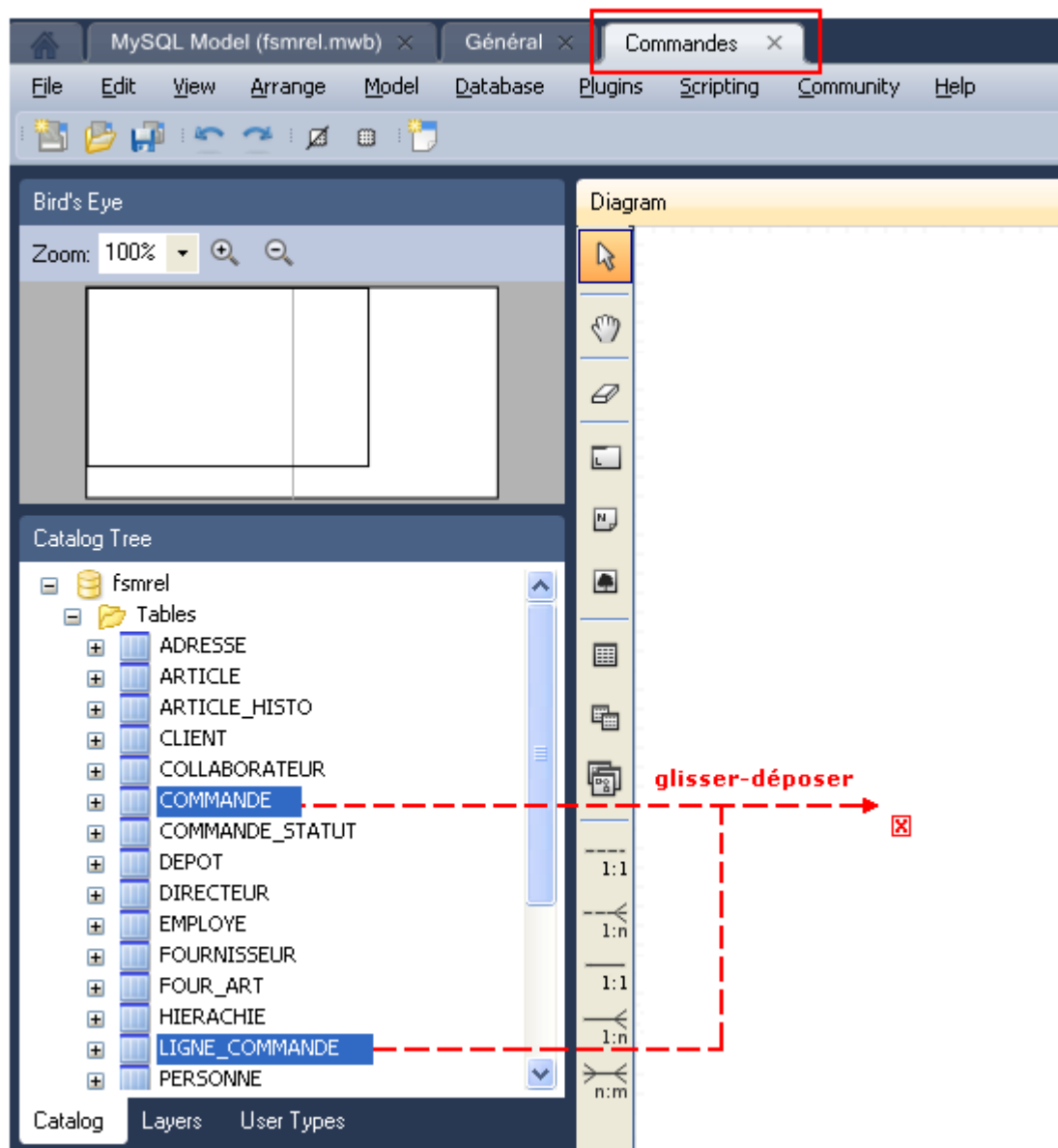


Figure 10.32 - Diagramme « Commandes » (b)

Au résultat (en notant bien que MySQL Workbench établit automatiquement les liens entre les tables) :

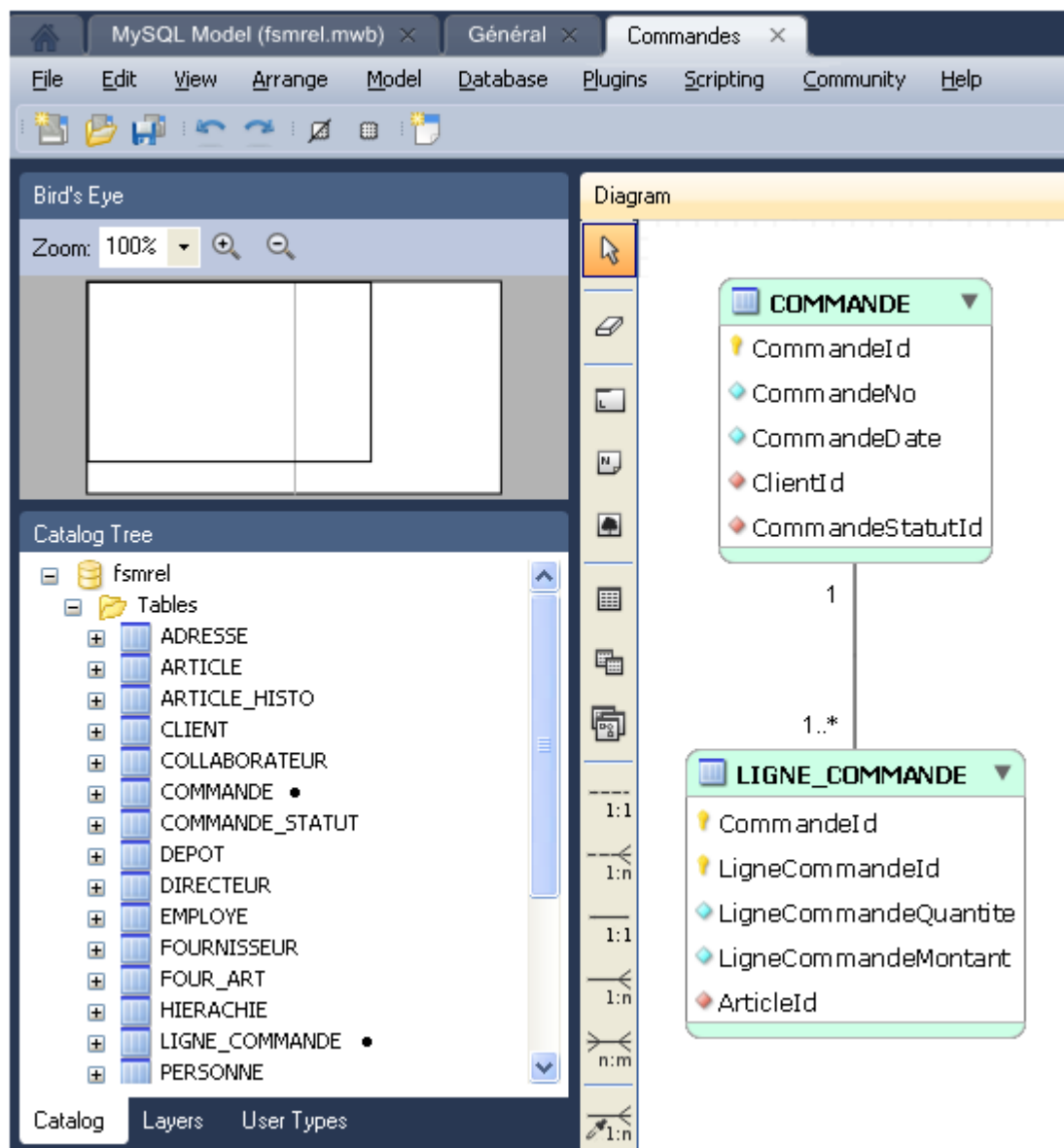


Figure 10.33 - Diagramme « Commandes » (c)

Il n'y a plus qu'à compléter le diagramme avec les autres tables parties prenantes dans les commandes :

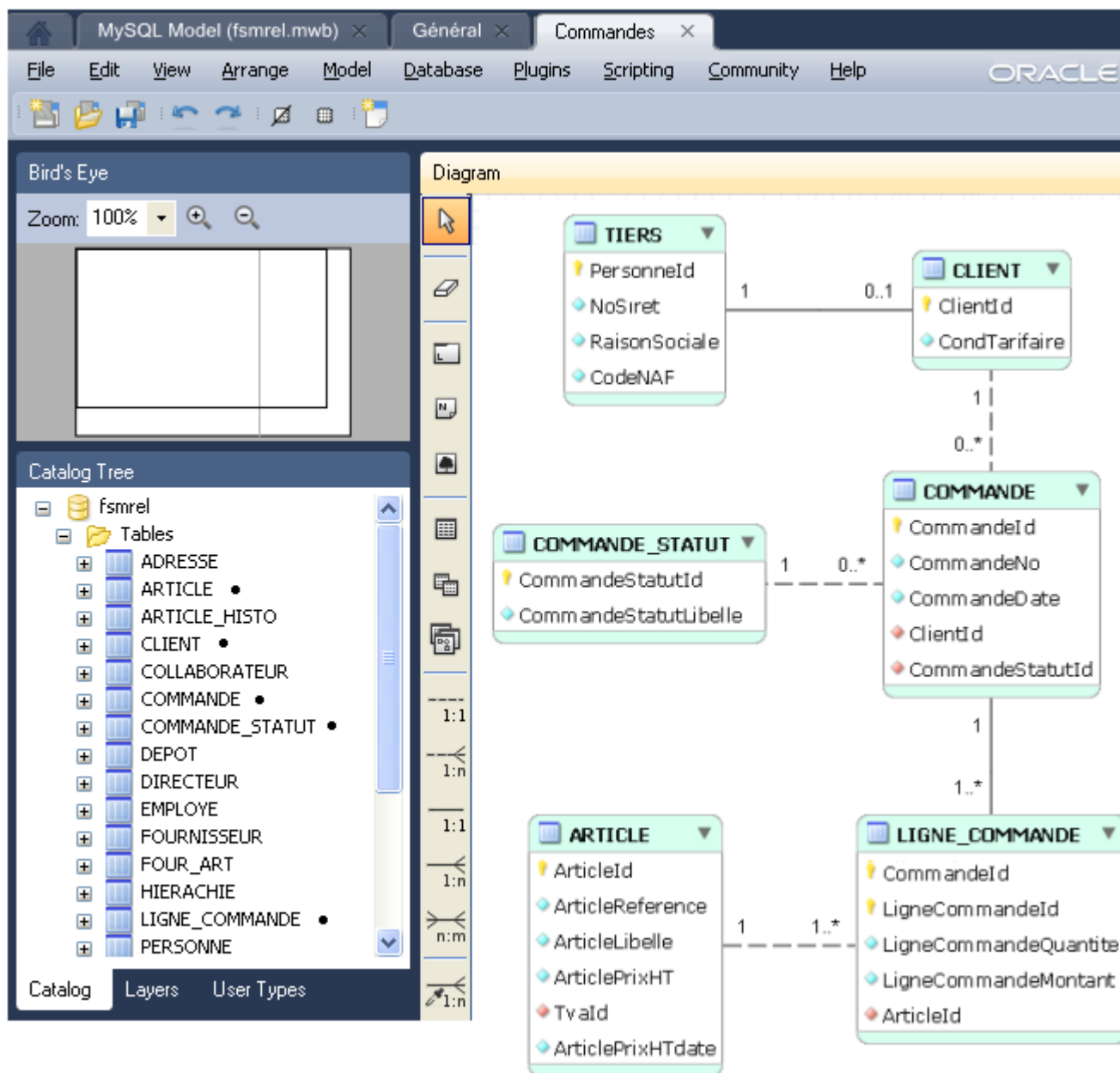


Figure 10.34 - Diagramme « Commandes » (d)

### 10.6.6. Masquer une table ou un lien dans un diagramme

L'objet de l'opération est de ne plus afficher un objet (table ou lien) dans un diagramme, sans pour autant le supprimer (ne serait-ce que parce que d'autres diagrammes s'en servent !) Par exemple pour masquer la table COMMANDE\_STATUT :

Sélectionner la table, puis faire un clic droit pour avoir accès à la commande « Delete » (ou passer par la commande « Menu > Edit > Delete ») :

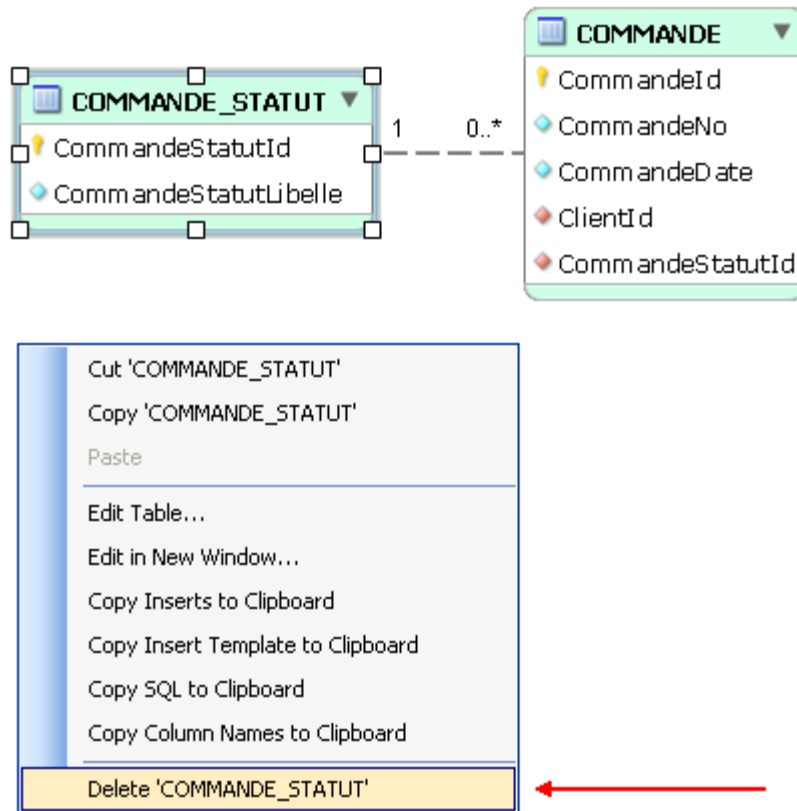


Figure 10.35 - Masquer une table (a)

Il y a ouverture de la fenêtre « Delete Object ». Choisir « Keep » pour effectivement masquer la table sans la supprimer :

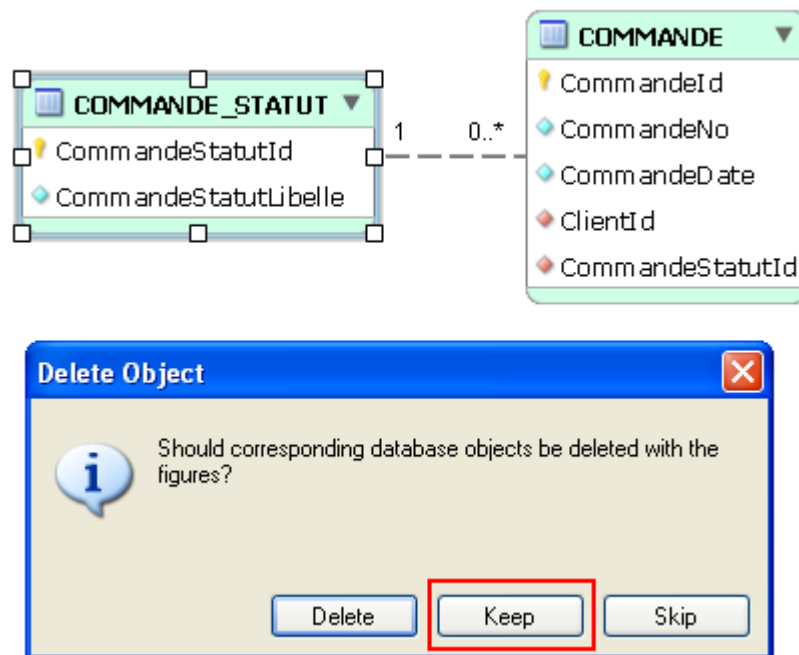


Figure 10.36 - Masquer une table (b)

Au résultat, la table **COMMANDE\_STATUT** n'est plus affichée (mais elle n'est pour autant supprimée, ouf !) :



Figure 10.37 - - Masquer une table (c)



Pour réafficher la table : effectuer un glisser-déposer de son nom à partir du catalogue des objets (*Catalog tree*) :

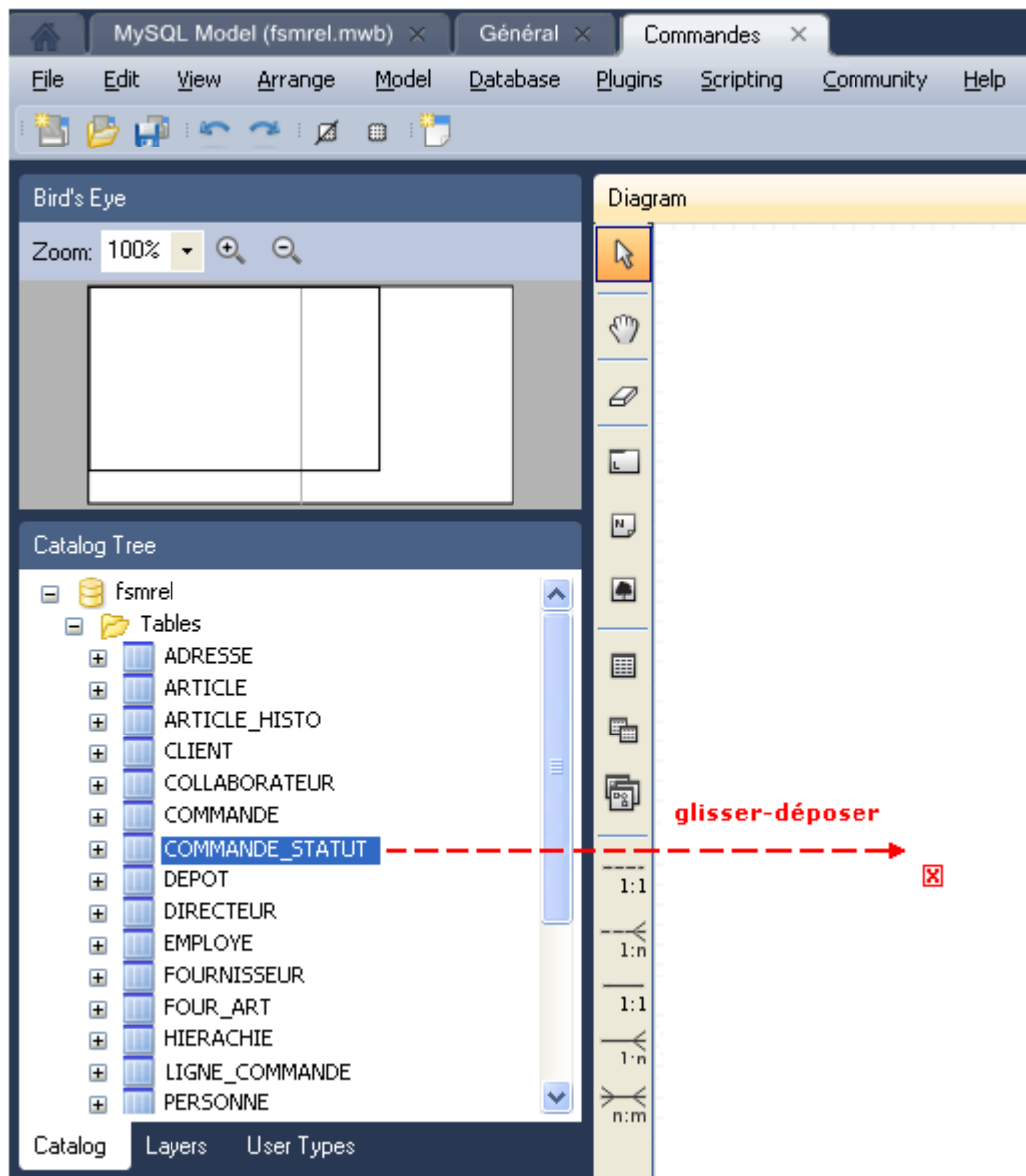


Figure 10.38 - Réafficher une table qu'on a masquée

### 10.6.7. Les couches de peinture (layers)

Histoire d'en rajouter une couche pour faire joli... Qu'il s'agisse du diagramme général (cf. paragraphe 10.6.2) ou d'un sous-diagramme ne contenant qu'un sous-ensemble de tables, on peut faire du coloriage et donner une sensation d'urbanisation. Partons d'un sous-diagramme où ne sont affichées que les commandes et leur livraison par les camions :

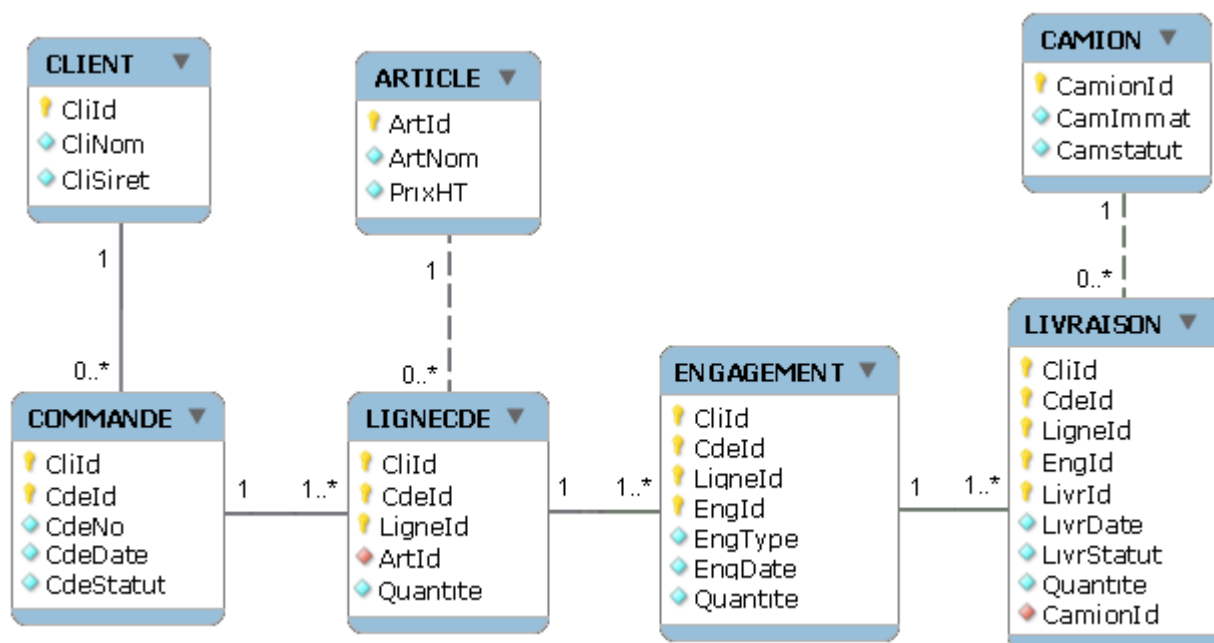


Figure 10.39 - Layers, situation initiale

On commence par cliquer sur l'icône ad-hoc :

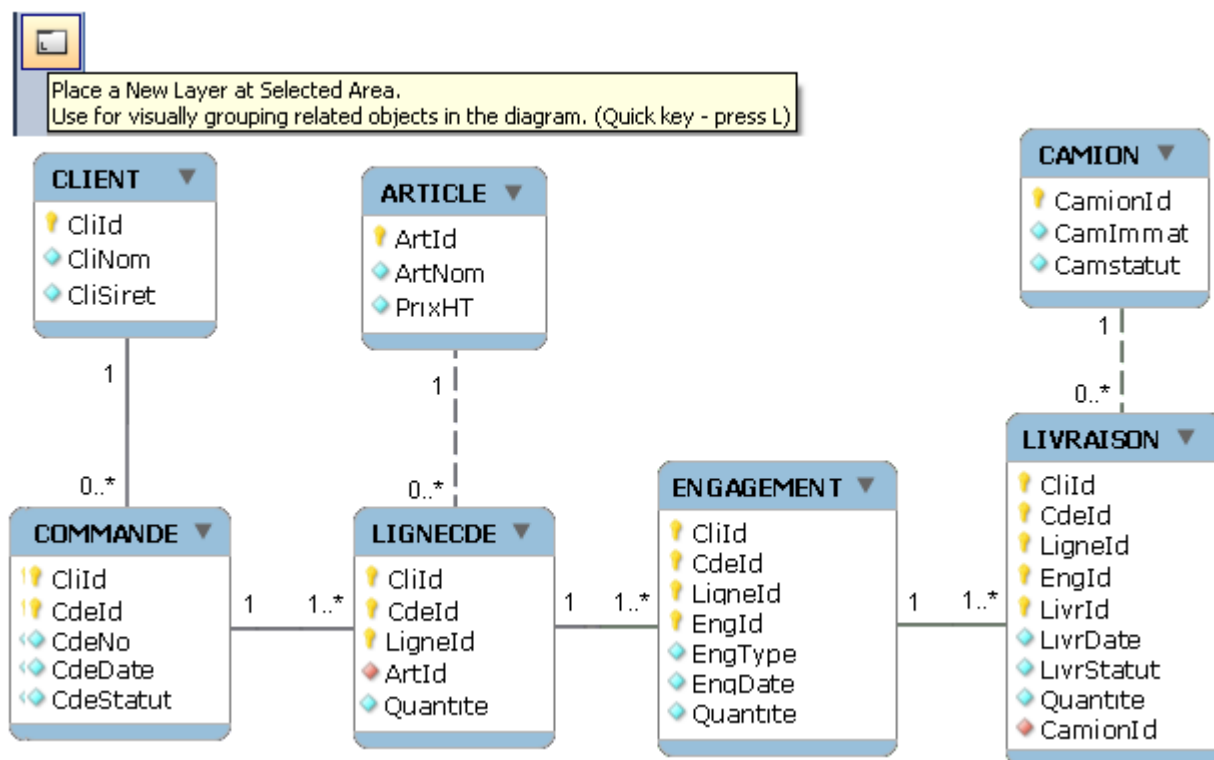


Figure 10.40 - Layers, l'icône ad-hoc

On entoure les objets à incorporer au layer (qui a été nommé « New Layer » suite au clic sur l'icône) :

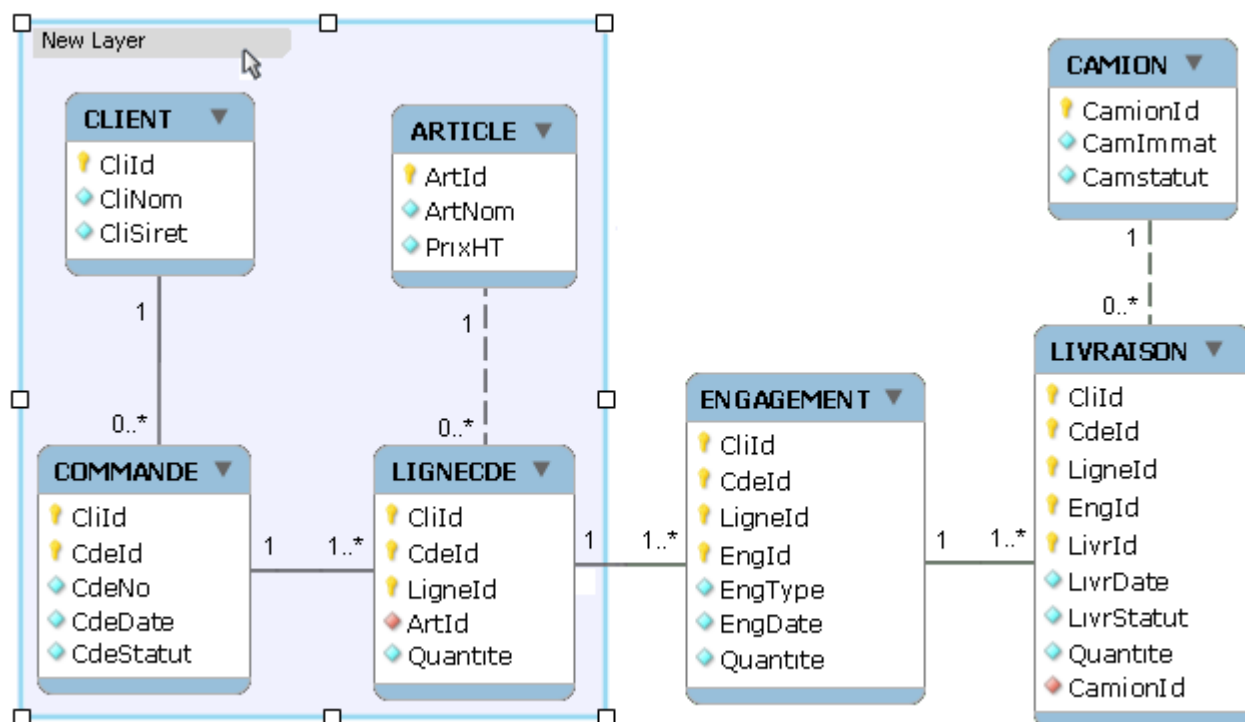


Figure 10.41 - Création d'un layer

Puis en double-cliquant, on renomme le layer comme ci-dessous (et au besoin on change le coloriage) :

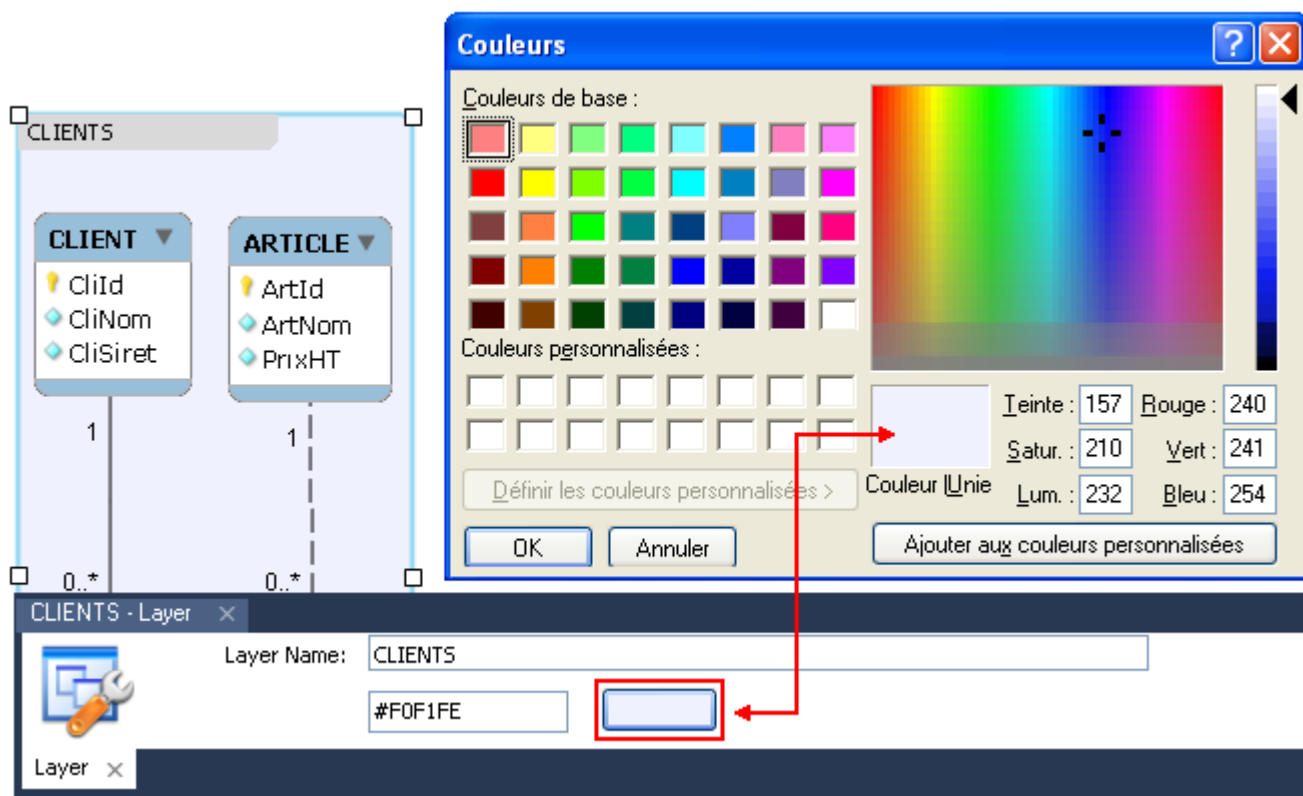
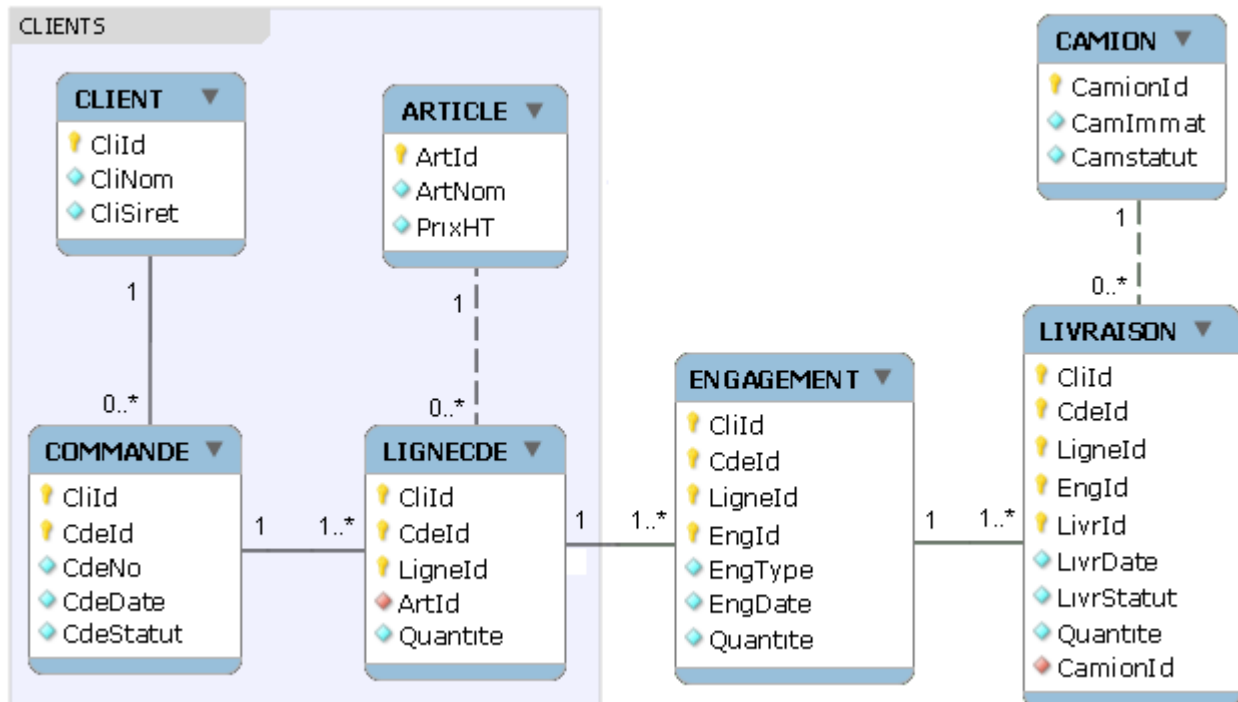
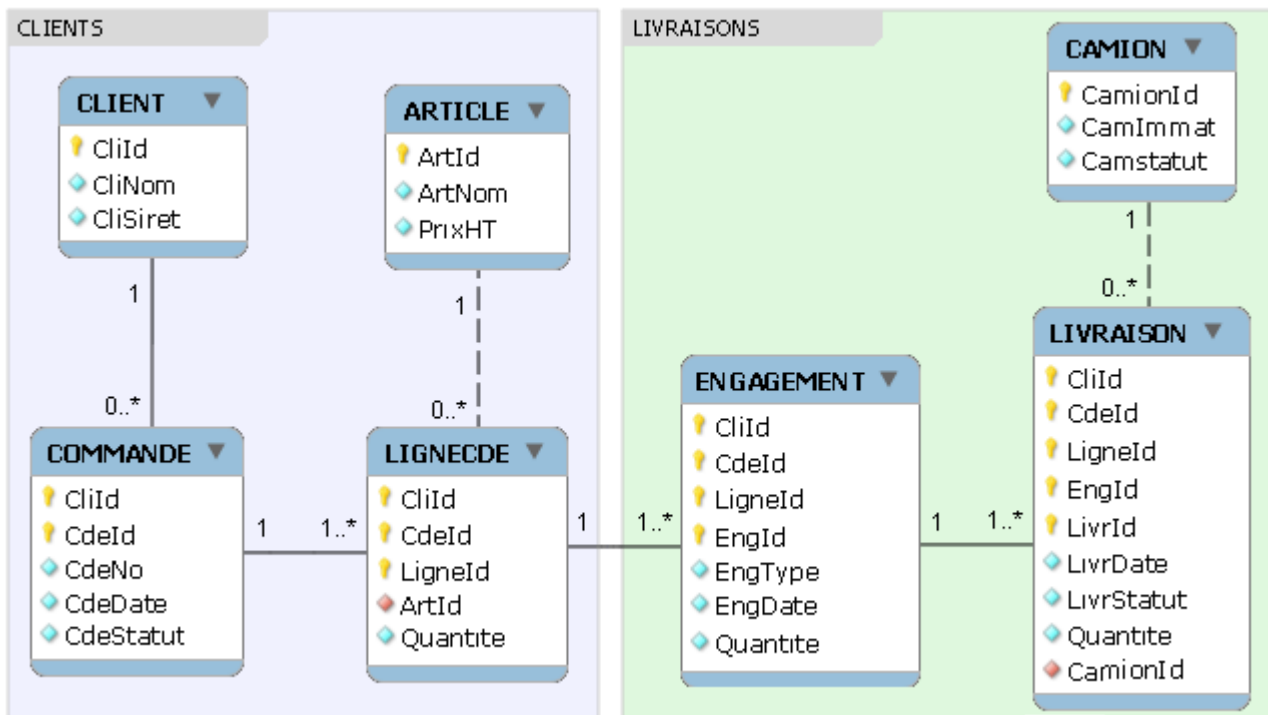


Figure 10.42 - Layer, renommage et coloriage

Au résultat :



Constitution d'un layer supplémentaire :



N.B. Quand on clique sur l'onglet associé à un layer, on peut le bouger, tout ce qu'il contient suit.

## 10.7. Définir une clé alternative

Dans un premier temps, on peut savoir si on a défini des clés alternatives pour une table et quelle en est la composition. Exemple (en France) du Siret des tiers (c'est-à-dire des clients et des fournisseurs), défini pour la table TIERS (attribut NoSiret). Supposons que la notation « Workbench simplifiée » soit active :

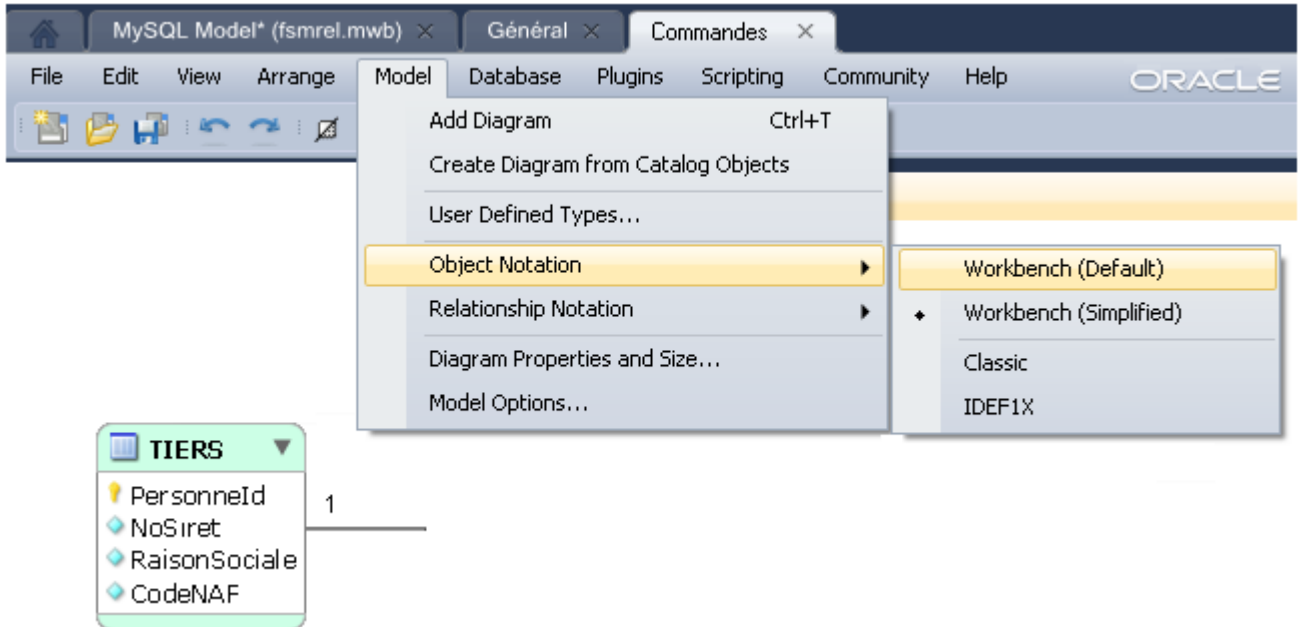


Figure 10.45 - Définir une clé alternative (a)

On passe à la notation « Workbench par défaut », ce qui permet de mettre en évidence le cartouche « Indexes » attaché à la table concernée (MySQL Workbench n'est pas avare en matière d'index...) :

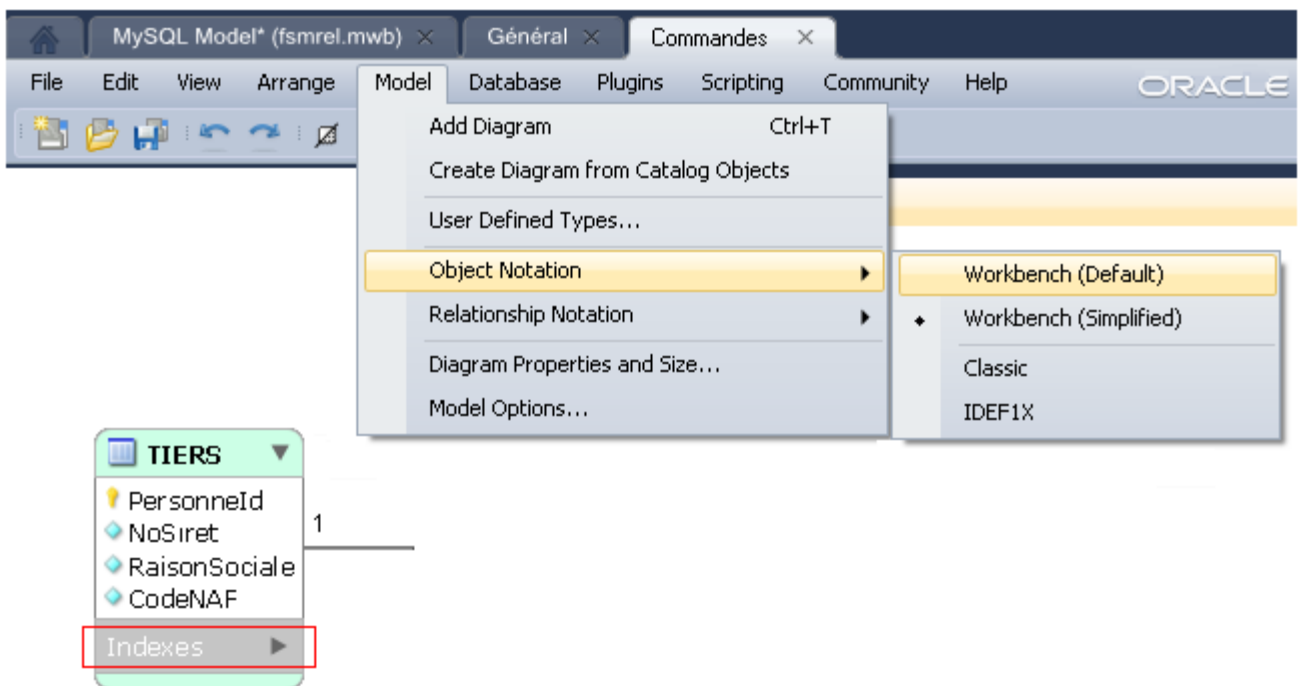


Figure 10.46 - Définir une clé alternative (b)

Après avoir cliqué sur le symbole « ► », par le truchement d'une infobulle, donc de façon fugace, on peut déjà savoir à la volée si on a défini des clés alternatives et quelle en est la composition :

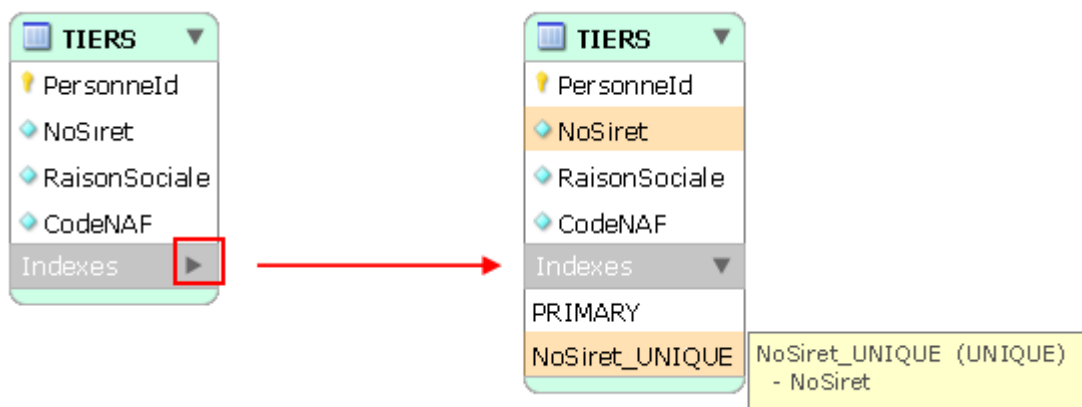


Figure 10.47 - Définir une clé alternative (c)

Si la clé alternative {NoSiret} entrevue ci-dessus n'a pas encore été définie, on le fait via l'onglet « Indexes », en ajoutant un index de type « UNIQUE » :

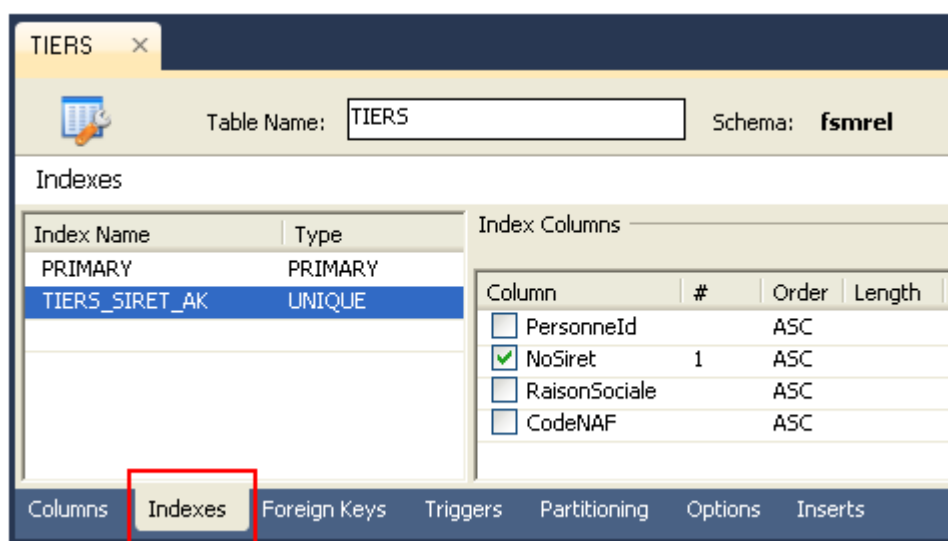


Figure 10.48 - Définir une clé alternative (d)

Cette façon de procéder vaut surtout pour les clés alternatives multi-colonnes. En passant par l'onglet « Columns », on peut aussi cocher la case qui va bien (« UQ ») pour les clés alternatives singletons (ce qui est quand même plus propre, puisqu'on ne descend pas au niveau physique et qu'on laisse l'outil se dépatouiller avec ses index, conformément à la huitième des douze règles de Codd) :

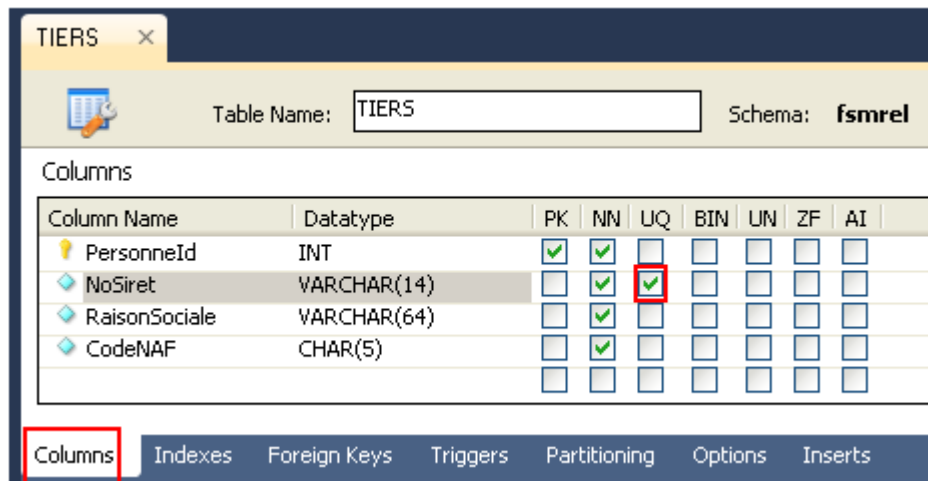


Figure 10.49 - Définir une clé alternative (e)

## 10.8. Changer l'ordre des colonnes d'une clé

Une clé est un ensemble, aussi l'ordre (des noms) des colonnes qui en sont les éléments n'est pas important. Néanmoins, ça le devient au niveau physique, c'est-à-dire quand les index sont en jeu : on peut donc être amené à changer cet ordre si la performance des requêtes en dépend (on change l'ordre en agissant sur la colonne « # ») :

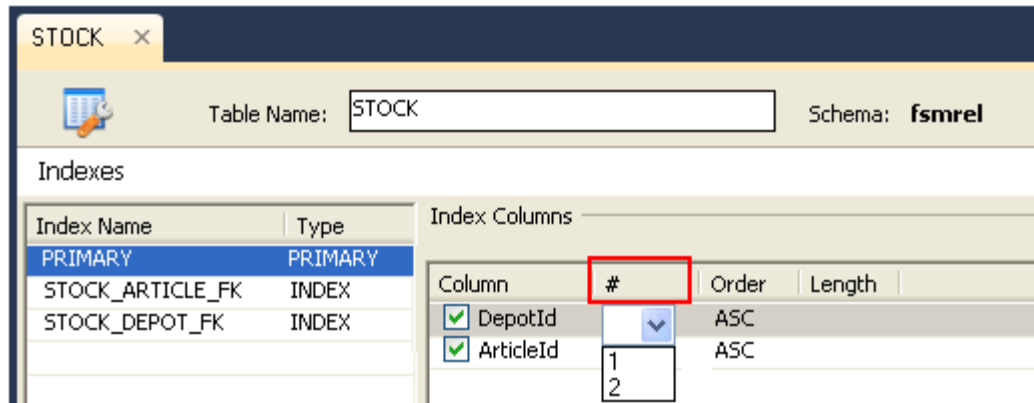


Figure 10.50 - Ordre des colonnes composant les clés

La performance des requêtes n'est pas notre propos, mais notons que l'on peut choisir d'avoir deux index, le 1er selon la séquence {DepotId, ArticleId}, utile par exemple pour les traitements d'inventaires, et le 2e selon la séquence {ArticleId, DepotId}, indispensable pour les traitements de prise des commandes.

## 10.9. Rendre un lien facultatif

Par défaut, MySQL Workbench force les cardinalités minimales (multiplicités en UML) à 1. C'est parfait dans le cas du lien entre COMMANDE et LIGNE\_COMMANDE car une commande comporte au moins une ligne, mais dans le cas du lien entre ARTICLE et LIGNE\_COMMANDE, il faut changer cette cardinalité pour signifier que des articles peuvent ne pas être référencés (provisoirement du moins) par des lignes de commande.

Pour faire passer la cardinalité minimale à 0, double-cliquer sur le lien connectant ARTICLE et LIGNE\_COMMANDE :

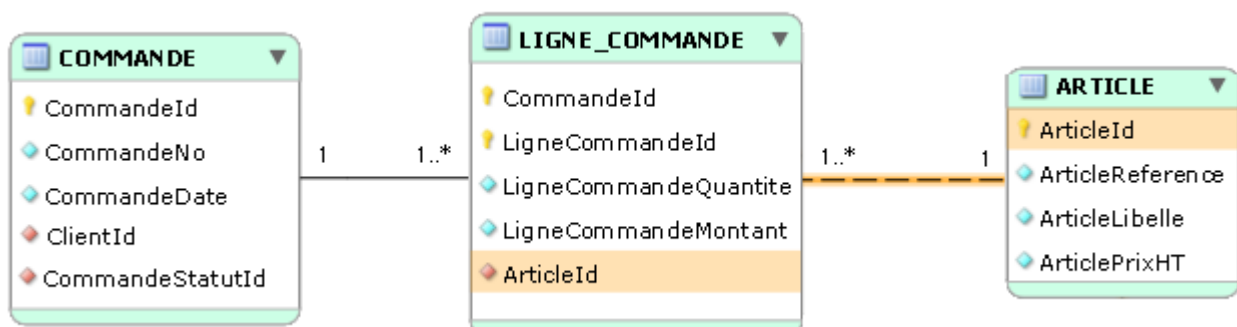


Figure 10.51 - Rendre un lien facultatif (a)



Une fois ouverte la fenêtre qui va bien :

Onglet « Foreign Key » : décocher la case « Mandatory » associée à LIGNE\_COMMANDE (Referencing Table) :

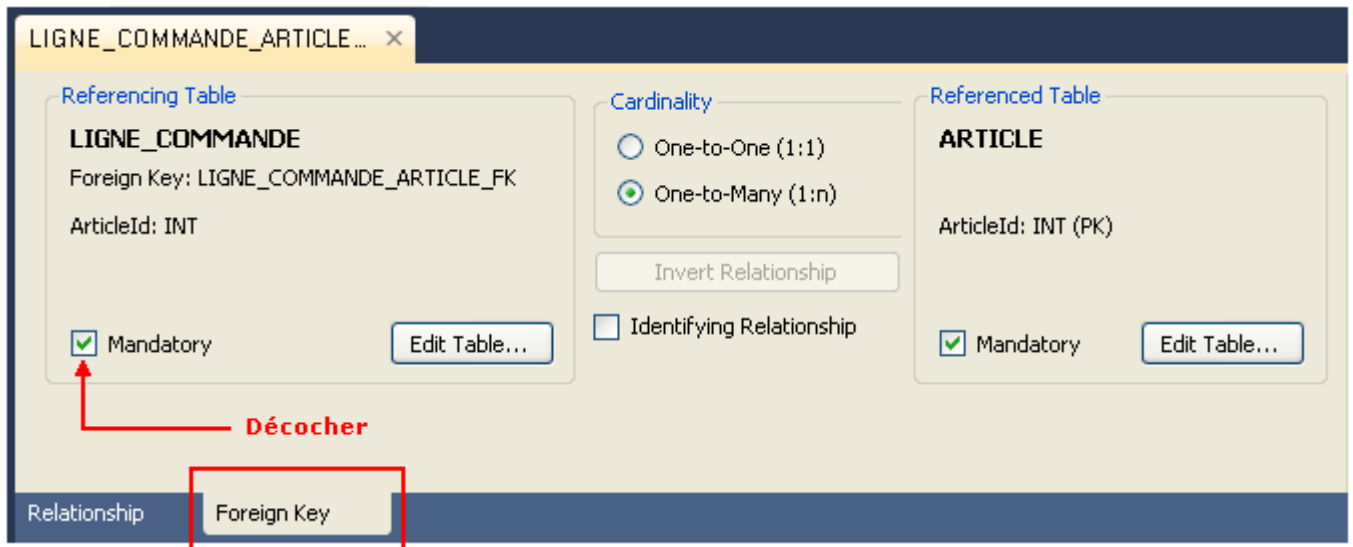


Figure 10.52 - Rendre un lien facultatif (b)

## 10.10. Supprimer un objet (sans s'énerver)

Il arrive que des forumers se lamentent de ne pas pouvoir supprimer un objet : colonne, table, lien entre tables, etc. D'une façon assez générale on peut en passer par un clic droit à la souris sur l'objet à supprimer. Ainsi, pour supprimer une colonne de table, un clic droit sur le nom de la colonne (onglet Columns) donne accès à la commande « Delete Selected Columns » :

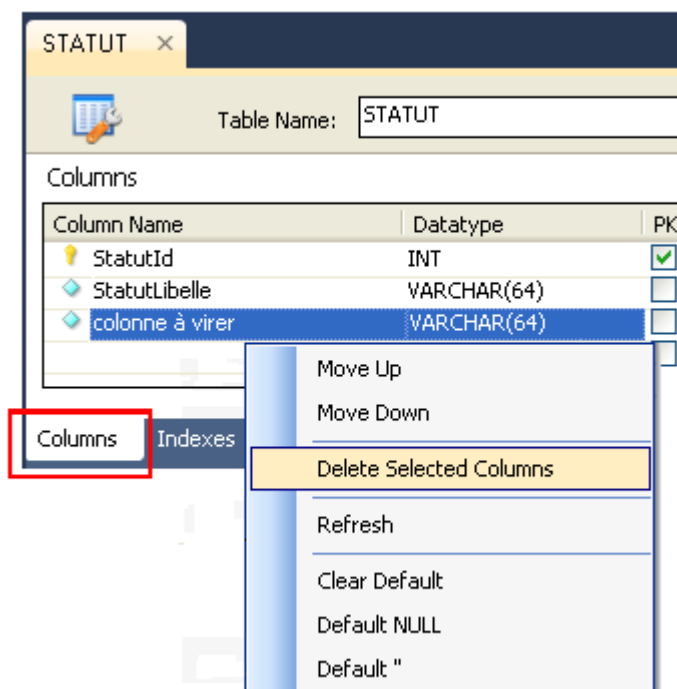


Figure 10.53 - Supprimer une colonne de table

Et bien sûr, en ce qui concerne les tables et les liens qu'elles entretiennent, on peut toujours passer par le menu « Edit ». Ainsi, quand on cherche à établir un lien entre deux tables, alors qu'on est encore un peu novice dans l'utilisation de MySQL Workbench, on crée assez facilement des associations réflexives non voulues et dont on a du mal à se débarrasser :

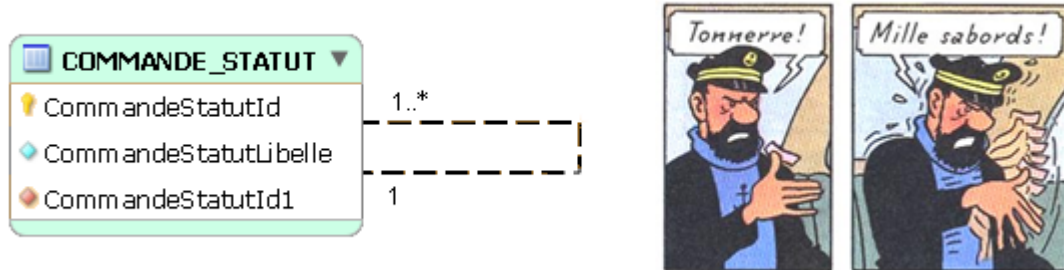


Figure 10.54 - Association réflexive intempestive et collante

Si le lien vient d'être créé, la commande « Annuler » (↶) de la barre d'outils permet de s'en sortir facilement. En tout cas, par un clic gauche sur le lien non voulu pour le sélectionner, puis en passant par le menu « Edit », on accède à la commande « Delete » de suppression :

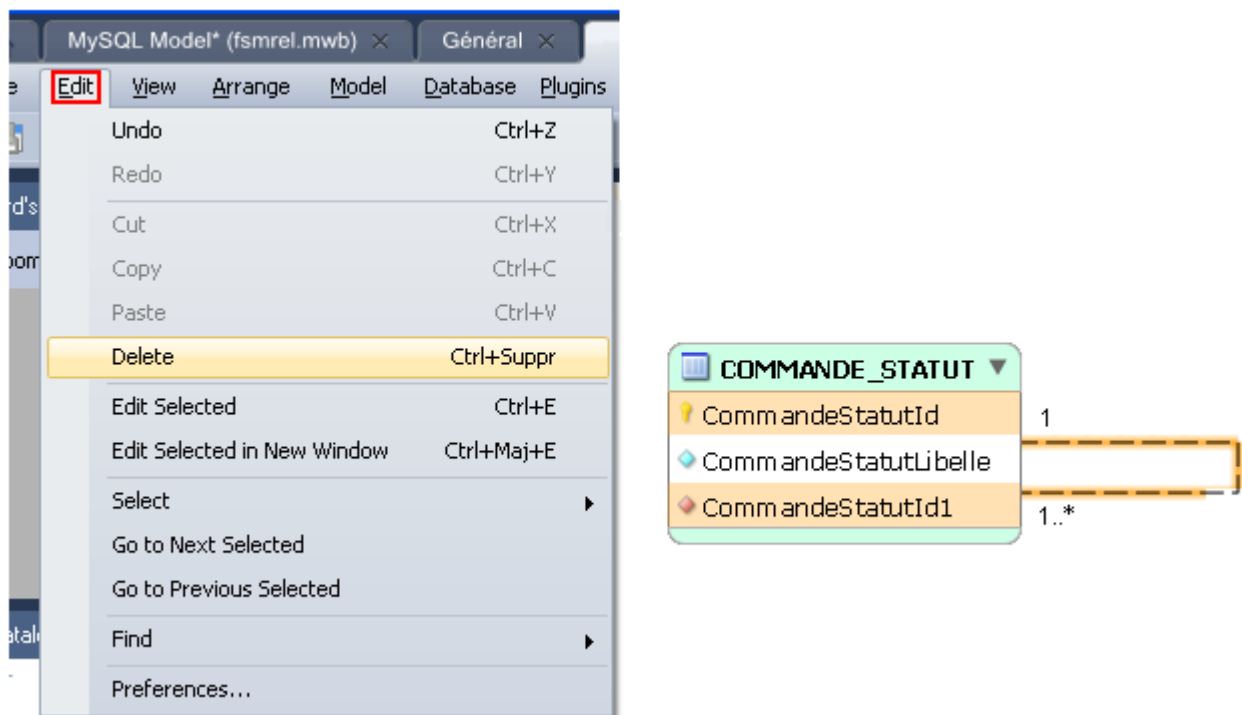


Figure 10.55 - Commande Delete du menu Edit

En sélectionnant la commande « Delete », on provoque l'ouverture de la fenêtre « Delete Foreign Keys » : il ne reste plus qu'à « tuer » le lien (bouton « Delete » de la fenêtre), ce qui provoque aussi la suppression de la colonne `CommandeStatutId1` automatiquement créée par MySQL Workbench pour servir de clé étrangère :

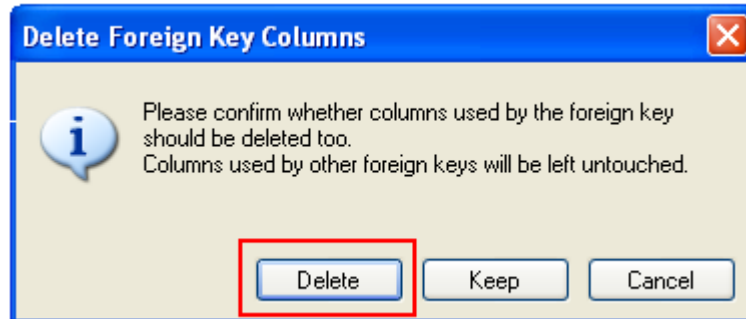


Figure 10.56 - Suppression du lien

Mais bien sûr, un clic droit sur le lien permet d'effectuer la même opération :

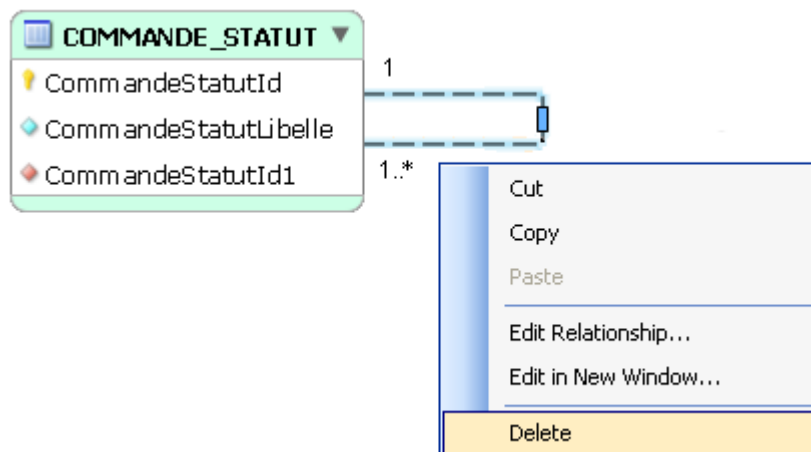


Figure 10.57 - Suppression par clic droit

### 10.11. Afficher le nom des associations entre tables

On peut afficher les noms des associations entre tables. Pour cela, il faut d'abord passer par le menu « Edit » et choisir « Preferences » :

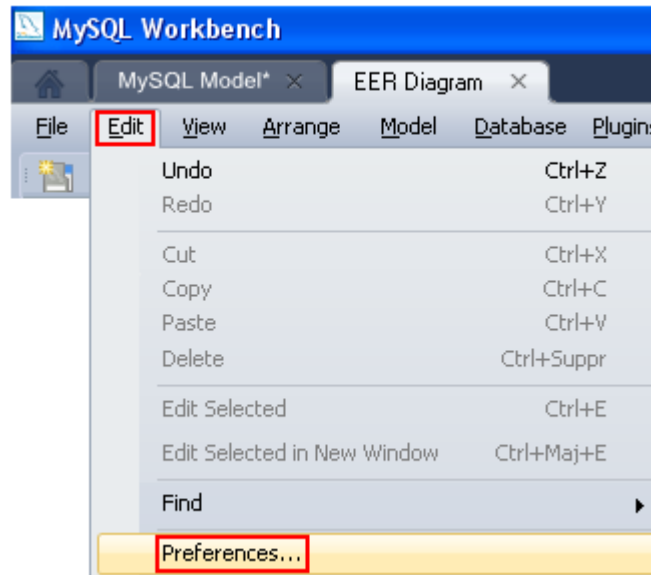


Figure 10.58 - Menu : Edit, Preferences

Puis ouvrir l'onglet « Diagram » et y décocher la case « Hide Captions » :

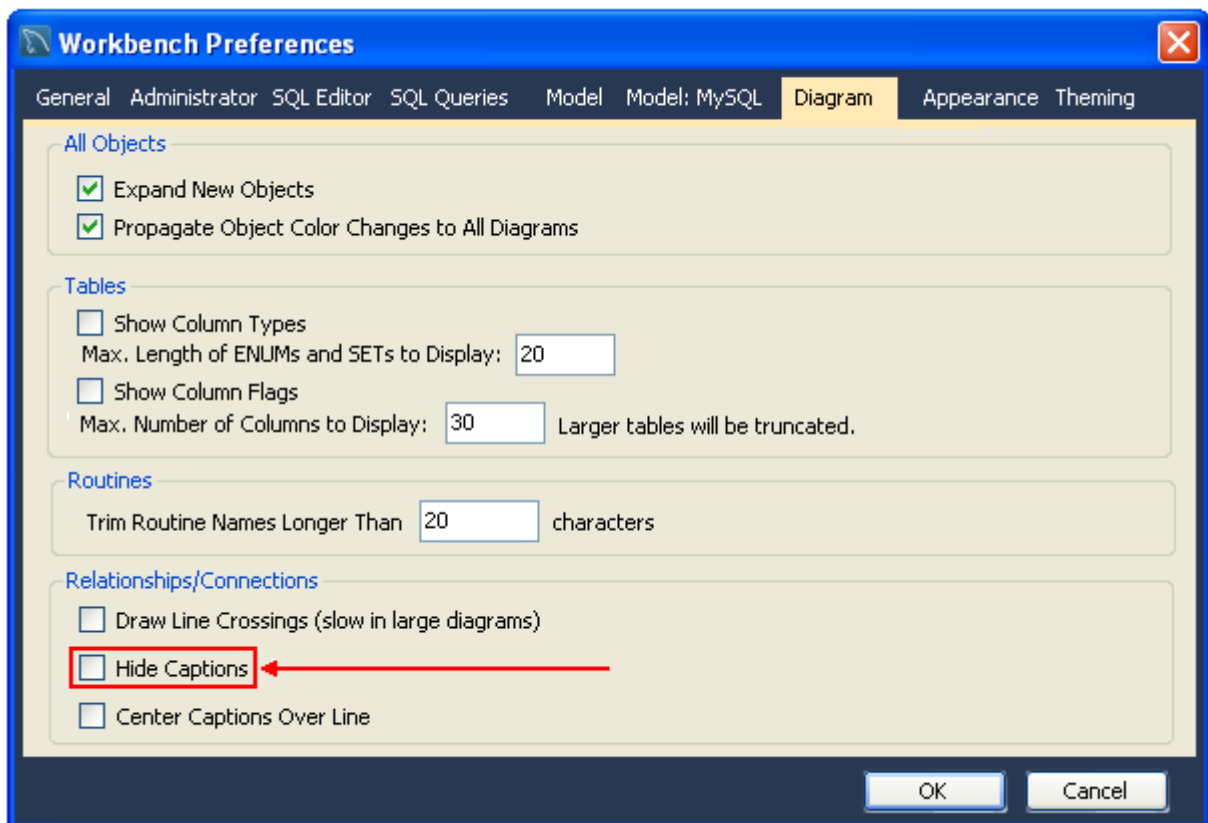


Figure 10.59 - Autoriser l'affichage du nom des liens

Mais alors chaque nom d'association est affiché. Par défaut, il s'agit du nom des clés étrangères :

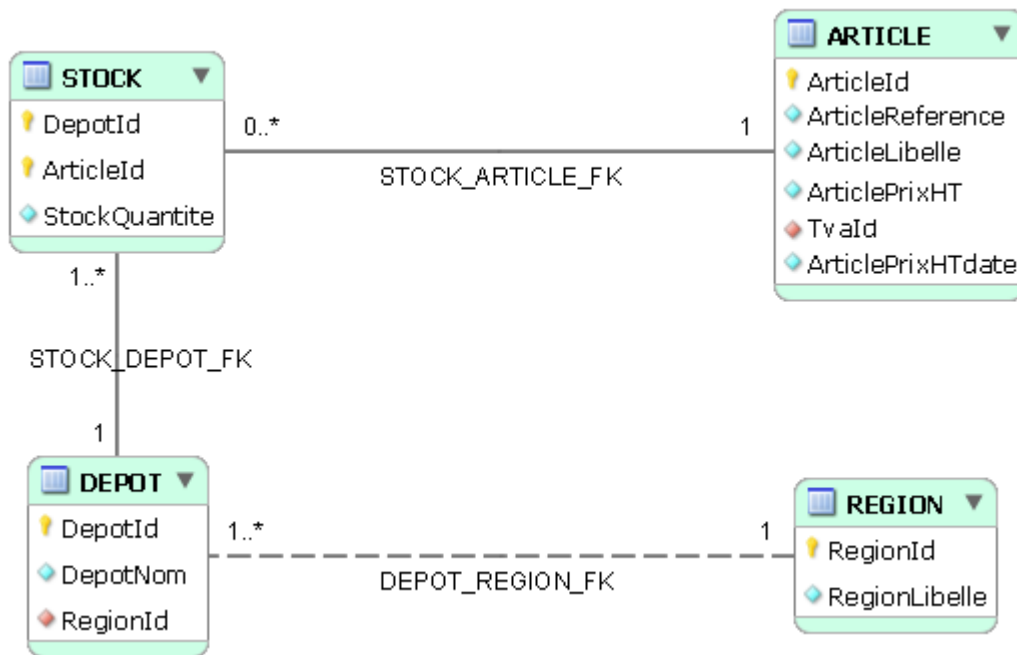
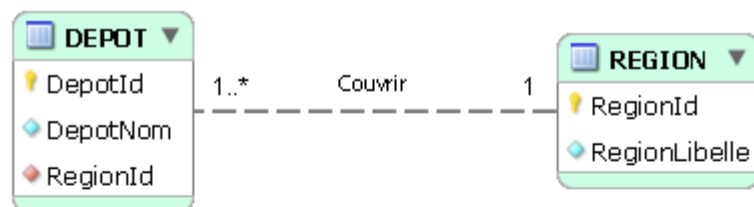


Figure 10.60 - Affichage du nom des associations

Sans modifier pour autant le nom des clés étrangères, on peut afficher autre chose que ce nom. Il suffit pour cela de valoriser chaque cartouche « Caption » de l'onglet « Relationship », voire en effacer le contenu pour ne rien afficher. Par exemple, une région couvre un ou plusieurs dépôts :



Relationship

Caption:
'DEPOT' () 'REGION'

2nd Capt.:

Couvrir

'REGION' (Couvrir) 'DEPOT'

Comments:

Visibility Settings

☒ Fully Visible
☐ Draw Split
☐ Hide

Relationship

Foreign Key

Figure 10.61 - Renommer les associations

## 10.12. MySQL Workbench 5.2 : Accueil

### 10.12.1. Écran d'accueil

L'écran d'accueil de la version 5.2 de MySQL Workbench est moins sinistre que celui de la version 6.0...

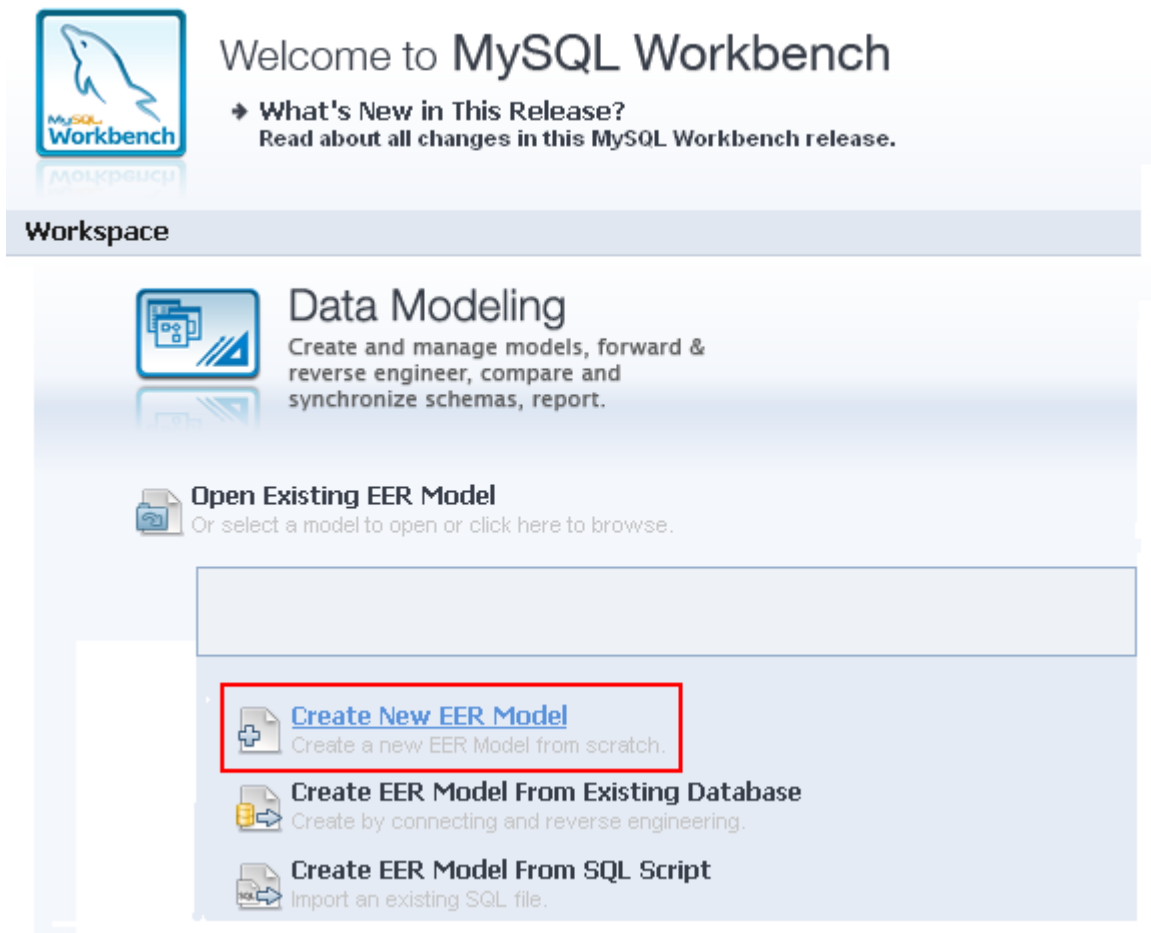


Figure 10.62 - Bienvenue de la part de MySQL Workbench 5.2

### 10.12.2. Créer un nouveau modèle

Il s'agit de créer un nouveau modèle. On sélectionne l'option « Create New EER Model »<sup>1</sup>.

<sup>1</sup> Pour la petite histoire, EER est l'abréviation de « Enhanced Entity-Relationship », c'est-à-dire à peu de choses près que MySQL Workbench se veut une extension du modèle Entité-Relation (E/R). En réalité, l'outil nous permet plutôt de créer des modèles logiques de données (MLD) au sens Merise du terme, car il nous propose par exemple le concept de clé étrangère, qui n'a pas lieu d'être au niveau E/R où il ferait double emploi (violant donc le principe d'essentialité) avec le concept de relation (*relationship*, association) tel que défini en 1976 par Peter Chen, père du modèle (*The Entity-Relationship Model-Toward a Unified View of Data*, dans ACM Transactions on Database Systems, Vol. 1, No. 1. March 1976, Pages 9-36.) En revanche, au niveau logique, le concept de clé étrangère est essentiel. Notons encore que, pour faire partie de la famille des modèles EER « enhanced », MySQL Workbench devrait en plus offrir des fonctionnalités telles que la généralisation/spécialisation (cf. paragraphe 6).

En réponse, MySQL Workbench propose la fenêtre suivante, permettant de créer un MLD (modèle logique de données) sous la forme d'un diagramme :

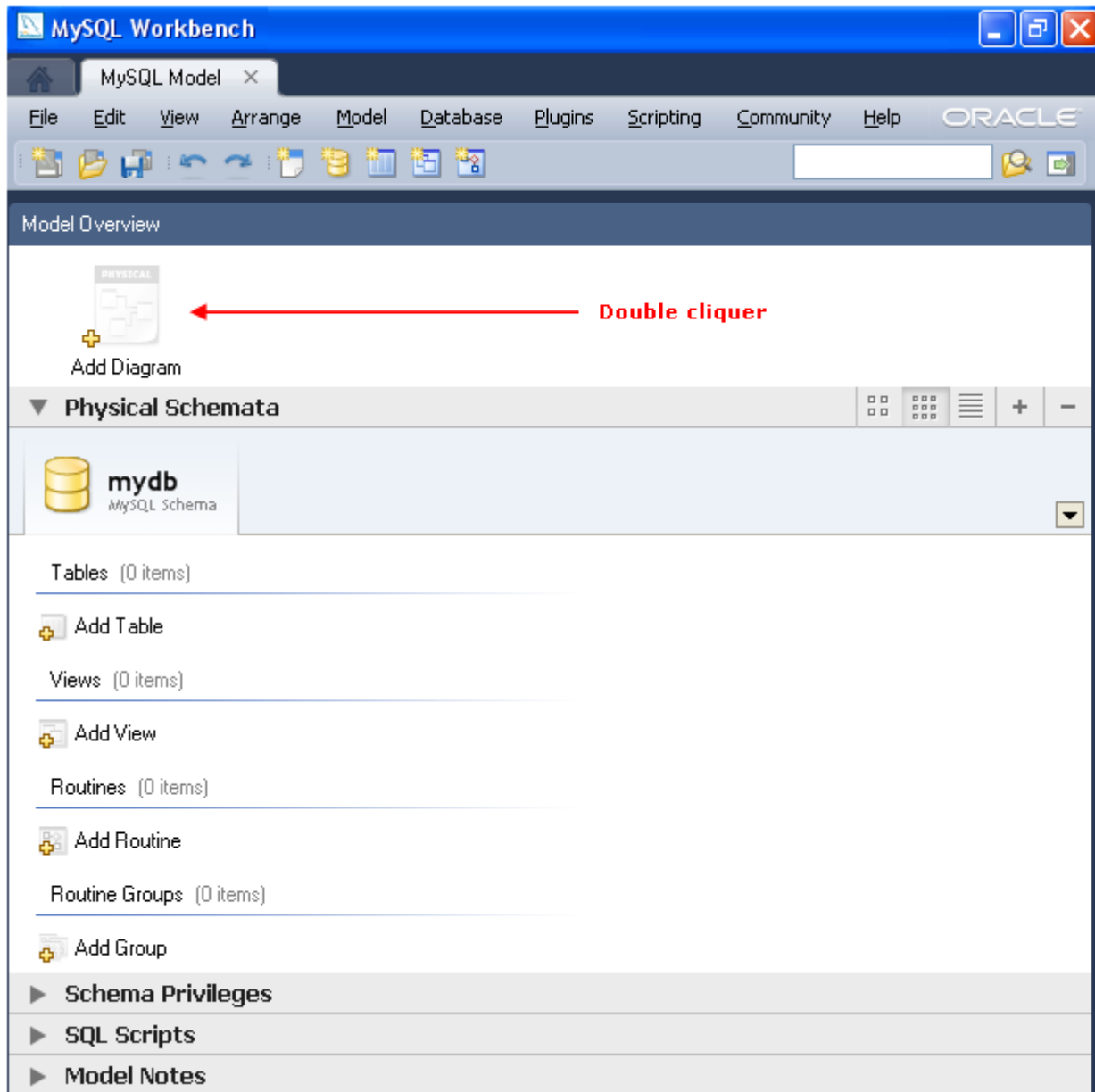


Figure 10.63 - Ajout d'un diagramme pour un MLD

Cette fenêtre est identique à celle qui est proposée par MySQL Workbench version 6 : à partir de là, on procède de la même façon (cf. paragraphe 2.2.2).