Création des tables de faits et de dimension

1. Création de l'entrepôt de données

Nous allons commencer par créer l'entrepôt de données. Nous appellerons cette base de données : *DistrisysDW*.

Pour information, les deux lettres DW sont le sigle de *Data Warehouse*, traduction anglaise d'*Entrepôt de données*.

Pour créer cette base de données nous devons utiliser l'outil SQL Server Management Studio (SSMS).

- Ouvrez la console **SQL Server Management Studio**.
- Créez une nouvelle base de données **DistrisysDW** avec le modèle de récupération **Simple**. En effet, une base de données décisionnelle ne doit pas enregistrer les logs de transaction. D'une part parce que les logs seraient trop volumineux, d'autre part parce que le système de recouvrement au quotidien sera géré par le système d'audit. Pour plus de détails sur ce sujet, reportez-vous au chapitre Alimenter l'entrepôt de données avec SSIS - L'audit des flux ETL.
- Assurez-vous que le compte de service de votre serveur SQL Server a les droits en lecture sur *DistrisysDW*. Si vous ne savez pas comment vous y prendre, vous avez à votre disposition un complément téléchargeable sur la page Informations générales.

Vous venez de créer un entrepôt de données et de vous assurer que le compte de service a bien les droits d'accès sur cette nouvelle base de données. Dans les prochaines étapes, nous allons nous atteler à la création de la table de faits et des dimensions.

2. Création d'une table de faits

D'abord quelques explications sur la construction d'une table de faits. Chaque table de faits sera construite en trois blocs.

Le premier bloc détaille les liaisons avec les tables de dimension :

DateFacturation_FK	int	☐
Site_FK	int	☐
Produit_FK	int	☐
Client_FK	int	☐

Les quatre axes pour analyser les factures sont les suivants :

- *DateFacturation_FK* permettra d'identifier la date de facturation et fera la liaison avec la dimension *Temps.*
- *Site_FK* permettra d'identifier le site de facturation et fera la liaison avec la dimension *Site*.
- *Produit_FK* permettra d'identifier le produit facturé et fera la liaison avec la dimension *Produit*.
- *Client_FK* permettra d'identifier le client facturé et fera la liaison avec la dimension *Client*.

Ces champs définissent la *granularité* de notre table de faits.

Dans notre cas, la granularité de la table de faits *FactFacture* correspond à une ligne : par jour (date de facturation), par site de facturation, par produit et par client. Cela signifie que, potentiellement, nous pourrons regrouper et sommer en une seule ligne, les lignes de facture ayant ces mêmes critères.

Ce regroupement est appelé un *agrégat*.

Veuillez noter que chaque champ de liaison ne tolère pas de valeur null.

Ces champs de liaison seront des clés étrangères. Une table de faits n'a pas de clé primaire, c'est une de ses caractéristiques.

Le second bloc détaille les mesures de la table de faits :

PrixCatalogue	numeric(9, 2)	☐
Remise	numeric(9, 2)	☐
CA	numeric(9, 2)	☐
Marge	numeric(9, 2)	☐
CoutDirectMatiere	numeric(9, 2)	☐
CoutDirectMainOeuvre	numeric(9, 2)	☐
CoutIndirect	numeric(9, 2)	☐
Quantite	numeric(9, 2)	☐

Ces mesures sont issues d'un travail conjoint avec le service contrôle de gestion de Distrisys. La facture est l'occasion de redéfinir les termes et le découpage des différents montants. Suite à l'atelier, nous avons posé les relations suivantes entre ces différentes mesures :

- Prix Catalogue = CA TTC + Remise
- CA TTC = CA HT + TVA
- CA HT = Coût Indirect + Coût Direct Main d'œuvre + Coût Direct Matière + Marge

Les mesures de la table des faits sont tous de type numeric(9,2) afin de gérer les nombres réels compris entre -1 000 000,00 et 1 000 000,00.

La précision de 9 représente le nombre de chiffres total et 2, le nombre de chiffres après la virgule. Pour mieux comprendre le fonctionnement du type numérique, veuillez-vous reporter au tableau ci-dessous :

	Mini	Maxi	Coût en octets
Numeric(9,1)	-10 000 000,0	10 000 000,0	5
Numeric(9,2)	-1 000 000,00	1 000 000,00	5
Numeric(9,3)	-100 000,000	100 000,000	5

Le type *numeric* (9,x) coûte donc 5 octets. Ce type de données représente le stockage de la valeur réelle, le moins coûteux en octets.

Les mesures de la table de faits ne doivent pas accepter de valeur null.

Le troisième bloc liste des champs dits de *dimensions dégénérées* :

NumFacture	varchar(6)	☑

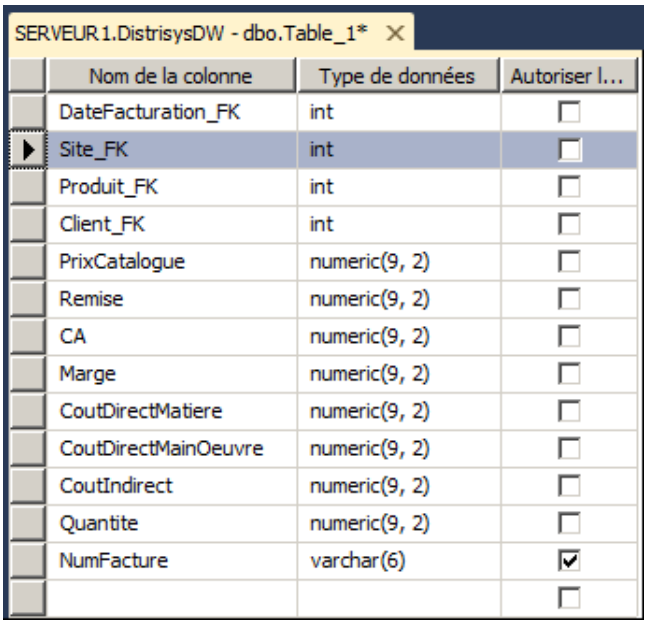
Ces champs n'ont pas d'utilité dans l'analyse. Ils représentent généralement une référence au grain de la table de faits. Ces champs permettront de faire le lien entre le système décisionnel et le système source.

En effet, nous n'analyserons jamais nos factures par le numéro de facture. En revanche, nos utilisateurs souhaiteront peut être connaître la liste des numéros de factures qui compose les ventes du mois d'un produit, pour un client et pour un site en particulier. Nous verrons l'usage concret des dimensions dégénérées au chapitre La modélisation dimensionnelle - Facturation et commande client.

Attention, ces champs sont assez coûteux en espace, car ils sont généralement en type varchar : 1 octet par caractère plus 2 octets. Un varchar(6) coûte donc entre 2 et 8 octets par ligne dans la table de faits. Il ne faut donc pas tomber dans l'excès. La création d'un tel champ doit être pesée.

Nous allons maintenant nous atteler à la création d'une table de faits : FactFacture.

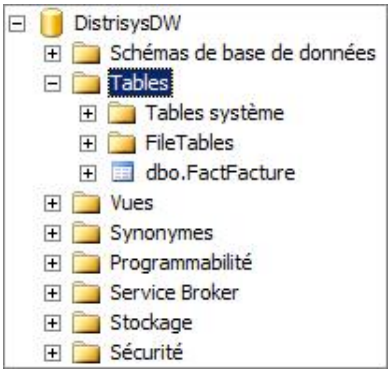
- Créez la table de faits des factures de la manière suivante :



- Enregistrez la table et nommez-la : **FactFacture**.

Toutes les tables de faits de l'entrepôt de données seront préfixées par Fact afin de les identifier comme telles.

La table doit apparaître comme ci-dessous :



3. Création des tables de type dimension

Maintenant que nous avons créé la table de faits des factures, nous allons nous atteler à construire les tables de type dimension utilisées dans l'analyse des factures. De même que les tables de faits sont préfixées par Fact, les tables de type dimension seront préfixées par Dim.

Nous allons donc créer les tables de type dimension suivantes :

- *DimProduit*, pour la dimension produit.
- *DimSite*, pour la dimension site.
- *DimClient*, pour la dimension client.

Commençons par la création de la table dimension *Produit*.

Comme pour la table de faits, quelques explications s'imposent quant à la construction d'une table de type dimension. La table de dimension Produit est construite en deux blocs. Ces deux blocs devront se retrouver dans chacune de nos tables de type dimension.

Le premier bloc identifie le champ de clé technique de la table de dimension *Produit*.

Produit_PK	int	☐

Cette clé technique ne doit pas être issue de votre système source. Elle ne doit pas non plus être une codification métier. Il est important que votre entrepôt de données utilise et gère ses propres identifiants de table de dimension. Nous aurons donc pour chacune de nos tables de dimension, une clé technique de type *int*, en incrémentation automatique.

Le deuxième bloc de colonnes liste les attributs de la dimension *Produit*.

ProduitCode	varchar(10)	☐
Produit	varchar(20)	☐
SousFamilleCode	varchar(10)	☐
SousFamille	varchar(20)	☐
FamilleCode	varchar(10)	☐
Famille	varchar(20)	☐

Nous remarquons que les attributs sont tous de type varchar, pour supporter une valeur sous forme de chaînes de caractères. Le nombre spécifié entre parenthèses correspondant au nombre de caractères maximum du champ.

La dimension *Produit* se décomposera en trois niveaux :

- Le niveau *Famille*.
- Le niveau *SousFamille*.
- Le niveau *Produit*.

Chacun des attributs *Famille*, *SousFamille* et *Produit* est décomposé en deux champs au sein de la table de dimension de l'entrepôt de données.

Le champ suffixé Code (*ProduitCode* par exemple) servira de clé d'identification unique de l'attribut, tandis que l'autre champ (*Produit* par exemple) correspondra à sa désignation : la valeur affichée pour l'utilisateur.

Par exemple, pour le champ ProduitCode LL1100, le champ Produit correspondant est LAGON LL 1100.

Cette façon de procéder est nécessaire dans le cas des attributs disposant déjà d'une codification ou des attributs générant de nombreuses valeurs comme les produits, les clients, les fournisseurs, les actions commerciales…

Créez la table de dimension Produit :

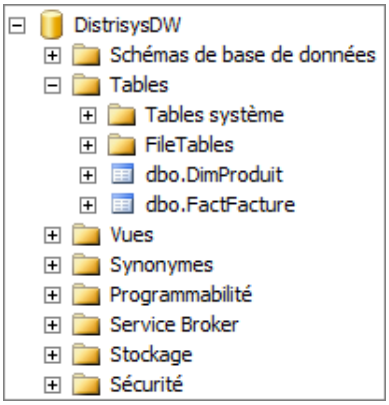
Créez les colonnes de table comme ci-dessous :

SERVEUR1.DistrisysDW - dbo.Table_1*

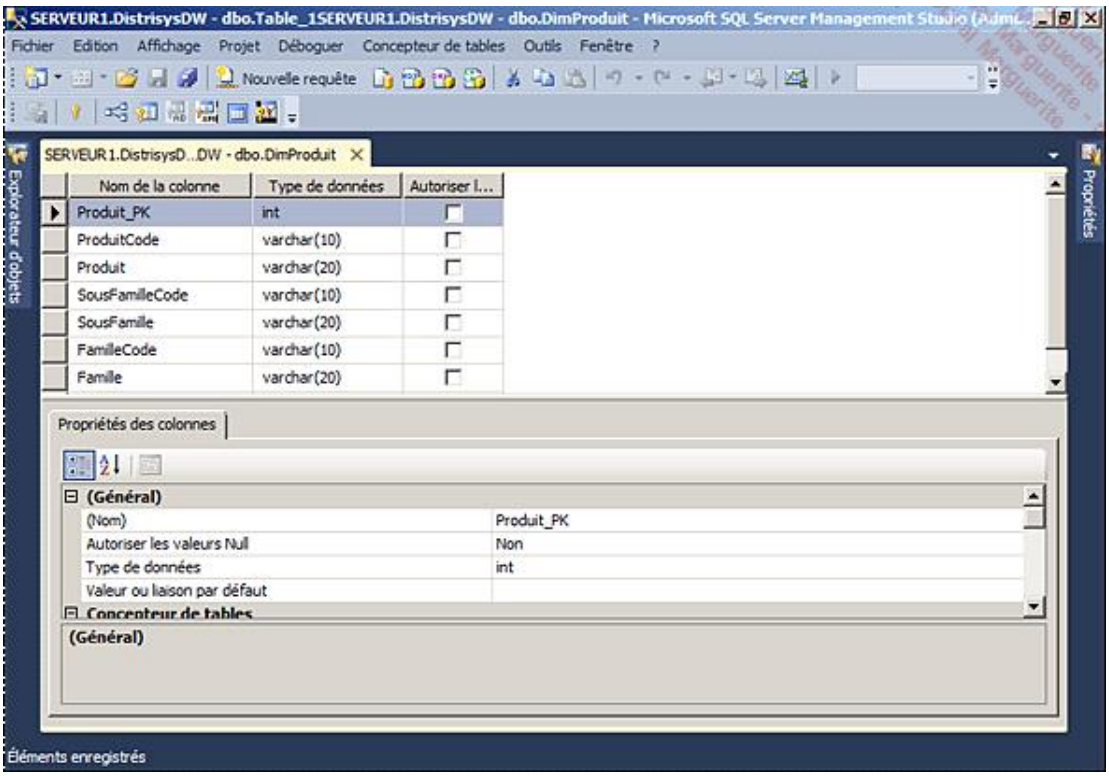
Nom de la colonne	Type de données	Autoriser l…
Produit_PK	int	☐
ProduitCode	varchar(10)	☐
Produit	varchar(20)	☐
SousFamilleCode	varchar(10)	☐
SousFamille	varchar(20)	☐
FamilleCode	varchar(10)	☐
Famille	varchar(20)	☐

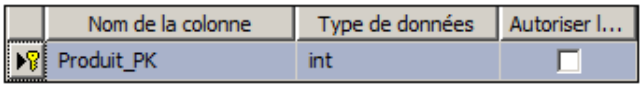
Enregistrez la table en la nommant **DimProduit**.

La table *DimProduit* est créée :



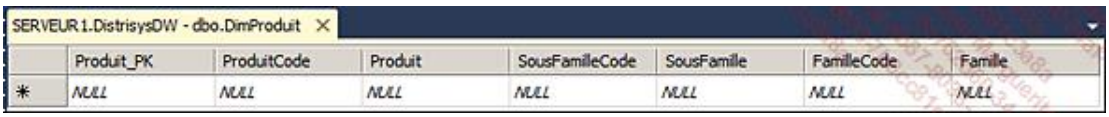
Positionnez une clé primaire sur la première colonne *Produit_PK* et enregistrez les modifications :





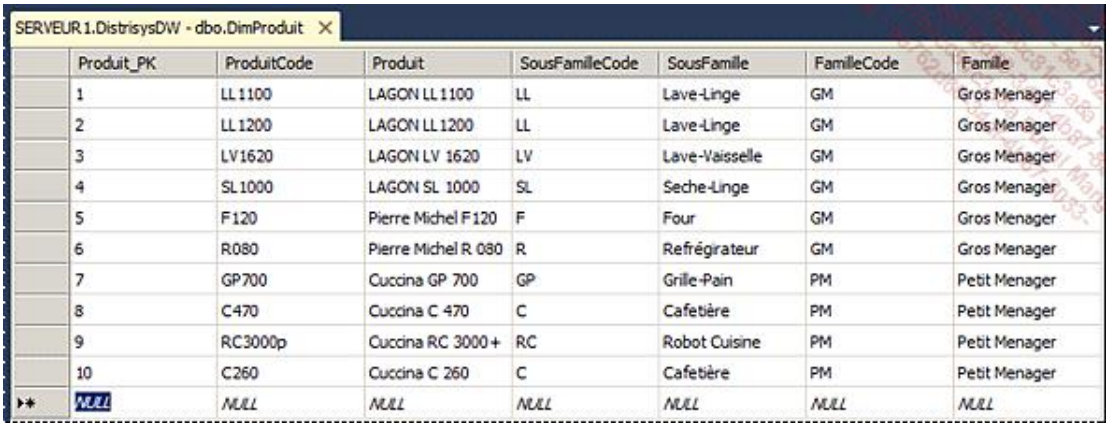
Pensez à enregistrer chacune de vos modifications.

Éditez le contenu de la table *DimProduit* :



La table est actuellement vide.

Saisissez directement dans l'interface les 10 nouvelles lignes comme ci-dessous :

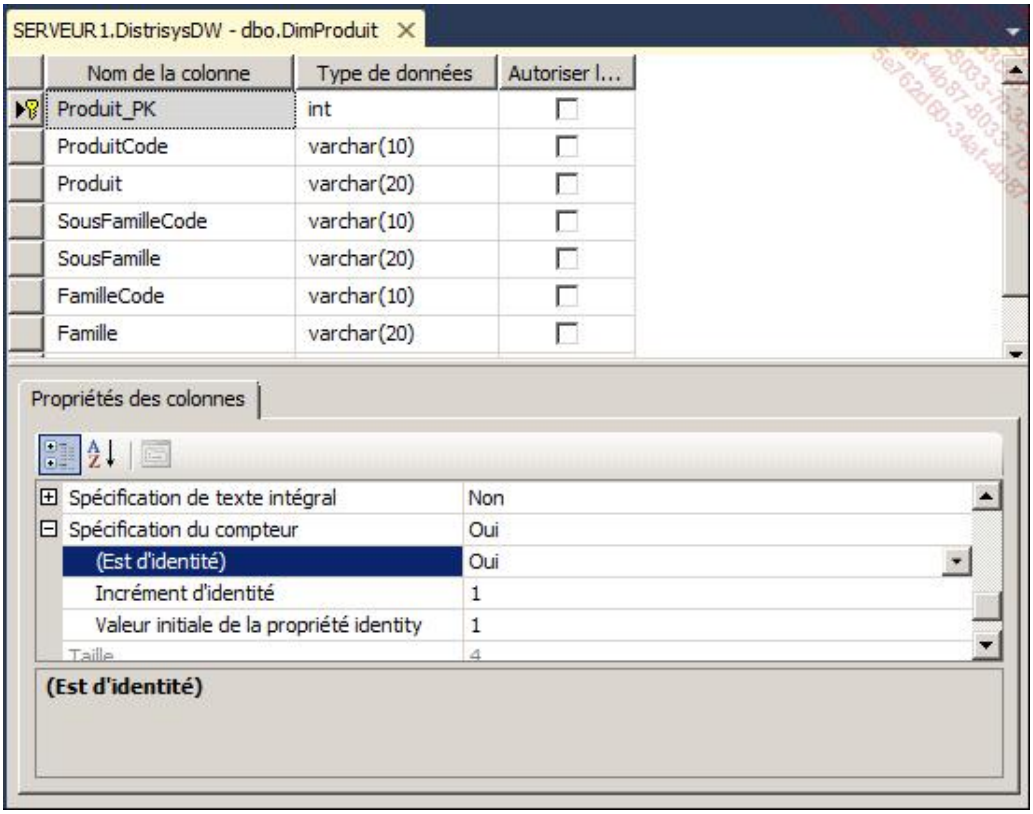


Vous trouverez en téléchargement sur la page Informations générales un script SQL permettant de générer ces

lignes. Le fichier de script se nomme RemplirDimProduit.sql.

Pour les besoins de notre cas, nous venons de saisir manuellement la valeur des données de **Produit_PK**. Néanmoins, il est préférable de laisser SQL Server gérer et générer la valeur de ce champ. Pour ce faire, il faut activer la propriété d'**incrémentation automatique**. Une fois en **incrémentation automatique**, vous ne pourrez plus saisir manuellement la valeur de ce champ. Pour les besoins de notre cas, nous activerons l'**incrémentation automatique** seulement après la saisie des données, pour nous assurer que les clés techniques aient bien les valeurs figurant dans les exemples proposés.

Créez une identité avec *incrémentation automatique*. Pour cela, changez la propriété **(Est d'identité)** à **Oui** en utilisant la liste déroulante.

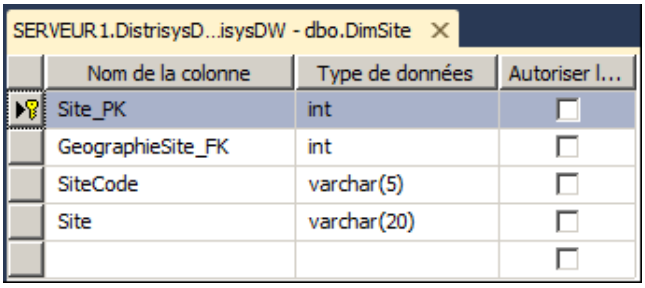


Un peu de vocabulaire :

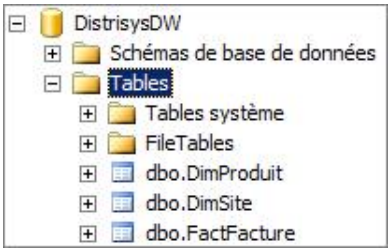
La valeur unique que prend chaque *attribut* est appelée un *membre*. Ainsi dans notre exemple, l'attribut *Produit* dispose de dix membres. De même l'attribut *Famille* dispose de deux membres : *Gros Ménager* et *Petit Ménager*. Le nombre de lignes de la dimension est appelé la *cardinalité* de la dimension. Dans notre exemple, la dimension *Produit* a une cardinalité de 10.

Vous savez maintenant créer une table et vous comprenez la composition d'une dimension.

Créez la table de dimension **DimSite** :



Vous devrez alors avoir :



Vous pouvez constater que la dimension *DimSite* est composée de trois blocs de colonnes distincts. Nous retrouvons les deux blocs obligatoires :

Le premier référence la *clé technique* :

Nom de la colonne	Type de données	Autoriser l...
Site_PK	int	☐

Le second bloc référence les *attributs de dimension* :

SiteCode	varchar(5)	☐
Site	varchar(20)	☐

Mais il existe un troisième bloc, qui liste les liaisons avec d'autres dimensions :

GeographieSite_FK	int	☐

Un peu à la manière de la table des faits, *GeographieSite_FK* est une clé étrangère qui fera la liaison avec une dimension *Geographie* et permettra de localiser le site sur un axe géographique : pays, département, ville…

Pourquoi faire la liaison avec une table de dimension distincte ? Pourquoi ne pas créer les attributs **Pays**, **Département** et **Ville** directement dans cette dimension Site ? Tout simplement, parce que la dimension géographie est un axe qui sera partagé par d'autres dimensions autres que **Site**. L'axe **Geographie** au sein de la dimension **Client** nous permettra de localiser les clients de Distrisys. Nous créerons donc une table **DimGeographie** qui centralisera toutes les informations concernant la localisation géographique.

Créez la table de dimension **DimGeographie** :

Nom de la colonne	Type de données	Autoriser l...
Geographie_PK	int	☐
PaysCode	varchar(10)	☐
Pays	varchar(20)	☐
DepartementCode	varchar(10)	☐
Departement	varchar(20)	☐
CodePostal	varchar(10)	☐
Ville	varchar(20)	☐

Notre dimension *Geographie* dispose donc des attributs nécessaires pour réaliser des analyses par pays, département et ville.

Dans la réalité, nous vous conseillons vivement d'apporter une attention particulière à cette dimension, en y intégrant la région et de descendre jusqu'au code INSEE. Cette dimension est importante car c'est la clé qui vous permettra d'analyser vos données à l'aide d'outils de cartographie.

- Éditez le contenu de la table **DimGeographie** (11 lignes) :

SERVEUR1.DistrisysD...- dbo.DimGeographie | RemplirDimGeographie...VEUR1\sfantini (54))

Geographie_PK	PaysCode	Pays	DepartementC...	Departement	CodePostal	Ville
1	FR	France	13	Bouches-du-Rhône	13001	Marseille
2	FR	France	69	Rhône	69001	Lyon
3	FR	France	75	Paris	75001	Paris
4	FR	France	33	Gironde	33001	Bordeaux
5	FR	France	59	Nord	59001	Lille
6	FR	France	31	Haute Garonne	31001	Toulouse
7	FR	France	67	Bas-Rhin	67000	Strasbourg
8	FR	France	13	Bouches-du-Rhône	13100	Aix-en-Provence
9	FR	France	25	Franche-Comté	25000	Besançon
10	DE	Allemagne	091	Haute-Bavière	58352	Munich
11	ES	Espagne	08	Catalogne	08006	Barcelone
NULL	NULL	NULL	NULL	NULL	NULL	NULL

- Éditez le contenu de la table **DimSite** (5 lignes) :

SERVEUR1.DistrisysDW - dbo.DimSite | RemplirDimSite.sql -...ERVEUR1\sfantini (54))

Site_PK	GeographieSite...	SiteCode	Site
1	3	D001	Siège social
2	8	D002	Agence Sud
3	4	D003	Agence Ouest
4	11	D004	Agence Europe Sud
5	10	D005	Agence Europe Est
NULL	NULL	NULL	NULL

Pour faciliter la saisie, vous trouverez le script de création des données en exemple, en le téléchargeant à partir de la page Informations générales. Les scripts se nomment RemplirDimGeographic.sql et RemplirDimSite.sql.

Au vu de ces exemples, il faut donc lire que l'agence Sud se situe à Aix-en-Provence. En effet, l'agence Sud fait référence à GeographieSite_FK qui est égal à 8. Et la table DimGeographie nous apprend que l'identifiant 8 fait référence à la ville d'Aix-en-Provence.

Finissons enfin par la création de la dimension *client*.

- Créez la table de dimension **DimClient** :

SERVEUR1.DistrisysD...ysDW - dbo.DimClient

Nom de la colonne	Type de données	Autoriser l...
Client_PK	int	☐
GeographieClient_FK	int	☐
ClientCode	varchar(10)	☐
Client	varchar(20)	☐
TypeClient	varchar(20)	☐
SegmentationClient	varchar(20)	☐

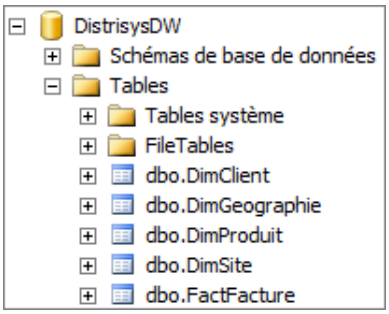
Les attributs **TypeClient** et **SegmentationClient** ne comprendront que très peu de valeurs et ne disposeront pas de codification métier. C'est pour cela que l'attribut avec le suffixe **Code** n'est pas nécessaire. Nous vous conseillons d'éviter toutefois autant que possible les attributs sans codification.

Éditez le contenu de la table **DimClient** (10 lignes) :

SERVEUR1.DistrisysDW - dbo.DimClient | RemplirDimClient.sql...RVEUR1\sfantini (55))

Client_PK	GeographieClie...	ClientCode	Client	TypeClient	SegmentationClient
1	1	C1	LaBoutiqueOnLine.com	Site Marchand	Bon Client
2	3	C2	Maison Discount	Discounteur	Bon Client
3	8	C3	Cuisine du sud	Spécialiste	Tiède
4	4	C4	Discount plus	Discounteur	Tiède
5	2	C5	EquiperSaMaison.com	Site Marchand	Très Bon Client
6	3	C6	Hypermarché Youpi	Grande surface	Très Bon Client
7	10	C7	EineKüche	Spécialiste	Bon Client
8	11	C8	Mercado Del Sol	Grande surface	Bon Client
9	1	C9	ElectroYoupa	Spécialiste	Bon Client
10	5	C10	Toutmoinscher.com	Site Marchand	Tiède
NULL	NULL	NULL	NULL	NULL	NULL

Au final, au sein de Management Studio, vous devriez avoir :



N'oubliez pas de spécifier l'incrémentation automatique des tables *DimSite*, *DimClient* et *DimGeographie*.

Notre entrepôt de données se dessine lentement. Malgré tout, il nous reste une dernière dimension essentielle à créer. Il s'agit de la dimension *Temps*. L'importance est telle que nous y consacrerons la partie suivante.