TP 1 : Magasin

Contexte d'étude "Magasin" L'entreprise que nous allons étudier est un commerce de proximité de type "supérette" organisé en rayons contenant des articles de diverses marques.

Lors du passage en caisse, on conserve la date, le détail des articles achetés ainsi que le montant total qui a été réglé pour cet achat.

On conserve également les informations concernant les fournisseurs et les livraisons. Pour chaque livraison, on mémorise le fournisseur ainsi que le ou les lots qui ont été livrés. Chaque lot concerné contient un seul article.

Un lot est identifié par son numéro de lot et par le code-barres de l'article qu'il concerne.

Un article possède un identifiant unique, le codebarres.

C'est son appartenance à un lot qui permet de retrouver les informations qui le concernent (dates de fabrication, de livraison et de péremption).
Il peut y avoir par ailleurs quatre articles ayant le même codebarres, mais des numéros de lot différents et donc des dates de péremption différentes.

Le MLD de la base magasin permet de visualiser les liens entre les différentes tables.

Par exemple, la table ***FOURNISSEUR*** est liée par sa **clé primaire (idFournisseur)** avec la **table LOT** qui elle contient une **clé étrangère *idFournisseur***.

C'est clé sont des champs (ou colonnes) des tables.

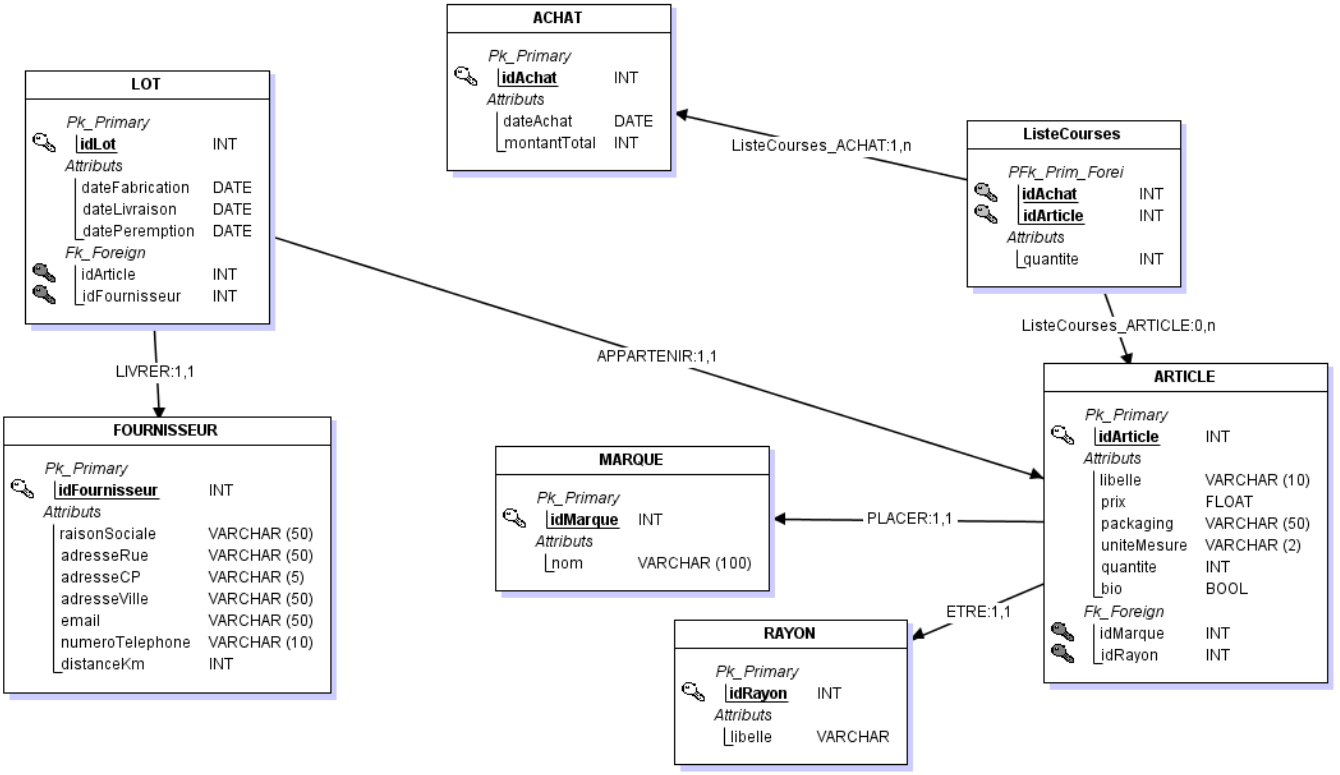

Schéma relationnel de la base de données "magasin"

```
FOURNISSEUR (idFournisseur, raisonSociale, adresseRue, adresseCP,
             adresseVille, email, numeroTelephone, distanceKm)
   idFournisseur : Clé primaire

LOT (idLot, idArticle, dateFabrication, dateLivraison,
     datePeremption, idFournisseur)
   idLot, idArticle : Clé primaire
   idArticle : Clé étrangère en référence à idArticle de article
   idFournisseur : Clé étrangère en référence à idFournisseur de
                   fournisseur

ARTICLE (idArticle, libelle, prix, packaging, uniteMesure,
         quantite, bio, idMarque, idRayon)
   idArticle : Clé primaire
   idMarque : Clé étrangère en référence à idMarque de marque
   idRayon : Clé étrangère en référence à idRayon de rayon

RAYON (idRayon, libelle)
   idRayon : Clé primaire

MARQUE (idMarque, nom)
   idMarque : Clé primaire

LISTECOURSES (idArticle, idAchat, quantite)
   idArticle, idAchat : Clé primaire
   idArticle : Clé étrangère en référence à idArticle de article
   idAchat : Clé étrangère en référence à idAchat de achat

ACHAT (idAchat, dateAchat, montantTotal)
   idAchat : Clé primaire
```

Un schéma relationnel est constitué par l'ensemble des schémas de relation

Dictionnaire de données de la base Magasin

C'est une étape intermédiaire qui peut avoir son importance, surtout si vous êtes plusieurs à travailler sur une même base de données, d'un volume conséquent.

Le dictionnaire des données est un document qui regroupe toutes les données que vous aurez à conserver dans votre base .

On y retrouve souvent :

-Code mnémonique : libéllé désignant une données (ex; titre_f , pour le titre d'un film)
-La designation : il s'agit d'une mention décrivant ce à quoi la données correspond (par exemple "titre d'une film").
-Le type de données (exemple ,date, varchar, numeric…) et la taille

A. Introduction au SELECT

Le choix des colonnes

Dans la zone SQL, vous pouvez voir que, lorsqu'il y a plusieurs champs à afficher, on sépare leurs noms par une virgule.

```
SELECT fournisseur.raisonSociale, fournisseur.email
FROM fournisseur
```

En SQL, un champ est identifié **par son nom et par la table** où il se situe, ce qui veut dire que, lorsqu'on utilise un champ, on doit normalement préciser la table à laquelle il appartient en préfixant le nom du champ par le nom de la table.
Dans l'exemple précédent, le moteur d'exécution SQL n'a pas besoin de ces « précisions » car on utilise qu'une seule table (fournisseur) et que dans cette table , il n'y a **qu'un seul champ « raisonSociale »** et **qu'un seul champ « email »**. Il n'y a donc pas de confusion possible.

Dans la zone SQL , modifiez directement la requête pour lui retirer les préfixes de noms de tables et réexécutez-la. Le résultat restera le même.

```
SELECT raisonSociale, email
FROM fournisseur
```

Prenons le cas où 2 attributs portent le même nom, dans des tables distinctes (ici on souhaite obtenir la liste des articles avec le rayon où ils sont placés) :

Sans préfixe
(attention, cette requête provoque une erreur[10])

```
SELECT libelle, libelle
FROM article, rayon
WHERE…
```

Avec préfixe

```
SELECT article.libelle,
       rayon.libelle
FROM article, rayon
WHERE…
```

Deux attributs portant le même nom doivent être préfixés par le nom de la table d'origine pour éviter toute ambigüité quant à leurs tables d'appartenance

Le caractère * ("étoile") : L'étoile est un caractère qui désigne "tout".

```
SELECT *
FROM article
```

Il faut toutefois utiliser ce "joker" avec prudence. En fonction de la structure de certaines tables, le résultat produit **peut s'avérer gigantesque et très consommateur de ressources systèmes ralentissant ainsi le serveur.**

Le mot clé "DISTINCT"

On souhaite afficher le numéro des fournisseurs qui nous ont déjà été fourni (idfournisseur présent dans la table "lot") :

Lorsque le moteur d'exécution SQL construit un résultat, par défaut, il prend toutes les lignes qui satisfont à la requête demandée, **y compris les lignes en double**. Ce qui nous donne en SQL, sans DISTINCT :

```
SELECT idFournisseur
FROM lot
```

Le mot clé DISTINCT, utilisé avec l'instruction SELECT, a pour **effet de ne pas afficher de doublons** (lignes en double) :

```
SELECT DISTINCT idFournisseur
FROM lot
```

Le mot clé "AS"

Le SQL nous permet d'**améliorer la lisibilité d'une requête en créant un "alias" de colonne**, c'est-à- dire un nom de substitution ou un pseudonyme.

Dans la zone SQL, on peut voir que la clause "AS" a été ajoutée après chaque champ pour spécifier un alias :

```
SELECT raisonSociale AS "Nom",
       adresseRue AS "Adresse",
       adresseCP AS "Code Postal",
       adresseVille AS "Ville"
FROM fournisseur
```

Les guillemets (simple ' Mon alias ') permettent de renommer les champs avec des mots avec des espaces ou d'utiliser des mots clé habituellement réservé par le SGBDR

L'alias de table.

Pour éviter une ambiguïté au niveau des champs, il est possible d'utiliser des **alias** de table.

```
SELECT f.raisonSociale, f.adresseRue FROM fournisseur f
```

Maintenant notre table fournisseur à comme alias "f" et il est possible de **préfixer** nos champs avec celui-ci. (voir exemple ci-dessus).

Exercice 1 :

Donnez **toutes** les clés primaires et **toutes** les clés étrangéres des tables de la base de données **magasin.**

Exercice 2

Réaliser les requêtes permettant d'obtenir les informations ci-dessous.

- La liste des fournisseurs ("n°F" doit apparaître à la place de "idFournisseur" et "Nom" à la place de "raison Sociale").
- La liste des références d'articles qui ont déjà été sur des listes de courses ("Code barre" doit apparaître à la place de "idArticle").
- La liste des marques ("m" doit être utilisée comme alias de la table "marque". Utilisez également le préfixage des champs).
- la liste des achats : "N°" doit apparaître à la place de "idAchat", "Date" à la place de "dateAchat" et "Prix à payer" à la place de "montantTotal", utilisez "a" comme alias de la table "achat", utilisez le préfixage des champs.