**Dossier d'analyse**

**Site Web Cava (application B2C)**

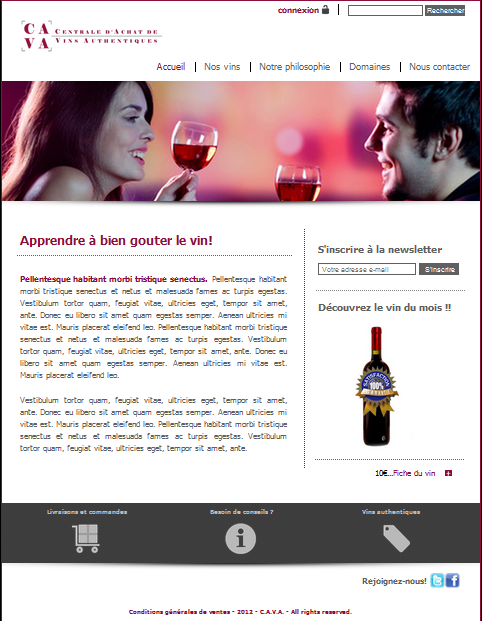
**Objectifs**

Migrer un site statique en site dynamique avec gestion Database

Site réalisé par un étudiant de la formation WebDesigner (maquette HTML + CSS) IramPs

**HomePage** du site

Vente de vins avec un site B2C



Les objets importants

a) lien pour se connecter comme Client

b) lien (cadenas) pour vérifier si variable session ou pas cadenas ouvert ou fermé)

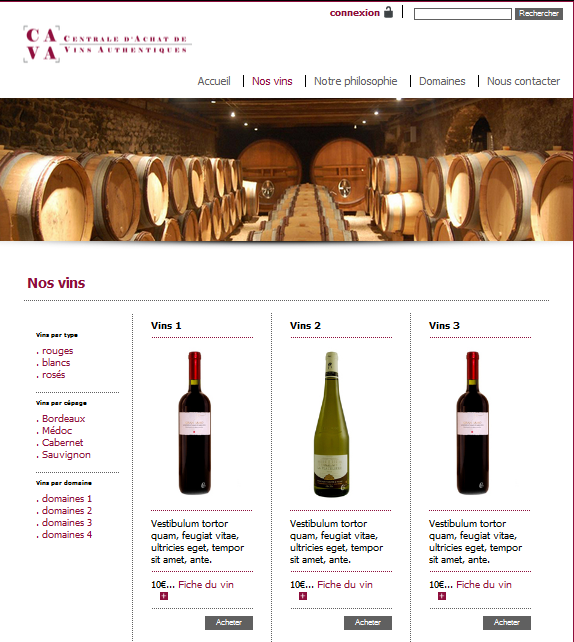
c) zone pour réaliser une recherche simple pour le vin (réaliser une interaction type ajax pour aider au choix du vin)

d) zone pour s'inscrire à la NewsLetter

**manque une icône (caddie) pour afficher le panier directement 'selon l'icône caddie vide ou plein)**

**manque possibilité à un client d'aller dans sa partie gestion … (à afficher si connecter)**

**Pagination Maître-Esclave**



**Revoir affichage lien de navigation, pour type de vin, cépage et domaines**

**Revoir le positionnement de l'affichage des vignettes**

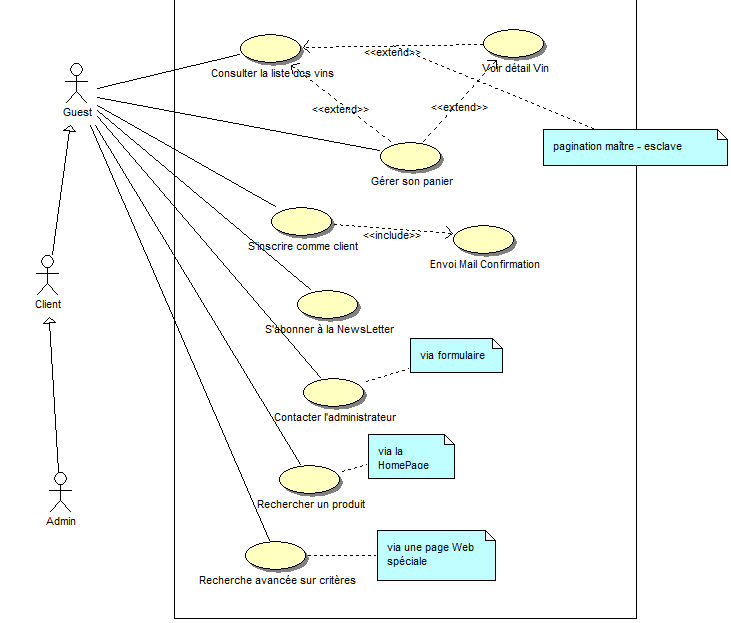
**Barre de navigation : position et choix affichage**

**Prévoir la gestion facile du panier de commande …**

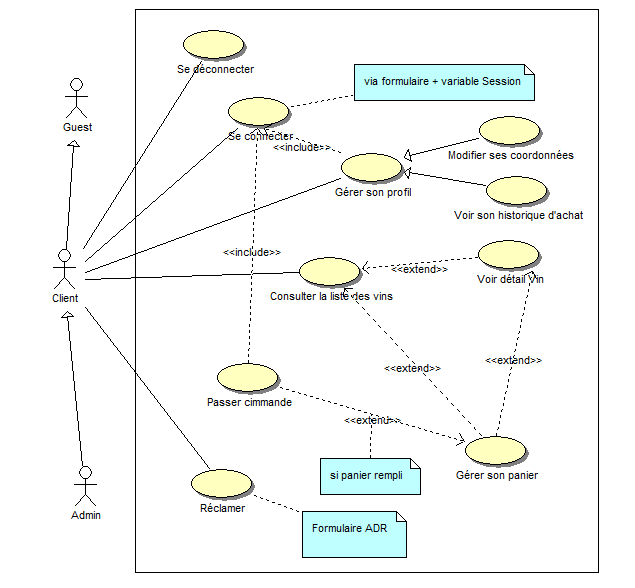
**Les Différents diagrammes Cas d'utilisation**

Les acteurs et différents UseCases (plusieurs diagrammes si nécessaire)

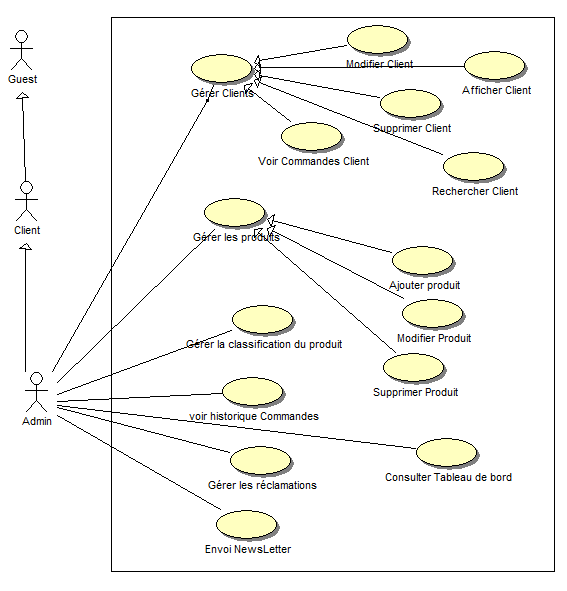
**Pour l'internaute Lambda**



**Pour le client**

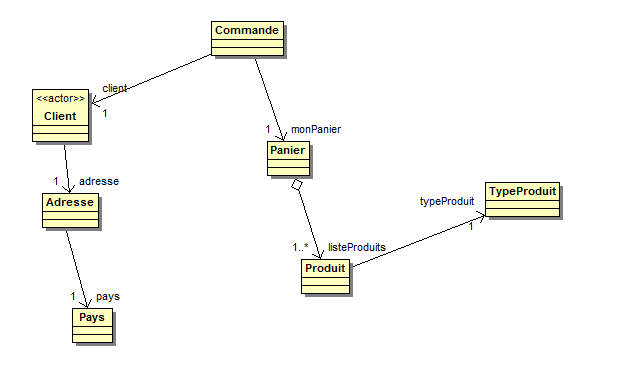


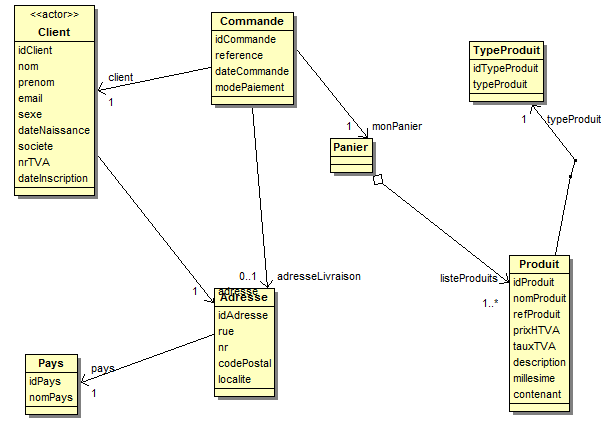
**Pour l'administrateur**



**Diagramme de classes**

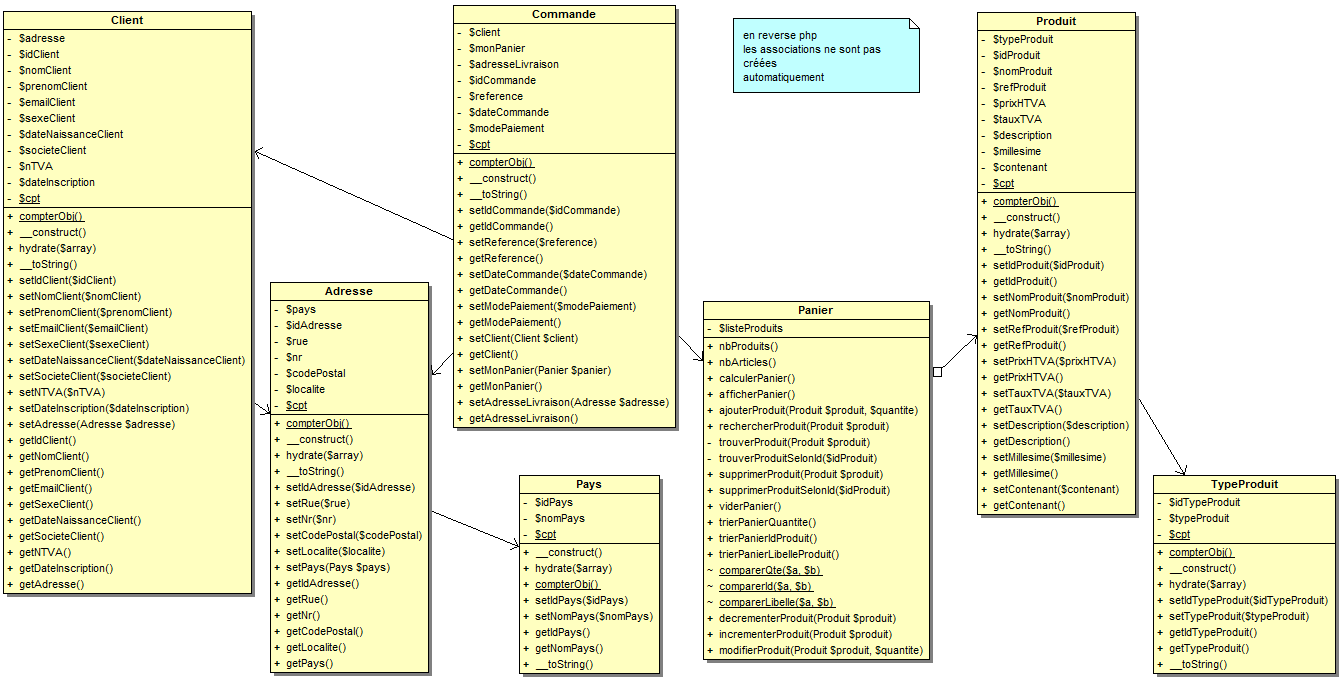
Ebauche Les différentes entités





Exporter en fichiers PHP dans un dossier classes du site Web

**Le diagramme complet**



**La classe la plus importante: la classe Panier**

Redéfinition des méthodes utilitaires pour gérer le panier …

<?php

require\_once 'Produit.php';

class Panier {

private $listeProduits = array(); // tableau 2 dim

// retourne un integer nombre de produits dans le panier

public function nbProduits(){

return count($this->listeProduits);

}

// retourne le nombre d'articles

public function nbArticles(){

$nbArticles = 0;

foreach($this->listeProduits as $value){

$nbArticles += $value[1];

}

return $nbArticles;

}

// return float

public function calculerPanier(){

$montant = 0; //float

foreach($this->listeProduits as $value){

$p = $value[0]; // objet produit

$q = $value[1]; // quantité

$montant += $p->getPrixHTVA() \* $q \*

(1 + $p->getTauxTVA());

}

return $montant;

}

// return string

// utiliser la fonction nl2br pour remplacer \n par la balise <br/>

public function afficherPanier(){

$txt = "";

foreach($this->listeProduits as $value){

$p = $value[0]; // objet produit

$q = $value[1]; // quantité

$txt .= $q . " unités x " . $p->getNomProduit() . " à " .

$p->getPrixHTVA() . " euros\n";

}

return $txt;

}

// return boolean

public function ajouterProduit(Produit $produit,$quantite){

// vérifier si le produit est absent du panier

$this->listeProduits[] = array($produit,$quantite);

return true;

}

// return boolean

public function rechercherProduit(Produit $produit){

for ( $i=0; $i < count($this->listeProduits);$i++ ) {

if ( $this->listeProduits[$i][0] === $produit ){

return true;

}

}

return false;

}

// return la position dans le tableau 2 dimensions

// retourne -1 si pas trouvé

private function trouverProduit(Produit $produit) {

$pos = -1;

for ( $i=0; $i < count($this->listeProduits);$i++ ) {

if ( $this->listeProduits[$i][0] === $produit ){

// === compare la valeur et le type

$pos = $i;

break;

}

}

return $pos;

}

// return la position en lui passant un id (integer)

// retourne -1 si pas trouvé

private function trouverProduitSelonId($idProduit) {

$pos = -1;

for ( $i=0; $i < count($this->listeProduits);$i++ ) {

if ( $this->listeProduits[$i][0]->getIdProduit() === $idProduit ){

// === compare la valeur et le type

$pos = $i;

break;

}

}

return $pos;

}

// return boolean

public function supprimerProduit(Produit $produit){

if(($pos = $this->trouverProduit($produit)) != -1) {

unset($this->listeProduits[$pos]); // supprime la ligne

array\_splice($this->listeProduits,$pos,0); // réaffecter l'index

return true;

}

return false;

}

// return boolean

public function supprimerProduitSelonId($idProduit){

if(($pos = $this->trouverProduitSelonId($idProduit)) != -1) {

unset($this->listeProduits[$pos]); // supprime la ligne

array\_splice($this->listeProduits,$pos,0); // réaffecter l'index

return true;

}

return false;

}

// return string

public function viderPanier(){

$this->listeProduits = array();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// selon le libellé, selon le nr référence ou autre possibilité

public function trierPanierQuantite(){

usort($this->listeProduits,array("Panier","comparerQte"));

}

public function trierPanierIdProduit(){

usort($this->listeProduits,array("Panier","comparerId"));

}

public function trierPanierLibelleProduit(){

usort($this->listeProduits,array("Panier","comparerLibelle"));

}

// créer les fonctions de comparaison

// fonction static qui doit retourner -1,0 ou 1

static function comparerQte($a,$b){

if ( $a[1] == $b[1] ) {

return 0;

}

else {

return ($a[1]<$b[1])?-1:1;

}

}

static function comparerId($a,$b){

if ( $a[0]->getIdProduit() == $b[0]->getIdProduit() ) {

return 0;

}

else {

return ($a[0]->getIdProduit()<$b[0]->getIdProduit())?-1:1;

}

}

static function comparerLibelle($a,$b){

if ( $a[0]->getNomProduit() == $b[0]->getNomProduit() ) {

return 0;

}

else {

return ($a[0]->getNomProduit()<$b[0]->getNomProduit())?-1:1;

}

}

// pour decrementer la quantité de 1

public function decrementerProduit(Produit $produit){

if(($pos = $this->trouverProduit($produit)) != -1) {

if ( ($this->listeProduits[$pos][1] - 1) > 0 )

$this->listeProduits[$pos][1] = $this->listeProduits[$pos][1] - 1;

if ( ($this->listeProduits[$pos][1] - 1) == 0 )

$this->supprimerProduit($produit);

return true;

}

return false;

}

// pour incrémenter la quantité de 1

public function incrementerProduit(Produit $produit){

if(($pos = $this->trouverProduit($produit)) != -1) {

$this->listeProduits[$pos][1] = $this->listeProduits[$pos][1] + 1;

return true;

}

return false;

}

//modifier produit

public function modifierProduit(Produit $produit,$quantite){

if ( $quantite > 0 && ($pos = $this->trouverProduit($produit)) != -1){

$this->supprimerProduit($produit);

$this->ajouterProduit($produit,$quantite);

return true;

}

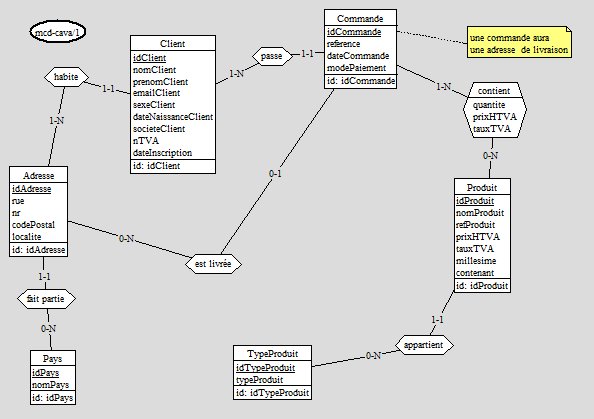
return false;

}

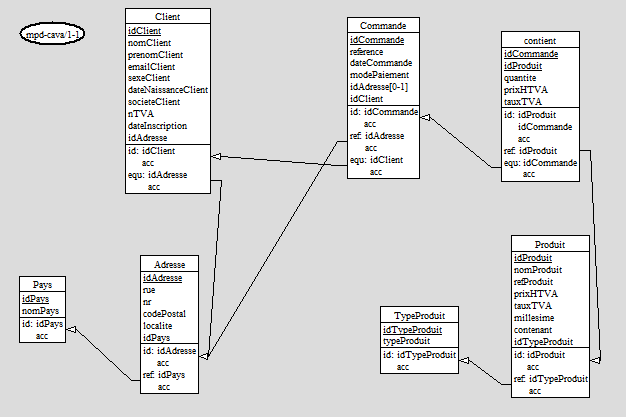
}

?>

**Le diagramme Merise MCD**



**Le diagramme Merise MPD**



**Le code généré par DBMain**

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- \* SQL MySQL generation

-- \*--------------------------------------------

-- \* DB-MAIN version: 9.1.6

-- \* Generator date: Feb 25 2013

-- \* Generation date: Thu Oct 17 19:10:45 2013

-- \* LUN file: D:\wamp\www\cava\analyse\Merise\cava.lun

-- \* Schema: sql/1-1-1

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Database Section

-- \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

create database cavaDBmain;

use cavaDBmain;

-- Tables Section

-- \_\_\_\_\_\_\_\_\_\_\_\_\_

create table Client (

idClient int not null,

nomClient varchar(1) not null,

prenomClient varchar(1) not null,

emailClient varchar(1) not null,

sexeClient varchar(1) not null,

dateNaissanceClient date not null,

societeClient varchar(1) not null,

nTVA varchar(1) not null,

dateInscription date not null,

idAdresse int not null,

constraint ID\_Client\_ID primary key (idClient));

create table Adresse (

idAdresse int not null,

rue varchar(1) not null,

nr varchar(1) not null,

codePostal varchar(1) not null,

localite varchar(1) not null,

idPays int not null,

constraint ID\_Adresse\_ID primary key (idAdresse));

create table Commande (

idCommande int not null,

reference varchar(1) not null,

dateCommande date not null,

modePaiement varchar(1) not null,

idAdresse int,

idClient int not null,

constraint ID\_Commande\_ID primary key (idCommande));

create table TypeProduit (

idTypeProduit int not null,

typeProduit varchar(1) not null,

constraint ID\_TypeProduit\_ID primary key (idTypeProduit));

create table Produit (

idProduit int not null,

nomProduit varchar(1) not null,

refProduit varchar(1) not null,

prixHTVA float(1) not null,

tauxTVA float(1) not null,

millesime varchar(1) not null,

contenant varchar(1) not null,

idTypeProduit int not null,

constraint ID\_Produit\_ID primary key (idProduit));

create table Pays (

idPays int not null,

nomPays varchar(1) not null,

constraint ID\_Pays\_ID primary key (idPays));

create table contient (

idCommande int not null,

idProduit int not null,

quantite int not null,

prixHTVA float(1) not null,

tauxTVA int not null,

constraint ID\_contient\_ID primary key (idProduit, idCommande));

-- Constraints Section

-- \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

-- Not implemented

-- alter table Client add constraint ID\_Client\_CHK

-- check(exists(select \* from Commande

-- where Commande.idClient = idClient));

alter table Client add constraint FKhabite\_FK

foreign key (idAdresse)

references Adresse (idAdresse);

-- Not implemented

-- alter table Adresse add constraint ID\_Adresse\_CHK

-- check(exists(select \* from Client

-- where Client.idAdresse = idAdresse));

alter table Adresse add constraint FKfait\_partie\_FK

foreign key (idPays)

references Pays (idPays);

-- Not implemented

-- alter table Commande add constraint ID\_Commande\_CHK

-- check(exists(select \* from contient

-- where contient.idCommande = idCommande));

alter table Commande add constraint FKest\_livree\_FK

foreign key (idAdresse)

references Adresse (idAdresse);

alter table Commande add constraint FKpasse\_FK

foreign key (idClient)

references Client (idClient);

alter table Produit add constraint FKappartient\_FK

foreign key (idTypeProduit)

references TypeProduit (idTypeProduit);

alter table contient add constraint FKcon\_Pro

foreign key (idProduit)

references Produit (idProduit);

alter table contient add constraint FKcon\_Com\_FK

foreign key (idCommande)

references Commande (idCommande);

-- Index Section

-- \_\_\_\_\_\_\_\_\_\_\_\_\_

create unique index ID\_Client\_IND

on Client (idClient);

create index FKhabite\_IND

on Client (idAdresse);

create unique index ID\_Adresse\_IND

on Adresse (idAdresse);

create index FKfait\_partie\_IND

on Adresse (idPays);

create unique index ID\_Commande\_IND

on Commande (idCommande);

create index FKest\_livree\_IND

on Commande (idAdresse);

create index FKpasse\_IND

on Commande (idClient);

create unique index ID\_TypeProduit\_IND

on TypeProduit (idTypeProduit);

create unique index ID\_Produit\_IND

on Produit (idProduit);

create index FKappartient\_IND

on Produit (idTypeProduit);

create unique index ID\_Pays\_IND

on Pays (idPays);

create unique index ID\_contient\_IND

on contient (idProduit, idCommande);

create index FKcon\_Com\_IND

on contient (idCommande);

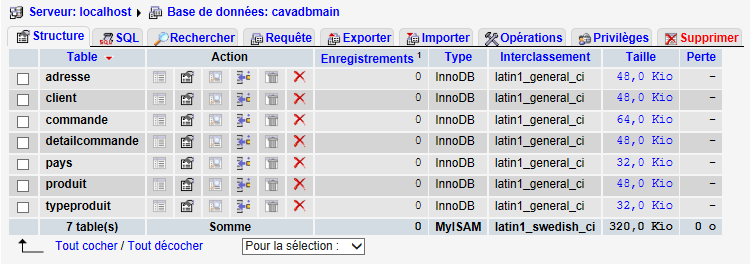
***On exécute le code dans une console SQL de phpMyAdmin …***



Vérifier le type de moteur et le charset utilisé (Collation)

*Changer le moteur myIsam vers InnoDB*

*Et définir les PK pour les attributs des tables*



Le code backup de la database avec PhpMyAdmin

-- phpMyAdmin SQL Dump

-- version 3.2.0.1

-- http://www.phpmyadmin.net

--

-- Serveur: localhost

-- Généré le : Jeu 17 Octobre 2013 à 20:47

-- Version du serveur: 5.1.36

-- Version de PHP: 5.3.0

SET SQL\_MODE="NO\_AUTO\_VALUE\_ON\_ZERO";

--

-- Base de données: `cavadbmain`

--

-- --------------------------------------------------------

--

-- Structure de la table `adresse`

--

CREATE TABLE IF NOT EXISTS `adresse` (

`idAdresse` int(11) NOT NULL,

`rue` varchar(100) CHARACTER SET latin1 NOT NULL,

`nr` varchar(10) CHARACTER SET latin1 NOT NULL,

`codePostal` varchar(10) CHARACTER SET latin1 NOT NULL,

`localite` varchar(100) CHARACTER SET latin1 NOT NULL,

`idPays` int(11) NOT NULL,

PRIMARY KEY (`idAdresse`),

UNIQUE KEY `ID\_Adresse\_IND` (`idAdresse`),

KEY `FKfait\_partie\_IND` (`idPays`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1\_general\_ci;

--

-- Contenu de la table `adresse`

--

-- --------------------------------------------------------

--

-- Structure de la table `client`

--

CREATE TABLE IF NOT EXISTS `client` (

`idClient` int(11) NOT NULL,

`nomClient` varchar(100) CHARACTER SET latin1 NOT NULL,

`prenomClient` varchar(100) CHARACTER SET latin1 NOT NULL,

`emailClient` varchar(100) CHARACTER SET latin1 NOT NULL,

`sexeClient` char(7) CHARACTER SET latin1 NOT NULL,

`dateNaissanceClient` date NOT NULL,

`societeClient` varchar(1) CHARACTER SET latin1 NOT NULL,

`nTVA` varchar(1) CHARACTER SET latin1 NOT NULL,

`dateInscription` date NOT NULL,

`idAdresse` int(11) NOT NULL,

PRIMARY KEY (`idClient`),

UNIQUE KEY `ID\_Client\_IND` (`idClient`),

KEY `FKhabite\_IND` (`idAdresse`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1\_general\_ci;

--

-- Contenu de la table `client`

--

-- --------------------------------------------------------

--

-- Structure de la table `commande`

--

CREATE TABLE IF NOT EXISTS `commande` (

`idCommande` int(11) NOT NULL,

`reference` varchar(10) CHARACTER SET latin1 NOT NULL,

`dateCommande` date NOT NULL,

`modePaiement` varchar(100) CHARACTER SET latin1 NOT NULL,

`idAdresse` int(11) DEFAULT NULL,

`idClient` int(11) NOT NULL,

PRIMARY KEY (`idCommande`),

UNIQUE KEY `ID\_Commande\_IND` (`idCommande`),

KEY `FKest\_livree\_IND` (`idAdresse`),

KEY `FKpasse\_IND` (`idClient`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1\_general\_ci;

--

-- Contenu de la table `commande`

--

-- --------------------------------------------------------

--

-- Structure de la table `detailcommande`

--

CREATE TABLE IF NOT EXISTS `detailcommande` (

`idCommande` int(11) NOT NULL,

`idProduit` int(11) NOT NULL,

`quantite` int(11) NOT NULL,

`prixHTVA` float NOT NULL,

`tauxTVA` int(11) NOT NULL,

PRIMARY KEY (`idProduit`,`idCommande`),

UNIQUE KEY `ID\_contient\_IND` (`idProduit`,`idCommande`),

KEY `FKcon\_Com\_IND` (`idCommande`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1\_general\_ci;

--

-- Contenu de la table `detailcommande`

--

-- --------------------------------------------------------

--

-- Structure de la table `pays`

--

CREATE TABLE IF NOT EXISTS `pays` (

`idPays` int(11) NOT NULL,

`nomPays` varchar(100) CHARACTER SET latin1 NOT NULL,

PRIMARY KEY (`idPays`),

UNIQUE KEY `ID\_Pays\_IND` (`idPays`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1\_general\_ci;

--

-- Contenu de la table `pays`

--

-- --------------------------------------------------------

--

-- Structure de la table `produit`

--

CREATE TABLE IF NOT EXISTS `produit` (

`idProduit` int(11) NOT NULL,

`nomProduit` varchar(100) CHARACTER SET latin1 NOT NULL,

`refProduit` varchar(20) CHARACTER SET latin1 NOT NULL,

`prixHTVA` float NOT NULL,

`tauxTVA` float NOT NULL,

`millesime` varchar(10) CHARACTER SET latin1 NOT NULL,

`contenant` varchar(100) CHARACTER SET latin1 NOT NULL,

`idTypeProduit` int(11) NOT NULL,

PRIMARY KEY (`idProduit`),

UNIQUE KEY `ID\_Produit\_IND` (`idProduit`),

KEY `FKappartient\_IND` (`idTypeProduit`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1\_general\_ci;

--

-- Contenu de la table `produit`

--

-- --------------------------------------------------------

--

-- Structure de la table `typeproduit`

--

CREATE TABLE IF NOT EXISTS `typeproduit` (

`idTypeProduit` int(11) NOT NULL,

`typeProduit` varchar(100) CHARACTER SET latin1 NOT NULL,

PRIMARY KEY (`idTypeProduit`),

UNIQUE KEY `ID\_TypeProduit\_IND` (`idTypeProduit`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1\_general\_ci;

--

-- Contenu de la table `typeproduit`

--

Le diagramme Merise Mysql Workbench

Faire le même processus avec WorkBench, avantages: meilleure conformité du code