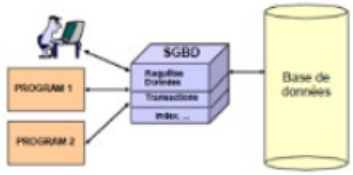
Base de données

Quelques définitions

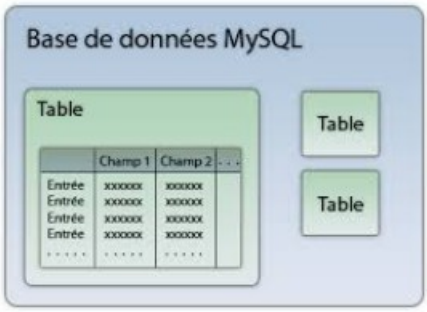
Base de données (BD) = ensemble de données accessibles et exploitables au moyen d'un ensemble de programmes



Système de gestion de bases de données (SGBD) = outil (logiciel) permettant d'accéder à des BD

Les bases de données s'appuient très souvent (mais pas toujours) sur une architecture client/serveur.

Structuration des données



Une base de données est constituée de tables.

– La table (relation) : vision tabulaire du relationnel exemple : la table FILM décrit un film.

- Les éléments constituants les colonnes des tables se nomment les champs. Ils prennent leur valeur dans un domaine (entier, chaine, date...)

Les tables sont souvent liées entre elles (mises en relation). Cela permet de supprimer la redondance.

Exemple de la table « film »

Une clé primaire : champ qui permet d'identifier de manière unique et formelle un enregistrement dans la table.

champs

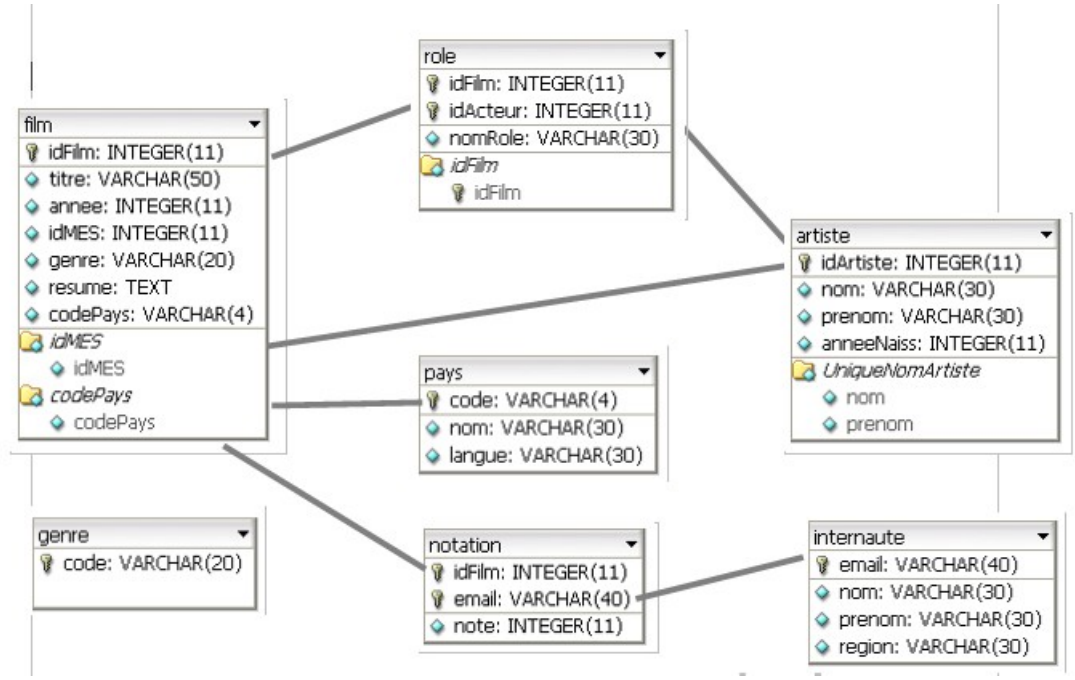
enregistrements

idFilm	titre	annee	idMES	genre	resume	codePays
1	Vertigo	1958	3	Drame	Scottie Ferguson, ancien inspecteur de police, est...	USA
2	Alien	1979	4	Science-fiction	Près d'un vaisseau spatial échoué sur une lointain...	USA
3	Titanic	1997	6	Drame	Conduite par Brock Lovett, une expédition américai...	USA
4	Sacrifice	1986	9	Drame		FR
5	Volte/Face	1997	10	Action	Directeur d'une unité anti-terroriste, Sean Archer...	USA

Etude d'une base de données cinémathèque

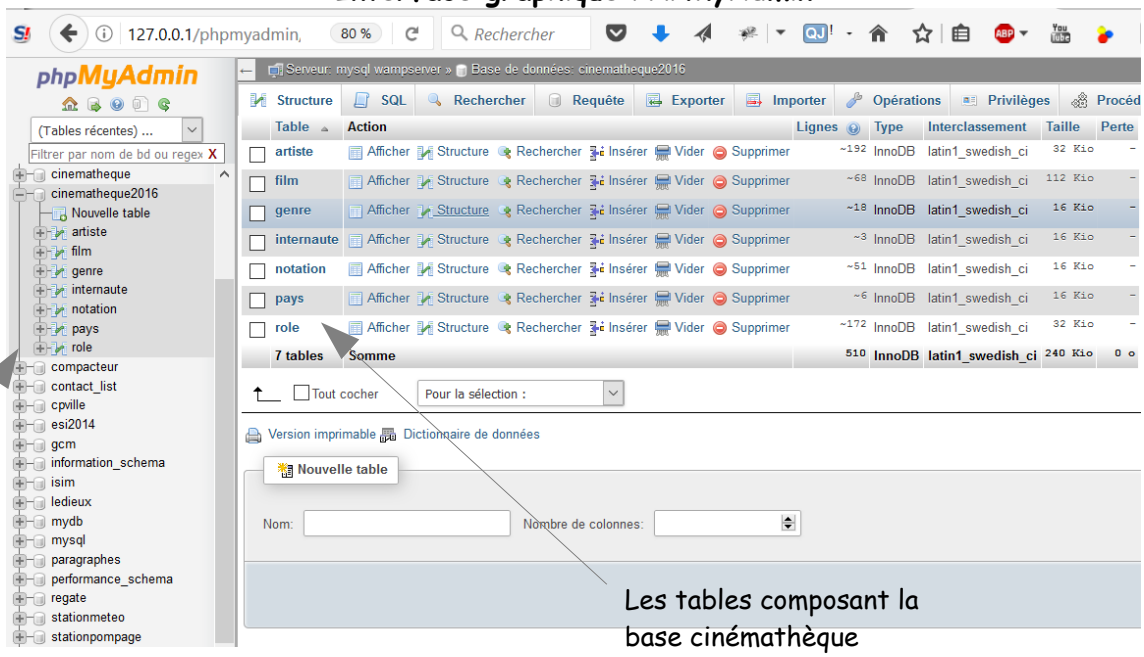
- Artiste (**idArtiste**, nom, prenom, anneeNaiss)
- Film (**idFilm**, titre, annee, *idMES*, genre, resume, *codePays*)
- Internaute (**email**, nom, prenom, region)
- Role (***idFilm***, ***idActeur***, nomRole)
- Notation (***idFilm***, ***email***, note)
- Pays (**code**, nom, langue)

Schéma logique de la base



Clé étrangère : toute clè primaire qui apparaît dans une autre table s'appelle clè étrangére.

Interface graphique PHPMyAdmin



Toutes les bases de données

Les tables composant la base cinémathèque

SELECT : requête d'interrogation

SELECT est la seule commande du SQL permettant **d'interroger une base de données**. Il existe d'autres commandes pour **manipuler les données** (ajouter, modifier), **manipuler les structures** (bases, vues, tables), **gérer les utilisateurs et leurs droits**, **créer des procédures stockées**.

Le résultat d'un **SELECT** est appelé un **jeu d'enregistrements** (**recordset** en anglais).

1/ Afficher le résultat d'une table

```
SELECT * FROM `artiste`
```

idArtiste	nom	prenom	anneeNaiss
3	Hitchcock	Alfred	1899
4	Scott	Ridley	1937
5	Weaver	Sigourney	1949

2/ Afficher seulement certains champs

Au lieu d'afficher toutes les **colonnes** (ou **champs**), on veut afficher uniquement le nom et le prénom de la table artiste.

```
SELECT `nom`,`prenom` FROM `artiste`
```

nom	prenom
Anglade	Jean-Hughes
Annaud	Jean-Jacques
Arquette	Rosanna

3/ Renommer les colonnes

Très utile pour clarifier l'affichage ou exploiter les données dans un langage de programmation.

```
SELECT `nom`,`prenom`,`anneeNaiss` as "année de naissance" FROM `artiste`
```

nom	prenom	année de naissance
Hitchcock	Alfred	1899
Scott	Ridley	1937
Weaver	Sigourney	1949
Cameron	James	1954

4/ Afficher certains enregistrements

Au lieu d'afficher tous les **enregistrements** (ou **lignes**), on veut afficher uniquement les artistes qui s'appellent *James*

```
SELECT `nom`,`prenom` FROM `artiste` WHERE `prenom`="James"
```

nom	prenom
Caan	James
Cameron	James
Gray	James
Mason	James
Stewart	James

La condition **WHERE** est suivie de n'importe quelle expression booléenne, c'est à dire qui a pour résultat "vrai" ou "faux". Par exemple :

```
WHERE condition1 AND condition2
WHERE condition1 OR condition2
WHERE NOT condition
WHERE champ1 = "valeur"
WHERE champ1 > champ2 AND champ3 > champ4
etc.
```

5/ Trier les résultats

Pour que les résultats soient triés, on utilise la clause **ORDER BY**, et les paramètres **ASC** (pour un tri croissant, ***asc****endant* en anglais) ou **DESC** (pour un tri décroissant, ***desc****endant* en anglais). Par exemple:

```
SELECT `nom`,`prenom` FROM `artiste`ORDER BY `nom` DESC
```

nom	prenom
Woo	John
Winslet	Kate
Willis	Bruce

Le tri par défaut est croissant, on n'est donc pas obligé de préciser **ASC**. Par exemple :

```
SELECT `nom`,`prenom`,`anneeNaiss` FROM `artiste`ORDER BY `anneeNaiss` ASC
```

nom	prenom	anneeNaiss
Van Cleef	Lee	*NULL*
Volonte	Gian Maria	*NULL*
Rains	Claude	1889
Hitchcock	Alfred	1899

NULL n'est pas une valeur. C'est la marque qu'il n'y en a pas.(vide)

On peut enchainer les tris

```
SELECT `nom`,`prenom`,`anneeNaiss` FROM `artiste` WHERE `anneeNaiss` BETWEEN 1980 AND 1990 ORDER BY `nom` ASC
```

nom	prenom	anneeNaiss
Osment	Haley Joel	1988
Portman	Natalie	1981
Ricci	Christina	1980

6/ La clause DISTINCT

Parfois, on est susceptible d'écrire une requête qui répond plusieurs fois la même réponse. Par exemple : quels sont les genres de films enregistrés dans la table films ?

```
SELECT DISTINCT `genre` FROM `film`
```

genre
Action
Aventures
Comédie
Comédie sentimentale
Documentaire
Drame

7/ La valeur NULL

On peut affecter une valeur spéciale à un champ : la valeur **NULL**.

A ne pas confondre avec une chaîne de caractères vide :

- sémantiquement : une chaîne vide peut signifier *"je connais cette information, mais elle est vide"*, alors que la valeur **NULL** signifie *"je ne connais pas cette information"* (par exemple pour un numéro de téléphone).
- Techniquement : une chaîne vide se teste avec `champ=""`, une valeur **NULL** se teste avec `champ IS NULL`. Une chaîne non vide se teste avec `champ <> ""`, une valeur non nulle se teste avec `champ IS NOT NULL`.

*Quand on construit une base de données, on doit préciser pour chaque champ s'il peut recevoir la valeur **NULL** ou pas*

```
SELECT `nom`,`prenom` FROM `artiste`WHERE `anneeNaiss` is NULL
```

nom	prenom
Van Cleef	Lee
Volonte	Gian Maria

8/ Le comparateur de chaînes de caractères LIKE

*Très utile pour les recherches, **LIKE** permet de comparer des chaînes entre elles en utilisant des caractères jokers, c'est à dire des caractères qui remplacent un ou plusieurs caractères.*

__%__ remplace 0, 1 ou plusieurs caractères. Par exemple :

- `champ like "méd%"` *est vrai pour les champs ayant la valeur **méd**, **méd**ecin, **méd**ical, **méd**ée, etc.*
- `champ like "%test%"` *est vrai pour les champs ayant la valeur grea**test** hits, **test**, con**test**, **test**ament, etc.*

_ (underscore) remplace 1 caractère. Par exemple :

- `champ like "par_"` *est vrai pour les champs ayant la valeur **part**, **pars**, **pare**, etc.*
- `champ like "199_-12-31"` *est vrai pour les champs ayant la valeur 1990-12-31, 1991-12-31, ... 1999-12-31*

```
SELECT `nom`,`prenom` FROM `artiste`WHERE `prenom` LIKE "jean%"
```

nom	prenom
Anglade	Jean-Hughes
Annaud	Jean-Jacques
Barr	Jean-Marc
Girault	Jean
Reno	Jean