

Internet et Outils  
L1/IO2 2006 - 2007  
S2-IO2  
Bases de données:  
Jointures, Transactions

**François Armand**  
**armand@ufr-info-p7.jussieu.fr**

# Plan Général Prévisionnel (1/2)

(non contractuel)

## • Cours Internet et Outils:

- [1/12] Intro, Internet, Web, XHTML (2H)
- [2/12] XHTML(2H)
- [3/12] CSS (2H)
- [4/12] PHP (2H)
- [5/12] PHP suite (2H)
- [6/12] Introduction MySQL, Table simple (2H)

• [7/12] ► **Partiel (2H)** ◀

**9 Mars Amphi 2A**

# Plan Général Prévisionnel (2/2)

(non contractuel)

- Bases de données

- [8/12] Tables multiples, Schémas (2H)
- [9/12] **Concepts un peu plus avancés (2H)**
- [10/12] Introduction aux réseaux(2H)
- [11/12] Aperçu de HTTP et d'autres outils (2H)
- [12/12] Logiciels clients, serveurs (2H)

# Plan

- Ce que vous allez découvrir
- Rappels
- Jointures, Unions
- Index
- Vues
- Transactions
- Sauvegardes
- Intégration PHP
- Ce qu'il faut retenir

# Ce que vous allez découvrir

- Extractions de données depuis plusieurs tables,
  - Jointures et variations sur le thème!
- Accélérer les requêtes
  - Index
- Faire "Tout ou Rien"
  - Laisser la base dans "l'état" où on l'on trouvé!
  - Transactions,
- Sauvegarder une table

# Plan

- Ce que vous allez découvrir
- **Rappels**
- Jointures, Unions
- Index
- Vues
- Transactions
- Sauvegardes
- Intégration PHP
- Ce qu'il faut retenir

# Rappels

- Utilisation de tables pour "structurer" la manière de représenter l'information
- Eviter d'avoir la même information présente plusieurs fois (on parle de "normalisation")
- Colonnes d'une table: attributs
- Lignes d'une table: un enregistrement
- Intersection colonne/ligne: une cellule
- Identifier les lignes par des "clés"
  - primaires, secondaires, étrangères

# Rappels

- SQL:
  - Langage déclaratif
  - Orientation: Retrouver et Mettre à disposition l'information demandée par l'utilisateur,
  - La manière de stocker les informations est cachée et laisser au libre arbitre du SGBD.
- Retrouver les informations:
  - `SELECT colonne,... FROM TABLE WHERE clause;`
  - colonne: découpage vertical des tables
  - clause `WHERE`: découpage horizontal des tables.

# Accéder à plusieurs tables

- Recomposer l'information dont on disposait avec une seule table: Titre, Réalisateur.

```
mysql> SELECT * FROM Artiste;
```

ident	Nom	Prenom	Naissance
1	Scott	Ridley	1943
2	Hitchcock	Alfred	1899
3	Woo	John	1946
4	Kurosawa	Akira	1910
5	Tarkovski	Andrei	1932
6	Cameron	James	1954

# Accéder à plusieurs tables

```
mysql> SELECT Titre, Annee, Id_Real FROM
  Films;
```

```
+-----+-----+-----+
| Titre      | Annee  | Id_Real |
+-----+-----+-----+
| Alien      | 1979   | 1       |
| Sacrifice  | 1986   | 5       |
+-----+-----+-----+
2 rows in set (0.00 sec)
```

- **SELECT** permet de référencer des champs de différentes tables.... Essai:
- **SELECT** Titre, Nom FROM Films, Artiste;

# Accéder à plusieurs tables

```
mysql>SELECT Titre, Nom FROM Films,Artiste;
```

```
+-----+-----+
| Titre      | Nom      |
+-----+-----+
| Alien      | Scott    |
| Sacrifice  | Scott    |
| Alien      | Hitchcock|
| Sacrifice  | Hitchcock|
| Alien      | Woo      |
| Sacrifice  | Woo      |
| Alien      | Kurosawa |
| Sacrifice  | Kurosawa |
| Alien      | Tarkovski|
| Sacrifice  | Tarkovski|
| Alien      | Cameron  |
| Sacrifice  | Cameron  |
+-----+-----+
```

Ne donne pas le résultat escompté!

Pas de lien explicité entre les deux tables.

# Accéder à plusieurs tables

```
mysql> SELECT Titre, Nom FROM Films, Artiste  
      -> WHERE Films.Id_Real=Artiste.Ident;
```

```
+-----+-----+  
| Titre      | Nom      |  
+-----+-----+  
| Alien      | Scott    |  
| Sacrifice  | Tarkovski|  
+-----+-----+  
2 rows in set (0.00 sec)
```

```
mysql>
```

- Accès à des informations contenues dans plusieurs tables: JOINTURE.

# Plan

- Ce que vous allez découvrir
- Rappels
- **Jointures, Unions**
- Index
- Vues
- Transactions
- Sauvegardes
- Intégration PHP
- Ce qu'il faut retenir

# Jointures

- Une "jointure" permet de "réunir" les informations stockées dans plusieurs tables différentes d'une même base.
- Différents types de jointures
  - Jointure Interne (**INNER Join**)
  - OUTER Joint
    - LEFT
    - RIGHT

# Jointure Interne

- **SELECT** col1, col2... **FROM** table\_1  
**INNER JOIN** table\_2  
**ON** table\_1.col = table\_2.cle\_etrangere;
- Renvoie les lignes de table\_1 qui correspondent à la condition,
- Les lignes de table\_1 qui ne sont pas référencées depuis table\_2 n'apparaissent pas dans le résultat.

# Jointure Interne

```
mysql> SELECT Films.Titre, Artiste.Nom  
-> FROM Films  
-> INNER JOIN Artiste  
-> ON Films.Id_Real=Artiste.Ident;
```

- Renvoie les "seuls" Titres de Films ayant un Artiste enregistré dans la table Artiste
- Les Films n'ayant pas d'artiste associé ne figurent pas dans le résultat d'une jointure interne.
  - Pour nommer une colonne d'une table sans ambiguïté: nom\_table.nom\_colonne

# JOINTURE INTERNE

```
mysql> SELECT Films.Titre, Artiste.Nom  
-> FROM Films INNER JOIN Artiste  
-> ON Films.Id_real=Artiste.ident;
```

```
+-----+-----+  
| Titre      | Nom      |  
+-----+-----+  
| Alien      | Scott    |  
| Sacrifice  | Tarkovski |  
+-----+-----+  
2 rows in set (0.03 sec)
```

```
mysql>
```

# Jointure Gauche

```
mysql> SELECT col1, col2... FROM table_1
```

```
-> LEFT JOIN table_2
```

```
-> ON table_1.col = table_2.cle_etrangere;
```

- Renvoie toutes les lignes de table\_1 même si la condition n'est pas remplie.
- Si une colonne de table\_2 doit être affichée, sa valeur sera vide pour les lignes de table\_1 pour lesquelles la condition n'est pas remplie.

# Jointure Gauche

```
mysql>SELECT Films.Titre, Artiste.Nom  
->FROM Films  
->LEFT JOIN Artiste  
->ON Films.Id_Real=Artiste.Ident;
```

- Renvoie tous les Titres de films,
- Ceux qui n'ont pas d'artiste associé, auront le champ "Nom" vide.

# Jointure Gauche: Table Films

```
mysql> SELECT * FROM films;
```

```
+-----+-----+-----+
| Titre | Annee | Id_Real |
+-----+-----+-----+
| Alien | 1979  | 1       |
| Sacrifice | 1986 | 3       |
| Match Point | 2005 | NULL    |
+-----+-----+-----+
```

```
3 rows in set (0.00 sec)
```

```
mysql>
```

# Jointure Gauche

```
mysql> SELECT Films.Titre, Artiste.Nom  
-> FROM Films LEFT JOIN Artiste  
-> ON Films.Id_real=Artiste.ident;
```

```
+-----+-----+  
| Titre | Nom |  
+-----+-----+  
| Alien | Scott |  
| Sacrifice | Tarkovski |  
| Match Point | NULL |  
+-----+-----+  
3 rows in set (0.00 sec)
```

```
mysql>
```

# Jointure Droite

```
mysql> SELECT col1, col2... FROM table_1  
-> RIGHT JOIN table_2  
-> ON table_1.col = table_2.cle_etrangere;
```

- Renvoie les lignes de table\_2 même si la condition n'est pas remplie,
- Si une colonne de table\_1 doit être affichée, sa valeur sera vide pour les lignes de table\_2 pour lesquelles la condition n'est pas remplie.

# Jointure Droite

```
mysql>SELECT Films.Titre, Artiste.Nom  
->FROM Films  
->RIGHT JOIN Artiste  
->ON Films.Id_Real=Artiste.Ident;
```

- Renvoie tous les Noms d'Artiste,
- Ceux qui n'ont pas de film associé, auront le champ "Titre" vide.

# Jointure Droite

```
mysql> SELECT Films.Titre, Artiste.Nom
-> FROM Films RIGHT JOIN Artiste
-> ON Films.Id_real=Artiste.ident;
```

Titre	Nom
Alien	Scott
NULL	Hitchcock
NULL	Woo
NULL	Kurosawa
Sacrifice	Tarkovski
NULL	Cameron

```
6 rows in set (0.00 sec)
```

# Jointure Droite

```
mysql> SELECT Films.Titre, Artiste.Nom
-> FROM Films RIGHT JOIN Artiste
-> ON Films.Id_real=Artiste.ident;
```

Titre	Nom
Alien	Scott
Vertigo	Hitchcock
Psychose	Hitchcock
NULL	Woo
NULL	Kurosawa
Sacrifice	Tarkovski
NULL	Cameron

# Exemple

- Trouver les clients qui ont commandé le produit dont la description est "Machin Vert"

Table: Produits

Produit_Id	Prod_Descr
------------	------------

Table: Cmd\_Unitaires

Cmd_Num	Produit_Id	Quantité	Coût	Prix
---------	------------	----------	------	------

Table: Cmdes

Cmd_Num	Client_Id	Date_Commande
---------	-----------	---------------

Table: Clients

Client_Id	Nom	Prénom
-----------	-----	--------

# Exemple

- Trouver le produit qui s'appelle Machin Vert:
- **SELECT** Produit\_Id **FROM** Produits **WHERE** Prod\_Descr='Machin Vert';
- Ensuite, parcourir le chaînage entre les tables jusqu'à ce que l'on trouve le "Produit" avec l'identificateur correspondant.

# Exemple

```
SELECT Clients.Nom
FROM Clients, Cmdes, Cmd_Unitaires, Produits
WHERE
  Clients.Client_Id = Cmdes.Client_Id
AND Cmdes.Cmd_Num = Cmd_Unitaires.Cmd_Num
AND Cmd_Unitaires.Produit_Id =
  (SELECT Produit_Id FROM Produits
   WHERE Prod_Descr = 'Machin Vert')
ORDER BY Clients.Nom;
```

# UNION

- Sélectionne les lignes de 2 tables;
  - Les colonnes doivent être de même type
- Si une information est présente dans les 2 tables, elle ne sera listée qu'une seule fois
- **SELECT** col\_x **FROM** table\_1  
**UNION**  
**SELECT** col\_y **FROM** table\_2;

# UNION

```
mysql>SELECT Titre FROM Films
```

```
-> UNION
```

```
->SELECT Titre FROM Vieux_Films;
```

- Liste les titres des films répertoriés dans la table Films et les titres des films répertoriés dans la table Vieux\_Films
- Les doublons (titres présents dans les deux tables) n'apparaissent qu'une fois.
- Si on veut voir les doublons, utiliser **UNION ALL**

# Plan

- Ce que vous allez découvrir
- Rappels
- Jointures, Unions
- **Index**
- Vues
- Transactions
- Sauvegardes
- Intégration PHP
- Ce qu'il faut retenir

# INDEX

- Tables peuvent devenir très grandes
- Rechercher une information dans une table
  - Parcourir linéairement la table
  - Comparer l'information de la ligne avec l'information recherchée...
  - Temps possiblement (très) long
- Utiliser des systèmes d'indexation pour retrouver l'information plus rapidement
  - Exemple: livres d'une bibliothèque

# INDEX

ID	Titre	Auteur	Editeur
1	Les Misérables	Victor Hugo	Ed. Minuit
2	Astérix	René Goscinny	Ed. Dargault
3	Vie Mode d'Emploi	Georges Perec	Ed. Laffont

- Retrouver un livre par le titre, l'auteur, l'éditeur... le numéro ISBN...
- On ne peut pas ordonner la table sur tous les critères simultanément
- Par contre, on peut créer des "listes" ordonnées pour les critères importants

# INDEX

ID	Titre	Auteur	Editeur
1	Les Misérables	Victor Hugo	Ed. Minuit
2	Astérix	René Goscinny	Ed. Dargault
3	Vie Mode d'Emploi	Georges Perec	Ed. Laffont

Par Titre

2
1
3

Par Auteur

3
2
1

Par Editeur

2
3
1

- On peut créer plusieurs index en fonction de critères différents,
- C'est le SGBD qui ensuite utilise l'index qui lui semble le plus approprié en fonction de la requête à traiter.

# INDEX

- Il faut indiquer au SGBD quel(s) index créer.
- Le SGBD maintient ensuite ces index à jour en fonction des insertions, suppressions et modifications des tables.
- Plus il y a d'index, plus les insertions, suppressions et modifications coûtent cher: il faut aussi mettre les index à jour!

# INDEX

- **CREATE UNIQUE INDEX** mon\_nouvel\_index  
**ON** ma\_table(ma\_colonne);
- Crée un index "mon\_nouvel\_index" accélérant les recherches sur "ma\_colonne",
  - UNIQUE indique que 2 lignes ne peuvent contenir la même valeur
- **CREATE INDEX** mon\_nouvel\_index  
**ON** ma\_table(ma\_colonne);
  - Sans UNIQUE, 2 lignes peuvent contenir la même valeur.

# INDEX

- En fait, on peut créer des index sur plus d'un attribut
- **CREATE UNIQUE INDEX** mon\_nouvel\_index  
**ON** ma\_table(colonne\_x, colonne\_y);
- On peut supprimer un index:
- **DROP INDEX** mon\_nouvel\_index **ON** ma\_table;

# Plan

- Ce que vous allez découvrir
- Rappels
- Jointures, Unions
- Index
- **Vues**
- Transactions
- Sauvegardes
- Intégration PHP
- Ce qu'il faut retenir

# VUES (Views)

- Une VUE est une table virtuelle basée sur le résultat d'une requête SELECT.
- Une VUE est comme une table: colonnes et lignes
- Les champs d'une VUE peuvent provenir de plusieurs tables (résultat d'une jointure)
- La création d'une VUE n'affecte pas les tables existantes.

# VUES

- **CREATE VIEW** ma\_vue **AS SELECT** col,...  
**FROM** table,... **WHERE** condition;
- On peut ensuite appliquer n'importe quel **SELECT** sur cette vue,

```
mysql>CREATE VIEW FilmSimple AS SELECT  
->Titre, Annee, Nom, Prenom, Naissance  
->FROM Films, Artiste  
->WHERE Films.Id_Real=Artiste.Ident;
```

- Reproduit la table "FilmSimple" initiale.

# Plan

- Ce que vous allez découvrir
- Rappels
- Jointures, Unions
- Index
- Vues
- **Transactions**
- Sauvegardes
- Intégration PHP
- Ce qu'il faut retenir

# Transactions

- Besoin de mise à jour de plus
  - d'un champ,
  - d'une ligne,
  - d'une table...
- Exemple
  - Transférer 100 euros du compte de Jules sur le compte de Julie...
  - A la fin:
    - Soit l'opération est effectuée,
    - Soit elle n'est pas effectuée.

# Transactions

- États possibles (Base cohérente)
  - Compte de Jules =  $X$ , Compte de Julie =  $Y$
  - OU
  - Compte de Jules =  $X-100$ , Compte de Julie =  $Y+100$ .
- États impossibles (Base incohérente)
  - Compte de Jules =  $X$ , Compte de Julie =  $Y + 100$ 
    - "Apparition spontanée" d'argent
  - Compte de Jules =  $X-100$ , Compte de Julie =  $Y$ 
    - "Disparition spontanée" d'argent

# Transactions

- Opérations:
  - **A**tomiques (tout est effectué ou rien)
  - Assurant la **C**ohérence des informations
  - Comme si l'utilisateur était seul, **I**solé, même si plusieurs utilisateurs accèdent simultanément à la base,
  - Et ayant un effet **D**urable
    - L'argent est définitivement transféré d'un compte à l'autre.

# Transactions

- Débuter une séquence d'opérations
  - MySQL: **BEGIN** ou **START TRANSACTION**
- Effectuer les opérations
- Si problème:
  - Défaire et retrouver l'état initial
  - MySQL: **ROLLBACK**
- Si OK
  - Confirmer les modifications
  - MySQL: **COMMIT**

# Transactions

**BEGIN;**

**UPDATE** Banque **SET** Cpte=Cpte-50

**WHERE** Nom='Jules';

**#Si Cpte < 0 ROLLBACK;**

**UPDATE** Banque **SET** Cpte=Cpte+50

**WHERE** Nom='Julie';

**COMMIT;**

# Plan

- Ce que vous allez découvrir
- Rappels
- Jointures, Unions
- Index
- Vues
- Transactions
- **Sauvegardes**
- Intégration PHP
- Ce qu'il faut retenir

# Copies / Sauvegardes

- Le résultat d'un **SELECT** peut-être affecté dans une autre table de la même base ou d'une autre base.
- **SELECT** col,... **FROM** table,... **INTO** nouvelle\_table [**IN** base];
- Ex:
  - **SELECT** \* **FROM** Artiste **INTO** Copie\_Artiste **IN** Film\_backup;

# Copies / Sauvegardes

- Le résultat d'un SELECT peut-être affecté dans un fichier.
- Opération inverse de "LOAD DATA"
- **SELECT** col,... **INTO OUTFILE** fichier.txt  
**FIELDS TERMINATED BY ','** **LINES**  
**TERMINATED BY '\n'** **FROM** table;

# Plan

- Ce que vous allez découvrir
- Rappels
- Jointures, Unions
- Index
- Vues
- Transactions
- Sauvegardes
- **Intégration PHP**
- Ce qu'il faut retenir

# Exemple (Cnx.php)

```
<?php
```

```
function Connexion($nom, $mdp, $base, $srv)
```

```
{
```

```
$connexion = mysql_pconnect($srv, $nom, $mdp)
```

```
if (!$connexion) {
```

```
    echo "Connexion au serveur $srv impossible\n";
```

```
    exit;
```

```
}
```

# Exemple (Cnx.php)

```
if (!mysql_select_db($base, $connexion)) {  
    echo "Accès à la base $base impossible\n";  
    echo "<br /><strong>Erreur: </strong>";  
    echo mysql_error($connexion);  
    exit;  
}  
  
return $connexion  
} // Fin de fonction Connexion
```

# Exemple (Requete.php)

```
<?php
function Requete($req, $cnx)
{
$resultat = mysql_query( $req, $cnx);
if ($resultat) {
    return $resultat;
} else {
    echo "<b>Erreur</b>" . mysql_error() . "<br />";
}
}
```

# Exemple (Requete.php)

```
function LigneSuivante ($res)
{
return mysql_fetch_array($res);
}
?>
```

# Exemple (AfficheFilms.php)

```
<?php
function AfficheFilms($cnx)
{
$res = Requete("SELECT * FROM FilmSimple",
    $cnx);
echo "<table><caption> Liste des Films
    </caption>";
echo "<tr><th>Titre</th> <th>Annee</th>
    <th>Real</th> <th>Ne en</th> <th>Action</th>
    </tr>";
```

# Exemple (AfficheFilms.php)

```
while ($film=LigneSuivante($res)) {  
    $titre_url=urlEncode($film['Titre']);  
    echo "<tr><td>$film['Titre']</td>  
        <td>$film['Prenom'] $film['Nom']</td>  
        <td>$film['Naissance']</td>  
        <td><a href='Film.php?mode=MAJ  
            &titre=$titre_url'>  
            Modifier ce film</a></td>  
    </tr>";  
}
```

# Exemple (AfficheFilms.php)

```
echo "</table>";
```

```
}
```

```
?>
```

# Plan

- Ce que vous allez découvrir
- Rappels
- Jointures, Unions
- Index
- Vues
- Transactions
- Sauvegardes
- Intégration PHP
- **Ce qu'il faut retenir**

# Ce qu'il faut retenir

- Comment faire des jointures
- Créer les bons index: assez mais pas trop
- La notion de transaction
- Sauvegarde de base
- Accès depuis PHP