MySQL

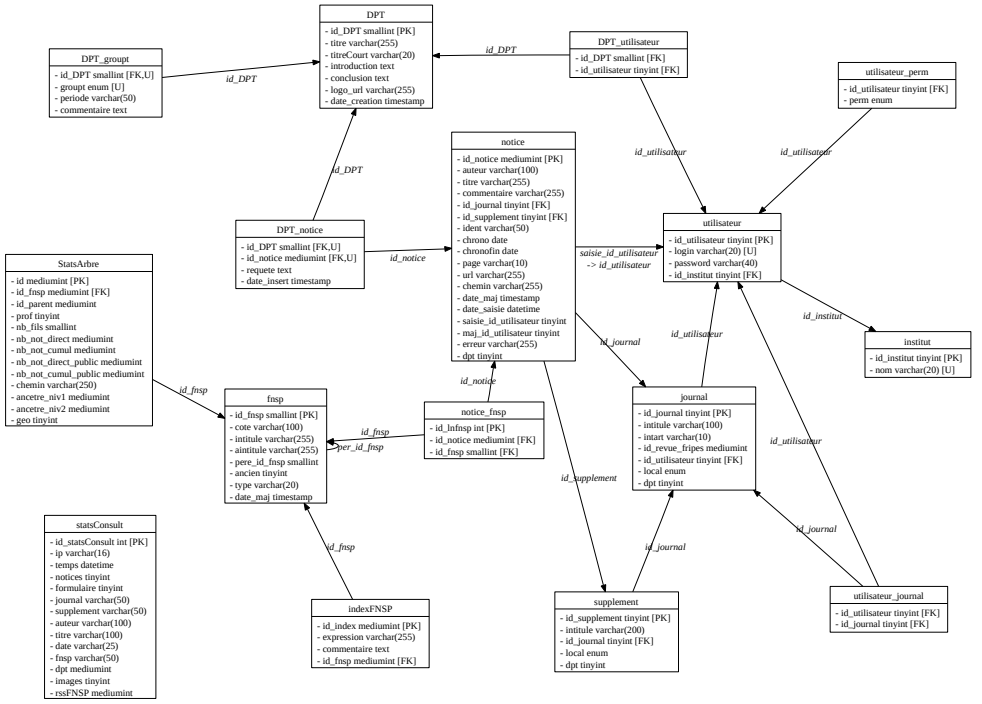
François Gannaz

Grenoble INP – Formation Continue

Plan

MySQL c'est. . .

- une base de donnée **relationnelle** (SGDBR) créée en 1995
- **libre** (GPL) et développée par une société suédoise (ABSoft)
  Racheté par Sun Microsytems début 2008,
  lui-même en cours de rachat par Oracle en 2009.
- le standard de facto des SGDB pour le web.
  Utilisé par Google, Facebook, Yahoo. . .
- multi plates-formes : Linux, Windows, OSX, etc.
- modèle client-serveur
- une très bonne doc en ligne (traduite) :
  `http://dev.mysql.com/doc/refman/5.0/en/index.html`
- plus ou moins fidèle aux standards SQL
  Compatibilité partielle avec PostGreSQL, SQLite. . .



Les tables

Une base de données (par ex. *discotheque*) est faite de **tables**.

Table **Disques**		
Titre	**Auteur**	**Date**
Cantates	Bach J.S.	2006
Sonates	Beethoven	2005
Concerto	Dvorak	2000

Chaque ligne est un **enregistrement**.
Le nom d'une colonne est dit **champ** ou **attribut**.

Les colonnes sont typées

Numérique	BOOLEAN, TINYINT, MEDIUMINT, INT, BIGINT, DOUBLE
Texte	VARCHAR(taille), MEDIUMTEXT, TEXT, LONGTEXT, ENUM(liste)
Date/Heure	DATE, TIMESTAMP

Plan

Conception d'une base

- Lister les données à stocker
- Structurer en entités–attributs (tables–champs)
- (1NF) Normaliser : un attribut contient une seule valeur
- (2NF) Fixer un identifiant unique pour chaque entité (Primary Key)
- (3NF) Normaliser : éviter les redondances d'un attribut
- Tracer un diagramme des relations

Exemple de la gestion d'une liste de CD.

Disques
Titre
Interprète
Genre musical
Label
Date de sortie

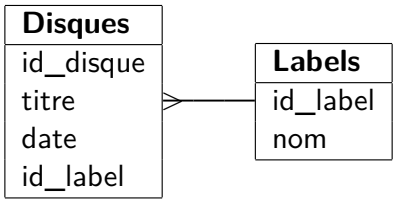
Comment organiser ces données ?

Disques
Titre

Relations

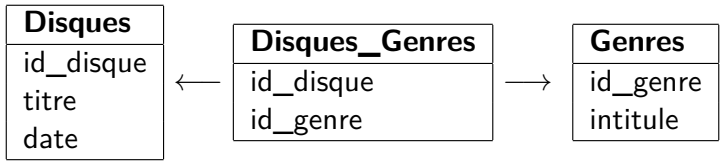
Relation 1:n

Chaque élément de *Disques* est lié à **au plus** un élément de *Labels*.



Relation multiple

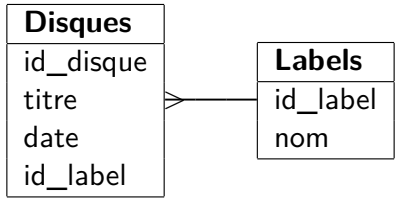
Chaque *disque* est lié à **plusieurs** *genres* et réciproquement.



Relations

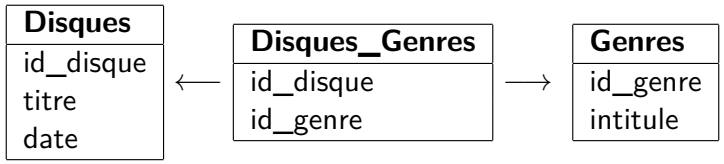
Relation 1:n

Chaque élément de *Disques* est lié à **au plus** un élément de *Labels*.



Relation multiple

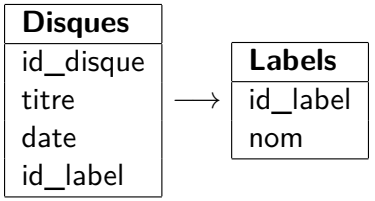
Chaque *disque* est lié à **plusieurs** *genres* et réciproquement.



Relations

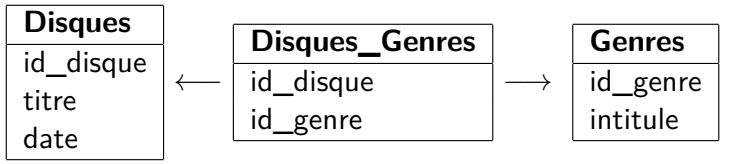
Relation 1:n

Chaque élément de *Disques* est lié à **au plus** un élément de *Labels*.



Relation multiple

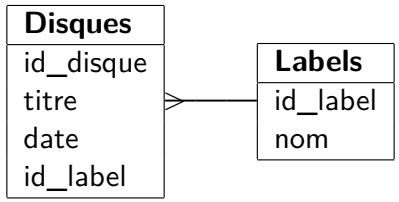
Chaque *disque* est lié à **plusieurs** *genres* et réciproquement.



Relations

Relation 1:n

Chaque élément de *Disques* est lié à **au plus** un élément de *Labels*.



Relation multiple

Chaque *disque* est lié à **plusieurs** *genres* et réciproquement.

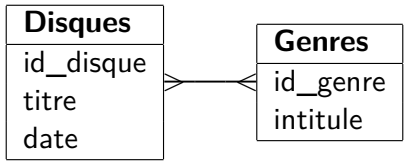
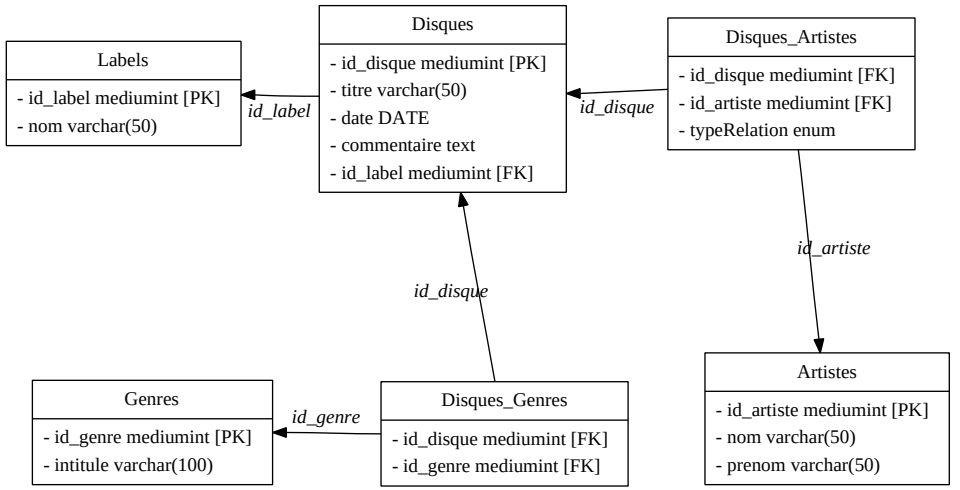


Schéma final pour l'exemple



Plan

Outils MySQL

Interfaces disponibles

Web : phpMyAdmin
Graphique : MySQL Administrator
Texte SQL : client mysql

Actions possibles

- Gestion des utilisateurs et permissions
- Création de bases de données
- Visualisation et recherche
- Requêtes SQL

Plan

Lire des données : SELECT

SELECT renvoie une "table" : résultat en lignes/colonnes.

Syntaxe simplifiée

SELECT expression **FROM** matable **WHERE** condition ;

Une expression (et une condition) est composée de
constantes : 3.14, 'chaine'
attributs : date, id_genre
fonctions : CONCAT(id_genre,'-',intitule)

Exemples :

- SELECT * FROM Disques ;
- SELECT titre FROM Disques WHERE date>'2006-01-01' ;

Compléments sur SELECT

- **ORDER BY** : Trier les résultats
  - SELECT * FROM Disques ORDER BY titre ASC
  - SELECT * FROM Disques ORDER BY date DESC, titre ASC
- **LIMIT** : Limiter le nombre de résultats
  - SELECT * FROM Disques LIMIT 3
  - SELECT * FROM Disques LIMIT 3,5
- **DISTINCT** : Supprimer tout doublon dans les résultats
  - SELECT DISTINCT prenom FROM Artistes

Quelques fonctions

- opérateurs : $= < > != * / +$ etc.
- la comparaison de texte est sans casse et sans accents
- **LIKE** : chaînes contenant un motif donné

```
SELECT titre FROM Disques
     WHERE titre LIKE 'Variations %'
```


Exercices

1. Trouver le disque de 2005 intitulé "souvenir de florence".
2. Lister les titres des disques, triés par date. Les trier par label, et par date pour un même label.
3. Quelle différence entre `SELECT intitule, id_genre FROM Genres` et `SELECT * FROM Genres` ?
4. Afficher tous les titres de 2006. Les 3 titres les plus récents.
5. Que donne `SELECT count(*) FROM Disques` ? Quelle différence avec `SELECT count(Disques.id_disque) FROM Disques` ?
6. Quelles sont les différentes relations entre artistes et disques ?
7. Combien de compositeurs y a-t-il ?
8. Quels disques ont été publiés par Harmonia Mundi ? Comment obtenir ce résultat en une seule requête ?

Jointures

Le but : interroger plusieurs tables à la fois

Exemple :

```
SELECT titre FROM Disques
  JOIN Labels ON Disques.id_label=Labels.id_label
  WHERE Labels.nom='Harmonia mundi'
```

Variantes

- `SELECT titre FROM Disques AS d JOIN Labels AS l ON d.id_label=l.id_label WHERE l.nom='Harmonia mundi'`
- `SELECT titre FROM Disques d JOIN Labels l USING (id_label) WHERE l.nom='Harmonia mundi'`
- `SELECT titre FROM Disques AS d JOIN Labels AS l WHERE d.id_label=l.id_label AND l.nom='Harmonia mundi'`

Jointures : exemple

`SELECT * FROM Joueurs`

nom	id_pays
Federer	1
Nadal	2
Ferrer	2

`SELECT * FROM Pays`

id_pays	pays
1	Suisse
2	Espagne
3	France

`SELECT * FROM Pays JOIN Joueurs USING (id_pays)`

id_pays	pays	nom
1	Suisse	Federer
2	Espagne	Nadal
2	Espagne	Ferrer

Exercices

1. Afficher tous les titres avec leur label associé.
2. Y a-t-il un disque de EMI intitulé "... Figaro" ?
3. Quels sont les disques de genre "Baroque" ?
4. Lister les interprètes ayant contribué à au moins un disque.
5. Est-ce que Phillips a déja produit de l'opéra ?
6. Afficher tous les titres avec leur compositeur associé. Avec leurs artistes associés ?

INSERT

Insérer une ligne dans une table

2 syntaxes :

- `INSERT INTO Artistes (nom,prenom) VALUES ('La tordue',")`
  Permet d'insérer plusieurs enregistrements
- `INSERT INTO Artistes SET nom='La tordue', prenom="`
  Syntaxe commune avec UPDATE

Si un champ n'a pas de valeur :

- s'il est en AUTOINCREMENT, il vaudra 1 de plus que pour le précédent
- sinon, il prend la valeur par défaut (souvent NULL ou '')

UPDATE et DELETE

UPDATE
Modifie des enregistrements

UPDATE Artistes **SET** nom = 'Les ogres'
  WHERE id_artiste = 2 ;
UPDATE Artistes **SET** nom = **UPPER**(nom) ;

DELETE
Supprime des enregistrements

DELETE FROM Artiste
  WHERE nom **LIKE** '%tordu%'

Améliorer drastiquement les performances

Performance de SELECT

SELECT d.titre, l.nom
  FROM Disques d **JOIN** Labels l **USING** (id_label)
  WHERE l.**date** > 2005

S'il n'y a pas d'index, MySQL parcourt

- toute la table *Labels* pour trouver les *id_label* (jointure)
- toute la table *Disques* pour trouver les *date* (condition)

Index
Permet à MySQL de trouver rapidement une valeur quand le nombre d'enregistrements est important.

Clé primaire $\Longrightarrow$ index

Pour aller plus loin

- **Clés étrangères** pour officialiser les relations.
- La valeur **NULL** : impact sur count(), IS NULL / IS NOT NULL
- **Collation** et comparaison des chaînes de caractères
- **Auto-jointures** : pour les arbres, les hiérarchies
  `FROM matable t1 JOIN matable t2 ON t1.pere=t2.id`
- **GROUP BY** : regrouper les résultats quand un champ est identique
  fonctions associées : count, sum, max, min, avg...
- HAVING : condition sur le GROUP BY
- **Sous-requêtes**
  `SELECT * FROM (SELECT CONCAT(nom,' ',prenom) AS nomC FROM Joueurs) WHERE nomC LIKE '%al%'`
- EXPLAIN : optimiser les requêtes

Informations utiles

Pour garder le contact :
`francois.gannaz@silecs.info`

Les documents utilisés sont disponibles en ligne :
`http://silecs.info/formations/PHP-MySQL/`

- Transparents
- Corrections des exercices

Licence

Schéma final pour l'exemple