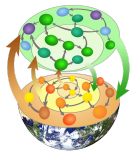


Séance 3: La gestion des données biologiques

BIO 500 - Méthodes en écologie computationnelle

Steve Vissault & Dominique Gravel
Laboratoire d'écologie intégrative

Séance 3

- ✓ Ces diapositives sont disponibles en **version web** et en **PDF**.
- ✓ L'ensemble du matériel de cours est disponible sur la page du portail **moodle**.

Les grandes étapes

1. Spécifier la connexion avec le serveur
2. Créer la base de données
3. Créer les tables et spécifier les clés
4. Ajouter de l'information dans les tables
5. Faire des requêtes pour extraire l'information

Retour rapide sur la séance de la semaine dernière

Connexion au serveur

```
library(RSQLite)

con <- dbConnect(SQLite(),dbname="./assets/donnees/films.db")

tbl_films <- "CREATE TABLE films (
    id_film     integer,
    titre       varchar(300),
    annee   integer,
    PRIMARY KEY (id_film)
);"

dbSendQuery(con,tbl_films)
```

Question: Sur ce script, où sont les instructions SQL? Òu sont les commandes R?

Création de la table `films`

```
tbl_films <- "CREATE TABLE films (
    id_film     integer,
    titre       varchar(300),
    annee   integer,
    PRIMARY KEY (id_film)
);"

dbSendQuery(con,tbl_films)
```

Création de la table **acteurs**

```
tbl_acteurs <- "CREATE TABLE acteurs (
    id_acteur   integer,
    nom         varchar(100),
    prenom      varchar(100),
    id_film     integer,
    PRIMARY KEY (id_acteur),
    FOREIGN KEY (id_film) REFERENCES films (id_film) ON DELETE CASCADE
);"

dbSendQuery(con,tbl_acteurs)
```

```
## Warning: Closing open result set, pending rows
```

Ajouter de l'information dans les tables

RSQLite - dbWriteTable

La librairie RSQLite peut nous aider plus facilement à accomplir cette tâche:

```
# Lecture des fichiers CSV
bd_films <- read.csv2(file='./assets/donnees/bd_beacon/bd_films.csv',stringsAsFactors=FALSE)
bd_acteurs <- read.csv2(file='./assets/donnees/bd_beacon/bd_acteurs.csv',stringsAsFactors=FALSE)

# Injection des enregistrements dans la BD
dbWriteTable(con,append=TRUE,name="films",value=bd_films, row.names=FALSE)
```

```
## Warning: Closing open result set, pending rows
```

```
dbWriteTable(con,append=TRUE,name="acteurs",value=bd_acteurs, row.names=FALSE)
```

Exercice 1 (10-15 minutes)

Ce premier exercice est important pour la suite de la séance.

1. Recréer la base de données `bd_films` avec ses deux tables `films` et `acteurs`
2. Insérer les données **bd_acteurs.csv** et **bd_films.csv** dans les deux tables à l'aide de la commande R `dbWriteTable()`

Les requêtes

Structure du requête

```sql
SELECT colonnes/champs
FROM table1
JOIN table2 ON table1.foreignKey = table2.primaryKey
WHERE criteres
ORDER BY colonne1 ASC
LIMIT 10;
```

- ✓ Les requêtes SQL sont une suite d'opérations séquentielles.
- ✓ On ne peut pas filtrer (`WHERE`) avant que les opérations `SELECT`, `FROM` et `JOIN` soit complétées.

Sélectionner des tables et des colonnes

```
sql_requete <- "
SELECT id_film, titre, annee
  FROM films LIMIT 10
;"

films <- dbGetQuery(con,sql_requete)
head(films)
```

```
##   id_film                 titre annee
## 1       1               ¡Átame!  1990
## 2       2         ¡Three Amigos!  1986
## 3       3 ...And Justice for All  1979
## 4       4       'Breaker' Morant  1980
## 5       5           'burbs, The  1989
## 6       6  'Crocodile' Dundee II  1988
```

con

- ✓ `SELECT` spécifie les colonnes.
- ✓ `FROM` spécifie la table.
- ✓ On peut également ajouter une `LIMIT`.
- ✓ **Documentation SQL Select**.

Sélectionner des tables et des colonnes

```
sql_requete <- "SELECT * FROM films LIMIT 10;"

films <- dbGetQuery(con,sql_requete)
head(films)
```

```
##   id_film                  titre annee
## 1       1                ¡Átame!  1990
## 2       2          ¡Three Amigos!  1986
## 3       3 ...And Justice for All  1979
## 4       4       'Breaker' Morant  1980
## 5       5            'burbs, The  1989
## 6       6  'Crocodile' Dundee II  1988
```

- ✓ * permet de ne pas spécifier une colonne en particulier.
- ✓ Cette requête retournera toutes les colonnes de la table `films`
- ✓ Note: L'instruction `LIMIT` est utilisée dans les prochaines diapos afin de permettre le rendu des requêtes sur une diapo.

Sélectionner des enregistrements unique

```
sql_requete <- "SELECT DISTINCT nom, prenom
FROM acteurs LIMIT 10;"

films <- dbGetQuery(con,sql_requete)
head(films)
```

```
##           nom    prenom
## 1 Fitz-Gerald     Lewis
## 2        Gage     Kevin
## 3    Carrasco    Carlos
## 4     Vasquez David (I)
## 5   Blackmore   Stephen
## 6    Anderson  Adam (I)
```

- ✓ L'instruction `DISTINCT` permettra de retourner la combinaison unique de noms et prénoms présent dans la table acteurs.

Ordonner la table

```
sql_requete <- "
SELECT titre, annee, id_film
  FROM films ORDER BY annee DESC
;"
derniers_films <- dbGetQuery(con,sql_requete)
head(derniers_films)
```

```
##                  titre annee id_film
## 1               Breach  2007     708
## 2 Bridge to Terabithia  2007     727
## 3         Dead Silence  2007    1275
## 4           Epic Movie  2007    1573
## 5          Ghost Rider  2007    1946
## 6      Hannibal Rising  2007    2155
```

- ✓ `ORDER BY` permet de trier par ordre croissant (`ASC`) ou décroissant (`DESC`).

Critères avec NULL

```
sql_requete <- "
SELECT id_film, titre, annee
  FROM films WHERE annee IS NOT NULL
  ORDER BY annee DESC
;"
annees_films <- dbGetQuery(con,sql_requete)
head(annees_films)
```

```
##   id_film                titre annee
## 1     708               Breach  2007
## 2     727 Bridge to Terabithia  2007
## 3    1275         Dead Silence  2007
## 4    1573           Epic Movie  2007
## 5    1946          Ghost Rider  2007
## 6    2155      Hannibal Rising  2007
```

- ✓ `WHERE`, spécifie les critères de la requête.
- ✓ `annee_prod IS NULL` permet d'obtenir seulement les films n'ayant pas d'année de production.

Combiner les critères

```
sql_requete <- "
SELECT id_film, titre, annee
  FROM films WHERE
  (annee >= 1930 AND annee <= 1940)
  OR (annee >= 1950 AND annee <= 1960)
  ORDER BY annee
;"
derniers_films <- dbGetQuery(con,sql_requete)
head(derniers_films)
```

```
##   id_film                        titre annee
## 1     157 All Quiet on the Western Front  1930
## 2     239                 Animal Crackers  1930
## 3     603               Blaue Engel, Der  1930
## 4     997                     City Lights  1931
## 5    1436                         Dracula  1931
## 6    1820                     Frankenstein  1931
```

- ✓ Multi-critères avec AND et OR
- ✓ Les parenthèses définissent les priorités d'opérations.
- ✓ Opérateurs de comparaison: >=,<=, = (Valeurs numériques)
- ✓ **Documentation sur les opérateurs de comparaisons**

Critères sur le texte avec LIKE

```
sql_requete <- "
SELECT id_film, titre, annee
  FROM films WHERE titre LIKE '%Voyage%'
;"
derniers_films <- dbGetQuery(con,sql_requete)
head(derniers_films)
```

```
##   id_film                          titre annee
## 1    1662                Fantastic Voyage  1966
## 2    3654                    Now, Voyager  1942
## 3    4770 Star Trek IV: The Voyage Home  1986
## 4    5330       Voyage dans la lune, Le  1902
```

- ✓ Rechercher dans le texte: `LIKE`
- ✓ `%`: n'importe quels caractères
- ✓ `_`: un seul caractère (exemple: `_1_` peut renvoyer `113` ou encore `A1C`)
- ✓ Le critère contraire est aussi possible avec `NOT` (exemple: `WHERE titre NOT LIKE '%voyage%'`)

Exercice 2 (10 minutes)

Dans ta table `acteurs`, essayer de trouver votre acteur préféré avec `LIKE` ou avec `= 'votre_acteur_pref'`

Agréger l'information (1 ligne)

```
sql_requete <- "
SELECT avg(annee) AS moyenne,
  min(annee), max(annee)
  FROM films;"

resume_films <- dbGetQuery(con,sql_requete)
head(resume_films)
```

```
##    moyenne min(annee) max(annee)
## 1 1989.853       1902       2007
```

- ✓ Pour faire une synthèse de l'information sur une seule ligne.
- ✓ Faire des opérations sur les champs numériques: `max`, `min`, `sum`, `avg`, `count`.
- ✓ Mais aussi les opérations classiques: `*`, `/`, `-` etc.
- ✓ Renommer les colonnes avec `AS`.

Agréger l'information (plusieurs lignes par groupe)

```
sql_requete <- "
SELECT count(titre) AS nb_films, annee
  FROM films
  GROUP BY annee;"

resume_films <- dbGetQuery(con,sql_requete)
head(resume_films)
```

```
##   nb_films annee
## 1     1278    NA
## 2        1  1902
## 3        1  1915
## 4        1  1916
## 5        1  1920
## 6        1  1921
```

- ✓ `COUNT` permet de dénombrer le nombre de lignes.
- ✓ `GROUP BY` définit les champs sur lequel se fera l'agrégation des données.

Exercice 3 (10 minutes)

À l'aide de la base de données `bd_films`, dénombrer le nombre d'acteurs par films

Quels sont les 10 acteurs les plus prolifiques?

Jointures entre tables

Jointures entre tables

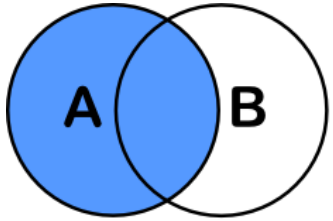
Le `INNER JOIN` est un type de jointure, renvoyant seulement les films et les acteurs ayant un identifiant `id_film` commun.

```
sql_requete <- "
SELECT titre, annee, films.id_film, acteurs.id_film
  FROM films
  INNER JOIN acteurs ON films.id_film = acteurs.id_film
  ;"

acteurs_films <- dbGetQuery(con,sql_requete)
head(acteurs_films,4)
```

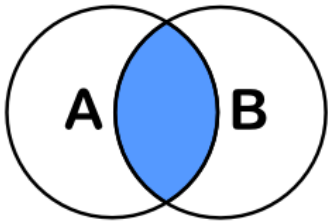
```
##                    titre annee id_film id_film
## 1        'Breaker' Morant  1980       4       4
## 2             'burbs, The  1989       5       5
## 3   'Crocodile' Dundee II  1988       6       6
## 4 *batteries not included  1987       7       7
```

Les type de jointures

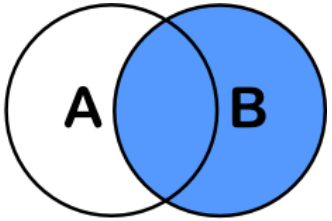


SELECT <auswahl>
FROM tabelleA A
LEFT JOIN tabelleB B
ON A.key = B.key

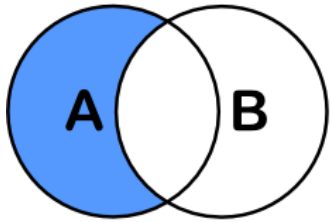
CHEATSHEET
SQL
JOINS



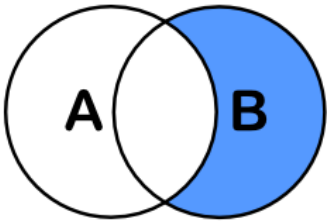
SELECT <auswahl>
FROM tabelleA A
INNER JOIN tabelleB B
ON A.key = B.key



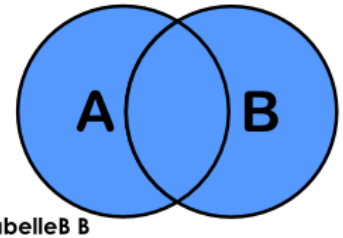
SELECT <auswahl>
FROM tabelleA A
RIGHT JOIN tabelleB B
ON A.key = B.key



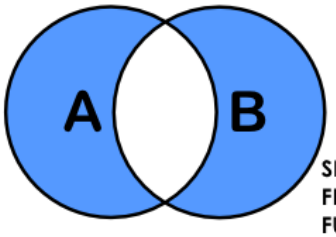
SELECT <auswahl>
FROM tabelleA A
LEFT JOIN tabelleB B
ON A.key = B.key
WHERE B.key IS NULL



SELECT <auswahl>
FROM tabelleA A
RIGHT JOIN tabelleB B
ON A.key = B.key
WHERE A.key IS NULL



SELECT <auswahl>
FROM tabelleA A
FULL OUTER JOIN tabelleB B
ON A.key = B.key



SELECT <auswahl>
FROM tabelleA A
FULL OUTER JOIN tabelleB B
ON A.key = B.key
WHERE A.key IS NULL
OR B.key IS NULL

Jointures entre tables

On peut spécifier la jointure avec `USING` seulement si les deux clés possèdent le même nom.

```
sql_requete <- "
SELECT titre, annee, nom, prenom
  FROM films
  INNER JOIN acteurs USING (id_film)
  ;"

acteurs_films <- dbGetQuery(con,sql_requete)
head(acteurs_films,4)
```

```
##                     titre annee         nom    prenom
## 1        'Breaker' Morant  1980 Fitz-Gerald     Lewis
## 2             'burbs, The  1989        Gage     Kevin
## 3   'Crocodile' Dundee II  1988    Carrasco    Carlos
## 4 *batteries not included  1987     Vasquez David (I)
```

Exercice 4 (5 minutes)

Exercice 5 (10 minutes)

Requêtes emboitées

```
sql_requete <- "
SELECT annee, avg(nb_acteurs) AS moy_acteurs FROM (
  SELECT titre, annee, count(nom) AS nb_acteurs
    FROM films
    INNER JOIN acteurs USING (id_film)
    GROUP BY annee, titre
) AS nb_acteurs_film
GROUP BY annee;"

moy_acteurs <- dbGetQuery(con,sql_requete)
head(moy_acteurs)
```

```
##   annee moy_acteurs
## 1    NA     25.2723
## 2  1902      9.0000
## 3  1915     55.0000
## 4  1916     59.0000
## 5  1920     11.0000
## 6  1921     59.0000
```

✓ On s'interroge sur le nombre moyen d'acteurs par années.

✓ Pour ce faire, on peut bâtir une requête à partir d'une autre requête.

Filtrer les requêtes à posteriori

```
sql_requete <- "
SELECT annee, avg(nb_acteurs) AS moy_acteurs FROM (
  SELECT titre, annee, count(nom) AS nb_acteurs
    FROM films
    INNER JOIN acteurs USING (id_film)
    GROUP BY annee, titre
) AS nb_acteurs_film
GROUP BY annee
HAVING avg(nb_acteurs) > 10;"

nb_acteurs <- dbGetQuery(con,sql_requete)
head(nb_acteurs)
```

```
##   annee moy_acteurs
## 1    NA     25.2723
## 2  1915     55.0000
## 3  1916     59.0000
## 4  1920     11.0000
## 5  1921     59.0000
## 6  1922     17.0000
```

- ✓ Il est possible de filtrer à posteriori sur la requête avec `HAVING`.

Sauvegarder les requêtes

Sauvegarder une requête

Afin de sauvegarder les requêtes obtenues dans R par `dbGetQuery()`, il est possible d'utiliser les fonctions d'écritures tels que `write.table()` ou encore `write.csv()`.

Manipuler les enregistrements

Mettre à jour des enregistrements

On peut mettre à jour des enregistrements d'une table avec des critères spécifiques.

```
UPDATE films SET genre = 'Dramatique' WHERE genre = 'Drame';
```

Note: On peut pas faire de modifications d'enregistrements sur des vues, seulement sur les tables directement.

Documentation sur la commande UPDATE

Supprimer des enregistrements

On peut supprimer des enregistrements d'une table avec des critères spécifiques.

```
DELETE FROM films WHERE genre <> 'Comédie musicale';
```

Ou sans critères, pour supprimer tous les enregistrements.

```
DELETE FROM films;
```

Documentation sur la commande DELETE

Travail de la semaine

Travail de la semaine

1. Créer la base de données
2. Injecter les données
3. Faire les requêtes suivantes :
    - ✓ Nombre de liens par étudiant
    - ✓ Décompte de liens par paire d'étudiants
4. Enregistrer le résultat des requêtes dans un fichier csv (voir la diapo 34 de ce cours)

Travail de la semaine

1. En post-traitement sur R :
    - ✓ Calculer le nombre d'étudiants, le nombre de liens et la connectance du réseau
    - ✓ Calculer le nombre de liens moyens par étudiant et la variance
2. Écrire un script qui réalise les étapes 1-3 d'un bloc

Vous devez remettre les 5 scripts pour chacune de ces étapes ainsi que le script final qui les exécute l'une après l'autre. Assurez vous que le script fonctionne sur la machine virtuelle et entre des utilisateurs différents.

Lectures

- ✓ Poisot et al. 2014. Moving toward a sustainable ecological science: don't let data go to waste ! Ideas in Ecology and Evolution 6: 11-19
- ✓ Mills et al. 2015. Archivin Primary Data: Solutions for Long-term Studies. Trends in Ecology and Evolution.

Discussion