

Séance 2: La gestion des données biologiques

BIO 500 - Méthodes en écologie computationnelle

Steve Vissault & Dominique Gravel
Laboratoire d'écologie intégrative

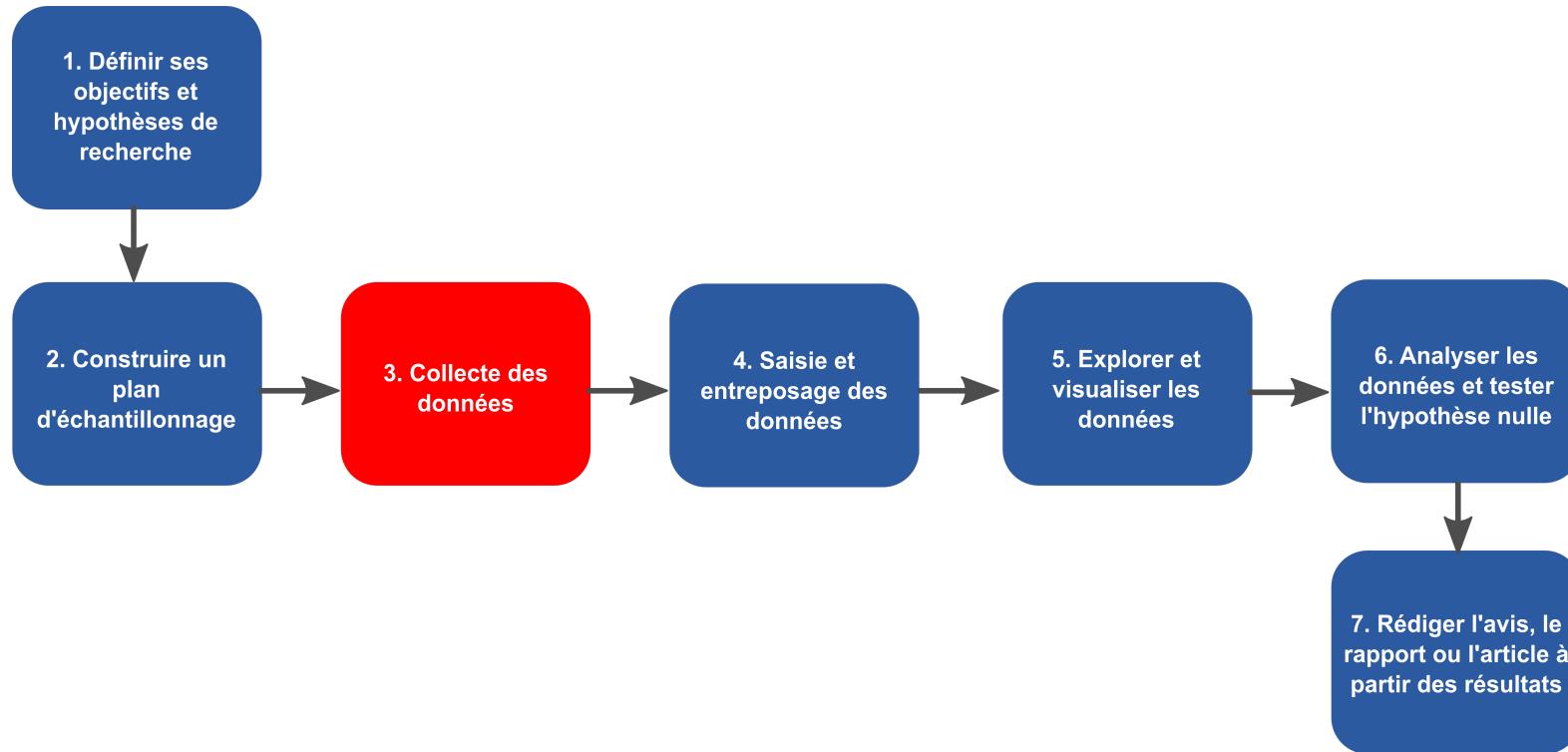


Séance 2

- ✓ 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 données en biologie

La collecte de données

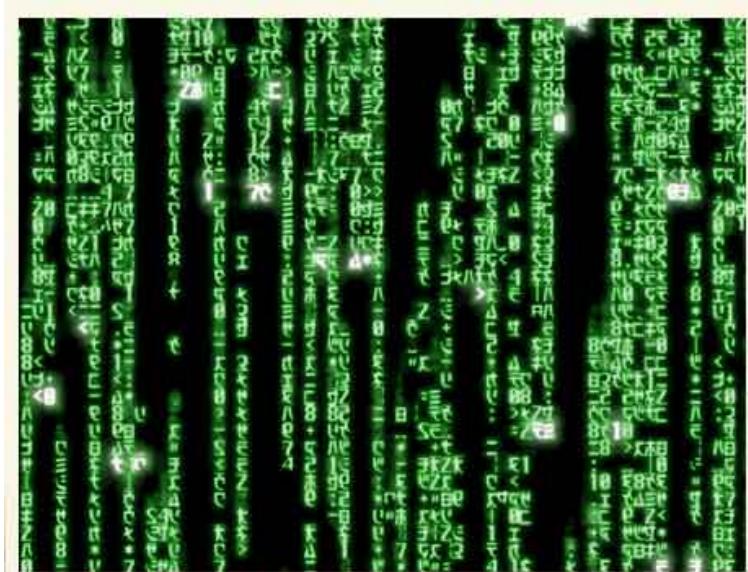


Le constat

Trop souvent en écologie, les données sont représentées et entreposées dans un format proche des analyses que l'on veut réaliser.

Par exemple, on utilise souvent une matrice *sites* \times *espèces* pour analyser la structure des communautés.

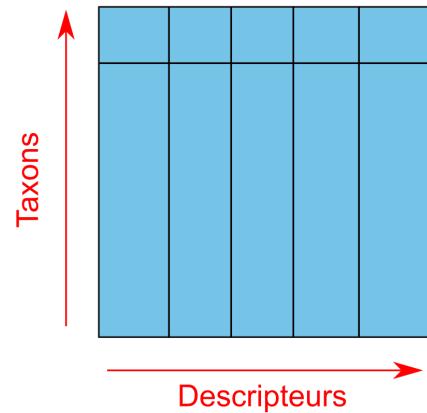
La question scientifique ne devrait jamais conditionner notre façon de stocker l'information sur un système écologique (données brutes).



La collecte de données en biologie

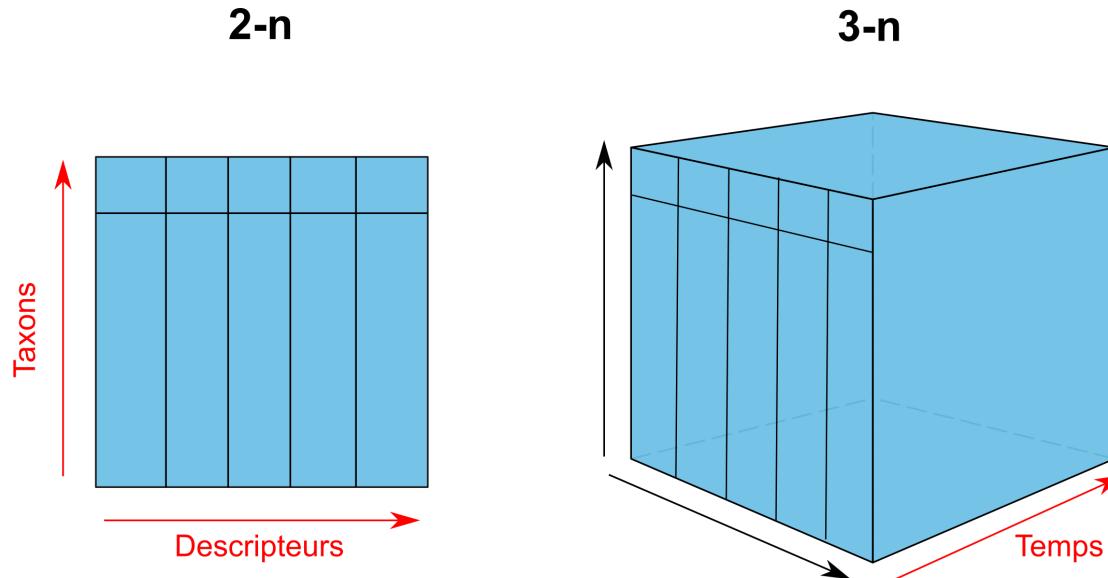
D'abord, qu'est ce qu'une donnée en écologie?

2-n



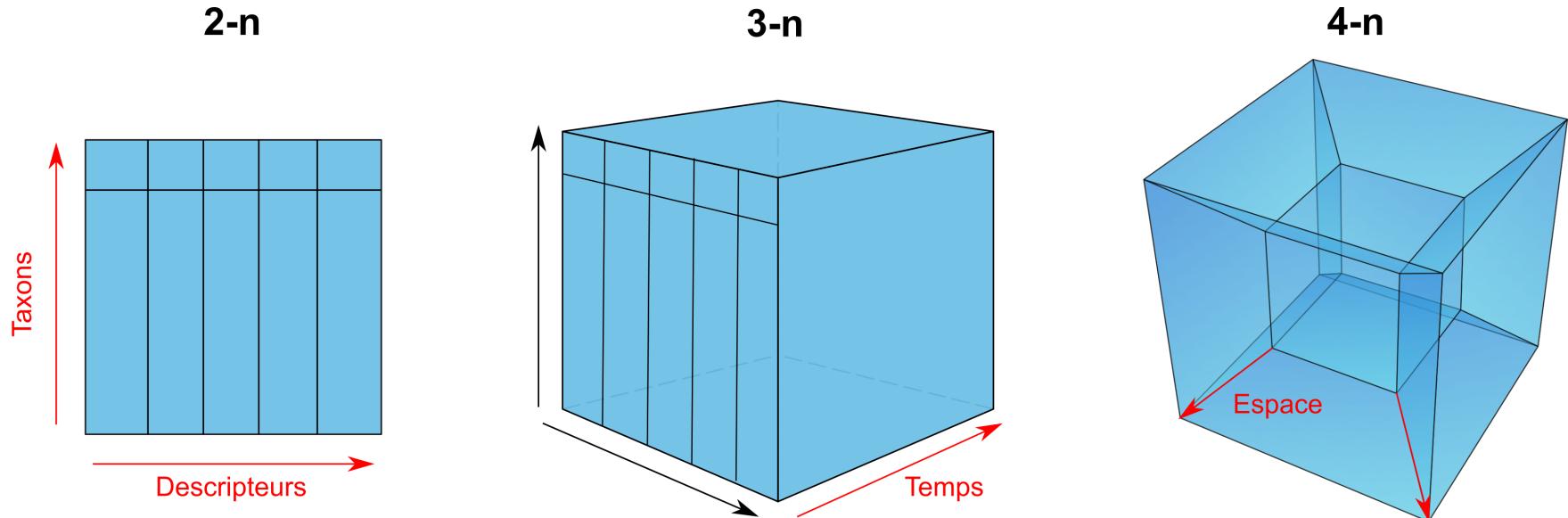
La collecte de données en biologie

Le problème de multi-dimensionnalité



La collecte de données en biologie

Le problème de multi-dimensionnalité



La collecte de données en biologie

En biologie, on classifie les données selon 4 dimensions/classes d'information:

1. Environnementale
2. Taxonomique
3. Temporelle
4. Spatial

Au sein de ce cours, nous nous attarderons à la façon de structurer ces données. Les spécificités propres à chacune de ces dimensions seront présentées. D'abord le format des données, puis les types de données.

Le format des données

Le format des données



Format large

ID	esp	2010	2011	2014
567-1	acsma	460	NA	NA
567-2	acsma	100	NA	NA
567-3	acsma	120	NA	NA
598	piru	NA	380	NA
876	abba	NA	NA	160

- ✓ Privilégier le format long
- ✓ Une ligne = une observation



Format long

ID	esp	annees	dhp_mm
567-1	acsma	2010	460
567-2	acsma	2010	100
567-3	acsma	2010	120
598	piru	2011	380
876	abba	2014	160

- ✓ Noms de colonnes courts, sans accents, sans espaces et explicites.
- ✓ Attacher les unités au nom de la colonne (si absence de métadonnées).

Le format des données: tableaux



Un tableau doit contenir un type d'information

ID_plot	ID_arbre	ID_multi	esp	annees	dhp_mm
A	567	1	acs	2010	460
A	567	2	acs	2010	100
A	567	3	acs	2010	120
B	598	NA	piru	2011	380
B	876	NA	abba	2014	160

ID_plot	annees	variable	valeur
A	2010	pp_tot_mm	880
B	2011	pp_tot_mm	560
B	2014	pp_tot_mm	900
A	2010	temp_max_deg	24
B	2011	temp_max_deg	26
B	2014	temp_max_deg	28

- ✓ Si l'on veut ajouter des données sur le climat, on ajoutera un nouveau tableau.

Le format des données: colonnes



Ne pas agréger l'information dans une seule colonne

ID_arbre	esp	annees	dhp_mm
567-1	acsa	2010	460
567-2	acsa	2010	100
567-3	acsa	2010	120
598	piru	2011	380
876	abba	2014	160

ID_arbre	ID_multi	esp	annees	dhp_mm
567	1	acsa	2010	460
567	2	acsa	2010	100
567	3	acsa	2010	120
598	NA	piru	2011	380
876	NA	abba	2014	160

- ✓ Une colonne = une information

Le format des données: colonnes

Important: votre fichier de données brutes (destiné au stockage à long terme) ne doit pas contenir de champ calculé (c.a.d. une nouvelle colonne avec une moyenne, etc..)

Les types de données

Les types de données en informatique

En informatique, on distingue plusieurs types de données:

Appellation	Type	Valeurs	Taille
BOOLEAN	Boléen	vrai/faux	1 octet
INTEGER	Entiers	-998, 123	1 à 4 octets
DOUBLE, FLOAT, REAL	Nombres réels	9.98, -4.34	4 à 8 octets
CHAR, VARCHAR	Chaine de caractères	lapin	n x 1 à 8 octets
TIMESTAMP, DATE, TIME	Dates et heures	1998-02-16	4 à 8 octets

- ✓ Ce sont ces types qui seront utilisés pour entreposer nos données.
- ✓ Le choix d'un type approprié permet de réduire la taille du fichier de données.

Les données temporelles

Les données temporelles

La plupart des langages/programmes disposent d'un type **TIMESTAMP**, **DATE** et **TIME** pour représenter une donnée temporelle.

On utilisera préférablement la norme **ISO8601** pour représenter ces données.

- ✓ **TIMESTAMP** (Heure et temps): **YYYY-MM-ddThh:mm:ss**. ex. **1977-04-22T01:00:00-05:00** ou **1977-04-22T06:00:00Z**
- ✓ **DATE**: **YYYY-MM-dd**. ex. **1997-04-22**
- ✓ **TIME**: **HH:mm:ss** dans un système de 24 heures. ex. **01:30:00**.

Gardez à l'esprit que vos données pourraient être réutilisées à travers le monde.

Les données taxonomiques

Les données taxonomiques

Un exemple avec l'érable à sucre

Selon vous quelle option est la meilleure?

Option	Exemple
1. Code spécifique à l'étude	ACSA
2. Code du ministère	ERS
3. Genre et espèce	<i>Acer saccharum</i>
4. Nom vernaculaire	Érable à sucre
5. Numéro Taxonomique (TSN - ITIS)	28731



Les données taxonomiques

Un exemple avec l'érable à sucre

Option	Exemple
1. Code spécifique à l'étude	ACSA
2. Code du ministère	ERS
3. Genre et espèce	Acer saccharum
4. Nom vernaculaire	Érable à sucre
5. Numéro Taxonomique (TSN - ITIS)	28731

- ✓ ✗ **Option 1 et 2:** Doit être associé à des métadonnées. Risque de perte du fichier attaché.
- ✓ ✗ **Option 3:** Le genre et l'espèce peuvent changer à travers le temps.
- ✓ ✗ **Option 4:** Le nom vernaculaire des espèces est le pire choix. Le nom vernaculaire est propre à un pays, à une région géographique, à une culture/dialecte.

Les données taxonomiques

Un exemple avec l'érable à sucre

Option	Exemple
1. Code spécifique à l'étude	ACSA
2. Code du ministère	ERS
3. Genre et espèce	Acer saccharum
4. Nom vernaculaire	Érable à sucre
5. Numéro Taxonomique (TSN - ITIS)	28731

- ✓ **Option 5:** Cette option couplée à l'option 3, est le meilleur choix.

Les données taxonomiques

On priviliege généralement l'utilisation de code d'espèce standardisée:

1. **ITIS**
2. **VASCAN** (Plantes vasculaires du Canada)
3. **NCBI**
4. **BOLD** (Projet code barre)

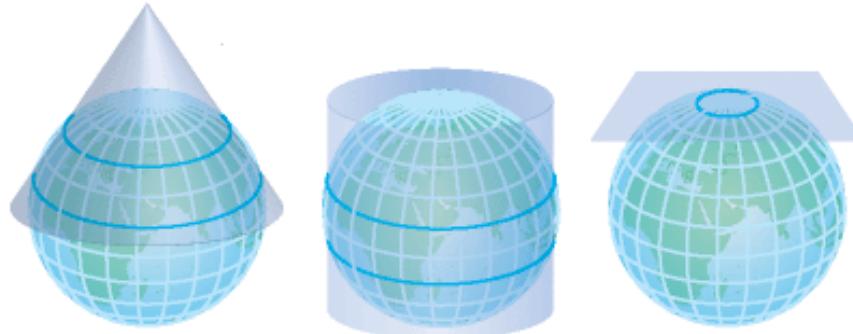
Avantage: Chacune de ces institutions/infrastructures nous permet de valider et retirer l'ensemble de la classification taxonomique d'une espèce à partir de son code. Même si l'identifiant change (nouvelle classification), nous serons en mesure de trouver le nouvel identifiant taxonomique à partir de l'ancien.

Exemple: **TSN - 28731**

Les données spatiales

Les données spatiales

Il existe plus de **65 familles de projections géographiques** pour représenter des coordonnées sur la planète, en voici 3 des plus connues:



- ✓ Il est important de choisir un bon système de projection pour minimiser la déformation spatiale (surtout à nos latitudes)
- ✓ À nos latitudes, on privilégiera l'utilisation d'une projection conique. Les ministères du Québec conseillent généralement l'utilisation d'une **projection conique conforme de Lambert**.

Les données spatiales

- ✓ **Ce qu'il est important de savoir:** des coordonnées spatiales sans système de projection ne veulent strictement rien dire.
- ✓ Ainsi, lorsque l'on entrepose des données spatiales, trois colonnes doivent être représentées:
 - La coordonnée en X
 - La coordonnée en Y
 - La projection écrite en texte (voir votre GPS), ou préférablement l'identifiant unique de la projection.

Les données spatiales

Une base de données connue permet de fournir des identifiants uniques:

- ✓ **SRID**: *Spatial reference system*.

Cet identifiant est unique et peut être trouvés à cette adresse: <http://spatialreference.org/>

Exemple: <http://spatialreference.org/ref/epsg/2138/>

L'absence de données

L'absence de données

On peut représenter l'absence de données de plusieurs façons:

- ✓ Laisser la cellule vide (**NULL**)
- ✓ Mettre un **NA** (*Not Available*)
- ✓ Mettre un **0**
- ✓ Mettre **-9999** dans une colonne numérique

Selon vous, quel est le choix le plus approprié ?

Le format des données

On peut représenter l'absence de données de plusieurs façons:

- ✓ Laisser la cellule vide: montre que l'information n'a pas été saisie (un oubli)
- ✓ Mettre un **NA** (*Not Available*): Montre que l'information est réellement absente (car le NA est saisie par un humain).
- ✓ ~~Mettre un 0~~: **JAMAIS** (empêche la distinction entre un vrai d'un faux 0, influence la moyenne)
- ✓ Mettre **-9999** dans une colonne numérique: Ce choix peut être utilisé seulement pour les jeux de données très importants (centaine de Megas-octet), et doit être référencé dans les métadonnées.

Choisir le bon type et format de données

Si l'on ne choisit pas le type de données approprié, cela aura diverses conséquences:

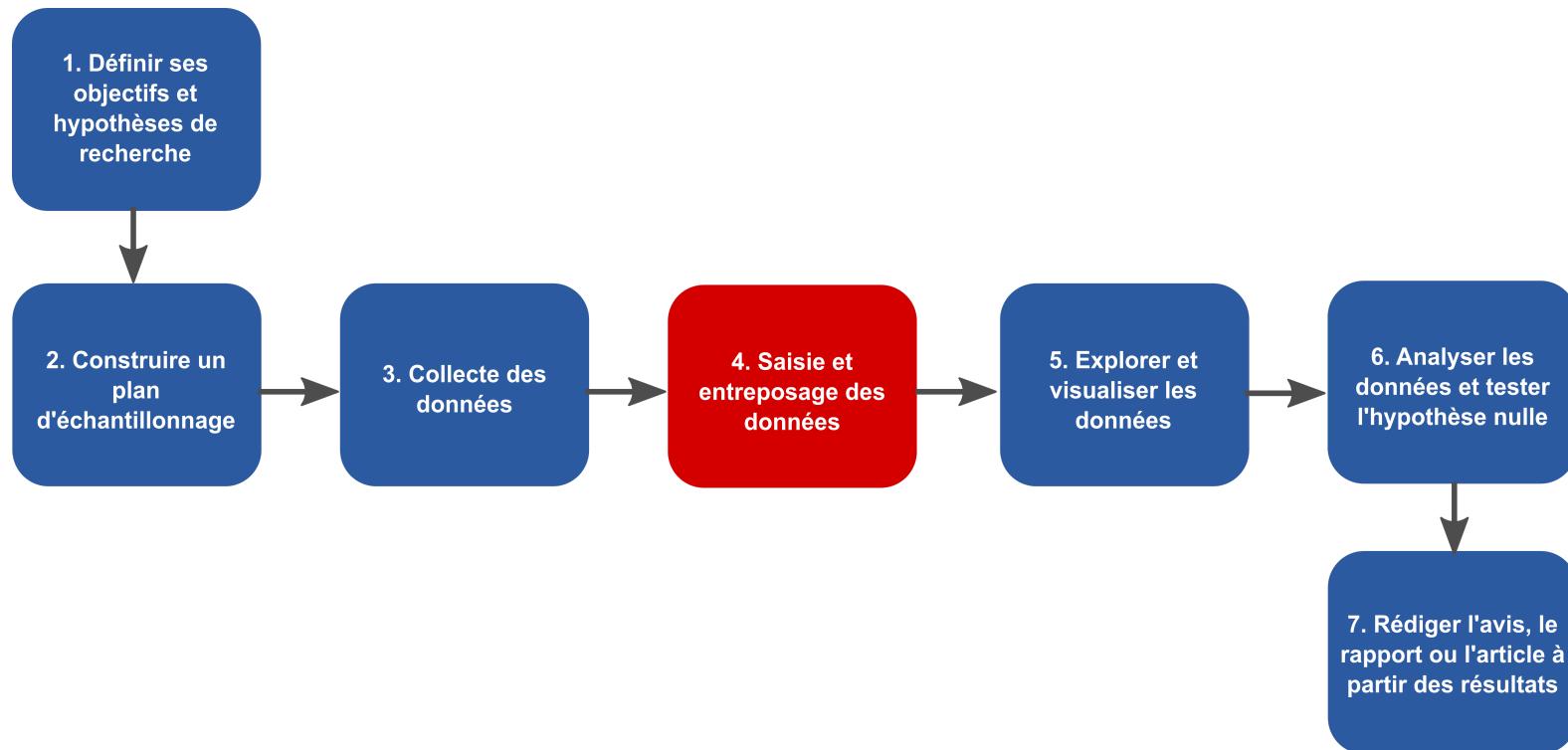
- ✓ Des problèmes de performance (ex. : il est plus rapide de faire une recherche sur un nombre que sur une chaîne de caractères)
- ✓ Un comportement contraire à celui attendu (ex. : trier sur un nombre stocké comme tel, ou sur un nombre stocké comme une chaîne de caractères ne donnera pas le même résultat)
- ✓ L'impossibilité d'utiliser des fonctionnalités propres à un type de données (ex. : stocker une date comme une chaîne de caractères vous prive des nombreuses fonctions temporelles disponibles).

Pour en savoir davantage:

- ✓ **Broman KW, Woo K (2017) Data organization in spreadsheets. The American Statistician.**
- ✓ **Hart EM, Barmby P, LeBauer D, Michonneau F, Mount S, Mulrooney P, et al. (2016) Ten Simple Rules for Digital Data Storage. PLoS Comput Biol**

Entreposer et archiver ses données écologiques

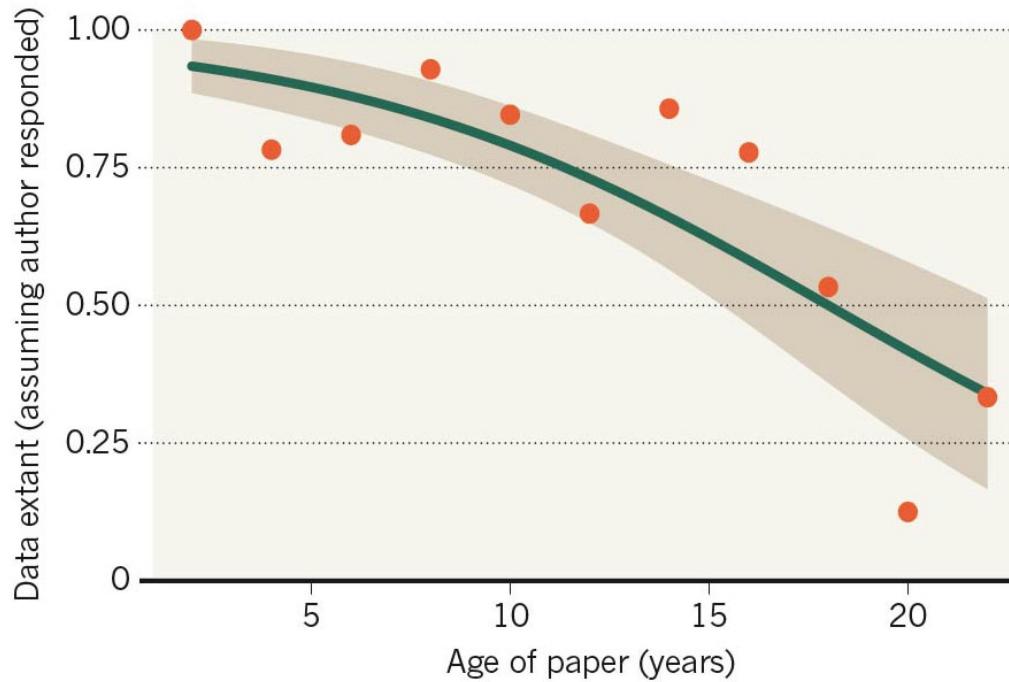
Où sommes-nous?



Pourquoi bien entreposer ses données?

MISSING DATA

As research articles age, the odds of their raw data being extant drop dramatically.



Vines et al., 2013

Les entrepôts existants

1. Les **fichiers textes** comme les CSV, TSV (Format libre et ouvert)
2. Les **tableurs** comme MS Excel (Logiciel propriétaire), Libre Office Calc. (Logiciel libre)
3. Les **fichiers hierarchiques/structurés** HDF, NetCDF (Format libre et ouvert)
4. Les **bases de données relationnelles**

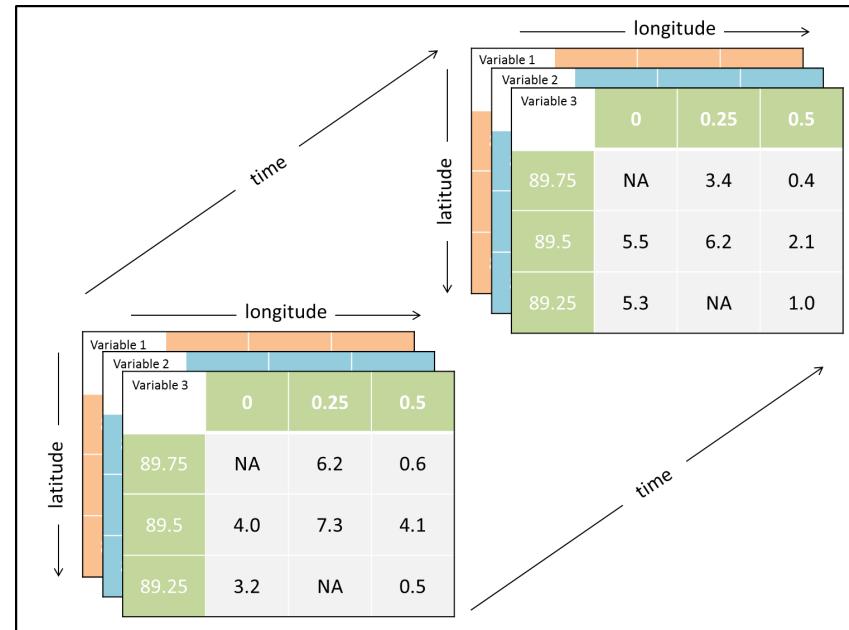
Exemple de fichier CSV

```
#Numero;X;Y;Z;profAtypique;cod_PGOC;Cod_Proj;Commentaire
1;528949.9333;151556.5564;135.6000;;C_BT;LZ3;
2;528949.9333;151556.5564;135.6000;;AFF_POS;LZ3;
3;528949.9601;151554.2897;135.8000;;C_BT;LZ3;
4;528950.6514;151553.0281;135.8200;;C_BT;LZ3;
5;528952.2443;151552.3536;135.8500;;C_BT;LZ3;
6;528960.0315;151552.3410;135.5000;;C_BT;LZ3;
7;528961.1426;151551.9886;135.4300;;C_BT;LZ3;
8;528961.4955;151551.1376;135.4000;;C_BT;LZ3;
9;528961.5159;151545.1582;135.1000;;C_BT;LZ3;
10;528961.5159;151545.1582;135.1000;;ACC_BJ;LZ3;
11;528961.6446;151536.1053;135.0200;;C_BT;LZ3;
12;528968.1905;151535.8394;134.9000;0.350;C_BT;LZ3;
13;528968.1931;151535.6266;134.8800;0.380;C_BT;LZ3;
14;528961.7701;151535.2689;135.2000;;C_BT;LZ3;
15;528961.5575;151526.1103;135.6000;;C_BT;LZ3;
16;528962.2934;151524.6042;135.6500;;C_BT;LZ3;
```

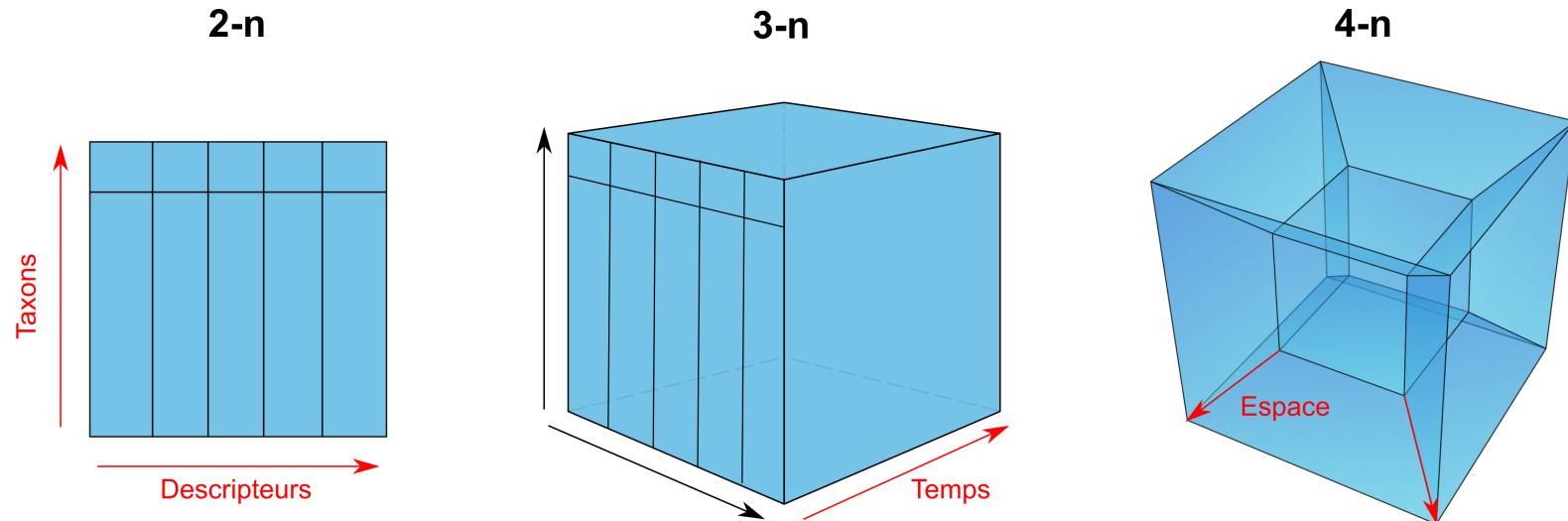
Les entrepôts existants

1. Les **fichiers texte** comme les CSV, TSV (Format libre et ouvert)
2. Les **tableurs** comme MS Excel (Logiciel propriétaire), Libre Office Calc. (Logiciel libre)
3. Les **fichiers hiérarchiques/structurés** HDF, NetCDF (Format libre et ouvert)
4. Les **bases de données relationnelles**

Structure NetCDF

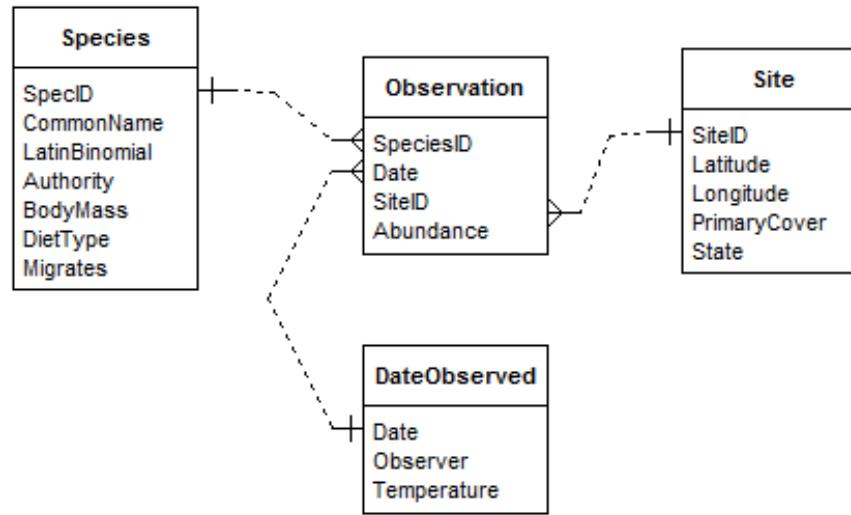


Le Tesseract de la biologie



- ✓ Il est difficile de stocker les données écologiques dans un tableau ($n-2$) lorsque les données écologiques ont ($n-4$).
- ✓ Conduit à une redondance dans l'information (par exemple. répéter les coordonnées de l'emplacement du site lorsqu'il est mesuré plusieurs fois).

Les bases de données (BDs) à la rescousse



- ✓ Les BDs permettent de redimensionner ce problème (plusieurs tableaux de n-2 avec des relations) grâce au modèle d'entités-relations.
- ✓ Chaque table correspond à une dimension. Les tables sont liées entre elles par des relations. Cette structure est appelée **schéma en étoile**.

Avantages des bases de données

- ✓ **Maintenir l'intégrité entre les enregistrements de nos tableaux.** Une observation ne peut être faite sur un site qui n'existe pas.
- ✓ **Normaliser et contrôler la qualité des données.** Chaque colonne est un type précis de données. Des contraintes peuvent être appliquées sur chaque colonne.
- ✓ **Éviter les redondances dans le stockage de l'information** (obtenir une **forme normale**), voir la section **Format de donnée du cours 2**.

Conceptualisation d'une base de données en 5 étapes

Étape 1. Faire une liste des variables

1. Dresser la liste des informations collectées par les différents groupes.
2. Regrouper les variables communes entre les équipes pour obtenir un format de données conjoint.

Étape 2. Regrouper les variables dans des tables

1. Déterminer les tables/entités:

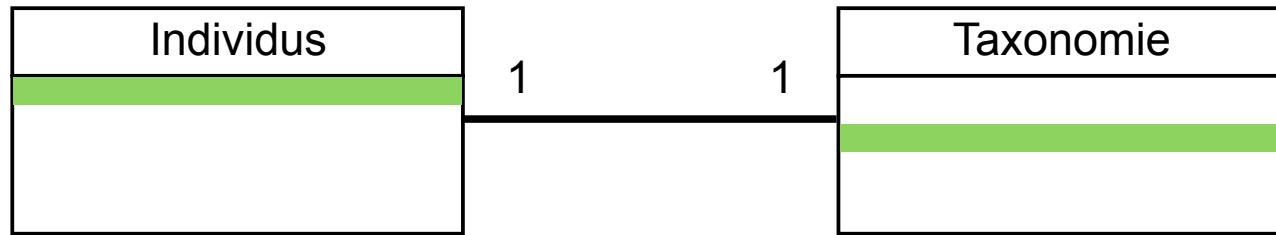
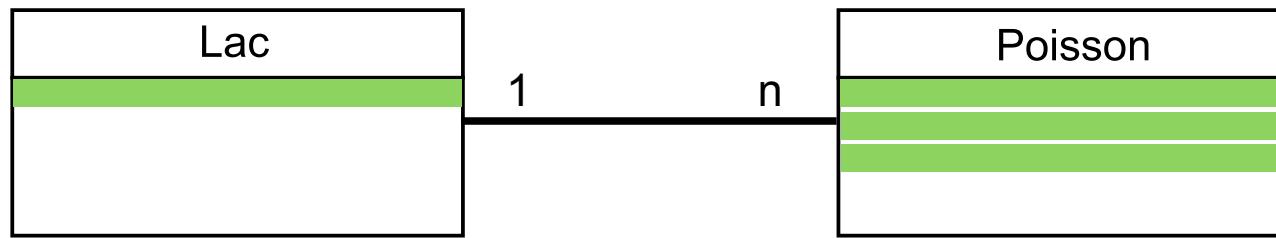
- ✓ Qu'elles sont les unités d'échantillonnage? Autrement dit, sur quelles entités portent nos mesures?

2. Remplir les tables avec les variables de l'étape 1.

À ce stade de la conceptualisation, une table est une entité possédant des attributs. Chaque attribut est une colonne.

Étape 3. Établir le type d'association entre les tables

Le concept d'association



Étape 3. Établir le type d'association entre les tables

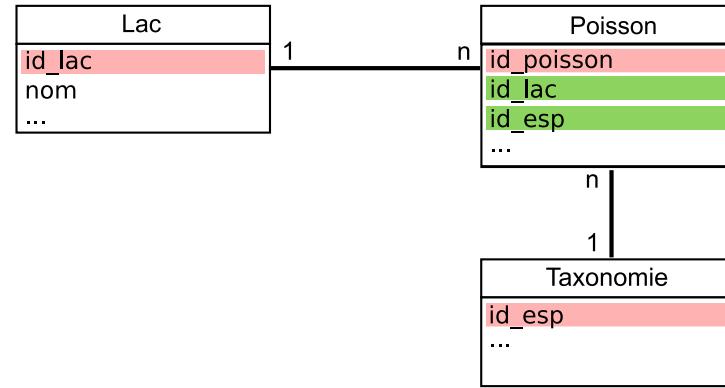
Il existe plusieurs types d'association:

Table 1	Table 2	Relation	exemple
1	1	<i>one-to-one</i>	personne ↔ permis de conduire
0..1	1	optionnel dans la table 1, <i>one-to-one</i>	permis de conduire ↔ personne
0..n ou n	0..n or n	optionnel dans les deux tables, <i>many-to-many</i>	personne ↔ livre
1..n ou n	1	<i>many-to-one</i>	personne ↔ lieu de naissance

Exercice (5 minutes): Quel(s) type(s) d'association retrouve-t-on entre nos tables?

Étape 4. Établir les clés primaires et étrangères

Le concept des clés primaires et des clés étrangères



Important:

- ✓ Une clé primaire garanti le caractère unique d'un enregistrement (ligne d'une table).
- ✓ Une clé primaire ne peut donc jamais être **NULL**.
- ✓ Une clé primaire peut être composite, une combinaison de colonnes.

Étape 4. Établir les clés primaires et étrangères

1. Déterminer quels sont les attributs/colonnes garantissant le caractère unique d'un enregistrement (ligne d'une table).
2. Déterminer quelles sont les clés étrangères.

Étape 5. Assigner un type de données aux attributs

Chaque attribut d'une table doit correspondre à un type de données:

Appellation	Type	Valeurs	Taille
BOOLEAN	Boléen	vrai/faux	1 octet
INTEGER	Entiers	-998, 123	1 à 4 octets
DOUBLE, FLOAT, REAL	Nombres réels	9.98, -4.34	4 à 8 octets
CHAR, VARCHAR	Chaine de caractères	lapin	n x 1 à 8 octets
TIMESTAMP, DATE, TIME	Dates et heures	1998-02-16	4 à 8 octets

Pour tous les types de données, **voir la documentation SQLite3**

En résumé

Finalement, qu'est-ce qu'un modèle conceptuel pour une base de données?

Une façon de représenter l'information dans un modèle de type entités-relations où chaque entité (table) possède des attributs (colonnes).

L'étape suivante est de se connecter à la base de données afin de transcrire ce modèle conceptuel en modèle logique (c.a.d compréhensible par l'ordinateur).

Les Systèmes de Gestion de Base de Données (SGBDs)

La diversité des SGBDs

Il en existe une multitude:



- ✓ Pour créer, interroger, gérer et maintenir des bases de données, on utilisera un **Système de Gestion de Base de Données** (souvent appelé **SGBD**).
- ✓ Mais ces systèmes disposent tous d'un dénominateur commun: le **langage SQL**
- ✓ Dans ce cours, nous utiliserons le système de gestion de données **SQLite3** (**Approche fichier de base de données**).

Le langage SQL

Définition

“ Le SQL (Structured Query Language) est le langage des SGBDs. Il permet de communiquer avec une base de données.”

Le langage SQL

Le **SQL** permet de:

1. Créer une base de données (**CREATE DATABASE**).
2. Créer des tables et établir des relations (**CREATE TABLE**).
3. Insérer des données (**INSERT**).
4. Interroger les données par requête (**SELECT**).
5. Supprimer des données ou des tables (**DROP, DELETE**).
6. Mettre à jour des données ou des tables (**UPDATE, ALTER**).
7. Supprimer la base de données (**DROP DATABASE**).

Chacune de ces commandes est une instruction **SQL** envoyée au serveur pour manipuler et interroger la base de données.

Le langage SQL

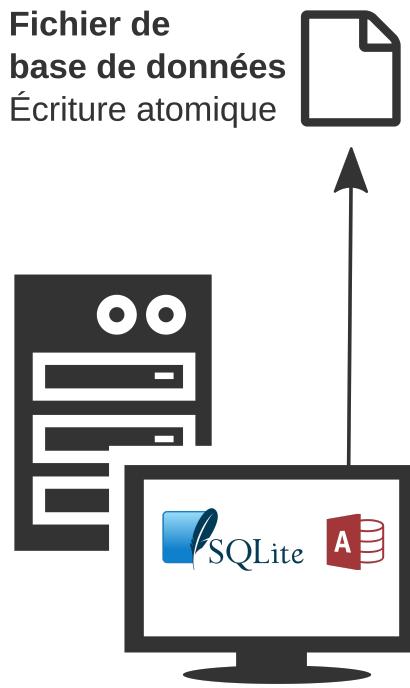
Pour cette séance, nous nous attarderons seulement à:

1. Créer une base de données (**CREATE DATABASE**).
2. Créer des tables et établir des relations (**CREATE TABLE**).
3. Supprimer ou modifier des tables (**DROP TABLE, ALTER TABLE**).
4. Supprimer la base de données (**DROP DATABASE**).

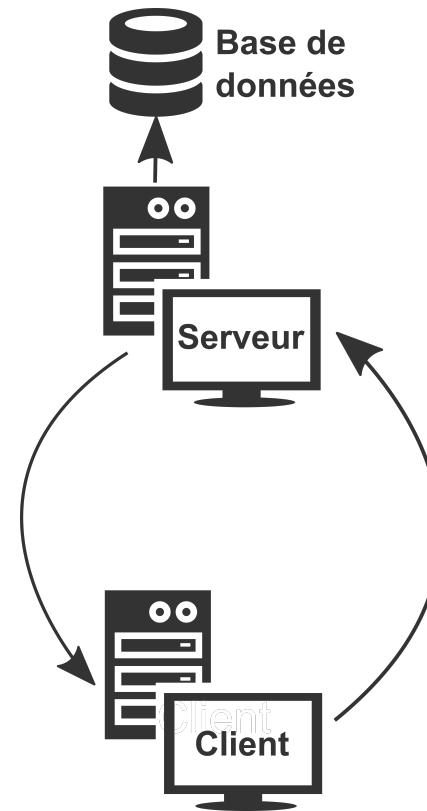
Soyez attentifs, car le travail de cette semaine consiste à écrire un script qui permet la création de la base de données (les tables et leurs relations) pour entreposer les données que vous aurez collectées pour le travail de session.

Deux approches avec les bases de données

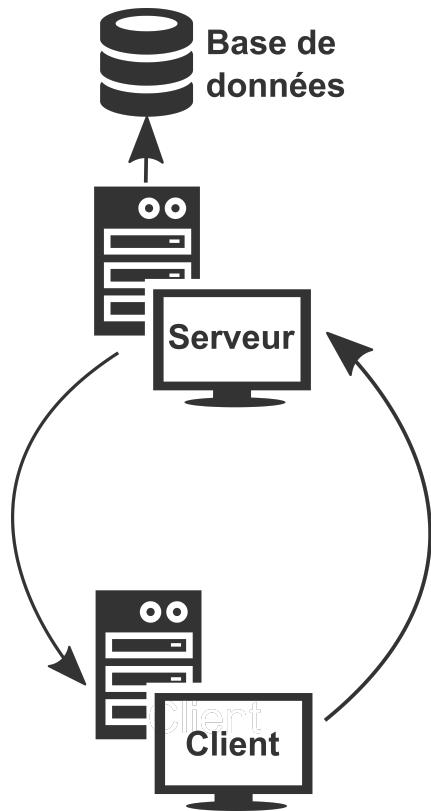
Fichier de base de données



Serveur de base de données

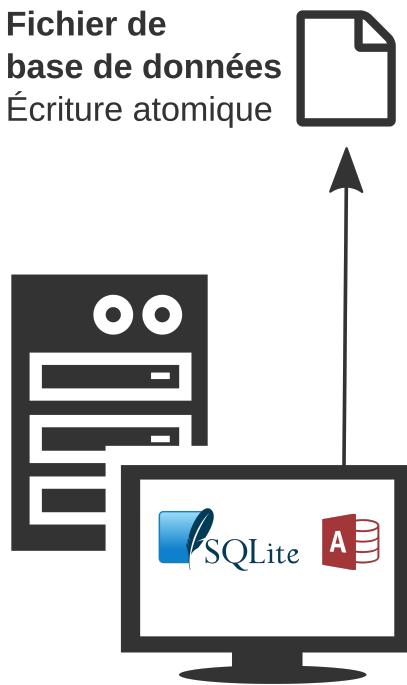


L'approche: Serveur de base de données



- ✓ Le **client** est un logiciel installé sur votre ordinateur.
- ✓ On se sert de ce logiciel pour interagir avec le serveur de base de données présent localement ou à distance.
- ✓ **Avantage:** On peut avoir plusieurs **clients** connectés sur un même serveur (contexte multi-utilisateurs).

L'approche: Fichier de base de données



- ✓ On se connecte au fichier a l'aide d'un **client** (logiciel)
- ✓ **Avantage:** Portabilité
- ✓ **Désavantage:** On ne peut pas se connecter à plusieurs utilisateurs en même temps.

Pratique: Du modèle conceptuelle vers le modèle logique

Les grandes étapes avec SQLite3

1. Créer et se connecter au fichier de base de données
2. Créer les tables et spécifier les clés
3. Ajouter de l'information dans les tables
4. Faire des requêtes pour extraire l'information

Se connecter au fichier la BD (SQLite3) via R

- ✓ **con** est un objet contenant la connexion avec le serveur/fichier de base de données.
- ✓ On utilisera la fonction **dbSendQuery()** pour envoyer les instructions SQL.
- ✓ Le deuxième argument de la fonction **dbSendQuery()** est une chaîne de caractères contenant les instructions SQL.

```
library(RSQLite)
con <- dbConnect(SQLite(), dbname=".assets/data/films.db")
## !ATTENTION!: Ceci est mon chemin d'accès vers le fichier!
## Astuces: getwd() et setwd()
```

```
dbSendQuery(con, "Instructions SQL à envoyer;")
```

Création d'une première table avec clé primaire

Voici un exemple d'instruction SQL pour créer la table **films**.

```
CREATE TABLE films (
    code      VARCHAR(5),
    titre     VARCHAR(40),
    did       INTEGER,
    date_prod DATE,
    genre     VARCHAR(10),
    duree     INTEGER,
    PRIMARY KEY(code,titre)
);
```

- ✓ **films** est le nom de la table
- ✓ Chaque attribut de la table (**code**,**titre** etc) dispose d'un type de données (**char(5)**, **varchar(40)** etc) **Type de données SQLite**
- ✓ La dernière ligne correspond aux contraintes de la table telle que la clé primaire.
- ✓ **Question:** Cette clé primaire est composite ou simple?

Création d'une table avec clé étrangère

Si l'on veut créer une table **acteurs** et référencer cette table à la table **films**.

```
CREATE TABLE acteurs (
    nom      VARCHAR(40),
    prenom   VARCHAR(40),
    naissance DATE,
    code     CHAR(5),
    titre    VARCHAR(40),
    PRIMARY KEY (nom, prenom),
    FOREIGN KEY (code, titre) REFERENCES
        films (code, titre) ON DELETE CASCADE
);
```

- ✓ On déclare **prenom** et **nom** comme étant la clé primaire de la table **acteurs**.
- ✓ On référence les attributs **code** et **titre** comme étant la clé étrangère.

Création d'une table avec clé étrangère

Si l'on veut créer une table **acteurs** et référencer cette table à la table **films**.

```
CREATE TABLE acteurs (
    nom      VARCHAR(40),
    prenom   VARCHAR(40),
    naissance DATE,
    code     CHAR(5),
    titre    VARCHAR(40),
    PRIMARY KEY (nom, prenom),
    FOREIGN KEY (code, titre) REFERENCES
        films (code, titre) ON DELETE CASCADE
);
```

Important:

- ✓ On ne peut plus insérer d'acteurs jouant dans un film qui n'est pas référencé dans la table **films**. C'est ce que l'on appelle l'intégrité référentielle.
- ✓ Lorsque l'on supprime un enregistrement dans **films**, les acteurs référencés à ce film vont être automatiquement supprimés grâce à l'instruction **CASCADE**.

Ajout de contraintes à une table

SQL présente également l'avantage de pouvoir mettre des contraintes sur les champs:

```
CREATE TABLE films (
    code      VARCHAR(5) NOT NULL,
    titre     VARCHAR(40) NOT NULL,
    did       INTEGER,
    date_prod DATE,
    genre     VARCHAR(10) DEFAULT "COMEDIE",
    duree     INTEGER CHECK( duree > 0 ),
    PRIMARY KEY(code,titre)
);
```

- ✓ Les contraintes **NOT NULL** sur la clé primaire ne sont pas obligées d'être définis.

Création d'une table avec R

On se sert de R pour envoyer l'instruction SQL de création de la table:

```
films_sql <- "
CREATE TABLE films (
  code      VARCHAR(5),
  titre     VARCHAR(40),
  did       INTEGER,
  date_prod DATE,
  genre     VARCHAR(10),
  duree    INTEGER,
  PRIMARY KEY(code,titre)
);"

dbSendQuery(con,films_sql)
dbListTables(con)
```

Création d'une table

Exercice pour le travail de session (20 minutes):

En vous inspirant des **exemples** et de la syntaxe SQL expliquée précédemment, écrivez le script contenant les instructions SQL permettant la création de table **personnes**.

- ✓ **Documentation SQL pour SQLite3**
- ✓ **Type de données SQLite**

Modifier la table existante

```
ALTER TABLE database_name.table_name RENAME TO new_table_name;  
ALTER TABLE database_name.table_name ADD COLUMN column_def...;
```

Il peut être parfois préférable supprimer la table et de la reconstruire plutôt que de la modifier à la volée.

Supprimer la table de données

```
dbSendQuery(con, "DROP TABLE films;")
```

- ✓ **DROP TABLE** supprime l'ensemble de la table et ses données.

Supprimer la base de données

```
dbSendQuery(con, "DROP DATABASE films;")  
dbDisconnect(con)
```

- ✓ **DROP DATABASE** fonctionne seulement avec d'autres SGBDs (approche serveur).
- ✓ Dans le cas de SQLite3, on supprime simplement le fichier *** .db**.
- ✓ **dbDisconnect (con)** permet de fermer la connection avec le fichier de base de données (permet à un autre utilisateur de se connecter).

Lectures et travail pour la semaine prochaine

Travail

Maintenant que vous en savez plus sur le format des données, vous devez écrire le script R pour créer votre base de données en spécifiant les tables, les champs et les clés liant les tables entre elles.

Lectures et travail pour la semaine prochaine

Travail

- ✓ 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.