SCI6005 Information numérique et informatique documentaire (A2024)

Anton Boudreau Ninkov, EBSI, UdeM

A2024 20 novembre 2024

Cours 11 - Structuration de l'information dans une base de données (2 de 2)
SCI6005

Table des matières

Cours 11 - Structuration de l'information dans une base de données (2 de 2)

1. + Au programme aujourd'hui

- Structuration des informations dans une base de données
- Exploitation de la structuration : formulaire de saisie
- Exploitation de la structuration : recherche d'information
- Exploitation de la structuration : rapport de sortie

2. + Alignement pédagogique

Objectifs visés, matériel du cours et évaluation : Examen final

Objectif général : Comprendre la place des technologies et de l'information numérique en contexte documentaire		
Objectifs spécifiques	**Compétences à développer**	**Matériel associé**
Décrire les principales technologies utilisées en milieux documentaires	Décrire l'utilisation d'une base de données pour structurer l'information	Section *Structuration des informations* *TP Structuration dans une base de données*

Lien entre les objectifs, les compétences à développer et le matériel du cours 11

Objectif général : Concevoir et implanter des systèmes et services d'information numérique		
Objectifs spécifiques	**Compétences à développer**	**Matériel associé**
Exploiter la structuration de l'information numérique dans différents contextes	Décrire la structure de l'information dans une base de données	Section *Structuration des informations* *TP Structuration dans une base de données*

	Exploiter la structuration de l'information dans une base de données à l'aide d'un formulaire de saisie	Section *Exploitation de la structuration : formulaire de saisie* *TP Structuration dans une base de données*
	Exploiter la structuration de l'information dans une base de données lors d'une recherche d'information	Section *Exploitation de la structuration : recherche d'information* *TP Structuration dans une base de données*
	Exploiter la structuration de l'information dans une base de données à l'aide d'un rapport de sortie	Section *Exploitation de la structuration : rapport de sortie* *TP Structuration dans une base de données*
Comparer des technologies utilisées en milieux documentaires	Distinguer la structuration dans un traitement de texte, un tableur et une base de données	*TP Structuration dans une base de données*

Objectifs visés, matière du cours et activités associées

Objectif général : Concevoir et implanter des systèmes et services d'information numérique		
Objectifs spécifiques	**Matière**	**Activités**
Exploiter la structuration de l'information numérique dans différents contextes	Structure de données dans une base de données	*TP Structuration dans une base de données*
	Formulaire de saisie	*TP Structuration dans une base de données*
	Rapport de sortie	*TP Structuration dans une base de données*
Comparer des technologies utilisées en milieux documentaires	Structuration dans un traitement de texte, dans un tableur, dans une base de données	*TP Structuration dans une base de données*

Lien entre les objectifs, la matière du cours 11 et les activités associées

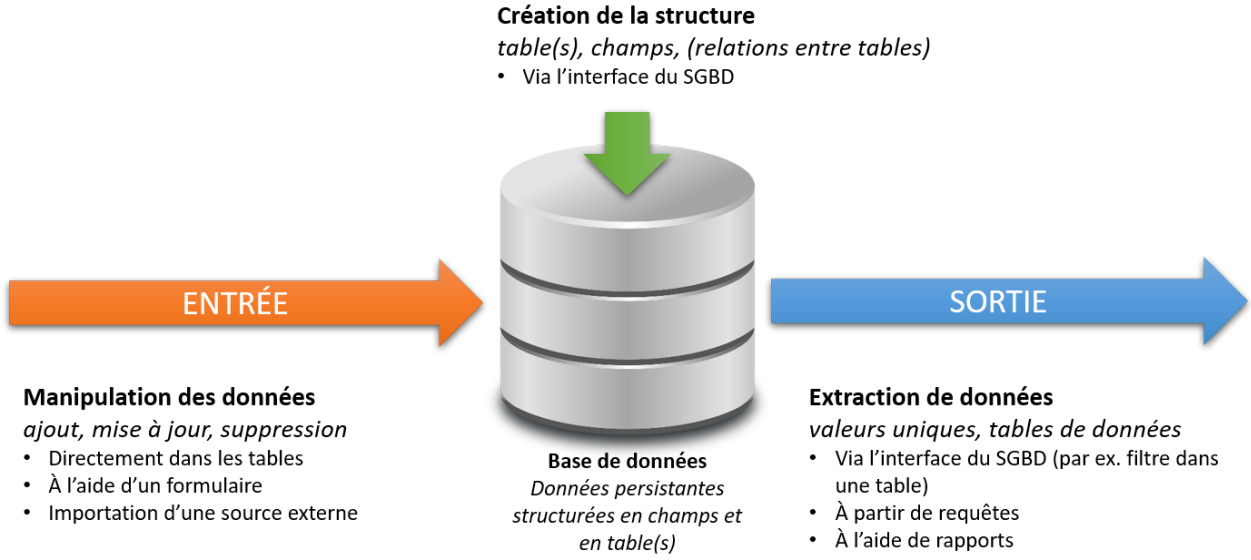
3. Introduction

Après avoir abordé les modèles de données entre autres textuels et relationnels, nous nous attarderons dans cette deuxième partie sur les systèmes de gestion de bases de données, en particulier sur la manière d'**exploiter la structuration de l'information** qu'ils permettent afin de soutenir de manière efficace les différents processus propres aux bases de données.

4. Structuration dans une base de données

4.1. Introduction

Nous nous sommes attardés la semaine dernière sur certains modèles de données pour comprendre la *structuration de l'information* et les *opérations* qui leur sont propres. C'est ce modèle de données qui influencera la modélisation de la structure des données dans une base de données, tâche centrale à son développement. Mais il n'y a pas que la structure de données à créer. Le concepteur ou la conceptrice d'une base de données doit prendre en considération l'**ensemble des processus** liés à la **création de la structure** de données, à la **manipulation des données** et à l'**extraction des données** :



Processus associés à un SGBD

Peu importe le type de base de données, la phase de conception de la base de données doit ainsi prévoir la modélisation de la structure de données, mais aussi s'assurer de **comprendre le contexte d'utilisation de la base de données** pour prévoir des mécanismes permettant de soutenir efficacement la *saisie des données* dans la base de données et les besoins liés à l'*exploitation des données* qui s'y trouvent. La bonne compréhension du **contexte organisationnel** et de la **culture informationnelle** des utilisateurs et utilisatrices de la base de données est cruciale pour offrir une base de données qui sera adoptée par le plus grand nombre. Il faut ainsi bien comprendre :

- Qui mettra les données à jour dans la base de données? Est-ce que le logiciel utilisé est connu de la ou des personnes responsables de la saisie?
- Qui consultera la base de données? Quel est le niveau de connaissance des personnes qui consulteront la base de données par rapport à son langage d'interrogation? Y a-t-il des rapports types attendus?

Ainsi, après avoir fait un rappel sur la manière dont l'information est structurée dans une base de données, nous aborderons les trois principaux processus où cette structuration est exploitée : (1) la *saisie de l'information*, (2) la *recherche d'information*, (3) l'*affichage de l'information*. Nous le ferons pour les deux principaux modèles de données décrits la semaine dernière soit le modèle textuel, avec *DB/TextWorks* comme exemple, et le modèle relationnel, avec *Base* comme exemple.

4.2. Structuration des informations

Rappel sur la structuration de l'information dans une base de données

Comme nous l'avons vu précédemment lors de la présentation de différents modèles de données, la structuration de l'information dans une base de données textuelle ou relationnelle passe par le découpage de l'information en **champs** (*colonnes*) et en **enregistrements** (*lignes*) dans **une table de données** (*modèle textuel*) ou **plusieurs tables de données** (*modèle relationnel*). Plus spécifiquement, les deux modèles proposent des manières parfois différentes de structurer l'information dans les champs, notamment :

- Le *modèle textuel* permet les **occurrences multiples** dans un champ, ce qui n'est pas le cas du modèle relationnel.
- Le *modèle relationnel*, en raison de la présence de plusieurs tables, demande la présence d'un champ ou d'une combinaison de champs dont les valeurs sont uniques dans une table de données (**clé primaire**) ainsi que la présence de champs servant à relier des tables (**clé externe**).

Tant le **tableur** que la **base de données** apportent une **structuration plus grande de l'information**, qui y est nécessairement strictement structurée, qu'un traitement de texte où l'information, à la base, n'est pas aussi strictement structurée. Toutefois, bien que l'on puisse faire un parallèle avec la structuration de l'information dans un **tableur** où l'information est aussi structurée en ligne (*enregistrement*) et colonne (*champ*), voire même en plusieurs feuilles de données (*tables*), cette structuration demeure **moins stricte** que dans une base de données. On ne peut en effet contrôler aussi finement et précisément les données saisies dans une cellule d'un tableur : rien ne nous empêche de laisser une cellule vide, par exemple, ou d'y saisir une date dans un format autre que celui attendu. La validation des données est ainsi moins bonne que dans une base de données où il est possible d'imposer, par exemple, un caractère obligatoire à un champ ou un format de date précis. On gagne ainsi sur le plan de la qualité des informations colligées, ce qui est un point crucial dans un contexte de gestion de l'information.

4.3. Exploitation de la structuration : Formulaire de saisie

Un des processus à conceptualiser pour une base de données est la manière dont l'information y sera **manipulée** (saisie, modification, suppression). Dans certains environnements, il est possible de *saisir directement les données dans la ou les tables de données*, mais ce mode de saisie est habituellement réservé à la personne qui administre la base de données comme il présuppose une bonne connaissance du SGBD. Lorsque la saisie est du ressort d'autres personnes que l'administrateur ou l'administratrice de la base de données, il faut habituellement prévoir des **formulaires de saisie** pour ces dernières. Ces formulaires reprendront la structure de la base de données pour offrir un *environnement facilitant la saisie des données*.

Remarque

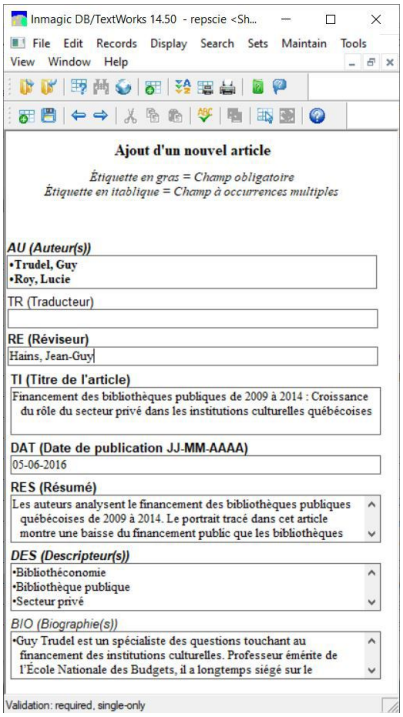
En sus d'une saisie ponctuelle dans la base de données à partir d'un formulaire de saisie ou directement dans les tables de données, il est aussi possible dans la majorité des SGBD d'effectuer une **saisie en lot**, c'est-à-dire d'**importer** dans la base de données simultanément plusieurs enregistrements. C'est entre autres la technique utilisée lorsque l'on migre d'un SGBD à un autre. Ce type de saisie relève habituellement de la personne qui administre la base de données et n'implique pas l'utilisation d'un formulaire de saisie.

a) Formulaire de saisie dans DB/TextWorks (modèle textuel)

Dans *DB/TextWorks*, la saisie ne peut se faire directement dans la table de données; il faut ainsi passer par un ou même plusieurs formulaires de saisie. C'est lors de la conception de la base de données qu'il importe de bien comprendre qui sera en charge de la saisie afin de prévoir le ou les formulaires nécessaires pour procéder. Comme illustré ci-dessous, un formulaire de saisie présente les **différents champs** où une saisie potentielle est attendue. Chacune des boîtes de saisie dans cet exemple est précédée d'une **étiquette** (courte chaîne de texte descriptive) rappelant le nom du champ ainsi que la

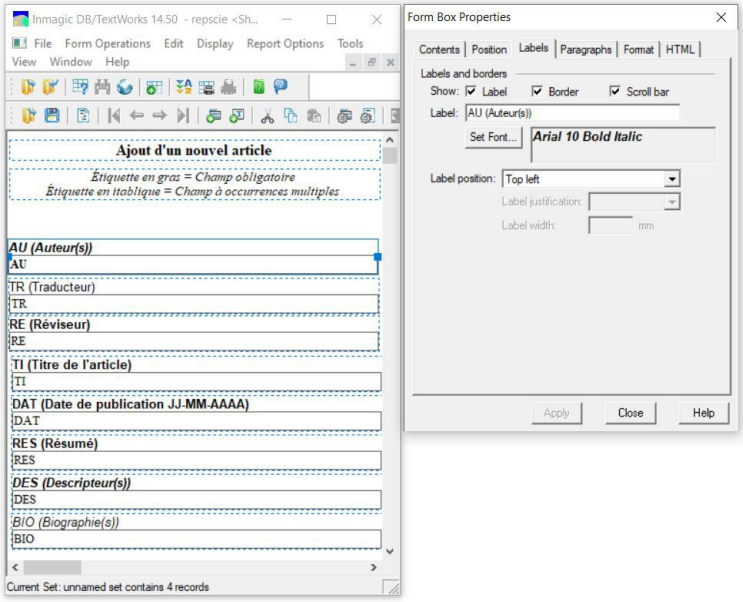
forme développée de ce dernier. De plus, ces étiquettes indiquent, par leur mise en forme, certaines **caractéristiques des champs** (caractère obligatoire et occurrences multiples). Finalement, la personne qui a conçu le formulaire de saisie a cru bon de **rappeler le format attendu** pour la saisie de la date.

L'objectif est de proposer des **formulaires de saisie facilitant le travail** et **augmentant la qualité des données saisies**. Bien que, lorsque le curseur est positionné dans une boîte de saisie, le SGBD rappelle certaines règles de saisie dans la barre d'état (ici *Validation: required, single-only* qui signifie que le champ RE est donc obligatoire et à occurrence simple), ce rappel demeure d'une part, un peu discret et, d'autre part, dans un vocabulaire qui n'est peut-être pas très explicite. Il a donc été décidé d'indiquer plus **explicitement** dans l'interface ces règles de saisie.



Formulaire de saisie dans DB/TextWorks

DB/TextWorks offre une seule manière de créer un formulaire de saisie, soit son **éditeur de formulaire**. Il est possible de débuter la conception d'un formulaire à partir d'un formulaire vide, comme on peut demander de débuter avec un canevas de base où tous les champs de la base de données sont inclus. L'édition consiste à **inclure** dans le formulaire les **champs désirés**, à **modifier** leurs **étiquettes**, à **ajouter titre et sous-titre** (s'il y a lieu) et à **ajuster la mise en forme**.



Éditeur de formulaire dans DB/TextWorks

b) Formulaire de saisie dans Base (modèle relationnel)

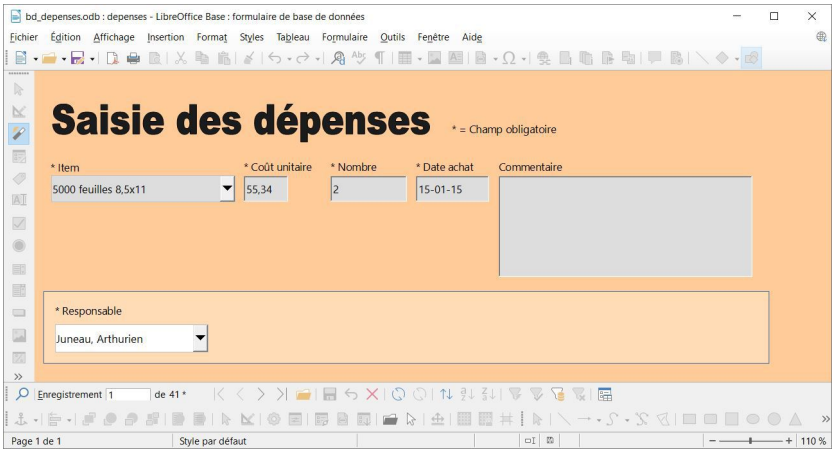
Bien que l'on puisse dans *Base* saisir directement des données dans une table de données, ce n'est habituellement pas la manière de procéder qui est la plus facile pour une personne autre que l'administrateur ou l'administratrice de la base de données. Tout comme pour *DB/TextWorks*, *Base* permet ainsi de créer un ou plusieurs formulaires de saisie. Toutefois, du fait qu'une base de données dans le modèle relationnel peut avoir plus d'une table de données et que ces tables de données peuvent être reliées, les **formulaires de saisie** peuvent devenir plus **complexes**.

L'exemple ci-dessous est un formulaire qui permet de saisir les informations pour une dépense ainsi que, par un *sous-formulaire*, d'associer à cette dépense une ou plusieurs personnes responsables de cette dernière. Le formulaire permet de saisir des informations dans deux tables :

- La table *Depenses* pour les informations sur les dépenses (c'est la table principale du formulaire)
- La table *Resp_achat* pour le numéro de la ou des personnes responsables de l'achat (c'est la table du sous-formulaire)

De plus, afin de faciliter la saisie, les deux autres tables de la base de données sont exploitées :

- La table *Items* est utilisée pour afficher les intitulés des items dans un *menu déroulant* pour faciliter la saisie.
- La table *Responsables* est utilisée pour afficher la liste des noms dans un *menu déroulant* pour faciliter la saisie.



Formulaire de saisie dans Base

Base offre deux modes pour la création d'un formulaire de saisie : (1) l'**Assistant Formulaire** et (2) le **mode Ébauche**. Le premier mode propose différentes étapes pour préparer un formulaire de base qu'il est possible de modifier par la suite. Le deuxième mode ouvre l'éditeur de formulaire et propose un formulaire vierge. Il peut ainsi être intéressant de préparer un **canevas de base** avec l'*Assistant Formulaire* pour ensuite **peaufiner** le résultat. C'est ainsi que le formulaire ci-dessus a été créé en suivant les différentes étapes de l'*Assistant*.

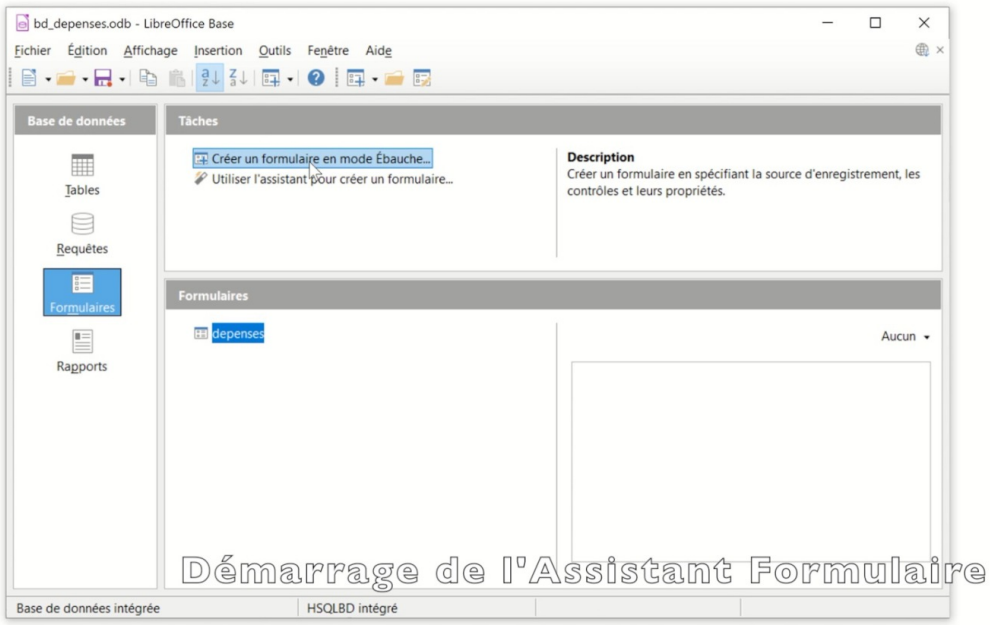
En résumé, les **différentes étapes** proposées par l'*Assistant Formulaire* vous permettront de :

- Choisir la **source de données** (table et champs) où l'information sera saisie;
- Préciser si le formulaire comprendra un **sous-formulaire** et, le cas échéant, de définir la source du sous-formulaire (table et champs) et le lien de ce dernier avec le formulaire principal;
- Choisir la **configuration physique des différents contrôles** (boîtes de saisie) du formulaire;
- Préciser le **type de saisie désirée** (par exemple, uniquement l'ajout de nouveaux enregistrements ou aussi la modification des enregistrements existants);
- Choisir un **visuel** parmi les modèles proposés.

La ressource qui suit illustre ce processus pour la création d'un *formulaire de base pour la saisie des items et des dépenses liées*. On y retrouvera donc : (1) les items comme formulaire principal et (2) les dépenses comme sous-formulaire lié au formulaire principal par le champ id_item (qui est la relation entre les deux tables).

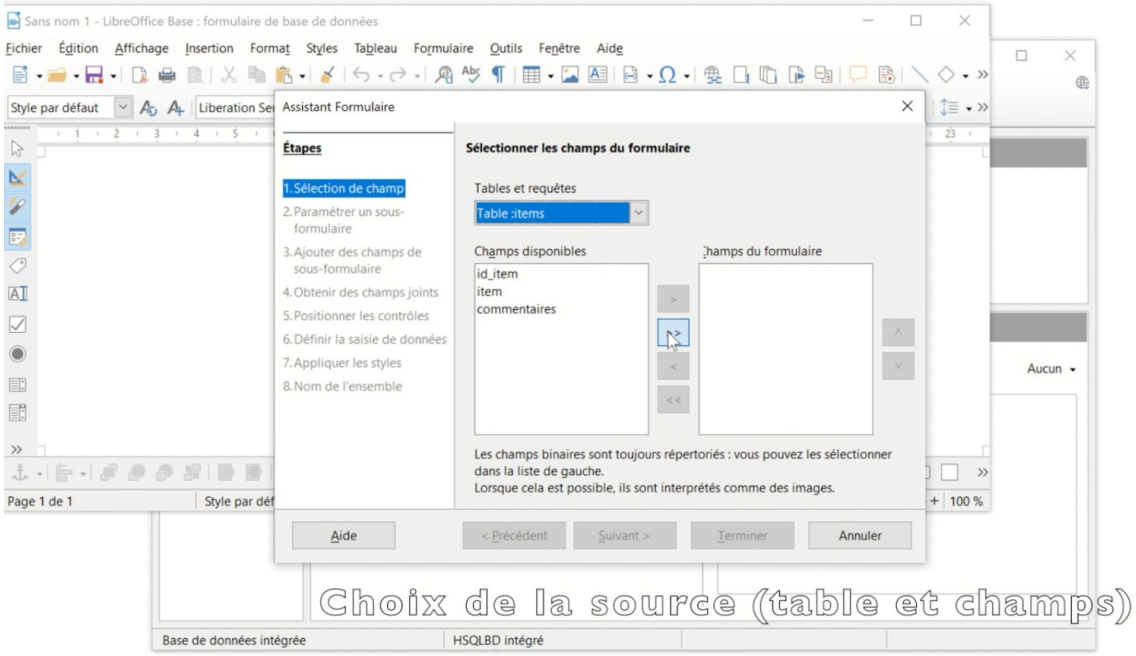
Note : Capsule vidéo accessible en ligne[1]

Première étape : **Démarrage** de l'*Assistant Formulaire*



Démarrage de l'Assistant Formulaire

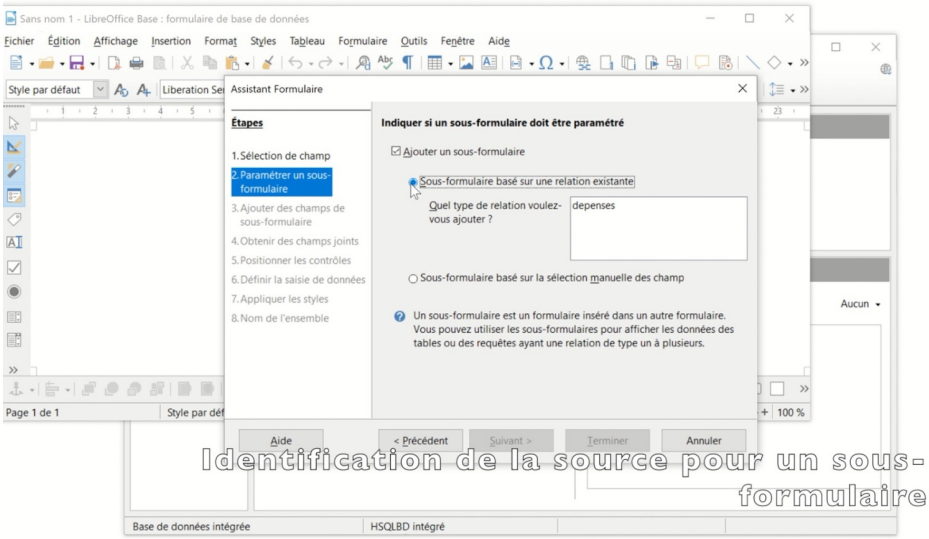
Deuxième étape : Choix de la **source du formulaire** principal (table et champs)



Choix de la source du formulaire principal (table et champs)

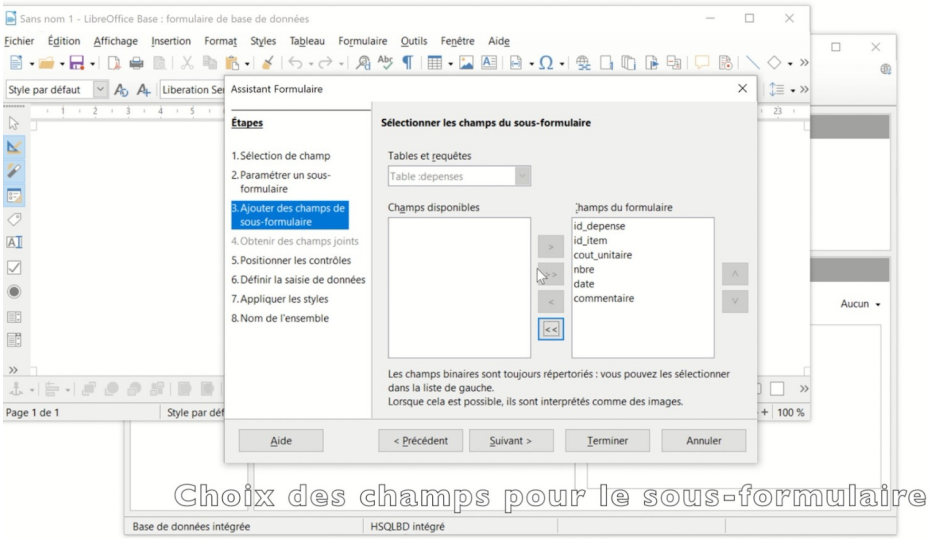
Troisième étape : Identification de la **source du sous-formulaire** (relation)

[1] http://cours.ebsi.umontreal.ca/sci6005/a2022/res/base_form_assist.mp4



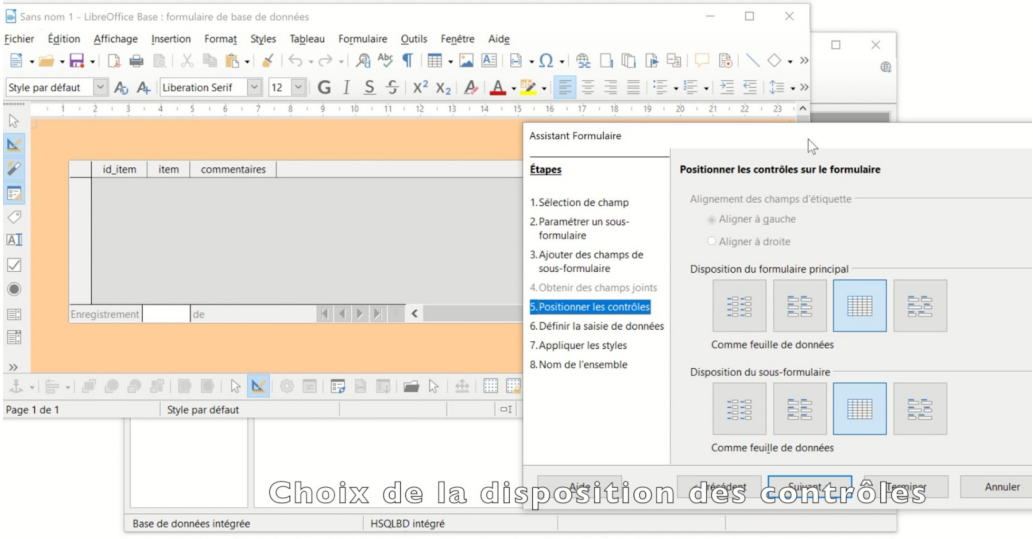
Choix de la source du sous-formulaire

Quatrième étape : Choix des **champs** pour le **sous-formulaire**



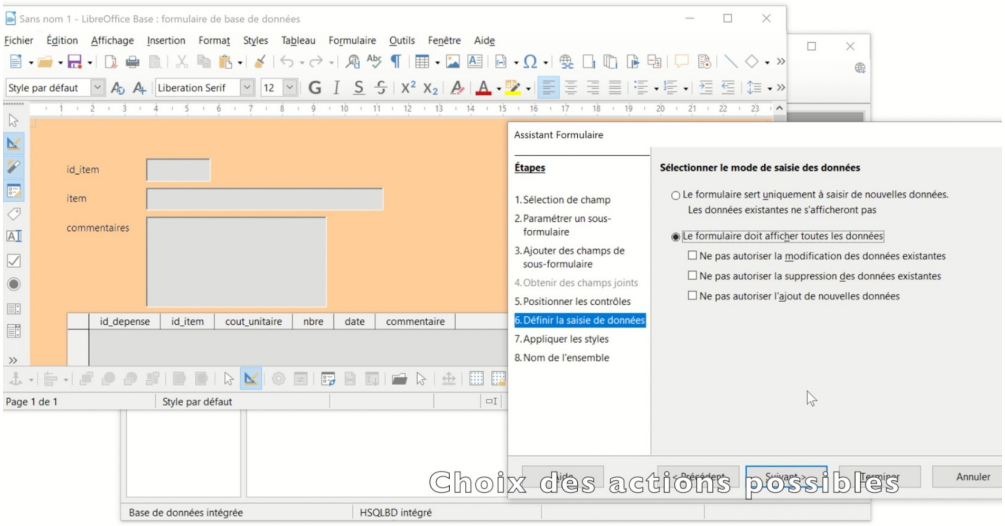
Choix des champs du sous-formulaire

Cinquième étape : Choix de la **position des boîtes de saisie** pour le formulaire et le sous-formulaire



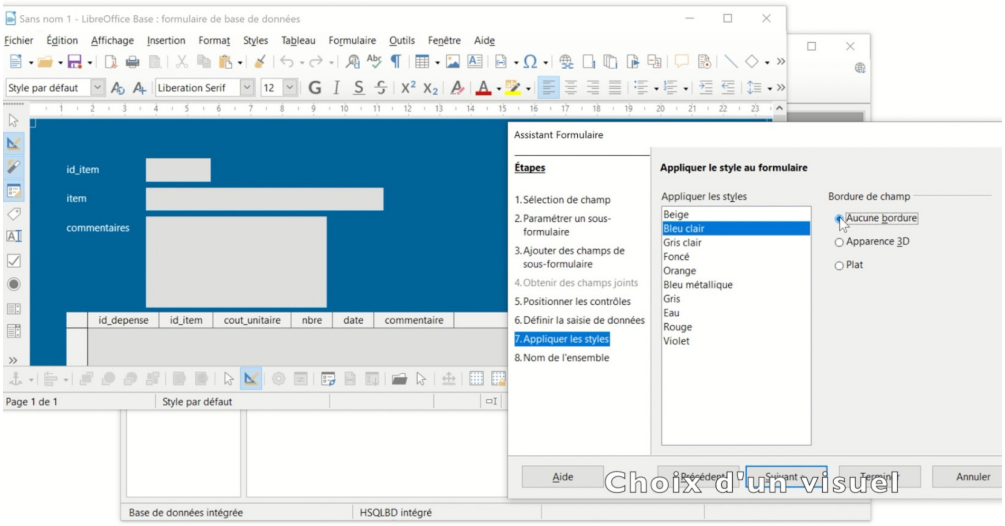
Positionnement des contrôles de saisie pour le formulaire et le sous-formulaire

Sixième étape : Choix du **type d'actions** possibles



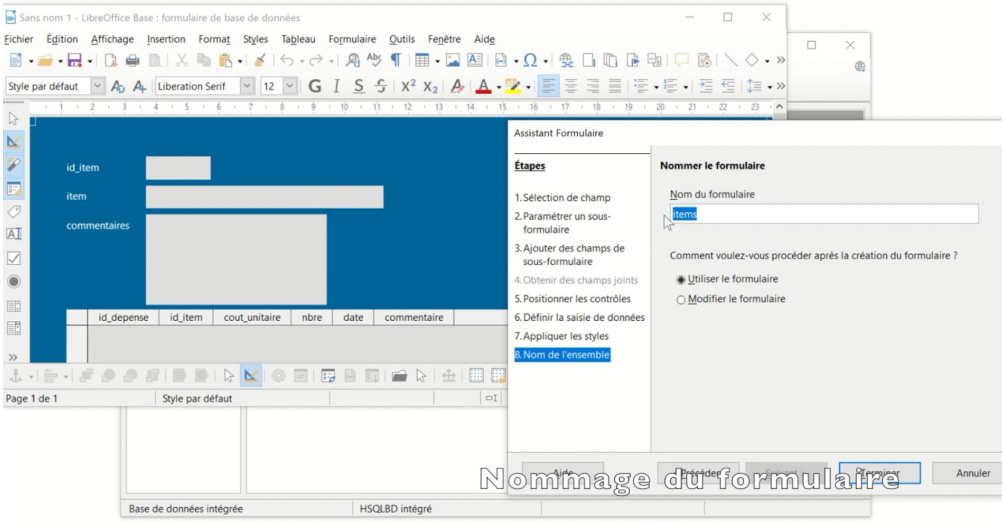
Choix des actions possibles

Septième étape : Choix d'un **visuel**



Choix d'un visuel

Huitième étape : **Nommage** du formulaire



Nommage du formulaire de saisie

L'Assistant Formulaire produit un formulaire de base qu'il est possible par la suite d'améliorer en éditant le formulaire créé. Du point de vue de la conception, le processus est similaire à celui présenté pour *DB/TextWorks*. Il s'agit de **modifier le formulaire** pour s'assurer de son **efficacité** à bien **accompagner** la personne responsable de la saisie dans son travail. On peut par exemple indiquer les *champs obligatoires* ou insérer des *zones de liste* pour proposer des menus déroulants. L'éditeur dans *Base* offre différentes possibilités dans les menus que l'on retrouve au pourtour du formulaire pour ajouter des contrôles ou des éléments visuels, ou pour modifier la mise en page. Lorsque l'on survole un bouton avec la souris, l'infobulle nous informe de son utilisation. De plus, il est possible de faire afficher la fenêtre des propriétés d'un contrôle par un clic droit sur ce dernier pour modifier certaines de ses caractéristiques.

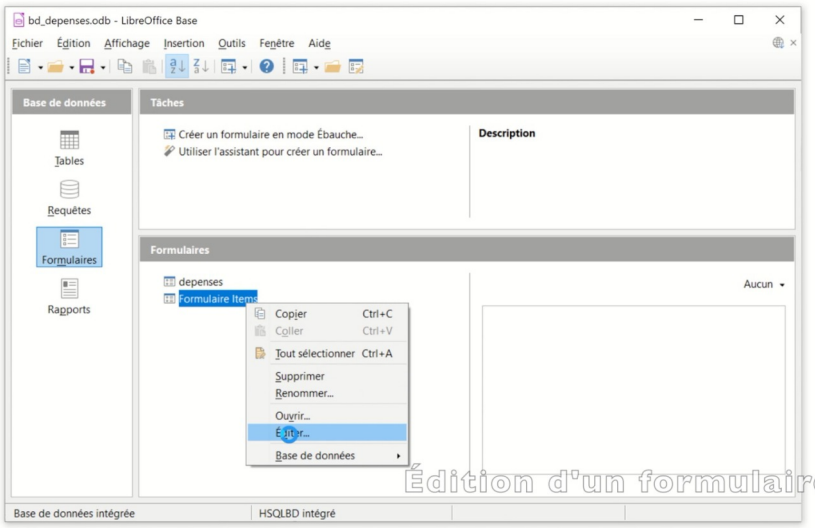
La ressource ci-dessous illustre quelques **modifications du formulaire**, notamment :

- Le *redimensionnement de certaines boîtes* de saisie et la *modification de leur étiquette*
    - *Truc* : Les étiquettes et les boîtes de saisie associées sont regroupées. Pour facilement les modifier sans les dégrouper, vous trouverez dans leur menu contextuel l'option *Entrer dans le groupe* qui permet, après l'avoir sélectionnée, de cliquer sur les éléments pour les modifier. Le redimensionnement se fait simplement avec la souris en bougeant les côtés d'un élément. La modification du texte d'une étiquette se fait en la double cliquant pour accéder à sa fenêtre de propriétés (c'est l'option *Étiquette* qu'il faut modifier dans les propriétés).
- L'ajout d'une *zone de texte* pour plus clairement identifier le sous-formulaire
- Le *masquage*, dans le sous-formulaire, du champ id_item qui se retrouve aussi dans le formulaire principal (information redondante)
- Le *redimensionnement*, dans le sous-formulaire, de la largeur de certaines colonnes

À la fin de la capsule vidéo, vous pourrez observer la **saisie de données** dans le formulaire. Portez attention notamment à l'utilisation des flèches dans le bas de l'écran qui permettent de naviguer soit entre les éléments du formulaire principal (items) ou ceux du sous-formulaire (dépenses). Elles s'adaptent en effet à la position de votre curseur : s'il est dans une des boîtes du formulaire principal, elles permettent de naviguer entre les éléments du formulaire principal. Cela peut se révéler parfois un peu mélangeant si on n'y porte pas attention! Le fait d'avoir utilisé le mode "feuille de données" pour la visualisation du sous-formulaire minimise cette désorientation.

Note : Capsule vidéo accessible en ligne[1]

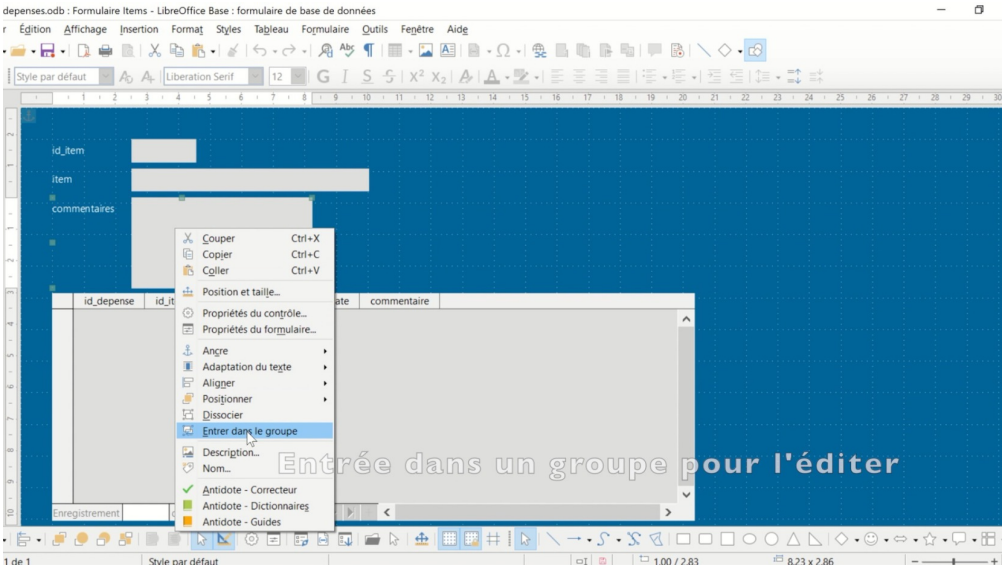
Première étape : Ouverture du formulaire en **mode édition**



Ouverture d'un formulaire en mode édition

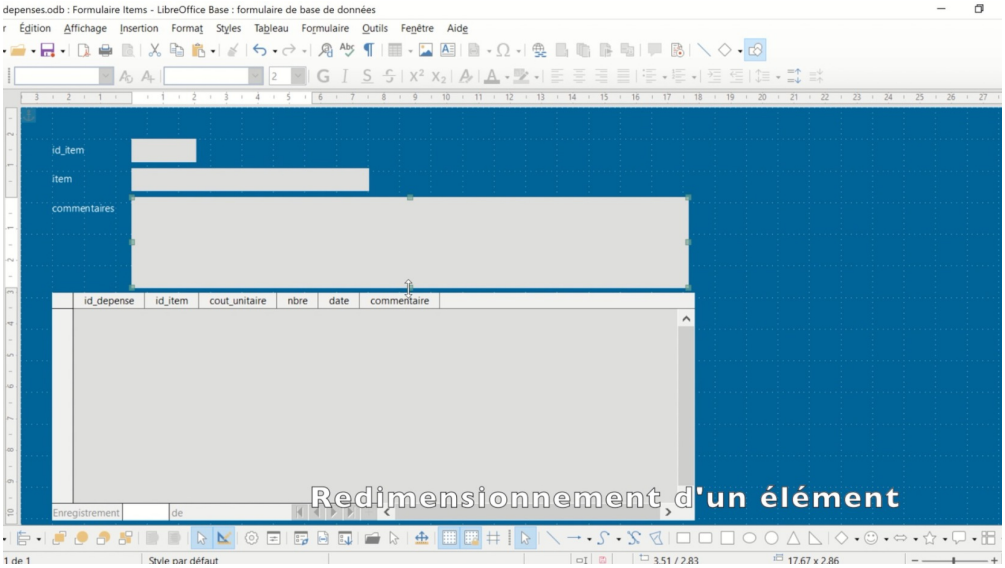
[1] http://cours.ebsi.umontreal.ca/sci6005/a2022/res/base_form_edition.mp4

Deuxième étape : Activation de l'option **Entrer dans le groupe**



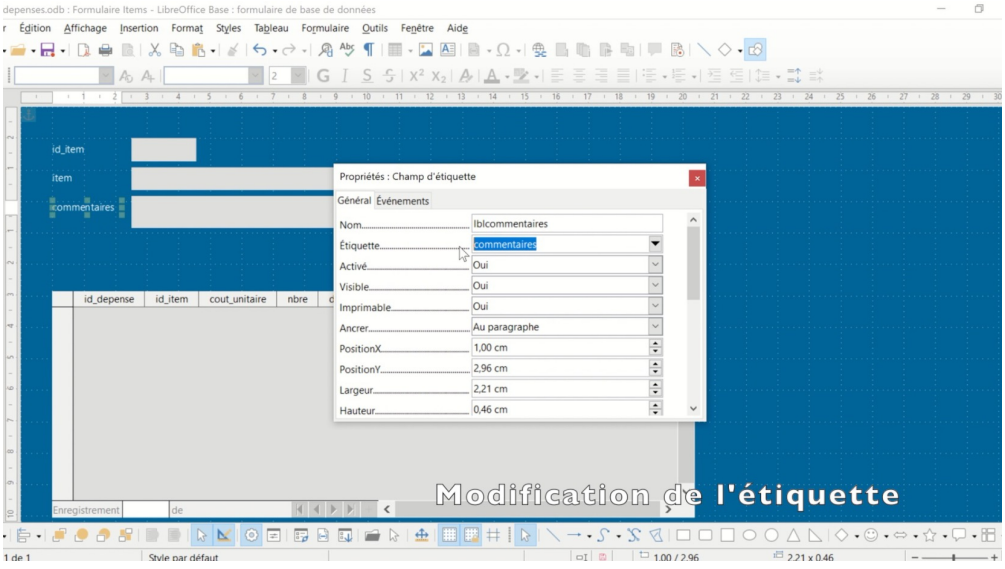
Option Entrer dans un groupe pour éditer les éléments du groupe

Troisième étape : **Redimensionnement d'une boîte** de saisie



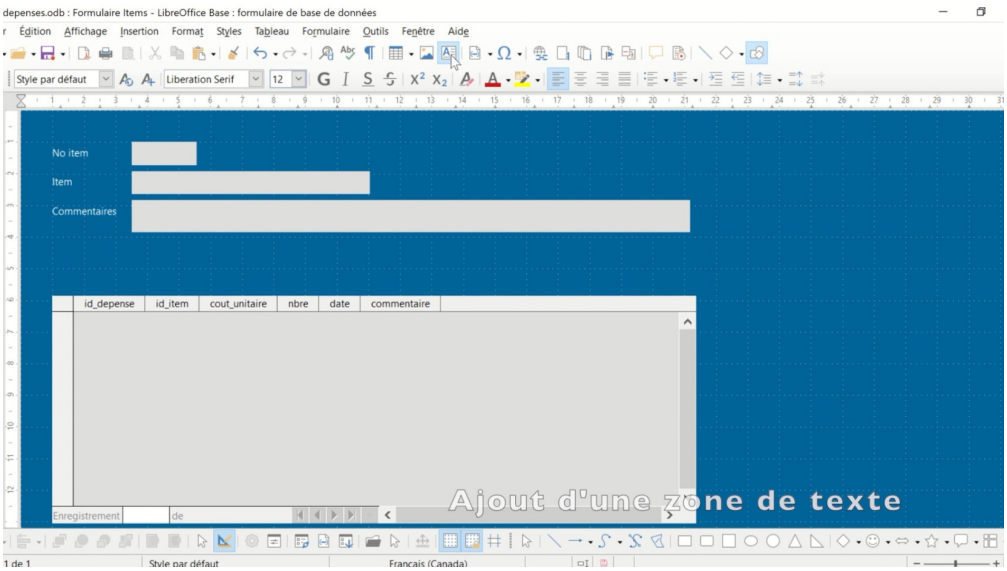
Redimensionnement d'une boîte de saisie

Quatrième étape : Modification d'une **étiquette**



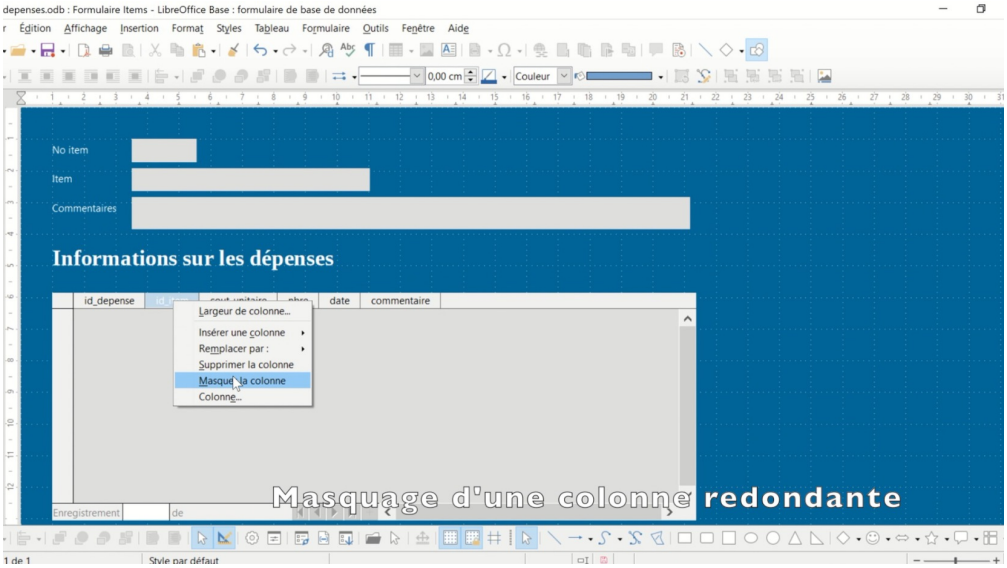
Modification d'une étiquette

Cinquième étape : Ajout d'une **zone de texte**



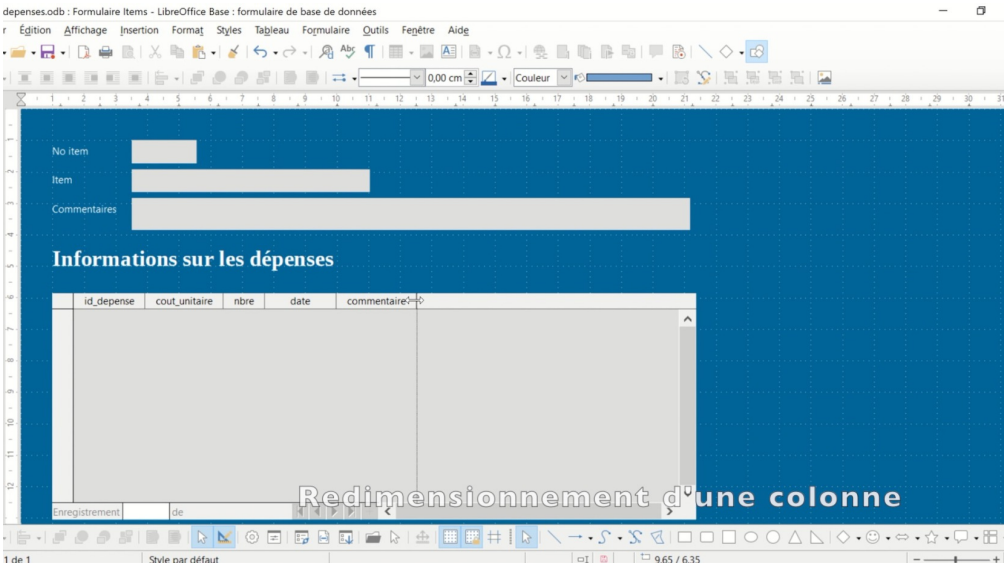
Ajout d'une zone de texte

Sixième étape : **Masquage** d'une colonne du sous-formulaire



Masquage d'une colonne du sous-formulaire

Septième étape : **Redimensionnement d'une colonne** du sous-formulaire



Redimensionnement d'une colonne du sous-formulaire

4.4. Exploitation de la structuration : Recherche d'information

La **structure en champs** dans une base de données donne une *prise plus complète* sur les données lorsqu'on y fait des **recherches** que dans un tableur. Bien que l'on puisse faire une recherche dans un tableur, celle-ci demeure très générique : si on peut limiter la recherche d'une chaîne de caractères dans une colonne précise, on ne peut faire des recherches plus complexes impliquant des valeurs différentes à chercher dans des colonnes différentes. On n'y retrouve pas non plus d'opérateurs pour lier les concepts dans une recherche. Le **langage d'interrogation** dans une base de données permet de faire des recherches **ciblant** certains **champs** et propose différents **opérateurs de recherche** comme, notamment, les opérateurs booléens.

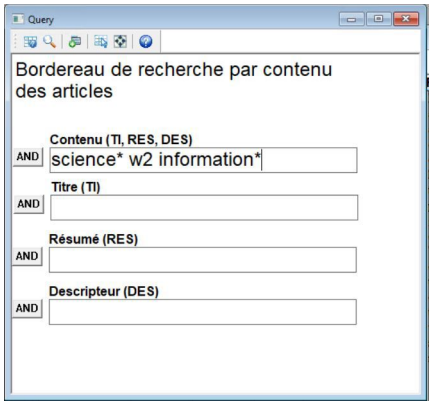
Selon le modèle de données du SGBD et le SGBD lui-même, les possibilités de recherche vont varier tant pour ce qui est des opérateurs de recherche que de la manière d'y construire une recherche. Nous ne nous attarderons pas sur les opérateurs de recherche, qui relèvent plutôt d'un cours sur la recherche d'information, mais nous examinerons plus avant la manière dont on peut faire des recherches. On retrouve en effet dans les SGBD très souvent **différents modes de recherche d'information**, certains pensés pour des chercheuses et des chercheurs d'information plus *novices* et d'autres pour des chercheuses et chercheurs plus *experts*.

a) Recherche d'information dans DB/TextWorks (modèle textuel)

Deux possibilités s'offrent pour faire des recherches dans *DB/TextWorks* : (1) la recherche à l'aide d'un **bordereau de recherche** et (2) la recherche en **mode commande**.

Recherche à l'aide d'un bordereau de recherche (niveau novice)

Il est possible dans *DB/TextWorks* de créer des **bordereaux de recherche** qui sont pensés pour des chercheuses et chercheurs d'information **novices**. Plusieurs bordereaux différents peuvent être conçus s'il s'avère que l'analyse du contexte et de la culture informationnelle révèle que les utilisateurs finaux de la base de données ont des besoins différents. L'exemple ci-dessous est celui d'un bordereau de recherche de la base de données *REPSCIE* conçu pour un besoin très précis, soit des recherches ciblant uniquement le contenu des articles.



Bordereau de recherche dans DB/TextWorks

Comme cet exemple le montre bien, un bordereau de recherche est un **formulaire** où la personne qui l'utilise pour faire une recherche indique dans les **boîtes de saisie** les informations recherchées.

Chacune des **boîtes de recherche** est *associée* à **un ou plusieurs champs** de la base de données. La première boîte de l'exemple a ainsi été associée à tous les champs où l'on peut retrouver des informations en lien avec le contenu d'un article (son titre, son résumé et ses descripteurs). Les boîtes suivantes reprennent ces mêmes champs, mais en les séparant cette fois pour permettre des recherches plus ciblées.

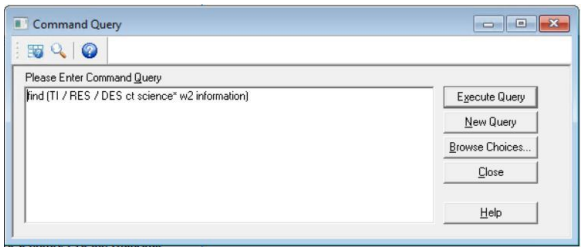
De plus, devant chacune des boîtes de recherche se retrouve un bouton permettant de choisir, pour le contenu de la boîte, **la manière dont on veut la relier aux autres** boîtes (par un *OU* booléen, par un *ET* booléen ou par un *SAUF* booléen).

Finalement, remarquez dans l'exemple ce qui est recherché dans la boîte *Contenu* : on n'y retrouve pas uniquement des mots, mais aussi un opérateur de recherche, soit le *w2* qui permet de chercher les deux mots à un maximum de deux positions, peu importe l'ordre (*opérateur de distance*).

Ainsi, même si le bordereau est pensé pour une personne plutôt novice en recherche d'information avec DBText/Works, il présuppose une **connaissance des opérateurs** de recherche ainsi que de la **structure de la base de données**. La **documentation** qui accompagne la base de données ainsi que la **formation** à son utilisation demeurent des éléments cruciaux dans la réussite de l'implantation de cette base de données.

Recherche en mode commande (niveau expert)

Le deuxième mode de recherche dans *DB/TextWorks*, la recherche en **mode commande**, vise cette fois les chercheuses et chercheurs d'information **experts** (par exemple, les professionnel(le)s de l'information). Comme illustré ci-dessous, il s'agit d'une **simple boîte** où l'on saisit l'**équation de recherche**. Une **connaissance fine du langage d'interrogation** (*syntaxe et sémantique*) ainsi que de la **structure de la base de données** est nécessaire pour pouvoir l'utiliser. Le mode commande pour la recherche experte permet de faire des recherches qui ne peuvent être faites à l'aide d'un bordereau de recherche; il est ainsi plus puissant.



Recherche en mode commande dans DB/TextWorks

Résultats de recherche

La recherche à l'aide d'un bordereau ainsi que celle en mode commande vont produire, comme **résultat**, un **ensemble d'enregistrements** qu'il sera possible de présenter sous différents formats en exploitant, entre autres, les **rapports de sortie** définis dans la base de données (nous y reviendrons plus tard). L'exemple ci-dessous présente les résultats sous forme tabulaire en mettant en exergue les mots de la requête.

NO	AU	TR	RE	DAT	TI	RES	DES	BIO
1	Roy, Lucie	Day, John	Hains, Jean Savard, Guy	01-01-2016	Les sciences de l'information aujourd'hui : une discipline à redéfinir / Information science today : the old, the new and what's to come	L'auteure propose un survol de l'état actuel des sciences de l'information ainsi qu'une définition revue des différentes branches qui la composent. Elle défend notamment l'idée que les avancées technologiques forcent une mise au point quant à la distinction entre la gestion de l'information et la gestion de données. The author describes the current state of information science and offers updated definitions of its many fields of study. She puts forward the notion that due to technological advances, we must rethink the boundaries of information management and data management.	Science de l'information Sciences de l'information Information science Information sciences Information management Gestion de l'information Data management Gestion de données	Lucie Roy est professeure titulaire à l'École des Sciences et de l'Information où elle enseigne depuis plus de vingt ans. Elle dirige la Chaire de recherche sur l'évolution des sciences de l'information et participe à de nombreux autres groupes de recherche internationaux. Ses nombreuses publications portent principalement sur l'information et les différentes sciences qui s'y rattachent. / Lucie Roy is a professor at the School of Sciences and Information. Her teaching career spans more than twenty years. She is a member of a number of international research groups and heads the Research Chair on the evolution of information sciences. She has published numerous books and articles on information and its sciences.
4	Clerc, Jean		Roy, Lucie Savard, Guy	25-12-2015	What helps and what doesn't : Efficiency measures in public libraries	Jean Clerc offers a thorough analysis of efficiency measures implemented in public libraries and their success (or lack thereof) in helping maintain a balanced budget. The author first suggests different criteria of success and then applies them to different real-life examples of management strategies in public libraries.	Information science Library science Librarianship Public library Efficiency Budget Management	Jean Clerc is a postdoctoral researcher at the School of Information of Northern Labrador. His PhD thesis The new librarian: A better way to manage libraries, written under the guidance of Lucie Roy, has recently been published by SINL Press. He has authored, coauthored and translated many articles on a wide array of subjects, including librarianship, library management, efficiency, and strategic measures.

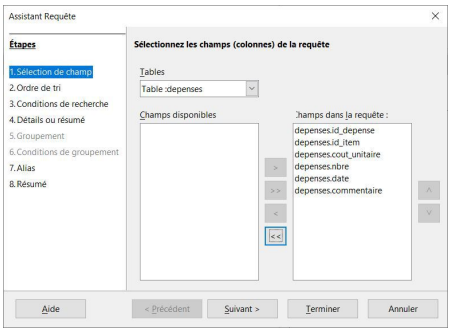
Utilisation de l'opérateur de distance dans le désordre (DB/TextWorks)

b) Recherche d'information dans Base (modèle relationnel)

Le SGBD *Base* de *LibreOffice* offre trois manières de concevoir une requête de recherche : (1) l'**Assistant Requête**, (2) le **mode Ébauche** et (3) le **mode SQL**. Ces trois manières, décrites ci-dessous, permettent à des chercheuses et chercheurs d'information de différents profils (niveau *novice*, *intermédiaire*, *expert*) de créer des requêtes. De plus, dans une logique un peu similaire à celle des bordereaux de recherche de *DB/TextWorks*, il est possible de créer des **requêtes paramétrées** demandant simplement à la personne qui les exécute de préciser certains paramètres de sa recherche.

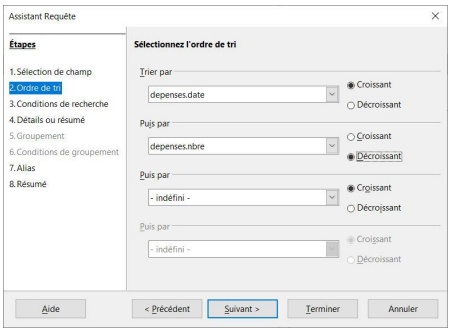
Assistant Requête (niveau novice)

La personne novice en recherche d'information dans *Base* peut utiliser l'**Assistant Requête**. Comme illustré ci-dessous, cet assistant permet de graduellement définir les caractéristiques de la recherche en précisant premièrement les champs à utiliser et leurs provenances (tables ou requêtes), par exemple les champs de la table *Depenses*.



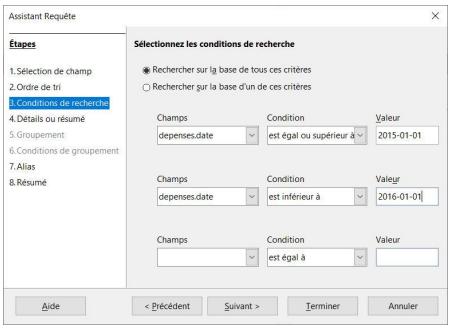
Assistant Requête dans Base : Choix des champs

Par la suite, il permet d'identifier **une ou des clés de tri** pour imposer un **ordre aux résultats de recherche**. Par exemple, on peut demander de trier les résultats premièrement par date en ordre croissant et, advenant que deux enregistrements aient la même date, par le nombre d'items acheté en ordre décroissant.



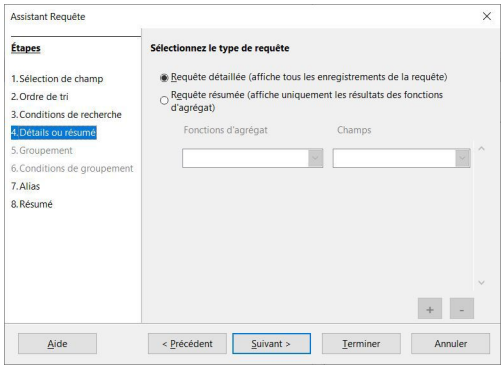
Assistant Requête dans Base : Clé(s) de tri

À l'étape suivante, il est possible de préciser des **critères de recherche**, c'est-à-dire des conditions à respecter pour retenir un enregistrement. Dans l'exemple ci-dessous, comme on ne veut que *les dépenses pour l'année 2015*, deux conditions ont été précisées qui doivent être **toutes les deux respectées** soit d'être une date (1) **plus grande ou égale au 1er janvier 2015** et (2) aussi plus **petite que le 1er janvier 2016**.



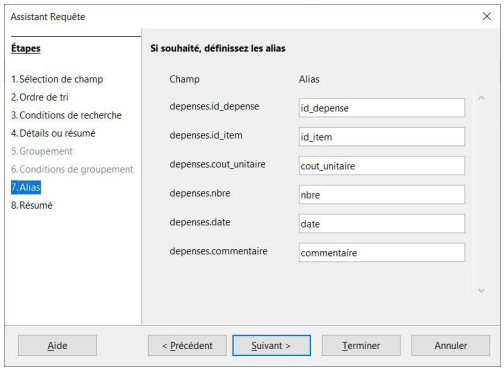
Assistant Requête dans Base : Critères de recherche

Ensuite, il est possible de décider d'aller vers une requête qui **détaillera** tous les enregistrements retrouvés ou une requête qui fera des **regroupements** pour faire certains calculs. Pour notre exemple, nous voulons tout le détail.



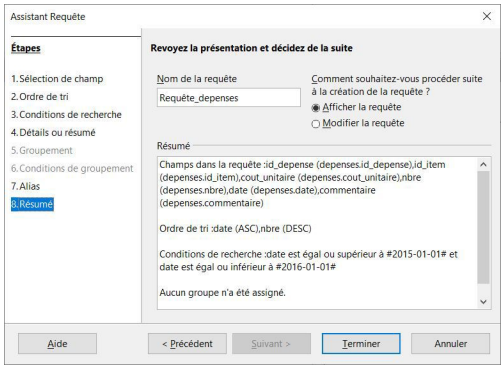
Assistant Requête dans Base : Type de requête

Finalement, si désiré, l'*Assistant* offre la possibilité de définir des *chaînes de caractères plus explicites* que les noms des champs (des **alias**) lors de l'affichage des résultats. Dans l'exemple ci-dessous, il a été décidé de conserver les noms des champs.



Assistant Requête dans Base : Alias

Un **résumé des choix** effectués est alors offert. Si on y trouve une erreur, il est possible de revenir aux étapes précédentes avec le bouton *Précédent*. C'est le temps de donner un nom significatif à la requête ainsi que de décider ce que l'on veut faire par la suite : afficher les résultats ou voir la requête en mode édition pour la modifier. Ce deuxième choix vient entre autres du fait qu'il n'est pas possible, avec l'*Assistant*, d'exploiter toutes les possibilités du langage d'interrogation. On peut ainsi utiliser l'*Assistant* pour faire un premier jet d'une requête pour la **peaufiner** par la suite soit en *mode Ébauche* ou en *mode SQL* (modes qui seront présentés plus loin).



Assistant Requête dans Base : Finalisation

L'*Assistant Requête* demande ainsi des **connaissances minimales de la recherche** dans le modèle relationnel. Pour pouvoir l'exploiter, il faut bien entendu avoir une **certaine connaissance de la structure de la base de données** (les champs) et de **certaines notions de base** (tri, critères de recherche, regroupements par exemple). **Documentation** et **formation** demeurent une valeur sûre même pour ce mode de recherche visant des novices.

Mode Ébauche (niveau intermédiaire)

Base offre un autre mode pour construire des requêtes qui vise cette fois des chercheuses et chercheurs intermédiaires. Il ne prend en effet pas autant la personne qui fait une recherche par la main que l'assistant présenté précédemment, mais sans lui demander de connaître le langage d'interrogation SQL. Il s'agit du **mode Ébauche**. L'exemple ci-dessous correspond au même besoin que celui illustré pour l'*Assistant Requête* : on veut repérer les dépenses ayant eu lieu en 2015 et les présenter en ordre croissant de date et, par la suite, décroissant de nombre d'items.

Comme illustré dans la ressource ci-dessous, lorsque l'on construit une requête en *mode Ébauche*, il faut dans un premier temps identifier la **source des données**. Le bouton représentant une table avec un signe plus permet d'accéder à la liste des tables et des requêtes de la base de données qu'il suffit de cliquer pour les ajouter dans la *partie supérieure* du *mode Ébauche*. Dans l'exemple ci-dessus, une seule table a été sélectionnée, soit la table *Depenses*.

Par la suite, il faut choisir les **champs** que l'on veut voir dans les résultats pour les glisser dans la *partie inférieure* du *mode Ébauche*. Comme on peut le voir, tous les champs ont été retenus.

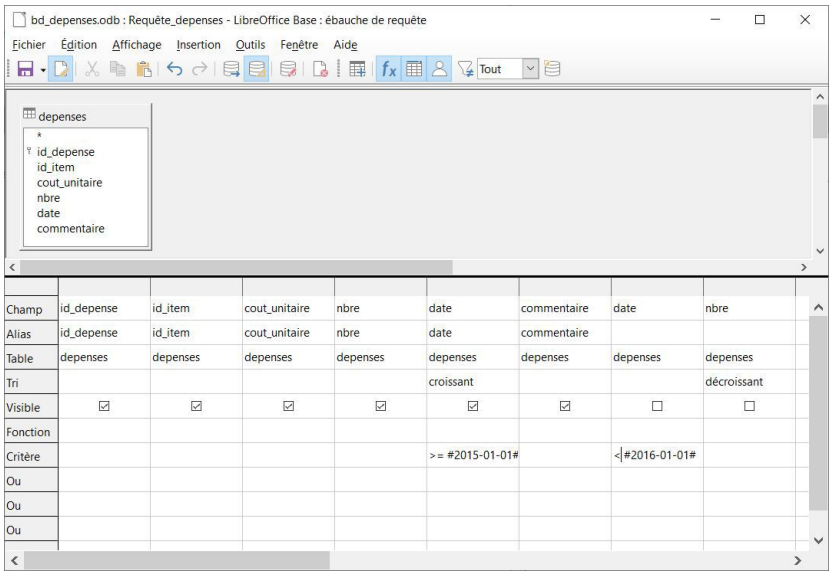
C'est la **ligne Tri**, dans la *partie inférieure*, qui permet de décider du ou des champs qui serviront à trier les résultats en indiquant si le tri est croissant ou décroissant. Remarquez que le champ *Nbre* apparaît en fait deux fois dans la partie inférieure. La première fois est pour qu'il soit affiché dans les résultats. La deuxième fois sert à l'identifier comme une clé de tri. Pour cette deuxième fois, la case de la ligne *Visible* n'a pas été cochée, comme il ne sert que pour le tri. Pourquoi ne pas avoir indiqué le tri à la première colonne où il apparaît? Simplement parce que, comme il arrive avant le champ *Date*, il aurait alors été la première clé de tri! Les clés de tri sont en effet exécutées de gauche à droite.

Finalement, les **critères** sont indiqués à la *ligne Critère*. Comme deux critères s'appliquent sur le champ *Date*, ce dernier apparaît deux fois, une fois par critère. Remarquez l'ajout des dièses (#) au début et à la fin d'une date; c'est la syntaxe à utiliser pour que Base comprenne qu'il s'agit bien d'une date.

L'exécution de la requête montre bien qu'on ne retrouve que les dépenses de 2015 et que l'ordre de tri demandé a bien été respecté.

Note : Capsule vidéo accessible en ligne[1]

Résultat de la construction de la requête en mode Ébauche



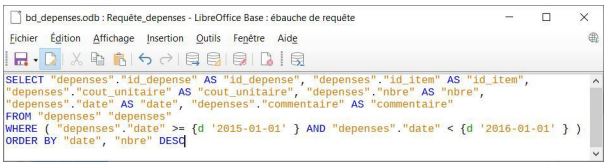
Requête en mode Ébauche dans Base

[1] http://cours.ebsi.umontreal.ca/sci6005/a2022/res/base_requete_ebauche.mp4

Mode SQL (niveau expert)

Le troisième mode pour faire des recherches est celui pour les **chercheuses et chercheurs experts** qui connaissent la syntaxe et la sémantique du **langage d'interrogation SQL**, soit le **mode SQL**. La copie d'écran ci-dessous présente la requête SQL qui correspond au même besoin que ce qui a été présenté précédemment. Sans entrer dans les détails, notez la présence des éléments suivants :

- **SELECT** : permet d'indiquer qu'il s'agit d'une requête pour extraire des données et est suivi des champs que l'on veut afficher dans la table de résultats.
- **FROM** : indique la source des données (ici la table *Depenses*).
- **WHERE** : sert à préciser les critères de la recherche. On y retrouve les deux critères sur le champ *Date* liés par un *ET* booléen (AND)
- **ORDER BY** : indique le tri. On y retrouve bien ici les deux clés de tri avec l'ordre de ces derniers.



Requête SQL dans Base

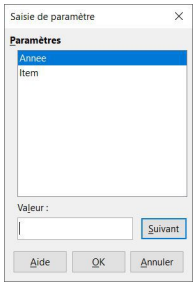
Le langage d'interrogation SQL est **puissant** et demande une **certaine pratique** pour arriver à le maîtriser. C'est la raison qui explique les deux modes précédents qui permettent de construire des requêtes de recherche sans avoir besoin de connaître SQL. Il est à noter que ces deux modes produisent aussi une requête SQL en arrière-plan.

Requête paramétrée

Vous l'aurez probablement remarqué : on ne retrouve pas dans *Base* l'équivalent des bordereaux de recherche de *DB/TextWorks*. Ce qui pourrait s'en approcher le plus est l'utilisation d'une **requête SQL paramétrée**, c'est-à-dire qui permet, lorsqu'elle est exécutée, de demander à la personne qui cherche de préciser certaines valeurs. On pourrait par exemple prévoir une requête SQL pour aller chercher les informations sur les dépenses pour certains items et une année précise. Cette requête paramétrée, comme illustrée ci-dessous, demandera à la personne qui exécute la recherche de saisir une année ainsi qu'un mot ou une expression à chercher dans les intitulés des items. Ce qui permet de transformer une requête en requête paramétrée, comme montré à la fin de l'exemple ci-dessous, est l'insertion dans les critères de variables (paramètres). La syntaxe dans *Base* pour les variables est de préfixer leur nom par un deux-points comme, par exemple, *:Annee* pour permettre de saisir une année. Si le critère pour la recherche dans le champ *item* est plus complexe, c'est parce qu'il faut ajouter la troncature (%) avant et après le mot saisi.

Note : Capsule vidéo accessible en ligne[1]

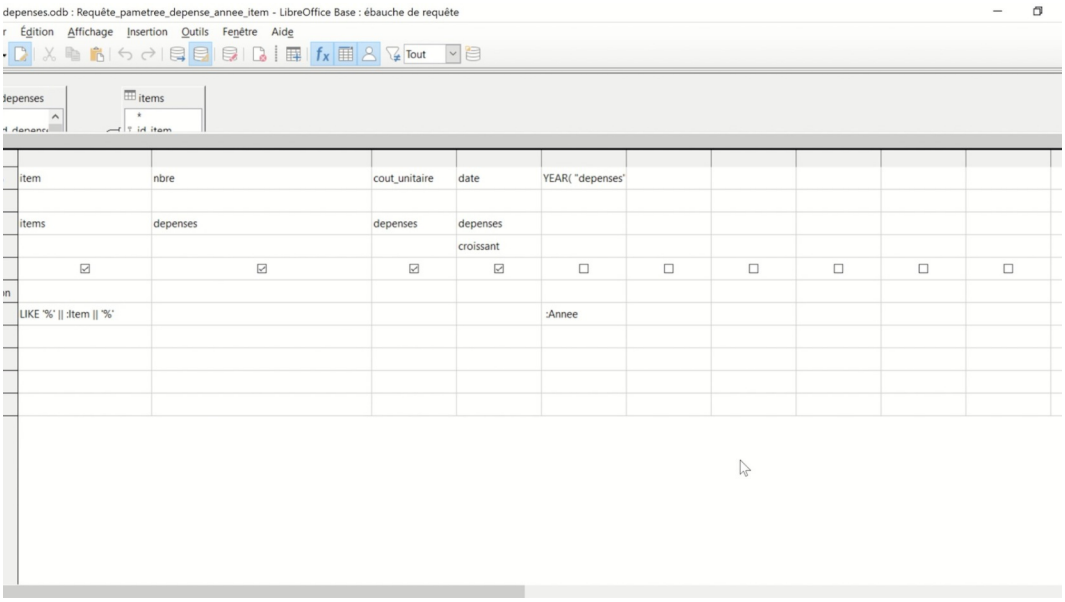
Interface pour la saisie des valeurs des paramètres



Saisie des paramètres pour une requête paramétrée

Requête paramétrée en mode édition

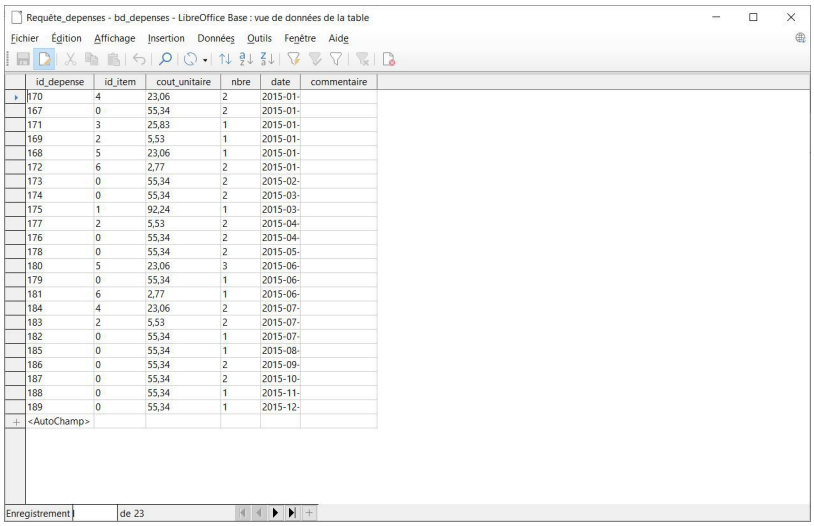
[1] http://cours.ebsi.umontreal.ca/sci6005/a2022/res/base_reqparam.mp4



Requête paramétrée en mode édition

Résultat d'une recherche

Le résultat de l'exécution d'une requête sera une table présentant les **enregistrements repérés** par la requête et les **champs retenus** dans la requête, comme illustré ci-dessous.



Assistant Requête dans Base : Résultat de la requête

4.5. Exploitation de la structuration : Rapport de sortie

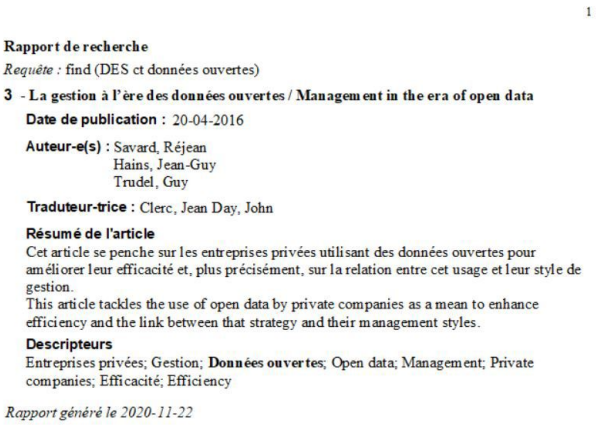
Finalement, il est aussi possible d'exploiter la structuration par champs afin de produire des **rapports sur mesure** à partir des données d'une table, par exemple, ou des résultats d'une requête. Un parallèle peut être fait avec la possibilité de faire du *publipostage* dans un traitement de texte à partir de données dans un tableur. Les **rapports de sortie** dans une base de données reproduisent cette logique : les données présentes dans la base de données peuvent être injectées dans un modèle de rapport. La différence est que tout se fait dans le SGBD.

a) Rapports de sortie dans DB/TextWorks (modèle textuel)

DB/TextWorks permet de créer un ou plusieurs rapports de sortie pour une base de données. Ces rapports de sortie servent pour **présenter les résultats d'une recherche**. Tout comme pour les formulaires de saisie, c'est lors de la conception qu'il faut s'assurer de comprendre quels seront les besoins des utilisateurs et utilisatrices de la base de données au niveau des rapports de sortie. Souhaiteront-ils ou elles une présentation sous forme de tableau ou plutôt de texte? Quelles informations voudront-ils ou elles voir dans leurs rapports de sortie?

L'exemple ci-dessous illustre un rapport de sortie pour la base de données REPSCIE. Tant le choix des informations présentées que la manière de les présenter relèvent du concepteur ou de la conceptrice du rapport. Il est ainsi possible, comme c'est le cas ici :

- De *numéroter* les pages du rapport
- De donner un *titre* au rapport
- D'insérer dans le rapport la *requête de recherche effectuée*
- De décider de l'*ordre de présentation* des informations, de leur *mise en forme* ainsi que de la *présence d'étiquettes* qui précèdent les contenus
- De la *forme* sous laquelle les *occurrences multiples* d'un champ sont présentées (avec un retour de ligne pour les auteurs et un point-virgule pour les descripteurs par exemple)
- De préciser la *date* à laquelle le rapport est *généré* (ce qui est une bonne pratique à adopter!)



1

Rapport de recherche

Requête : find (DES ct données ouvertes)

3 - La gestion à l'ère des données ouvertes / Management in the era of open data

Date de publication : 20-04-2016

Auteur-e(s) : Savard, Réjean
Hains, Jean-Guy
Trudel, Guy

Traduteur-trice : Clerc, Jean Day, John

Résumé de l'article
Cet article se penche sur les entreprises privées utilisant des données ouvertes pour améliorer leur efficacité et, plus précisément, sur la relation entre cet usage et leur style de gestion.
This article tackles the use of open data by private companies as a mean to enhance efficiency and the link between that strategy and their management styles.

Descripteurs
Entreprises privées; Gestion; **Données ouvertes**; Open data; Management; Private companies; Efficacité; Efficiency

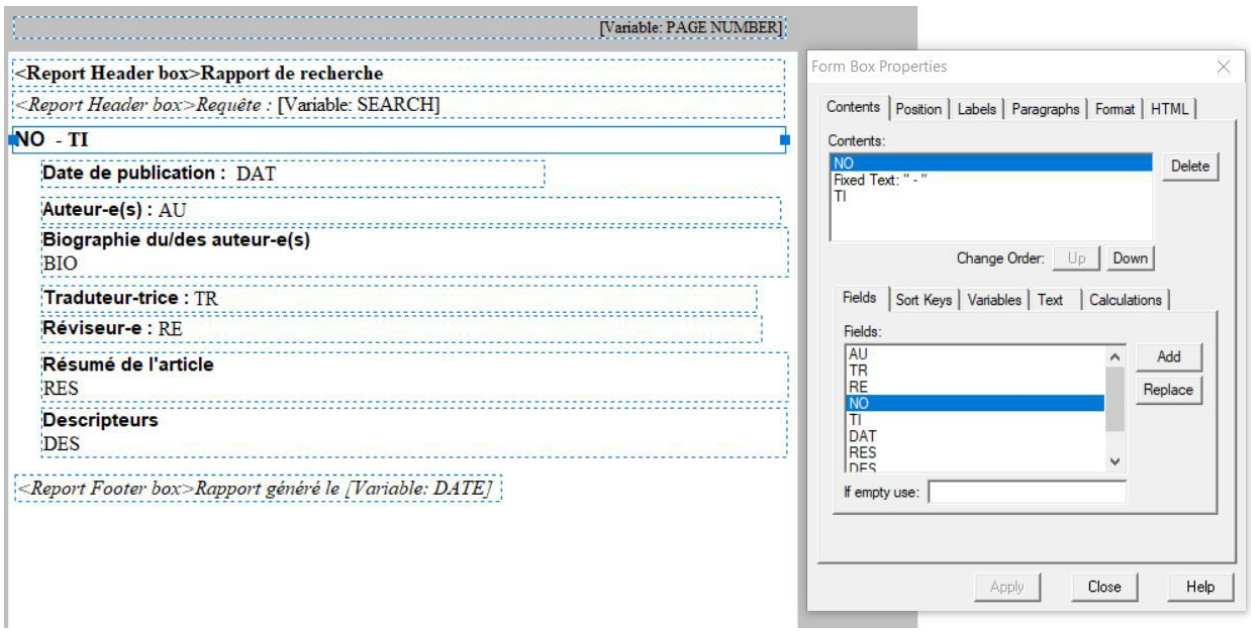
Rapport généré le 2020-11-22

Rapport de sortie dans DB/TextWorks

L'éditeur de rapport suit une logique similaire à celle de l'éditeur de formulaire. Il est possible de *partir de zéro*, comme on peut partir d'un *rapport incluant l'ensemble des champs*.

Comme il s'agit de rapports, on y retrouve en plus des notions d'*entête* et de *pied de page* et de *rapport* ainsi que des *marges* où l'on peut indiquer, par exemple, le numéro de page.

Chacune des **boîtes** que l'on ajoute dans le rapport peut être **associée à plusieurs types d'éléments** : des *champs*, des *chaînes de caractères fixes*, des *variables*. Dans l'exemple ci-dessous, on voit que pour la boîte sélectionnée, son contenu consiste à ce qui se trouve dans le champ *NO* suivi d'un tiret et suivi de ce qui se trouve dans le champ *TI*.

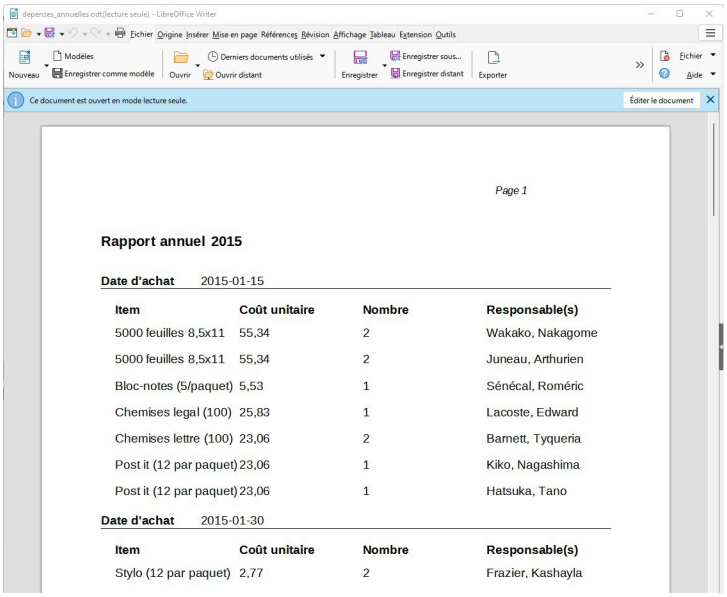


Éditeur de rapport dans DB/TextWorks

b) Rapports de sortie dans Base (modèle relationnel)

Base permet aussi la définition d'un ou plusieurs rapports de sortie pour présenter les *résultats d'une requête* ou le *contenu d'une table*. Tout comme pour les formulaires de saisie, du fait de la structuration plus complexe des informations, les rapports de sortie peuvent être **plus complexes** que ceux de *DB/TextWorks*.

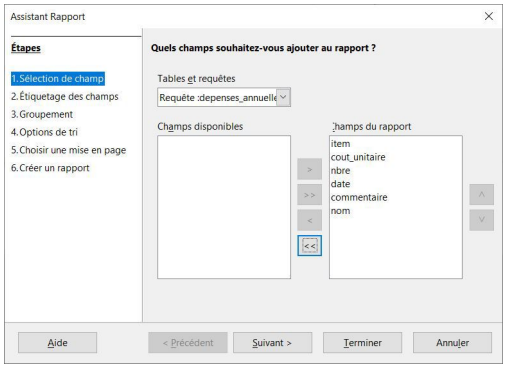
L'exemple ci-dessous illustre un rapport de sortie pour la base de données *Dépenses Papeterie*. Remarquez dans l'entête de la fenêtre que le rapport de sortie est généré dans *Writer*. Ce rapport est basé sur une **requête** qui va extraire des données sur les dépenses ainsi que sur les personnes responsables associées à ces dépenses pour une certaine année. On retrouve les mêmes notions propres à un document mentionnées précédemment : un *entête*, des *titres*, une *pagination*, de la *mise en forme*.



Rapport de sortie dans Base

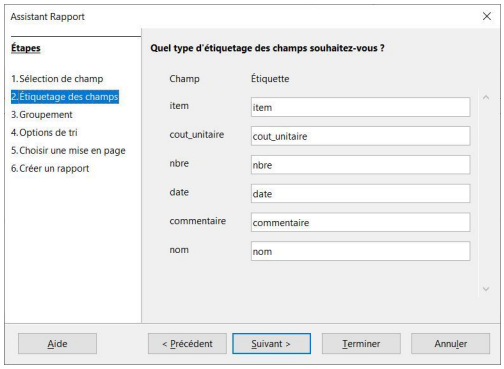
Tout comme pour les formulaires, *Base* offre deux manières pour créer un rapport de sortie : (1) le **mode Ébauche** où vous partez d'un rapport vide et (2) l'**Assistant Rapport** qui offre une assistance pas à pas dans la construction d'un rapport. Il peut être intéressant d'exploiter l'*Assistant Rapport* pour faire un *premier jet* du rapport que l'on *modifie* par la suite. C'est ce qui a été fait dans l'exemple ci-dessous.

À la première étape, **tous les champs** de la **requête paramétrée** permettant d'extraire les dépenses pour une certaine année ont été sélectionnés.



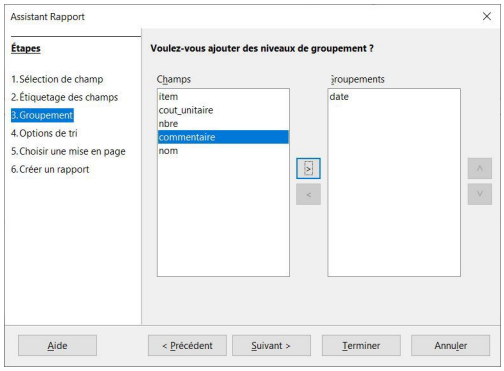
Assistant Rapport dans Base : Choix des champs

Par la suite, des **étiquettes plus significatives** peuvent être proposées en lieu et place des noms des champs.



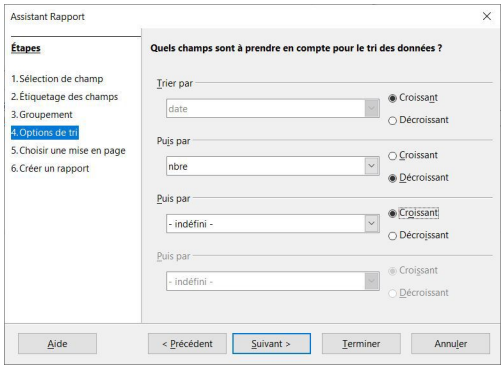
Assistant Rapport dans Base : Définition des étiquettes

Il est possible de **regrouper visuellement** les données par rapport à un ou des champs dans un rapport. Il a été décidé de regrouper toutes les entrées pour une même date.



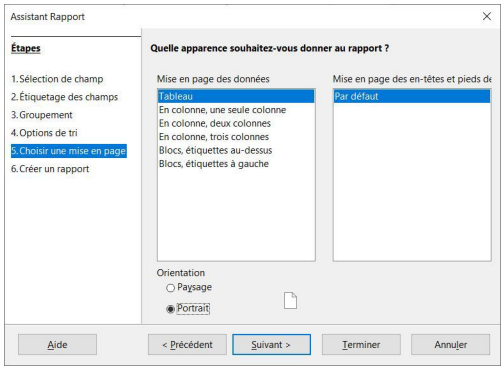
Assistant Rapport dans Base : Définition des regroupements

Comme un regroupement a été choisi, le champ de regroupement est défini par défaut comme première clé de tri. Il est possible d'ajouter **d'autres clés de tri**.



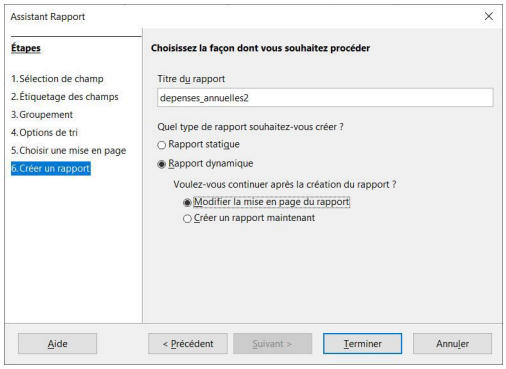
Assistant Rapport dans Base : Définition du tri

Base offre différentes possibilités pour la mise en page. Dans cet exemple, c'est une mise en page en tableau qui a été retenue.



Assistant Rapport dans Base : Choix de la mise en page

Finalement, l'*Assistant* permet de nommer le rapport ainsi que de décider si on veut un *rapport statique* (généré une seule fois) ou un *rapport dynamique* (généré à la demande). Une fois ces choix effectués, il est possible de décider d'ouvrir le rapport en *visualisation* ou en *modification* pour continuer à le peaufiner.



Assistant Rapport dans Base : Finalisation

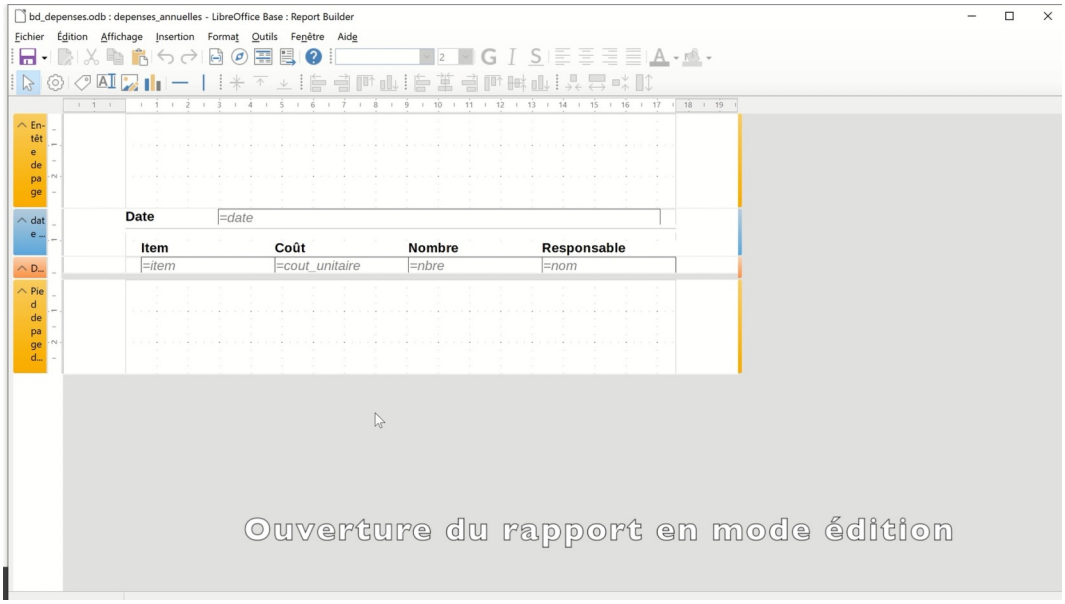
Une fois un rapport de base généré à partir de l'*Assistant*, il est possible de l'**ouvrir en édition** pour l'améliorer. L'exemple ci-dessous illustre quelques modifications de base, notamment :

- Ajout de l'**en-tête et du pied de rapport**
  - On peut retrouver deux niveaux d'en-tête et de pied dans un rapport : (1) au niveau de la *page* (répétition à chaque page) et (2) au niveau du *rapport* (une seule fois au tout début et à la toute fin).
- Ajout d'une **pagination**
- **Redimensionnement** des différentes zones pour éviter, d'une part, une perte d'espace et, d'autre part, aérer un peu le corps du rapport
- Ajout d'une **étiquette** et de la **date** et de l'**heure** à la fin du rapport
  - Comme les rapports sont générés à partir de données changeantes dans le temps, indiquer la date et l'heure de la création d'un rapport est une bonne pratique à adopter.

Il est à noter que les modifications se font dans *Base*, directement dans le rapport, et non dans *Writer* où les rapports spécifiques sont générés.

Note : Capsule vidéo accessible en ligne[1]

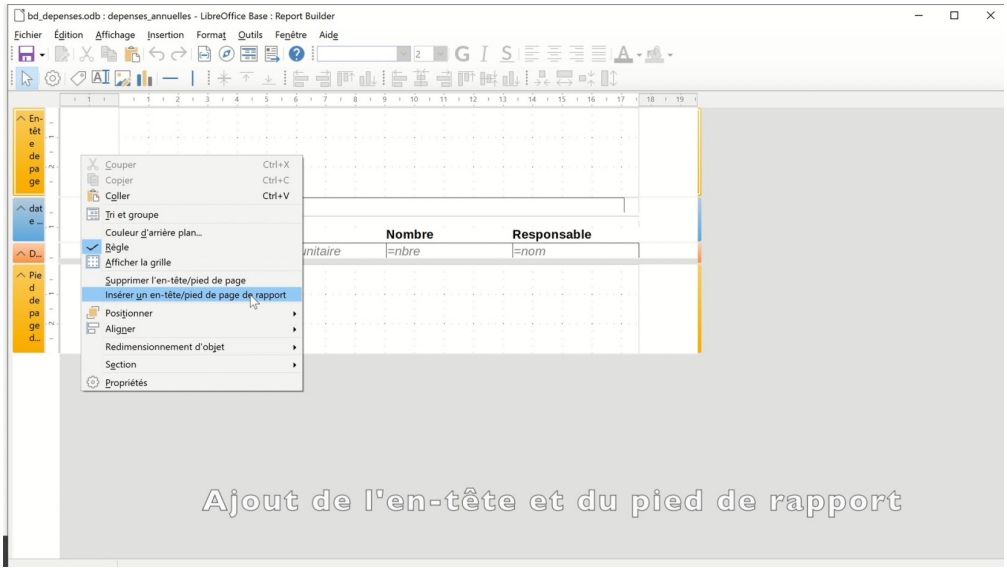
Première étape : Ouverture du rapport en **mode édition**



Ouverture du rapport dans l'éditeur de rapport (Report builder)

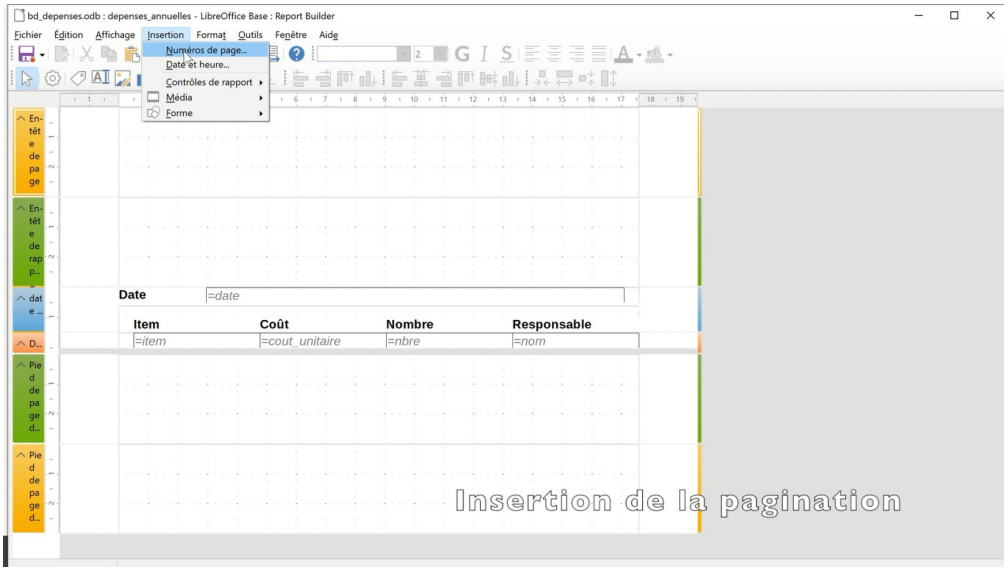
[1] http://cours.ebsi.umontreal.ca/sci6005/a2022/res/base_rapport_edition.mp4

Deuxième étape : Ajout de l'**en-tête** et du **pied** de **rapport** (clic-droit dans l'en-tête de la page)



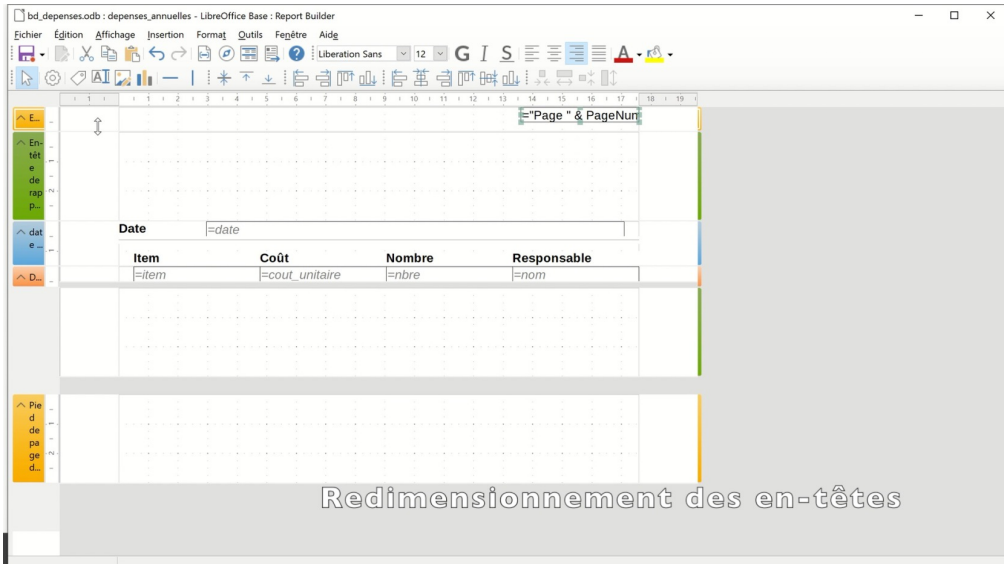
Ajout de l'en-tête et du pied du rapport

Troisième étape : Ajout de la **pagination**



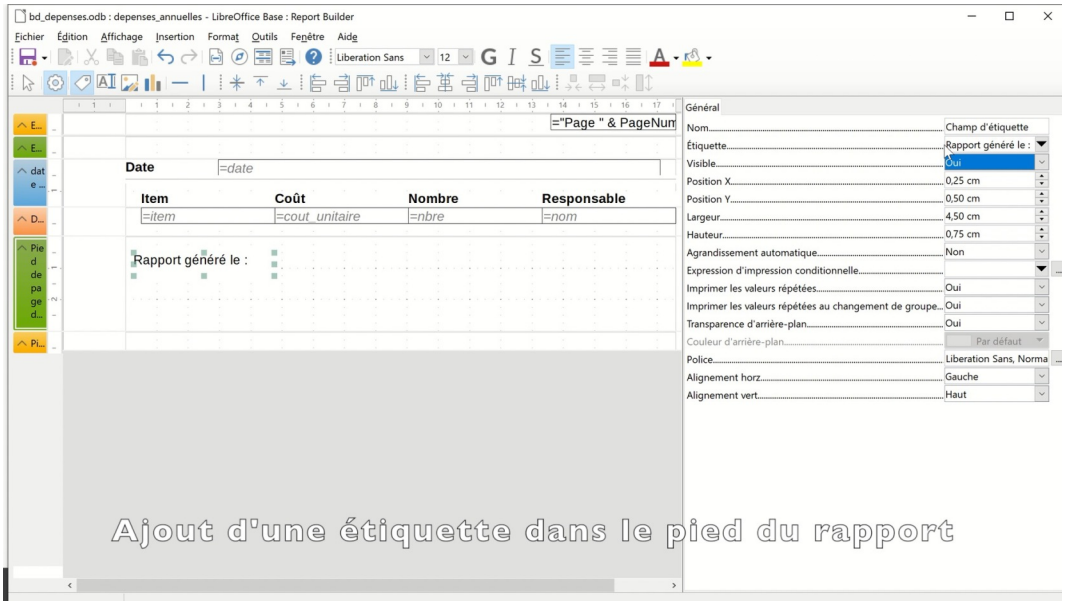
Insertion de la pagination dans l'en-tête des pages

Quatrième étape : **Redimensionnement** des zones



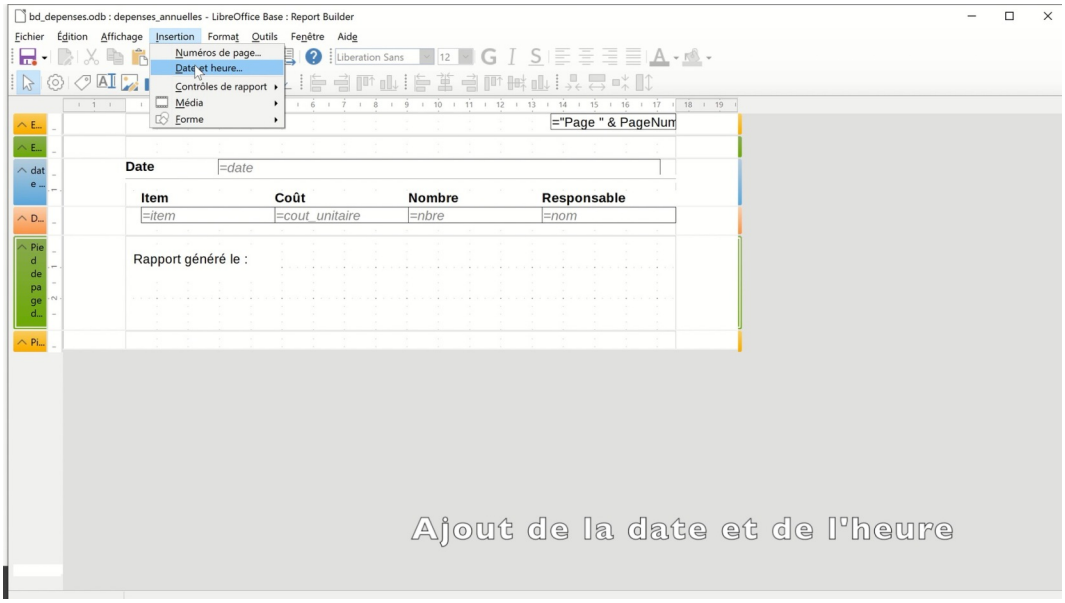
Redimensionnement des en-têtes pour éviter la perte d'espace

Cinquième étape : Ajout d'une **étiquette**



Ajout d'une étiquette dans le pied du rapport pour préciser la nature de la date et de l'heure

Sixième étape : Ajout de la **date** et de l'**heure**



Ajout de la date et de l'heure dans le pied du rapport

5. Ressources en lien avec le cours

Matériel de cours

- *Notes de cours [cf.]*

Note : Du matériel complémentaire est précisé dans le protocole du *TP Structuration dans une base de données* en lien avec les manipulations dans *Base* (*Libre Office*).

Index

Crédits des ressources

p. 3

http://creativecommons.org/licenses/publicdomain/4.0/fr/, johnny_automatic

p. 4

http://creativecommons.org/licenses/publicdomain/4.0/fr/, maoriveros