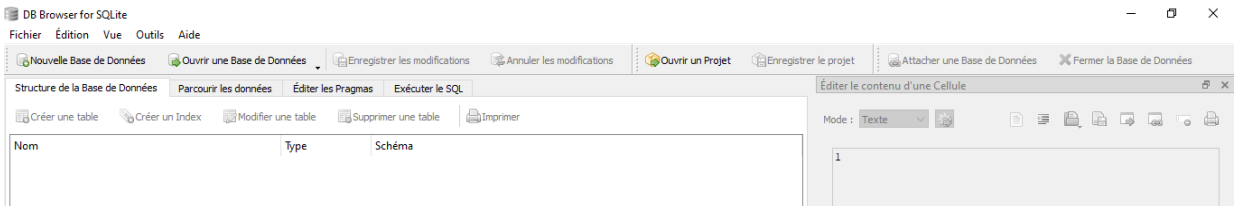
Manipulons SQL

Un exemple de base de données

Vous trouverez le fichier « **exemple** » qui contient une base SQLite déjà remplie avec les informations visibles dans le fichier **Tables_Commandes.pdf**.

Description de la base de données

Pour ouvrir et visualiser la base, nous allons utiliser un logiciel dédié, **DB_Browser(sqlite)**. Cela permet d'avoir une vue graphique sur structure de la base de données.
Ouvrir le logiciel DB Browser((sqlite) 



Cliquer sur « fichier / Ouvrir une base de données » pour ouvrir la base de données : « **exemple** »
La structure de la base de données, montre qu'elle est constituée de 4 Tables (relations) :

CLIENT (RefC, NomC, Ville, CAT)
COMMANDE (RefCom, RefC, DateCom)
DETAIL (RefCom, RefP, Quantite)
PRODUIT (RefP, TypeP,Prix, QStock)

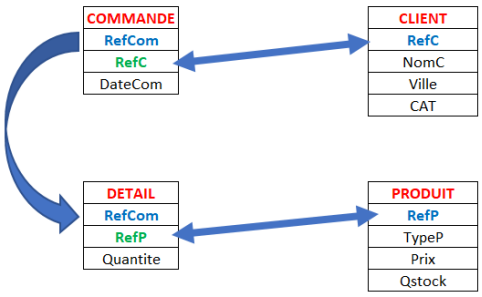
Cette base concerne les commandes des clients d'une entreprise.
Les attributs RefC, RefCom et RefP désignent respectivement la référence d'un client, d'une commande et d'un produit.
L'attribut NomC désigne le nom d'un client.
Une commande porte généralement sur plusieurs produits.
La relation DETAIL donne, pour chaque commande, les produits concernés et pour chacun la quantité commandée.

En cliquant sur chaque relation, on peut observer les attributs associés.

Créer une table | Créer un Index | Imprimer

Nom	Type	Schéma
Tables (4)		
CLIENT		CREATE TABLE CLIENT (RefC INT NOT NULL, NomC VARCHAR(20) NOT NULL, Ville VARCH
RefC	INT	"RefC" INT NOT NULL
NomC	VARCHAR(20)	"NomC" VARCHAR(20) NOT NULL
Ville	VARCHAR(20)	"Ville" VARCHAR(20) NOT NULL
CAT	VARCHAR(2)	"CAT" VARCHAR(2)
COMMANDE		CREATE TABLE COMMANDE(RefCom INT NOT NULL, RefC INT NOT NULL, DateCom DATE
RefCom	INT	"RefCom" INT NOT NULL
RefC	INT	"RefC" INT NOT NULL
DateCom	DATE	"DateCom" DATE NOT NULL
DETAIL		CREATE TABLE DETAIL(RefCOM INT NOT NULL, RefP VARCHAR(5) NOT NULL, Quantite INT
RefCOM	INT	"RefCOM" INT NOT NULL
RefP	VARCHAR(5)	"RefP" VARCHAR(5) NOT NULL
Quantite	INT	"Quantite" INT NOT NULL
PRODUIT		CREATE TABLE "PRODUIT" (RefP VARCHAR(5) NOT NULL, TypeP VARCHAR(20) NOT NULL,
RefP	VARCHAR(5)	"RefP" VARCHAR(5) NOT NULL
TypeP	VARCHAR(20)	"TypeP" VARCHAR(20) NOT NULL
Prix	INT	"Prix" INT NOT NULL
QStock	INT	"QStock" INT
Index (0)		
Vues (0)		

Utilisez le fichier Excel « « Schema_Base_Vierge » pour réaliser le schéma relationnel entre les relations CLIENTS, PRODUIT, COMMANDES et DETAIL.



En cliquant sur l'onglet « **Parcourir les données** », on accède au contenu de chaque relation.

Structure de la Base de Données | Parcourir les données

Table : CLIENT

	RefC	NomC	Ville	CAT
	Filtre	Filtre	Filtre	Filtre
1	1	GOFFIN	Namur	B2
2	2	HANSENNE	Poitiers	C1
3	3	MONTI	Geneve	B2
4	4	GILLET	Toulouse	B1
5	5	AVRON	Toulouse	B1
6	6	FERARD	Poitiers	B2
7	7	MERCIER	Toulouse	
8	8	TOUSSAINT	Poitiers	C1
9	9	PONCELET	Toulouse	B2
10	10	JACOB	Bruxelles	C2
11	11	VANBIST	Lille	B1
12	12	NEUMAN	Toulouse	
13	13	FRANCK	Namur	C1
14	14	VANDERKA	Namur	C1
15	15	GUILLAUME	Paris	B1

Structure de la Base de Données | Parcourir les données

Table : COMMANDE

	RefCom	RefC	DateCom
	Filtre	Filtre	Filtre
1	1	14	2005-12-21
2	2	9	2005-12-22
3	3	14	2005-12-23
4	4	9	2005-12-23
5	5	12	2006-01-02
6	6	9	2006-01-02
7	7	7	2006-01-03

Structure de la Base de Données | Parcourir les données

Table : PRODUIT

	RefP	TypeP	Prix	QStock
	Filtre	Filtre	Filtre	Filtre
1	CH262	Cheville	75	45
2	CH264	Cheville	120	2690
3	CH464	Cheville	220	450
4	CL45	Clou	105	580
5	CL60	Clou	95	134
6	PL222	Planche	230	782
7	PL224	Planche	185	1220

Structure de la Base de Données | Parcourir les données

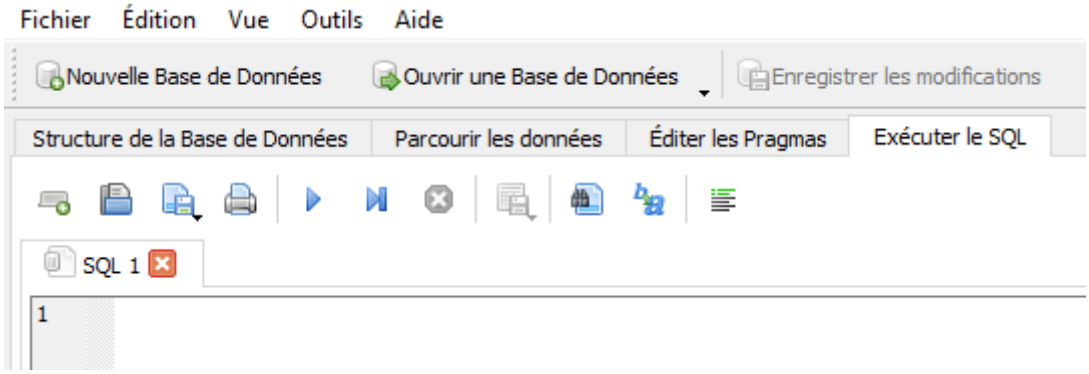
Table : DETAIL

	RefCOM	RefP	Quantite
	Filtre	Filtre	Filtre
1	1	CH464	25
2	2	CH262	60
3	2	CL60	20
4	3	CL60	30
5	4	CH464	120
6	4	CL45	20
7	5	CH464	260
8	5	CL60	15
9	5	PL224	600
10	6	CL45	3
11	7	CH264	180
12	7	CL45	22
13	7	CL60	70
14	7	PL224	92

Manipulons cette base de données : Les requêtes SQL

Vous insérerez les captures d'écran de chaque résultat obtenu.

Pour exécuter une requête (une recherche dans la base de données), on utilise l'onglet « Exécuter le SQL »



— Villes où habite au moins un client;

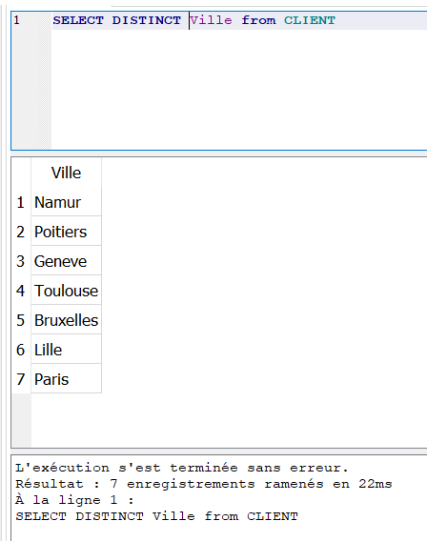
SELECT Ville from CLIENT

```
1    SELECT ville from CLIENT
```

L'utilisation la plus courante de SQL consiste à lire des données issues de la base de données. Cela s'effectue grâce à la commande **SELECT**, qui retourne des enregistrements dans un tableau de résultats. Cette commande peut sélectionner une ou plusieurs colonnes d'une table.
Cette commande SELECT va renvoyer **toutes** les villes où il y a un client. La ville sera écrite **autant de fois** qu'il y a de clients.

SELECT DISTINCT Ville from CLIENT

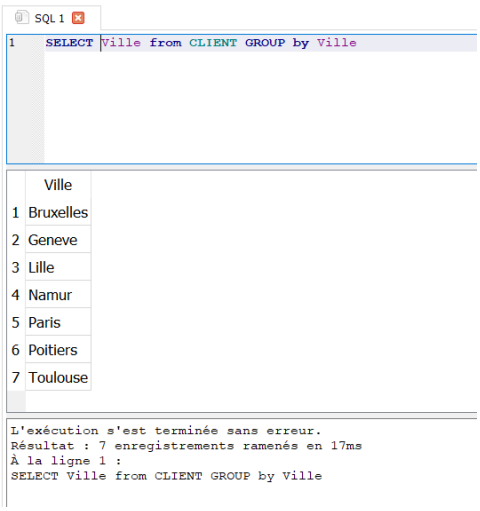
```
1    SELECT DISTINCT ville from CLIENT
```



L'utilisation de la commande SELECT en SQL permet de lire toutes les données d'une ou plusieurs colonnes. Cette commande peut potentiellement afficher des lignes en doubles. Pour éviter des redondances dans les résultats il faut simplement ajouter **DISTINCT** après le mot SELECT.

SELECT Ville from CLIENT GROUP by Ville

```
1    SELECT ville FROM CLIENT GROUP by ville
```

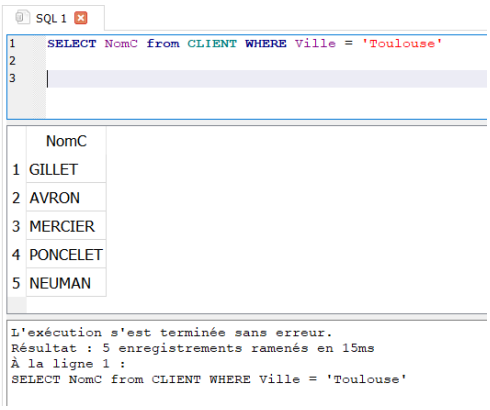


On peut aussi utiliser la commande **GROUP by** qui va ici regrouper par ordre alphabétique les villes identiques.

— Noms des clients qui habitent Toulouse;

SELECT NomC from CLIENT WHERE Ville = 'Toulouse'

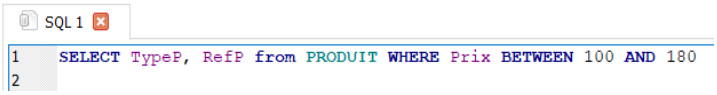
```
1    SELECT NomC from CLIENT WHERE Ville = 'Toulouse'
2
```



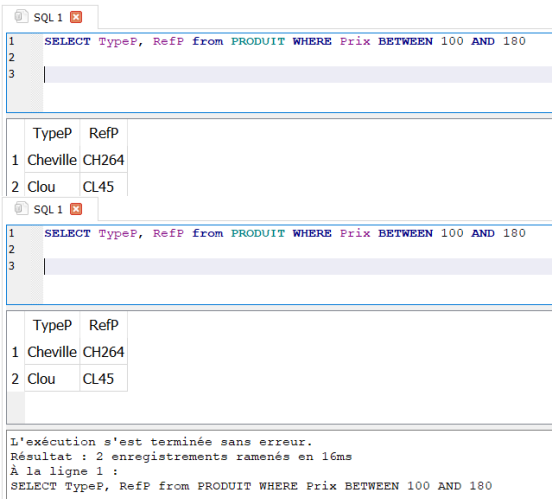
La commande **WHERE** dans une requête SQL permet d'extraire les lignes d'une base de données qui respectent une condition. Cela permet d'obtenir uniquement les informations désirées.

— Références et types des produits dont le prix est compris entre 100 et 180 euros;

SELECT TypeP, RefP from PRODUIT WHERE Prix BETWEEN 100 AND 180



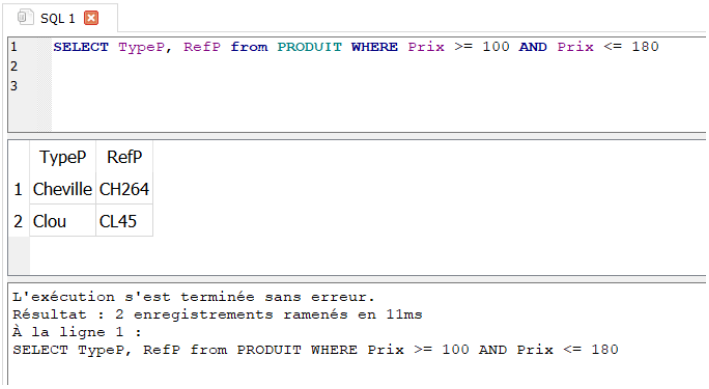
L'opérateur **BETWEEN** est utilisé dans une requête SQL pour sélectionner un intervalle de données dans une requête utilisant WHERE. L'intervalle peut être constitué de chaînes de caractères, de nombres ou de dates.



```
SELECT TypeP, RefP, Prix FROM PRODUIT WHERE prix>=100 AND prix<=180
```

SELECT TypeP, RefP from PRODUIT WHERE Prix >= 100 AND Prix <= 180

Une autre façon de faire en utilisant la commande **AND** associée à « >= » et « <= »

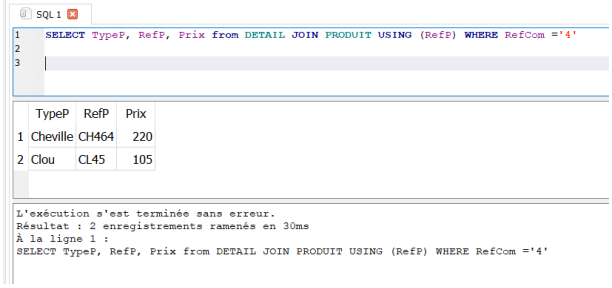


— Types et prix des produits de la commande de référence 4;

SELECT TypeP, RefP, Prix from DETAIL JOIN PRODUIT USING (RefP) WHERE RefCom ='4'

```
SELECT TypeP, RefP, Prix FROM DETAIL JOIN PRODUIT USING(RefP) WHERE RefCom = '4'
```

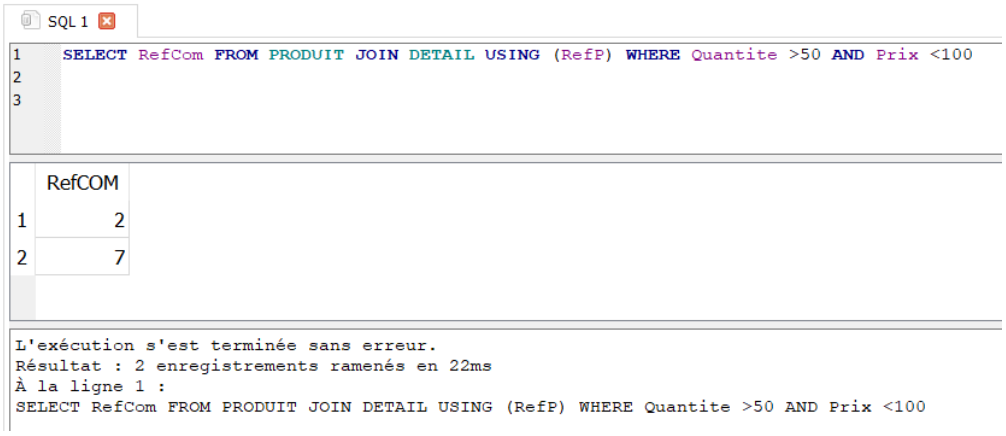
La commande **table1 JOIN table2 USING (clé primaire commune)** permet de relier différentes tables entre elles grâce aux clés primaires.



— Références des commandes qui comportent au moins un produit commandé à la fois pour une quantité supérieure à 50 unités et de prix (prix unitaire) inférieur à 100 euros;

SELECT RefCom FROM PRODUIT JOIN DETAIL USING (RefP) WHERE Quantite >50 AND Prix <100

```
SELECT RefCom FROM PRODUIT JOIN DETAIL USING (RefP) WHERE Quantite >50 AND Prix <100
```



— Références des clients qui ont commandé des clous;

SELECT RefC FROM COMMANDE JOIN DETAIL USING(RefCom) JOIN PRODUIT USING(RefP)WHERE TypeP = 'Clou'

```
SELECT RefC FROM COMMANDE JOIN DETAIL USING(RefCom) JOIN PRODUIT USING(RefP)WHERE TypeP = 'Clou'
```

SQL 1

```
SELECT RefC FROM COMMANDE JOIN DETAIL USING(RefCom) JOIN PRODUIT USING(RefP)WHERE TypeP = 'Clou'
```

	RefC
1	9
2	14
3	9
4	12
5	9
6	7
7	7

```
L'exécution s'est terminée sans erreur.
Résultat : 7 enregistrements ramenés en 19ms
À la ligne 1 :
SELECT RefC FROM COMMANDE JOIN DETAIL USING(RefCom) JOIN PRODUIT USING(RefP)WHERE TypeP = 'Clou'
```

Que faut-il ajouter si on veut les classer par ordre croissant de référence ?

SELECT RefC FROM COMMANDE JOIN DETAIL USING(RefCom) JOIN PRODUIT USING(RefP)WHERE TypeP = 'Clou' ORDER by RefC

SQL 1

```
SELECT RefC FROM COMMANDE JOIN DETAIL USING(RefCom) JOIN PRODUIT USING(RefP)WHERE TypeP = 'Clou' ORDER by RefC
```

	RefC
1	7
2	7
3	9
4	9
5	9
6	12
7	14

```
L'exécution s'est terminée sans erreur.
Résultat : 7 enregistrements ramenés en 15ms
À la ligne 1 :
SELECT RefC FROM COMMANDE JOIN DETAIL USING(RefCom) JOIN PRODUIT USING(RefP)WHERE TypeP = 'Clou' ORDER by RefC
```

Et pour obtenir le nom des clients qui ont commandé des clous ? :

SELECT NomC FROM CLIENT JOIN COMMANDE using(RefC) JOIN DETAIL USING(RefCom) JOIN PRODUIT USING(RefP)WHERE TypeP = 'Clou'

SQL 1

```
SELECT NomC FROM CLIENT JOIN COMMANDE using(RefC) JOIN DETAIL USING(RefCom) JOIN PRODUIT USING(RefP)WHERE TypeP = 'Clou'
```

	NomC
1	PONCELET
2	VANDERKA
3	PONCELET
4	NEUMAN
5	PONCELET
6	MERCIER
7	MERCIER

```
L'exécution s'est terminée sans erreur.
Résultat : 7 enregistrements ramenés en 18ms
À la ligne 1 :
SELECT NomC FROM CLIENT JOIN COMMANDE using(RefC) JOIN DETAIL USING(RefCom) JOIN PRODUIT USING(RefP)WHERE TypeP = 'Clou'
```

— Villes où habite au moins un client qui a commandé un produit à 220 euros.

SELECT Ville FROM CLIENT JOIN COMMANDE using(RefC) JOIN DETAIL USING(RefCom) JOIN PRODUIT USING(RefP)WHERE Prix >=220

```
SELECT Ville
FROM CLIENT JOIN COMMANDE USING(RefC) JOIN DETAIL USING(RefCom) JOIN PRODUIT USING(RefP)
WHERE Prix>=220
```

SQL 1

```
SELECT Ville FROM CLIENT JOIN COMMANDE using(RefC) JOIN DETAIL USING(RefCom) JOIN PRODUIT USING(RefP)WHERE Prix >=220
```

	Ville
1	Namur
2	Toulouse
3	Toulouse

```
L'exécution s'est terminée sans erreur.
Résultat : 3 enregistrements ramenés en 18ms
À la ligne 1 :
SELECT Ville FROM CLIENT JOIN COMMANDE using(RefC) JOIN DETAIL USING(RefCom) JOIN PRODUIT USING(RefP)WHERE Prix >=220
```

Que faut-il ajouter si on veut éviter les doublons ?

SELECT DISTINCT Ville FROM CLIENT JOIN COMMANDE using(RefC) JOIN DETAIL USING(RefCom) JOIN PRODUIT USING(RefP)WHERE Prix >=220

SQL 1

```
SELECT DISTINCT Ville FROM CLIENT JOIN COMMANDE using(RefC) JOIN DETAIL USING(RefCom) JOIN PRODUIT USING(RefP)WHERE Prix >=220
```

	Ville
1	Namur
2	Toulouse

```
L'exécution s'est terminée sans erreur.
Résultat : 2 enregistrements ramenés en 14ms
À la ligne 1 :
SELECT DISTINCT Ville FROM CLIENT JOIN COMMANDE using(RefC) JOIN DETAIL USING(RefCom) JOIN PRODUIT USING(RefP)WHERE Prix >=220
```

Et pour obtenir la référence et le nom des clients qui ont commandé des clous, toujours sans doublons ?

SELECT DISTINCT Ville, NomC FROM CLIENT JOIN COMMANDE using(RefC) JOIN DETAIL USING(RefCom) JOIN PRODUIT USING(RefP) WHERE Prix >=220

SQL 1

```
SELECT DISTINCT Ville, NomC FROM CLIENT JOIN COMMANDE using(RefC) JOIN DETAIL USING(RefCom) JOIN PRODUIT USING(RefP) WHERE Prix >=220
```

	Ville	NomC
1	Namur	VANDERKA
2	Toulouse	PONCELET
3	Toulouse	NEUMAN

```
L'exécution s'est terminée sans erreur.
Résultat : 3 enregistrements ramenés en 20ms
À la ligne 1 :
SELECT DISTINCT Ville, NomC FROM CLIENT JOIN COMMANDE using(RefC) JOIN DETAIL USING(RefCom) JOIN PRODUIT USING(RefP) WHERE Prix >=220
```

PySQLITE(py pour python)

On peut aussi faire quelques manips en Pysqlite, c'est-à-dire créer un programme python qui permet de réaliser une requête.

Écrire une fonction **Clients** qui renvoie la **liste des clients d'une ville donnée** en argument. Essayer d'appliquer cette fonction à la ville 'Toulouse'.

File Edit Format Run Options Window Help

```
                                    #---IMPORTAION BIBLIOTHEQUES---

import sqlite3



                                    #---CREATION DES FONCTIONS---

def Clients_ville_donnee():
    print("Liste des clients d'une ville donnée: ")
    nom_ville=input("Nom de la ville : ")
    curseurNomVille = connexion.execute('SELECT * FROM CLIENT WHERE Ville =?', (nom_ville,))
    for tupleVille in curseurNomVille:
        print(tupleVille[1])



                                    #---PROGRAMME PRINCIPAL---

connexion = sqlite3.connect('exemple.bdd')
connexion.execute('PRAGMA foreign_keys = ON')
Clients_ville_donnee()
connexion.close()
```