

# **NSI Terminale - Bases de données relationnelles**

BDD : introduction

---

qkzk



# Bases de données relationnelles

---

Une *base de donnée* est un concept qui permet de stocker efficacement une immense quantité d'information. Elles permettent de croiser facilement les informations et d'en extraire le contenu.

Les bases de données *relationnelles* sont apparues dans les années 60 et sont encore les plus utilisées à ce jour.

Tous les informaticiens doivent maîtriser un minimum de concepts clés relatifs aux bases de données et doivent être capables de réaliser les manipulations de base que nous allons présenter.

Depuis une décennie, les données sont omniprésentes et parfois moins organisées. On a vu un usage croissant d'un autre type de base de données *NoSQL* très utilisées par les grands acteurs du web.

# Définitions

**Base de donnée (définition large)** : tout ensemble de données stockées numériquement et pouvant servir à un (ou plusieurs) programmes.

**Base de donnée (définition restreinte)** : on appellera base de données un ensemble de données numériques qui possède une structure ; c'est à dire dont l'organisation répond à une logique systématique.

## Exemple

Table :     

	code ▾	region	departement	nom	coordonnees
	Filtre	Fil...	Filtre	Filtre	Filtre
1	59001	31	59	Abancourt	50.2368696873,3.20731301738
2	59002	31	59	Abscon	50.3283410844,3.29787586969
3	59003	31	59	Aibes	50.233709404,4.09622701433
4	59004	31	59	Aix	50.498421043,3.30536772065
5	59005	31	59	Allennes-les-Marais	50.5417063572,2.94865877262

- **BDD** : ensemble des tables.
- **Table** (parfois **relation**) : c'est l'ensemble des **enregistrements** qui existent sur les données
- **Colonne** (parfois **Champs** ou **attributs**) : “departement”, “code” etc. : les différents champs à remplir
- **Ligne** (parfois **Enregistrement** ou **relations**) : “1”, “2” etc. les données elles-mêmes.
- **Cellules** (parfois **cases**) : la valeur elle même.

Dans une même base de donnée, on rencontre souvent plusieurs tables.

Par exemple, pour un compte bancaire :

1. table des transactions du compte courant avec comme champs :

date, numero\_transaction, montant, libelle

Exemple d'enregistrement :

- date : 2019-06-23
- numero\_transaction : "TR123455667"
- montant : "-123.45"

2. table des différents comptes avec comme champs :

numero\_compte, nom\_compte, date\_ouverture

3. table des différents soldes avec comme champs :

numero\_compte, date, solde

## Identifier les enregistrements

Chaque fois qu'on enregistre quelque chose dans la base de donnée il faut s'assurer que la donnée n'est pas *déjà* présente.

On résout ce problème avec la notion de “clé”

### **Clé primaire = Primary Key (PK)**

- Identifie de manière unique une ligne
- Ne doit pas être NULL (vide)
- Peut être composée d'une ou plusieurs colonnes
- Ajout d'une colonne dédiée si besoin

### **Clé étrangère = Foreign Key (FK)**

- Référence une ou plusieurs colonnes d'une autre table (représentant une clé primaire)
- Les colonnes référencées doivent pré-exister dans la table référencée

Il arrive qu'un ordre de transaction ne parvienne pas jusqu'au bout... et qu'on doive le relancer. Comment éviter de facturer plusieurs fois la même chose ?

Chaque enregistrement se voit attribuer une clé primaire unique et quand on essaie d'enregistrer la même transaction, la base de donnée empêche cet ajout.

- Pour notre exemple des comptes, la clé primaire est le numéro de compte. etc.
- Pour notre exemple des transactions, la clé primaire est le numéro de transaction. Le numéro de compte est alors une clé étrangère (le compte doit déjà exister avant qu'on n'y réalise des transactions).

Comment choisir convenablement la présentation des données ?

Imaginez-vous à la tête du service informatique de la sécurité sociale.

Comment enregistrer :

- les patients (des dizaines de millions),
- les actes médicaux (des centaines de milliers par jour),
- sans occuper un espace monstrueux ?

Il faut au moins deux tables (en pratique sûrement une centaine. . . )

- pour les patients avec leur numéro de sécu (on suppose que tous les individus en ont un pour simplifier)
- pour les actes

Dans la table des patients on enregistre les données “permanentes” :

- n° de sécu, nom, prénom, date de naissance, date de décès éventuel etc.

Dans la table des actes médicaux, on ne reprend pas toute la fiche du patient, seulement son numéro de sécu.

Ainsi, le numéro de sécu est à la fois :

- *la clé primaire* de la table “patients”
- *une clé secondaire* de la table “actes”

# Construire une structure optimisée

## Regrouper les données en tables

- Mettre dans une même table les données relatives à un même sujet
- Créer de nouvelles tables pour éviter la redondance des données
  - Limite les incohérences lors des mises à jour
  - Facilite la construction des requêtes et améliore la pertinence des résultats

## Établir les relations entre tables

- Définir les clés primaires
  - Uniques et non NULL
- Définir les clés étrangères
  - Référencent les clés primaires

## Définir des colonnes pertinentes

- Facilité d'interrogation des colonnes
- Données cohérentes au sein d'une colonne
- Ne pas conserver des données qui peuvent être calculées

## Cas des catégories socioprofessionnelles du Nord.

Code g	Rég	Dépt	Libellé géo	Coordonnées	Date	var	Populat	Sexe	Tranche	Catégorie Socio-Professionnelle	catégorie
59001	31	59	Abancourt	50.2368696873.3.20	2012	15+ - Prof. intermédiaires	28	Femmes	15+	Professions Intermédiaires	Prof. Intermédiaires
59001	31	59	Abancourt	50.2368696873.3.20	2012	15+ - Employés	64	Total	15+	Employés	Employés
59001	31	59	Abancourt	50.2368696873.3.20	2012	55+ - Agriculteurs exploitants	0	Total	55+	Agriculteurs Exploitants	Agriculteurs
59001	31	59	Abancourt	50.2368696873.3.20	2012	15-24 - Employés	16	Total	15-24	Employés	Employés
59001	31	59	Abancourt	50.2368696873.3.20	2012	25-54 - Autres	12	Total	25-54	Autres	Autres
59001	31	59	Abancourt	50.2368696873.3.20	2012	25-54 -	176	Total	25-54	Total	Total
59001	31	59	Abancourt	50.2368696873.3.20	2012	15+ -	196	Hommes	15+	Total	Total
59001	31	59	Abancourt	50.2368696873.3.20	2012	15+ - Artisans, Comm., Chefs entr.	44	Total	15+	Artisans, Commerçants ,Chefs d'entreprises	Chefs d'entreprises
59001	31	59	Abancourt	50.2368696873.3.20	2012	15+ - Autres	32	Total	15+	Autres	Autres
59001	31	59	Abancourt	50.2368696873.3.20	2012	15+ - Ouvriers	8	Femmes	15+	Ouvriers	Ouvriers
59001	31	59	Abancourt	50.2368696873.3.20	2012	25-54 - Artisans, Comm., Chefs entr.	28	Total	25-54	Artisans, Commerçants ,Chefs d'entreprises	Chefs d'entreprises
59001	31	59	Abancourt	50.2368696873.3.20	2012	25-54 - Ouvriers	28	Total	25-54	Ouvriers	Ouvriers
59001	31	59	Abancourt	50.2368696873.3.20	2012	55+ - Autres	0	Total	55+	Autres	Autres
59001	31	59	Abancourt	50.2368696873.3.20	2012	15+ - Artisans, Comm., Chefs entr.	24	Hommes	15+	Artisans, Commerçants ,Chefs d'entreprises	Chefs d'entreprises
59001	31	59	Abancourt	50.2368696873.3.20	2012	Population en 2012 (princ)	445	Total	Population	Total	Total
59001	31	59	Abancourt	50.2368696873.3.20	2012	15+ - Cadres, Prof. intel. sup.	16	Total	15+	Cadres, Professions Intellectuelles Supérieures	Cadres, PIS
59001	31	59	Abancourt	50.2368696873.3.20	2012	15+ - Retraités	48	Femmes	15+	Retraités	Retraités
59001	31	59	Abancourt	50.2368696873.3.20	2012	55+ - Prof. intermédiaires	8	Total	55+	Professions Intermédiaires	Prof. Intermédiaires
59001	31	59	Abancourt	50.2368696873.3.20	2012	15+ - Employés	20	Hommes	15+	Employés	Employés

Figure 1: Fichier d'origine : CSV

# Construire une structure pertinente

## Regrouper les données en tables

- Tables = données relatives à un même sujet
  - Données sur les villes et sur les effectifs des catégories

## Éviter la redondance des colonnes

- var = Tranche + categorie
- categorie = abréviation Catégorie socioprofessionnelle

## Éviter la redondance des valeurs

- Les colonnes `departement` et `region` ne contiennent qu'une seule valeur
- À conserver uniquement si volonté d'étendre à d'autres données

## Données cohérentes

- Les colonnes `Sexe`, `Tranche`, ... contiennent aussi des totaux !
  - Les totaux peuvent être calculés à l'aide de fonctions et d'agrégats
- Décomposer `coordonnees` en deux `REAL`

# SQL : Structured Query Language

---

## Langage informatique servant à exploiter des bases de données relationnelles

- Manipulation des données
  - Recherche de données : `SELECT`
  - Ajout de données : `INSERT`
  - Modification de données : `UPDATE`
  - Suppression de données : `DELETE`

- Définition des données
  - Manipule les structures de données de la base
  - Création de tables et autres structures : `CREATE`
  - ...

- Contrôle des données et des transactions
  - Gestion des autorisations d'accès aux données par les différents utilisateurs
  - Gestion de l'exécution de transactions
    - Transaction = suite d'opérations de modification de la base de données

### **SGBDR = Système de Gestion de Bases de Données Relationnelle**

- Logiciel permettant de manipuler le contenu des bases de données relationnelles
- Garantit la qualité, la pérennité et la confidentialité des informations
- Exemple : SQLite est un SGBDR dont le code source est dans le domaine public



### **C'est un langage déclaratif**

- Décrit le résultat voulu sans décrire la manière de l'obtenir
- Les SGBDR déterminent automatiquement la manière optimale d'effectuer les opérations nécessaires à l'obtention du résultat

- SQLite : open source. n'utilise pas de serveur. Stocke la base dans un unique fichier. Très rapide pour des bases "modestes" (< 1 million d'enregistrements)
- MySQL, PostgreSQL, Oracle etc. : (Oracle n'est pas open source). Utilisent un serveur. Très rapides pour des bases "conséquentes".

La majorité de ces logiciels sont écrits en C ou en C++.

- Certains sont spécialisés (graphiques, données numériques précises etc)
- D'autres sont fournis avec un serveur dans le cloud etc.

Tous implémentent les fonctions de base de SQL + quelques fonctions "perso" parfois commodes.

Nous utiliserons SQLite qui est implémenté :

- en Python (`import sqlite3`)
- sous windows / linux / osx (DB Browser, `$ sqlitebrowser`)
- ainsi qu'en ligne.

mais aussi

- iOS,
- Android etc.

## Extraction des données d'une table

```
SELECT noms_colonnes_séparés_par_virgules  
FROM nom_table;
```

Sélectionne toutes les lignes d'une table

## Précisions sur les colonnes affichées

- \* pour toutes les colonnes

```
SELECT *  
FROM nom_table;
```

- `DISTINCT` pour sélectionner une seule occurrence de chaque valeur de la colonne en question

```
SELECT DISTINCT nom_colonne  
FROM nom_table;
```

Exemple :

```
SELECT DISTINCT categorie, genre  
FROM evolution;
```

```
SELECT noms_colonnes_séparés_par_virgules  
      FROM nom_table  
      WHERE nom_colonne op_comp valeur op_bool nom_colonne op_comp valeur;
```

Sélectionne uniquement les lignes qui respectent la clause du WHERE

## La clause porte sur les valeurs des colonnes

- Utilisation d'opérateurs de comparaison (op\_comp) : =, <>, !=, >, >=, <, <=
- Utilisation d'opérateurs booléens (op\_bool) : AND, OR
  - AND : combinaisons de conditions sur des colonnes différentes
  - OR : plusieurs valeurs possibles pour une même colonne

Exemples :

```
SELECT code, effectif
FROM evolution
WHERE categorie="Agriculteurs Exploitants" AND genre="Femmes";
```

```
SELECT code, categorie, effectif
FROM evolution
WHERE categorie="Agriculteurs Exploitants" OR categorie="Ouvriers";
```

## Extraction des données d'une table

```
SELECT abbrev.nom_colonne AS nom_affiché  
FROM nom_table AS abbrev  
ORDER BY nom_colonne [DESC];
```

Change l'affichage et le nommage des données

### AS

- Associé à un nom de colonne : change le nom affiché de la colonne dans le résultat.
- Associé à un nom de table : permet d'abrévier le nom de la table pour préciser de quelle table provient une colonne dont le nom est utilisé par plusieurs tables. Cette abréviation **doit** être utilisée dans le reste de la requête.

### ORDER BY

- Trie les données selon la colonne précisée.
- Par défaut, le tri est dans l'ordre croissant, DESC permet d'obtenir l'ordre décroissant.

```
SELECT FONCTION(nom_colonne)
  FROM nom_table;
```

Applique une fonction sur les valeurs d'une colonne

- COUNT : compte le nombre de lignes sélectionnées.
- MIN, MAX : renvoie la valeur minimum ou maximum de la colonne, parmi les lignes sélectionnées
- SUM, AVG : calcule la somme ou la moyenne des valeurs **numériques** de la colonne, parmi les lignes sélectionnées

Exemple :

```
SELECT AVG(effectif) AS Moy_employes
FROM evolution
WHERE categorie="Employés";
```

- Pas au programme, GROUP BY : agrège ensemble les valeurs identiques d'une colonne pour appliquer une fonction à chacun des sous-ensembles

Exemple :

```
SELECT code, AVG(effectif) AS Moy_employes
FROM evolution
WHERE categorie="Employés"
GROUP BY code;
```

Calcule la moyenne des effectifs des Employés (Hommes et Femmes) pour chaque ville.

## Exemples avec ou sans agrégat

moy_employes	
1	260.214615384615

```
Result: 1 enregistrements ramenés en 13ms  
At line 29:  
SELECT AVG(effectif) AS moy_employes  
      FROM evolution  
     WHERE categorie="Employés";
```

	code	moy_employes
1	59001	32.0
2	59002	233.0
3	59003	14.0
4	59004	76.0
5	59005	267.5
6	59006	18.0

```
Result: 650 enregistrements ramenés en 36ms  
At line 16:  
SELECT code, AVG(effectif) AS moy_employes  
      FROM evolution  
     WHERE categorie="Employés"  
     GROUP BY code;
```

## Extraction des données de deux tables

### Produit cartésien

- Comme son nom l'indique, génère de façon exhaustive toutes les associations possibles entre les lignes des deux tables
  - $Nb\_total\_lignes = Nb\_lignes\_ville * Nb\_lignes\_evolution = 650 * 10400$
- Non pertinent

	code	code	categorie	genre	effectif
1	59001	59120	Professions Intermédiaires	Femmes	128
2	59002	59120	Professions Intermédiaires	Femmes	128
3	59003	59120	Professions Intermédiaires	Femmes	128
4	59004	59120	Professions Intermédiaires	Femmes	128
5	59005	59120	Professions Intermédiaires	Femmes	128
6	59006	59120	Professions Intermédiaires	Femmes	128

Result: 6760000 enregistrements ramenés en 243ms

## Extraction des données de deux tables

### JOIN ON

- Génère uniquement les associations entre les lignes qui sont liées par des clés primaires et étrangères identiques.
  - $Nb\_total\_lignes = Nb\_lignes\_table\_clé\_étrangère = NB\_lignes\_evolution$
- À utiliser pour associer deux tables

	code	code	categorie	genre	effectif
1	59120	59120	Professions Intermédiaires	Femmes	128
2	59137	59137	Professions Intermédiaires	Hommes	8
3	59648	59648	Artisans, Commerçants ,Chefs d'entreprises	Hommes	173
4	59288	59288	Retraités	Hommes	230
5	59033	59033	Employés	Femmes	878
6	59596	59596	Cadres, Professions Intellectuelles Supérie...	Hommes	28

```
Result: 10400 enregistrements ramenés en 108ms
```

```
At line 29:
```

```
SELECT v.code, e.code, categorie, genre, effectif
```

### Syntaxe

- INSERT : ajoute une nouvelle ligne de données dans une table

```
INSERT INTO nom_table VALUES (liste_valeurs_dans_ordre_colonnes_table);
```

```
INSERT INTO nom_table (liste_nom_colonnes_à_remplir)
```

```
VALUES (liste_des_valeurs_à_insérer_dans_ordre_liste_colonnes);
```

- UPDATE : met à jour la ou les lignes qui respectent la clause du WHERE

```
UPDATE nom_table SET nom_colonne1=valeur1, nom_colonne2=valeur2
WHERE nom_colonne op_comp valeur op_bool nom_colonne op_comp valeur;
```

- DELETE : efface la ou les lignes d'une table qui respectent la clause du WHERE

```
DELETE FROM nom_table WHERE nom_colonne op_comp valeur op_bool nom_colonne
```

## Respect de l'intégrité des données

- Une clé primaire doit être unique et non NULL
  - On ne peut pas insérer une ligne avec une clé primaire qui existe déjà.
  - On ne peut pas modifier la valeur d'une clé primaire en une autre valeur qui existe déjà.
- Une clé étrangère doit référencer une clé primaire existante
  - Il faut créer la ligne contenant la clé primaire avant une ligne contenant une clé étrangère la référençant.
  - On ne peut pas modifier une clé primaire si elle est déjà référencée.
  - On ne peut pas effacer une ligne contenant une clé primaire déjà référencée.
- Il est possible de mettre des contraintes sur les clés pour gérer les cascades de modifications (interdiction ou gestion automatique)