Introduction aux bases de données
Cours 2 : Les tables ou relations

Jean Francis Michon [1]

[1]Université Nationale d'Economie, Kharkov

23 Octobre 2021

Bases de données relationnelle : les tables

- Les **données** sont stockées sous formes de **tables** appelées aussi **relations** (on va voir pourquoi). Les SGBD qui stockent et manipulent les données sous forme de tables sont dits **relationnels**.
- Des outils doivent être disponibles dans le SGBD pour importer, mettre à jour, classer, relier ces tables. Ces opérations s'appellent des **requêtes**.
- Les SGBD relationnels les plus connus : SQL Server, Oracle, MySQL, MariaDB, Postgres SQL, SQL Lite, IBM DB2,...

Table, attributs, enregistrements

Supposons que nous voulions enregistrer des écrivains et leurs œuvres. Nous concevons une table **ecrivains**.

- Les noms de colonnes sont appelés **attributs** ou les **champs**
- Chaque ligne de la table s'appelle un **enregistrement** ou un **tuple**

nom_auteur	prenom_auteur	titre_ouvrage
Baudelaire	Charles	Les fleurs du mal
Corneille	Pierre	Médée
Racine	Jean	Britannicus
Orwell	George	1984
Dostoïevski	Fiodor Mikhaïlovitch	Les frères Karamazov

Таблица : ecrivains

Valeur de champ NULL

Lorsque la valeur du champ n'est pas connue sa valeur est mise à NULL. Cela ne signifie pas zéro !
Cela peut poser des problèmes lors de certaines opérations (concaténation de chaînes: que vaut "rue" + NULL ?)
Evidemment il faut éviter les valeurs NULL dans les clés (voir la suite)

Le modèle relationnel : Edgar Frank Codd (1923-2003)

- Codd est un mathématicien-informaticien britannique qui a formalisé (mathématiquement) les opérations sur les données alors qu'il travaillait au centre de recherche de IBM.
- Il invente le modèle relationnel qui est publié en 1970 : **A Relational Model of Data for Large Shared Data Banks** ("Un modèle de données relationnel pour de grandes banques de données partagées"), CACM 13, No. 6, June 1970.
- Elison fonde Oracle en 1977 qui utilise cette théorie.

Table = Relation ?

Mathématiques : Soient A et B sont deux ensembles quelconques (finis ou infinis). Une **relation binaire** $\mathcal{R}$ entre A et B est un sous-ensemble quelconque de $A \times B$:

$$\mathcal{R} \subset A \times B$$

c'est-à-dire un sous-ensemble de l'ensemble de couples (donc ordonnés) (a, b) avec $a \in A$ et $b \in B$.
Exemple: $A = \{Pierre, Paul, Elisabeth\}$, $B = \{0, 1, \ldots, 10\}$ et

$$\mathcal{R} = \{(Pierre, 5), (Elisabeth, 9)\}$$

On représente $\mathcal{R}$ par le tableau

A	B
Pierre	5
Élisabeth	9

Таблица : $\mathcal{R}$

Table = Relation k-aire?

Généralisation : $A_1, \ldots, A_k$ une famille de n ensembles quelconques (finis ou infinis). Une **relation k-aire** sur ces ensembles $\mathcal{R}$ est un sous-ensemble quelconque de $A_1 \times \cdots \times A_k$ et k est son **degré** :

$$\mathcal{R} \subset A_1 \times \cdots \times A_k$$

Si vous reprenez la table **ecrivains**, on peut l'interpréter comme une relation 3-aire sur les ensembles

- A_1= ensemble de noms
- A_2= ensemble de prénoms
- A_3= ensemble de titres

$$\textbf{ecrivains} \subset A_1 \times A_2 \times A_3$$

c'est une relation de degré 3.

Le langage SQL

Après la théorie de Codd (basée sur les mathématiques) , c'est Chamberlain et Boyce qui développent SQL en 1970 (en fait SEQUEL: Structured English QUEry Language) pour manipuler les données du System R, une base de donnée expérimentale d'IBM. Il est fait pour être facile à comprendre.

- 1979, Oracle qui s'appelle alors Relational le diffuse commercialement
- Normalisation en 1986
- Dernière norme SQL : 2016

Langage déclaratif (on dit ce qu'on veut obtenir mais pas la manière de l'obtenir) : il permet de définir, manipuler, contrôler les données et de faire des suites de traitements (transactions). L'objectif de SQL est l'**accès rapide aux données**.

Le domaine d'un attribut

Ici commencent les difficultés :

- Le **domaine** est l'ensemble des valeurs possibles pour un attibut. Ils dépendent (souvent, malheureusement) du logiciel SGBD employé (Oracle, SQLServer, MariaDB,...)
- Cela correspond exactement au **type** d'une variable dans un langage de programmation (C, Python,...)
- L'objectif est d'avoir une taille fixe pour chaque enregistrement ce qui facilite l'accès à son contenu
- C'est le langage SQL qui impose la dénomination des types, mais les variantes de ce langage imposent des traductions dangereuses lorsqu'on passe d'un système à l'autre !

schéma d'une relation = nom de la table + noms des attributs

Les différents domaines

- Les chaînes de caractères (combien, UTF8 ?)
- les nombres (entiers, réels, positifs , bornes ?)
- Les chaînes structurées : tél, mél, code postal
- Les valeurs booléennes (Vrai, Faux, Oui, Non...)
- Les longs textes, les images, etc...

Dans une table, on respecte les **contraintes de domaine**: Respect du domaine +**Pas d'attribut multivalué** : plusieurs valeurs, plusieurs mots dans un attribut. On dit que la valeur d'un attribut est **atomique**.

Table $\neq$ Objet ?

Les objets des langages de la POO (C++, Python, Java,...) ont des attributs , mais aussi des **méthodes**. Ils peuvent aussi avoir des attributs identiques. Ce n'est pas le cas des enregistrements d'une table.
Les enregistrements d'une table T sont donc bien des objets instances de la "classe" T (voir le code ci-dessous)
Mais un ensemble d'objets instances d'une classe X est, en général, plus complexe qu'une table.

Table (simple) en Python

```
#Table ecrivain vue comme une liste d'objet en Python
#
class ecrivain:
    def __init__(self,nom_aut,prenom_aut,titre_ouv,an_publi):
        self.nom_aut=nom_aut
        self.prenom_aut=prenom_aut
        self.titre_ouv=titre_ouv
        self.an_publi=an_publi
a=ecrivain('Baudelaire','Charles','Les fleurs du mal',1857)
b=ecrivain('Corneille','Pierre','Médée',1639)
c=ecrivain('Булгаков','Михаил Афанасьевич',
'Белая гвардия',1927)
maliste=[a,b,c]
for x in maliste :
    print(x.nom_aut,x.prenom_aut,x.titre_ouv,x.an_publi)
```

MariaDB : types numériques

Consultez https://mariadb.com/kb/en/data-types/
Types numériques usuels : TINYINT (1 octet) = BOOLEAN (avec TRUE=1, FALSE=0), INT (32 bits signés), BIGINT (64 bits), DOUBLE (64 bits).
Exemple d'utilisation en SQL : Nous allons effectuer 3 instructions en langage SQL ci-dessous permettent de :

- créer une table nommée **t1** avec un attribut nommé **d**
- créer 3 enregistrements dans cette table
- afficher **t1**

MariaDB : table avec type numérique

```
/* on suppose que la base de données est créée. */
/* création de la table */
CREATE TABLE t1 (d DOUBLE(5,0) ZEROFILL);
/* insertion de 3 tuples */
INSERT INTO t1 VALUES (1),(2),(3);
/* affichage de t1 */
SELECT * FROM t1;
+-------+
| d     |
+-------+
| 00001 |
| 00002 |
| 00003 |
+-------+
```

MariaDB : types chaînes de caractères

Vous trouverez des définitions dans
https://mariadb.com/kb/en/data-types/

- Attention : utilisez l'interclassement : **utf8mb4_unicode_ci** si vous mélangez plusieurs alphabets.
- CHAR(n) n octets, pertes des caractères si plus de n, remplissage avec espaces
- BINARY(n) n octets comme CHAR mais remplissage avec des 0x00
- VARCHAR(n) n caractères UTF8, taille <64 Ko.
- BLOB binaire (<64 Ko), TEXT caractères <64Ko, LONGTEXT <4Go, JSON.

Ces attributs de grandes tailles servent à stocker des images, des programmes, etc...Attention au stockage des apostrophes, retour chariot, etc. **Exemple d'utilisation:**

MariaDB : type VARCHAR()

Trois commandes SQL pour créer une table d'une colonne avec 3 lignes et l'afficher.

```
/* on suppose que la base de données est créée */
/* création de la table : */
CREATE TABLE t2 (nom VARCHAR(10));
/* insertion de 3 tuples */
INSERT INTO t2
VALUES ('Михаил Афанасьевич Булгаков'),
('A. d'Aubigné'),
('Archimède');
/* affichage de la table */
SELECT * FROM t2;
```

MariaDB : types dates

Consultez https://mariadb.com/kb/en/data-types/
Types dates usuels :

- DATE format (YYYY-MM-DD) de 1000-01-01 à 9999-12-31 (rappel : l'année 0 n'existe pas, mais rien à voir avec une vulnérabilité du jour zéro)
- TIME format (HH:MM:SS.ssssss) de -838:59:59.999999' à '838:59:59.999999'. Attention aux conversions par défaut :

  '12 09' = '(297:00:00)'.
- DATETIME (YYYY-MM-DD HH:MM:SS) , TIMESTAMP (YYYY-MM-DD HH:MM:SS),
- YEAR (année sur 4 chiffres)

Petit Dictionnaire

attribute	attribut
field	champs
timestamp	horodatage
char	caractère
blob	données binaires
collate	interclassement

Alignement en mémoire des enregistrements

Pierre	Corneille	2002-5-17	1125
Jean	de la Fontaine	2002-5-17	19
...	...	...	...

Tous les enregistrements d'une table occupent le même espace en mémoire. Les types BLOB ou TEXTE sont stockés séparément. Cela permet des accès rapides aux données
Par exemple et **pour simplifier** si : Prénom VARCHAR(15) , Nom VARCHAR(30), naissance DATE, code INT, la taille occupée en mémoire est

$$4 \times (15 + 30) + 4 + 4 = 188 \text{ octets.}$$