Partie 1

Bases de données relationnelles - cours prof

1.1 Modèle relationnel

Le modèle relationnel est une manière de *modéliser* les relations entre plusieurs informations et de les ordonner entre elles. Modélisation qui repose sur des principes mathématiques. voir ausi wikipedia.org => modèle relationnel

1.2 Principes mathématiques

1.2.1 Définitions

Relation : attributs qui caractérisent une proposition. C'est une *table*.

Exemple : Dupond a un métier, un matricule, il porte une moustache ronde, il a un Sosie-jumeau, un employeur et des amis. Il apparait dans certains albums de Tintin.

uplet : collection ordonnée d'objets, un enregistrement dans la BDD.

Domaine : un pour chaque colonne

Algèbre relationnelle : langage de manipulation des données SQL (Structured Query Langage)

- il n'y a pas d'ordre dans la présentation des lignes du tableau
- il ne peut pas y avoir de doublets

1.2.2 Opérateurs

C'est l'objet du précedent cours sur le langage SQL.

- Union : L'union de deux relations R1 et R2 de même schéma est une relation R3 de schéma identique qui a pour n-uplets les n-uplets de R1 et/ou R2
- Différence : La différence entre deux relations R1 et R2 de même schéma est une relation R3 de schéma identique ayant pour n-uplets les n-uplets de R1 n'appartenant pas à R2.
- Sélection (restriction)
- Projection (produit cartesien) : La projection d'une relation R1 est la relation R2 obtenue en supprimant les attributs de R1 non mentionnés. Correspond à un decoupage vertical
- Intersection : L'intersection entre deux relations R1 et R2 de même schéma est une relation R3 de schéma identique ayant pour n-uplets les n-uplets communs à R1 et R2.

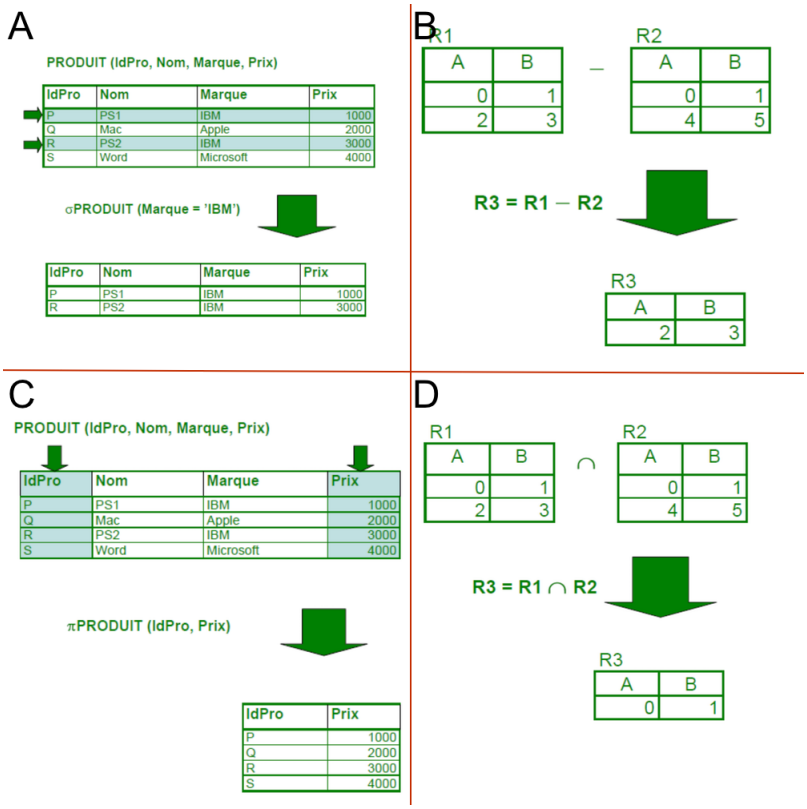

FIGURE 1 – exemples illustrés de quelques opérations

Quels sont les clients de Nice ayant acheté un produit de marque 'Apple' :

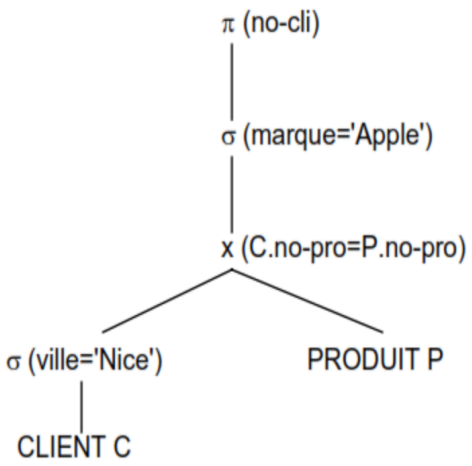

FIGURE 2 – opérations mises sous forme d'un arbre algébrique

Un *arbre algébrique* correspond à une décomposition de la requête en opérateurs élémentaires avec introduction de tables intermédiaires (: c'est donc l'équivalent d'une réécriture de la requête avec des vues.). Les feuilles de l'arbre représentent les tables de départ. La racine de l'arbre représente la table résultat. Tous les autres nœuds de l'arbre sont des opérateurs de l'algèbre relationnelle.

1.2.3 Règles de gestion d'un SGBD

- Unicité : toute information dans la bdd est représentée d'une SEULE et UNIQUE manière

- garantie d'accès : chaque valeur doit être accessible à partir de l'énoncé du nom de la table, de m'attribut clé primaire et de la valeur de cette clé
- Traitement des valeurs nulles : le sgbd doit disposer d'une manière spécifique pour traiter l'absence de valeur (NULL). Cette information doit être différente des autres valeurs
- Contraintes d'intégrité (developpé plus loin)

Partie 2

BDD2 - Cours élève

Une **base de données** est une *collection de données* qui vont être partagées entre plusieurs services, serveurs, utilisateurs.

2.1 Le modèle entité-association

La première étape pour aboutir à un modèle permettant de stocker les données dans une base consiste à *identifier les objets* et définir leurs *liens*.

Cette modélisation qui repose sur des principes mathématiques mis en avant par E.F. Codd (1923 - 2003, un informaticien britannique), et c'est cette modélisation qui est implémentée dans les bases de données.

2.1.1 Entité

on désigne par **entité** tout objet identifiable et pertinent pour l'application. Par exemple, pour la base de donnée précédente, les entités sont :

- *Films* : les films
- *Réalisateurs* : les réalisateurs

Pour chacune des entités, les individus ou objets ont en commun les même propriétés. Ce sont des **tables**, aussi appelées **relations**.

2.1.2 Attributs

Les entités sont caractérisées par des propriétés appelées attributs. Un attribut est désigné par un **nom** et prend ses valeurs dans un **domaine** énumérable. (notion à rapprocher du *type* en Python).

On exprime ici chaque attribut par son couple *nom : domaine* (*domaine = type*). Ces attributs doivent être *insécables*, ce qui empêche d'en choisir un du genre *adresse* (qui se decomposerait alors en n° + rue + ville).

Pour l'entité *Films*, ces attributs sont *Titre* et *Annee*.

2.1.3 Domaines

Les valeurs mises dans une cellule sont *élémentaires*. Il ne peut s'agir de types construits comme des listes.

- INT ou INTEGER : un entier
- FLOAT(x) : un nombre décimal avec x définissant la précision (nombre de bits de codage de la mantisse)
- REAL est un synonyme standard de FLOAT(24)
- CHAR(n) : chaine d'au plus n caractères

- VARCHAR(n)
- DATE une date

2.1.4 Association

Une association représente un **lien** entre les entités.

Il y a un lien représenté par le **verbe** *a réalisé* entre les entités *Réalisateurs* et *Films*.

On distingue :

- **L'association binaire fonctionnelle** : chaque occurence de l'entité A est liée à au plus une occurence de l'entité B. C'est équivalent à dire **A=>B** (la connaissance de A détermine celle de B). Dans ce cas, c'est la table A qui possède un attribut FOREIGN KEY qui pointe vers un attribut clé primaire de la table B.

{{< img src="../images/a_implique_b.png" alt="association binaire fonctionnelle" caption="association binaire fonctionnelle" >}}

- **L'association non fonctionnelle** : Une occurence de A peut être liée à plus d'une occurence de B.

{{< img src="../images/b_implique_a.png" alt="association binaire non fonctionnelle" caption="association binaire NON fonctionnelle" >}}

2.1.5 Vocabulaire employé dans le domaine des bases de données

Voici l'ensemble des mots utilisés, avec leur correspondance

Terme du modèle	Terme de la représentation par table
Relation	Table
n-uplet	ligne
Nom d'attribut	Nom de colonne
Valeur d'attribut	Cellule
Domaine	Type

Partie 3

Anomalies dans une base de données

La redondance d'informations dans une table va entrainer des anomalies de modification, et de suppression. Cela va entrainer un coût d'espace et des preformances moindres de la base de données.

- **Anomalie d'insertion** : Rien n'empêche dans la gestion d'une table par un tableur de faire plusieurs entrées pour le même film. Il pourrait alors y avoir une répétition des données. Il peut aussi y avoir des valeurs non renseignées, ce qui présente aussi une anomalie d'insertion.

La base de données en une seule table présente plusieurs problèmes de redondance*, ce qui va entrainer d'autres* anomalies.

- **Anomalie de modification** : Une modification sur une ligne peut nécessiter des modifications sur d'autres lignes.

La tenue d'une table de ce type peut entrainer des problèmes lors de la modification : Par exemple, la modification de l'année de naissance d'Hitchcock pour Vertigo et pas pour Psychose entraîne des informations incohérentes. Ou alors il faudrait modifier la table à plusieurs endroits, ce qui n'est pas réaliste si celle-ci est très grande...

- **Anomalie de suppression** : Certaines informations dépendent de l'existence d'autres informations.

On ne peut pas supprimer un film sans supprimer son metteur en scène, ... et les informations sur celui-ci.

Partie 4

La structuration des données

Cette **structuration** des données va permettre d'eviter les problèmes d'anomalie évoqués plus haut. La structuration repose sur l'existance de clés primaires et étrangères (permettant la **garantie d'accès**), mais aussi du respect des règles d'**unicité**, de **traitement des valeurs nulles**, ou autres... Ce sont les règles de gestion d'une BDD.

4.1 Règles de gestion d'un SGBD

La définition d'une relation comme un *ensemble (au sens mathématique)* entraîne que :

- l'ordre des lignes n'a pas d'importance,
- on ne peut pas trouver deux fois la même ligne, la table ne contient pas deux n-uplets identiques. (contrainte d'unicité)
- chaque valeur doit respecter le domaine (contrainte de domaine)
- il n'y a pas de case vide (contrainte existentielle/ contrainte de valeur par defaut). Chaque entrée (n-uplet) du tableau doit renseigner TOUS les attributs.

Ces règles ont été énoncées comme des théorèmes mathématiques, et sont retranscrites ici comme des *contraintes* à appliquer lors de la création d'une BDD.

4.2 Clés (ou identifiant)

Une clé est un attribut d'une relation.

clé primaire : une clé conçue pour identifier de manière unique les éléments d'une table. Si un attribut est considéré comme clef primaire, on ne doit pas trouver dans toute la relation 2 fois la même valeur pour cet attribut.

Dans la situation fréquente où on a du mal à déterminer quelle est la clé d'un type d'entité ; on crée un identifiant abstrait appelé id (un numéro séquentiel) indépendant de tous les autres attributs.

Dans une table, l'un des attributs peut être la clé primaire d'une autre table. Il s'agit alors d'une **clé étrangère**. Cet attribut permet de faire reference à un enregistrement dans une autre table, et caractérise parfaitement cette autre entité.

Exemple : **idMES** est la clé étrangère dans la relation *Films* :

Titre	Annee	**idMES**
Hana-bi	1997	1
Big fish	2003	2
Edward aux mains d'argent	1990	2

La relation *Réalisateur :*

idMES	NomMES	PrenomMES	AnneeNaiss
1	Kitano	Takeshi	1947
2	Burton	Tim	1958
3	Tarantino	Quentin	1963

La base de données doit se conformer aux **contraintes d'intégrité référentielles** : c'est à dire utiliser une *clé étrangère* pour spécifier le lien entre les entités.

4.3 Contraintes d'intégrité liées aux clés

Une contrainte d'intégrité est une règle qui définit la cohérence d'une donnée ou d'un ensemble de données de la BD.

Les contraintes d'intégrité liées aux *clés primaires et étrangères* sont :

1. Contrainte **de relation** : chaque relation doit comporter une clé primaire.
2. Clé **etrangère** : traduit une **association**. Elle doit être la clé primaire de l'autre relation à laquelle elle se réfère. Cette autre relation doit contenir l'élément auquel on veut se référencer AVANT de faire référence.
3. Modification : on ne doit pas faire de modification de la clé primaire d'une occurence si celle-ci est liée.

En pratique, ces contraintes sont définies **au moment de la création** d'une table.

PRIMARY KEY : contrainte **de clé primaire**. Définit l'attribut comme la clé primaire

FOREIGN KEY : contrainte de **clef étrangère**. Assure l'intégrité de référence. Cette clé étrangère n'est pas primaire pour la relation étudiée mais elle est la clé primaire d'une autre relation.

En option, on peut ajouter des contraintes d'unicité ou de validité avec UNIQUE ou CHECK, ou autres :

NOT NULL : contrainte **d'obligation de valeur**. Les éléments de la colonne doivent forcément être renseignés

UNIQUE : la contrainte **d'unicité**, permet de s'assurer qu'une autre clef pourrait remplacer la clef primaire : interdit que deux tuples de la relation aient la même valeur pour l'attribut.

REFERENCES `nom table` ('nom colonne) : contrôle l'intégrité référentielle entre l'attribut de la table et ses colonnes spécifiées

CHECK (`condition`) : contrainte de **validation**. Il s'agit d'une *assertion*, qui contrôle la validité de la valeur de l'attribut spécifié dans la condition. Permet de restreindre les valeurs de la-les colonnes qui la-les composent.

```
CREATE TABLE
AUTEURS
(id INT, nom TEXT, prenom TEXT, ann_naissance INT, langue_ecriture TEXT
   , PRIMARY KEY (id));
```

{{< img src="../images/table_brows.png" caption="creation de la table AUTEURS sous sqldbrowser" >}}

D'après la règle énoncée sur la contrainte de clé étrangère, la table LIVRES doit être créée APRES celle AUTEURS (à cause de la contrainte REFERENCES) :

```
CREATE TABLE LIVRES
(id INT, titre TEXT, id_auteur INT, ann_publi INT, note INT, PRIMARY
   KEY (id), FOREIGN KEY (id_auteur) REFERENCES AUTEURS(id))
```

4.4 Schéma d'une relation

Une relation possède un nom, et se compose de colonnes désignées par un nom d'attribut avec des valeurs d'un certain domaine.

Le **schéma d'une relation** : peut être représenté par un diagramme (une table), donnant les noms des relations, les attributs, leur domaine, et la mention de la clé primaire. Mais on préfère donner ce schéma sous forme d'un ensemble de tuples : $S = ((A_1, domaine_1, (A_2, domaine_2)...(A_n, domaine_n))$ où les A_i sont les attributs.

Exemple : le schéma de la relation Films est :

`films((id_film,INT),(titre,TEXT),(date,INT),(id_rea,INT))`

Lorsqu'une base de données comporte **plusieurs tables**, l'ensemble des schémas de ces relations s'appelle le **schéma relationnel** de la base de données.

Donner le schéma relationnel : *Le schéma relationnel correspond à l'ensemble des relations présentes dans une base de données. On y décrit les tables, les attributs, leur domaine, les **clés primaires** et les clés **étrangères**.*

Avec la représentation symbolique, on peut utiliser des marqueurs pour représenter ces clés :

- La clé primaire : soulignée
- clé secondaire : précédées d'un #

Dans un diagramme SQL, on représente cette relation (clé primaire -> clé secondaire par une flèche)

Partie 5

Correction des exercices

5.1 Ex 4 :

Question 3 : la table *Sandwichs* comporte à priori un attribut clé primaire qui est *Nom_sandwich*. Deux sandwichs différents devraient avoir des noms différents. Elle ne comporte aucun attribut qui pourrait être clé étrangère : en effet, il n'y a aucune reference dans la table *Sandwichs*.

Question 4 : La clé primaire est *Numero_client*. Aucune clé étrangère. La table *Commandes* ne comporte pas de clé primaire. Seul l'attribut *Numero_commande* pourrait différencier naturellement deux enregistrements (un client peut passer plusieurs commandes, un sandwich peut être commandé plusieurs fois, etc.). Il semblerait que le couple (`Nom_sandwich`, `Numero_commande`) puisse être une clé primaire. Ceci à la condition qu'aucun sandwich ne soit commandé plusieurs fois lors de la même commande, ce que l'attribut *Quantité* laisse présager.

Les attributs `Nom_sandwich`, `Numero_commande` sont des clé etrangères : ils premettent de faire référence à un enregistrement d'une autre table à l'aide de la clé primaire de celle-ci.

Question 5 : Cette base de données n'est pas bien modélisée : il y a redondance de la date et du numero de client dans la relation *Commandes*.

Il y a une relation binaire fonctionnelle entre *Numero_commande* et *Date*, ainsi qu'entre `Numero_commande` et `Numero_client`.

La relation *Commandes* peut ainsi être décomposée en 2 relations avec comme schémas :

```
1 Commandes((Numero_client, int), (Numero_commande, int), (Date, date))
2 ContenuCommandes((Nom_sandwich, str), (Quantité, int), (Numero_commande
    , int))
```

Partie 6

TP python SQL

6.1 run a simple server in python

what is a web server ? It's a program that runs persistently, listens to incoming HTTP traffic on a specific port, and handles incoming HTTP requests by parsing them and delegating them to the appropriate handlers. In the typical MAMP setup, Apache is the web server and your PHP scripts are the delegates that produce content.

Now, according to the above definition, you don't need a dedicated web server. All you need is a program which runs persistently, listens on a port and is able to handle HTTP requests and produce responses. Python has many frameworks that allow it to do that by itself. You just start a Python program, it binds itself to a port and produces responses for incoming HTTP requests.

6.2 connexion python à une base de données sql (mamp ou autre)

il faut definir un socket... assez compliqué, à creuser dans l'idée de se créer son propre LMS

6.3 connexion python à sqLite

voir les fichiers dans applications/python/serveur

> demarrer le fichier server.py avec **`python3 -m server.py`** puis ouvrir les autres fichiers en localhost :8800 (paramétrable depuis le fichier server.py)