CI 4 : Bases de données

Chapitre 1 – Introduction aux bases de données

Savoir

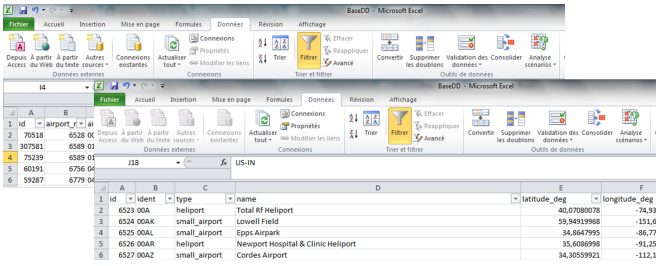
- BDD – C1 : utiliser une application offrant une interface graphique pour créer une base de données et l'alimenter ;
- BDD – C2 : utiliser une application offrant une interface graphique pour lancer des requêtes sur une base de données ;
- BDD – C3 : distinguer les rôles respectifs des machines client, serveur, et éventuellement serveur de données ;
- BDD – C4 : traduire dans le langage de l'algèbre relationnelle des requêtes écrites en langage courant ;
- BDD – C5 : concevoir une base constituée de plusieurs tables, et utiliser les jointures symétriques pour effectuer des requêtes croisées.

1 Présentation

1.1 Exemples de bases de données

Si on considère l'ensemble des données présentes sur un disque dur, semblent aisément gérées par un ordinateur. Le temps pour accéder et ouvrir un fichier semble en effet assez court. Mais qu'en est-il lorsqu'il s'agit de trouver un fichier sur le disque ? Qu'en est-il lorsqu'il s'agit de trouver une information dans chacun des fichiers du disque ?

Lorsqu'il s'agit de faire de nombreuses recherche sur un grand nombre de fichiers, un stockage « à plat » ne permet plus un temps d'accès satisfaisant. Il s'agit alors d'organiser les données sous une autre forme. On parle de base de données.



Les bases de données sont omniprésentes dans l'industrie en général et il s'agit de présenter comment ces données sont traitées.

On peut commencer par recenser des bases de données libres. Musicbrainz ou freedb sont par exemples des bases de données qui recensent des informations sur les disques de musiques (Auteurs, compositeurs, interprètes, titres des albums, titres des chansons, dates de sorties ...). IMDb est une base de donnée cinématographique. OpenStreetMap met à disposition des internautes des données cartographiques.

La FNAC, ou d'autres sites commerciaux disposent d'une base de données de leurs produits. Ainsi, grâce à un champ de recherche, l'internaute peut interroger la base. Il peut avoir des informations sur la disponibilité d'un produit, le délai de livraison ...

Une organisation des données permet aux utilisateurs d'avoir un accès rapide à tout type d'informations. On appelle requête la demande d'un utilisateur formulée auprès d'une base de donnée.

1.2 Bases de données à «plat»

Une première solution envisageable pour stocker des données est l'utilisation de bases de données dites plates. Les informations sont par exemple stockées dans un tableau. Prenons par exemple une base de données contenant la liste des aéroports du monde ainsi que diverses informations :

Nom	Ville	Pays	Continent	Type
Biarritz-Anglet-Bayonne Airport	Biarritz/Anglet/Bayonne	France	Europe	medium_airport
Milhaud Heliport	Toulon	France	Europe	heliport
Toulon-Hyères Airport	Toulon/Hyères/Le Palyvestre	France	Europe	medium_airport
Lake Hood Seaplane Base	Anchorage	États-Unis	Amérique du Nord	seaplane_base
Ted Stevens Anchorage International Airport	Anchorage	États-Unis	Amérique du Nord	large_airport
Mandalay International Airport	Mandalay	Myanmar	Asie	large_airport

Plusieurs remarques peuvent déjà être formulées :
- plusieurs informations sont stockées à plusieurs reprises : ainsi, l'information «France» est stockée à multiple reprise. Il en est de même pour le champ Continent ;
- des couples d'informations sont redondants : le couple (Pays, Continent) sera toujours identique pour un pays donné. Ainsi, stocker une seule fois que les États-Unis sont en Amérique du Nord.

Dans le cas d'une telle table, des requêtes simples sont aisées. Ainsi, faire la liste de tous les aéroports français ne pose pas de problème. Faire la liste de tous les héliports français est un peu plus difficile.

Si maintenant une seconde table comprend l'ensemble des fréquences sur lesquelles chacun des aéroports peut communiquer, les tables à plat vont rapidement trouver leur limite.

Enfin, lorsque les bases de données deviennent importantes (on recense plus de 40 000 installations aéroportuaires), l'accès aux données situées dans un tableau peut devenir considérablement lent.

1.3 Système de Gestion de Base de Données – SGBD

Pour stocker les données, on utilise des systèmes de gestion de base de données (SGBD). Le marché des SGBD est dominé par les entreprises Oracle, IBM ou Microsoft. Il existe par ailleurs des solutions libres telles que PostgreSQL ou MySQL.

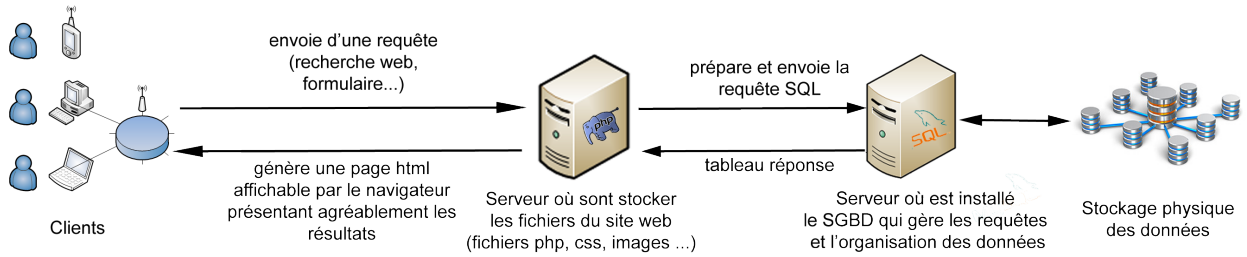
Une SGBD permettent d'assurer le stockage et l'organisation des informations ainsi que les gestions d'accès par des utilisateurs ayant des droits différents. La quantité de données peut dépasser plusieurs TéraOctets.

1.4 Structure client – serveur

Classiquement, les données ne sont pas stockées sur l'ordinateur de l'utilisateur (appelé client) utilisant la base de données mais sur un serveur, voire même un «nuage» (*cloud computing*).

Pour simplifier, pour faire du (*cloud computing*) les entreprises répartissent les informations sur plusieurs ordinateurs en réseau. Suivant les performances nécessaires ou suivant la quantité de stockage nécessaire, la base de donnée peur donc être répartie sur plusieurs ordinateurs physiques, la limite de répartition pouvant être modifié dynamiquement en fonction du besoin.

L'architecture classique d'un site we est l'architecture trois tiers :



2 Structure des base de données : le modèle relationnel

2.1 Tables

Dans une première approche, une base de données est constituée de tables. Une table est elle-même constituée de lignes rassemblant les informations (valeurs) que l'on désire stocker. On appellera entité chacune des lignes de cette table. Lorsque les valeurs d'une ligne ont les même propriétés, on les regroupe par colonnes.

Lors de la conception de la base de données, on définit, pour une table, chacune des colonnes. On peut alors renseigner chacune des lignes.

Exemple

Base de données des installations aéroportuaires

Aéroports				
Identifiant	Nom	Ville	iso_country	Type
4077	Biarritz-Anglet-Bayonne Airport	Biarritz / Anglet / Bayonne	FR	medium_airport
43537	Milhaud Heliport	Toulon	FR	heliport
4241	Toulon-Hyères Airport	Toulon/Hyères/Le Palyvestre	FR	medium_airport
21567	Lake Hood Seaplane Base	Anchorage	US	seaplane_base
5388	Ted Stevens Anchorage International Airport	Anchorage	US	large_airport
26727	Mandalay International Airport	Mandalay	MM	large_airport

Pays		
Identifiant	Code	Nom
302 687	FR	France
302 649	MM	Myanmar
302 755	US	United States

Remarque

Dans une table il n'y a pas de notion d'ordre a priori. Les données d'une ligne ne peuvent donc pas être désignées par un numéro de ligne.

2.2 Notion de clef

Afin de ne pas stocker des des doublons dans une base de donnée, on a recours au concept de clef primaire.

Dans certains cas, si on est certain que la valeur d'un attribut sera différent pour chaque ligne de la table, cet attribut peut tenir compte de clef primaire. Dans d'autres cas, si on est persuadé que pour une ligne donnée, la combinaison de n attributs est unique, la combinaison de ces attributs peut constituer une clef primaire.

Enfin, dans certains cas, on définit un attribut de type «identifiant». Lorsqu'on ajoute une ligne dans la table, on s'assure qu'un identifiant unique (un nombre entier par exemple) sera affecté.

2.3 Création de base de données en langage SQL

Le SQL (*Structured Query Language*) est un langage informatique permettant :
– de créer des bases de données ;
– d'interroger des bases de données.
Parmi les possibilités pourinterroger une base de donnée au format SQL on peut citer les interfaces graphiques qui vont permettre d'interroger une base de donnée avec le langage naturel. Il est aussi possible d'utiliser le langage SQL directement pour faire des opérations sur une base.

Exemple

Ainsi, il existe des API (*Application Programming Interface* – Interface de programmation) dans de nombreux langages de programmation, dont le Python, qui permettent de se connecter à une base de données et à l'interroger.

Remarque

De manière classique, les instructions SQL sont saisies en majuscules. Les requêtes se terminent par des points-virgules.

2.3.1 Création d'une base de données

SQL

Pour créer une base de données en langage SQL, la syntaxe est la suivante :

```
CREATE DATABASE Aeroports;
```

Il est à noter qu'il existe une gestion des utilisateurs et de leurs droits. La façon la plus aisée de les gérer est d'utiliser une interface graphique (par exemple phpmyadmin avec des bases de type mysql).

2.3.2 Création d'une table

Une fois la base de données crée, il est nécessaire de créer les tables. Chaque table dispose d'un ou de plusieurs attributs. Il existe plusieurs types d'attributs parmi lesquels :
– des entiers entiers. Il en existe plusieurs types selon les nombres à stocker (de `TINYINT` (codé sur un octet) et ayant une capacité de -128 à 127 à `BIGINT` (codé sur 8 octets)) ;

– la date et l'heure, permettant de stocker par exemple l'heure (`TIME` au format HH :MM :SS) ou l'année (`YEAR` au format AAAA).
– du texte : `CHAR` permet de stocker jusqu'à 255 caractères.
– *etc.*

SQL

Ainsi, pour créer une table, on a recours à la syntaxe suivante :

```
CREATE TABLE nom_table (nom_attribut_1 type_attribut1, nom_attribut_2 type_attribut_2, ...);
```

Exemple

Étant dans la base de données Aéroports, pour créer la table des pays avec comme attributs un identifiant (de type entier), un code (de type chaîne de caractères) un nom (de type chaîne de caractères), on a :

SQL

```
CREATE TABLE Pays (Identifiant INT, Code CHAR, Nom CHAR);
```

Lors de la création des tables il est de plus possible de définir des contraintes :
– `NOT NULL` : empêche d'avoir un 0 dans un champ ;
– `UNIQUE` : permet de n'avoir aucun doublon dans une colonne ;
– `DEFAULT` permet d'attribuer une valeur par défaut lorsque le champ n'est pas défini lors de l'insertion d'un élément ;
– `PRIMARY KEY` permet de définir un attribut comme clef primaire ;
– *etc.*

Exemple

Pour la même table que précédemment, il est possible de déclarer que l'identifiant est une clef unique primaire dont les occurrences sont uniques. On impose que le code et le nom des aéroports soient obligatoirement renseignés.

SQL

```
CREATE TABLE Pays (Identifiant INT UNIQUE PRIMARY KEY , Code CHAR NOT NULL , Nom CHAR NOT NULL);
```

Remarque : Le code d'aéroport étant un code unique, il devrait normalement être inutile d'avoir un identifiant. La clef primaire pourrait alors être le code.

2.4 Suppléments

Il est possible de supprimer une table, une base de données, de modifier la structure d'une table...

3 Algèbre relationnelle

La raison d'être d'une base de données étant de disposer de données, il est nécessaire de disposer d'outils d'interrogation. Pour cela, il existe un mathématique formel appelé algèbre relationnelle. Chaque opération formelle est transposable dans un langage de programmation.

Définition

Requêtes – Algèbre relationnelle

On entend par algèbre relationnelle, une collection d'opérations (requêtes) formelles qui agissent sur des relations et produisent une relation en résultat : $R_3 \leftarrow R_2 Op R_1$.

Ceci signifie que dans l'algèbre relationnelle, le résultat des requêtes effectuées sur les relations (tables) sera toujours une nouvelle relation.

3.1 Opérations relationnelles

3.1.1 Projection

Définition

Projection

Opération notée π au cours de laquelle on sélectionne certaines des colonnes (on élimine donc des attributs).

$$R_2 \leftarrow \pi_{\text{attribut 1, attribut 2, ...}}(R_1)$$

SQL

```sql
SELECT attribut_1,attribut_2 FROM nom_table;

SELECT * FROM nom_table;
```

Exemple

Lister les noms d'aéroports ainsi que les codes IATA correspondant :

$$\pi_{\text{name,iata\_code}}(\text{airports})$$

SQL

```sql
SELECT name,iata_code FROM airports
```

3.1.2 Sélection

Définition

Sélection

On appelle sélection de R_1 selon $A = a$, et on note $\sigma_{A=a}(R_1)$, la relation obtenue en sélectionnant dans R_1 uniquement les valeurs e telles que $e.A = a$.

$$R_2 \leftarrow \sigma_{\text{attribut=condition}}(R_1)$$

Le schéma relationnel R_2 est identique au schéma R_1. Les opérateurs permettant d'exprimer une condition sont $=$, $\neq$ (`!=` ou `<>`), $<$, $\leq$ (`<=`), $\geq$ (`>=`), $\neg$ (négation, `NOT`), $\vee$ (ou, `OR`), $\wedge$ (et, `AND`).

Définition

SQL

```sql
SELECT att_1,att_2 FROM nom_table WHERE att_3="xx" AND att_4=0 ...
```

Exemple

Sélectionner le nom de tous les aéroports situés au niveau de la mer :

$$\pi_{\text{name}}\left(\sigma_{\text{elevation\_ft}=0}(\text{airports})\right)$$

SQL

```sql
SELECT name FROM airports WHERE elevation_ft=0
```

Sélectionner le nom et le code iata de tous aéroports situés en-dessous du niveau de la mer en Europe :

$$\pi_{\text{name,iata\_code}}\left(\sigma_{\text{elevation\_ft}<0\wedge\text{continent=EU}}(\text{airports})\right)$$

SQL

```sql
SELECT name,iata_code FROM airports WHERE elevation_ft<0 AND continent="EU";
```

Cette requête laisse apparaître 2 fois la même ligne :

```
Mashland  Airfield
```

Cela s'explique par le fait que l'aéroport apparait 2 fois dans la table airports. Mais, pour la liste d'attributs sélectionnés, tous les champs sont les mêmes (l'identifiant est bien différent). En faisant apparaître l'id, on réaliserait que les deux champs sont différents. Si on souhaite éliminer les doublons dans, la requête est la suivante :

SQL

```sql
SELECT DISTINCT name,iata_code FROM airports WHERE elevation_ft<0 AND continent="EU";
```

3.1.3 Renommage

Définition

Renommage d'attribut

Cette opération est utilisée pour des raisons pratiques pour lever une ambigüité ou pour simplifier le nom d'un attribut de façon temporaire.

Soit $S=(A_1,...,A_n)$ un schéma, $i\in[1;n]$ et B un attribut tel que $dom(B)=dom(A_i)$. On note :

$$R_2(S)\leftarrow\rho_{\text{ancien attribut}\rightarrow\text{nouvel attribut}}(R_1(S))$$

Définition

SQL

```sql
SELECT att_1,att_2 as att_3 FROM nom_table
```

Exemple

On désire sélectionner le nom des aéroports et les altitudes et renommer temporairement l'attribut `elevation_ft` en `altitude` :

$$\pi_{\text{elevation\_ft}}\left(\rho_{\text{elevation\_ft}\to\text{altitude}}(\text{airports})\right)$$

SQL

```sql
SELECT name, elevation_ft AS altitude FROM airports;
```

3.2 Opérations sur les ensembles

Remarque

Les opérateurs ensemblistes sont dédiés à des relations de même schéma relationnel.

3.2.1 Opérations cartésiennes

Définition

Produit cartésien

Soient $R_1(S_1)$ et $R_2(S_2)$ deux relations de schémas disjoints. L'opération produit cartésien est notée $\times$.

$$R_3 \leftarrow R_1 \times R_2$$

La relation R_3 contient toutes les combinaisons d'association possibles entre les valeurs de R_1 et de R_2.

SQL

```sql
SELECT * FROM table_1,table_2
```

Définition

Division cartésienne

Soient $R_1(S_1)$ et $R_2(S_2)$ deux relations de schémas disjoints. L'opération division cartésienne est notée $\div$.

$$R_3 \leftarrow R_1 \div R_2$$

Dans ce cas, toutes les combinaisons de chaque tuple de R_3 et de chaque tuple de R_2 est contenue dans R_1.

Exemple

On désire sélectionner les continents qui ont tous types d'aéroports :

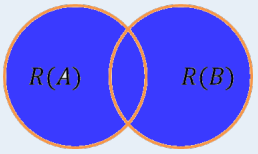
$$\pi_{\text{name}}\left(\text{airports} \div \pi_{\text{type}}(\text{airports})\right)$$

Il n'y a pas de transcription en langage SQL.

3.2.2 Union

Définition

Union
L'union de deux relations $R_1(S)$ et $R_2(S)$ est l'ensemble des valeurs comprises dans R_1 ou R_2.
On peut donc noter la relation $R_3(S)$ définie par : $R_3(S) \leftarrow R_1(S) \cup R_2(S)$



SQL

En langage SQL pour pouvoir faire l'union de deux relations, elles doivent avoir le même schéma (même nombre de colonne(s) et même type de colonnes). Il faut prêter attention à l'ordre des attributs dans la requête.

```
SELECT attribut_11, attribut_12 FROM table_1 UNION attribut_21, attribut_22 FROM table_2
```

Exemple

Ainsi, pour connaître les noms des régions et des pays associés à leur code, on utilise la requête suivante :

SQL

```
SELECT code,name FROM Countries UNION SELECT code,name FROM Regions;
```

La relation résultant contient donc l'ensemble des couples (code,name) des régions et des pays.

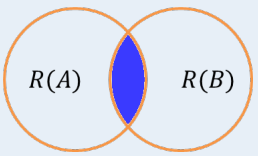
Remarque

Cette opération est à utiliser avec attention car la relation résultante peut avoir une forme absurde.

3.2.3 Intersection

Définition

Intersection
L'intersection de deux relations $R_1(S)$ et $R_2(S)$ est l'ensemble des valeurs comprises dans R_1 et dans R_2.
On peut donc noter la relation $R_3(S)$ définie par : $R_3(S) \leftarrow R_1(S) \cap R_2(S)$



SQL

```
SELECT (expression 1) INTERSECT SELECT (expression 2);
```

Exemple

Sélectionner le nom de l'ensemble des aéroports européens dont l'altitude est en dessous du niveau de la mer :

$$\pi_{\text{name}}\left(\sigma_{\text{continent}="EU"}(\text{airports}) \cap \sigma_{\text{elevation\_ft}\leq 0}(\text{airports})\right)$$

SQL

```
SELECT name FROM airports WHERE continent="EU"
    INTERSECT SELECT name FROM airports WHERE elevation_ft<=0;
```

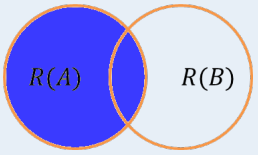
3.2.4 Différence

Définition

Différence

La différence de deux relations $R_1(S)$ et $R_2(S)$ est l'ensemble des valeurs comprises dans R_1 et qui ne sont pas comprises dans R_2.

On peut donc noter la relation $R_3(S)$ définie par : $R_3(S) \leftarrow R_1(S) - R_2(S)$



SQL

```
SELECT (expression 1) EXCEPT SELECT (expression 2);
```

3.2.5 Jointures

Définition

Jointure

La jointure est une combinaison de de tuples de deux relations en un seul tuple. On ne s'intéresse ici qu'à la jointure symétrique simple qui permet de recoller deux relations ayant un attribut en commun.

$$R_3 \leftarrow \sigma_{\text{R1.attribut\_1=R2.attribut\_2}}(R_1 \times R_2) \qquad R_3 \leftarrow \underset{\text{R1.attribut\_1=R2.attribut\_2}}{R_1 \bowtie R_2}$$

SQL

```
SELECT att_1,... FROM table_1 JOIN table_2 ON attribut.R1=attribut.R2
```

Exemple

Dans la table airrpots, les pays sont référencés par un attribut nommé iso_country. La table Countries fait le lien entre l'attribut code et l'attribut name.

L'utilisateur ne connaissant pas le code pays, il souhaite en une seule requête connaître la liste de tous les aéroports français.

$$\pi_{\text{name}}\left(\pi_{\text{name,code}}(\text{Countries}) \underset{\text{airports.iso\_country=Countries.code}}{\bowtie} \pi_{\text{name,iso\_country}}(\text{airports})\right)$$

Exemple

SQL

```sql
SELECT name FROM
    (SELECT name, iso_country FROM airports) AS aeroports
    INNER JOIN
    (SELECT Countries.code from Countries WHERE name="France") AS pays
        ON aeroports.iso_country=pays.code
```

3.3 Fonctions d'agrégations

Définition

Ces fonctions qui ne sont pas couvertes par l'algèbre relationnelle classique, permettent d'effectuer des calculs statistiques basiques sur les valeurs : `MIN`, `MAX`, `SUM` (somme), `AVG` (moyenne), `COUNT` (comptage du nombre de lignes).

Exemple

Combien existe-t-il d'aéroports en France ?

SQL

```sql
SELECT COUNT(airports.name)
        FROM airports
    INNER JOIN
        (SELECT Countries.code from Countries WHERE name="France") as pays
    ON airports.iso_country=pays.code
```

Quel est l'aéroport le plus élevé ?

SQL

```sql
SELECT MAX(elevation_ft) FROM
    (SELECT elevation_ft, name, iso_country FROM airports WHERE elevation_ft!='''') AS aeroports
    INNER JOIN
    (SELECT Countries.code from Countries WHERE name="France") AS pays
        ON aeroports.iso_country=pays.code
```

Quel est l'aéroport le plus élevé en mètres ?

SQL

```sql
SELECT MAX(elevation_ft)*0.30478513 FROM
    (SELECT elevation_ft, name, iso_country FROM airports WHERE elevation_ft!='''') AS aeroports
    INNER JOIN
    (SELECT Countries.code from Countries WHERE name="France") as pays
        ON aeroports.iso_country=pays.code
```

Remarque

Formalisme général de l'instruction SQL « SELECT » :

SQL

```
SELECT <liste d expressions>
    FROM <liste de tables>
    WHERE <conditions>
    GROUP BY <liste d attributs>
    HAVING <conditions>
    ORDER BY <liste d attributs>
```

Références

[1] Serge Abiteboul, Benjamin Nguyen, Yannick Le Bras, *Introduction aux Bases de Données Relationnelles Programme de Classes Préparatoires Scientifiques, Première année.*

[2] Christope Revy, *Concepts des bases de données*, Cours de STS IRIS, Lycée Janot de SENS.

[3] Patrick Beynet, *Supports de cours de TSI 2*, Supports de cours de TSI 2, Lycée Rouvière, Toulon.

[4] http://sqlpro.developpez.com/cours/divrelationnelle/