Bases de données

Introduction

Les organisations (entreprises, hôpitaux, universités…) manipulent diverses ressources (humaines, financières, logistiques, matérielles...) et font ainsi transiter des flux d'informations. La gestion d'une organisation implique un accès à ces données pour les consulter, traiter, modifier...

Base de données (BD) = Ensemble de données utilisées par une organisation.

Système de gestion de bases de données (SGBD) = Ensemble de programmes dédiés à la manipulation des données; c'est à dire ajouter des données, accéder à des données, mettre à jour des données, supprimer des données.
En Anglais, on utilise le terme DBMS pour database management system.

Les données peuvent être informatisées et stockées sur un disque dur.

Elles sont organisées de manière physique sur le disque par le SGBD. Des utilisateurs, bénéficiant de droits potentiellement différents, y accèdent grâce au SGBD. Il existe différents types de SGBD, hébergés sur des serveurs ou en local (sur sa machine).

Exemple concrets d'utilisation de base de données : les systèmes de réservations de billets sur Internet, les sites de courses sur Internet, le logiciel du secrétariat médical de votre dentiste...

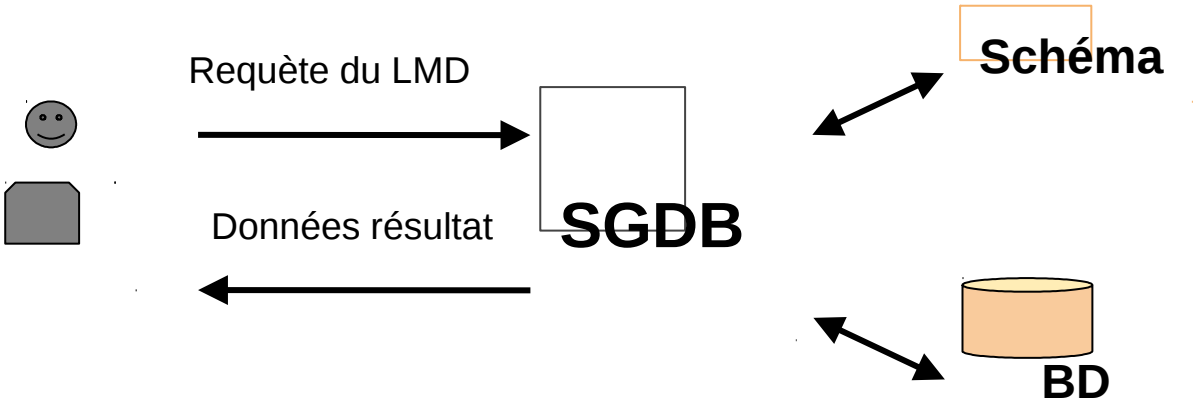
Les avantages d'une base de données : compacité, rapidité, exactitude et maintenance facile des données.

Naissance d'une base de données :

Conception : recueillir les informations sur l'organisation du système étudié → Création du schéma de la base de données : définition d'entités reliées entre elles par des associations → Création du schéma physique : implémentation du schéma de la base de données sur le disque au travers du SGBD.

Pour décrire le contenu d'une base de données, on utilise un modèle de données qui définit la manière dont on structure les données et les règles d'utilisation de ces données.
Exemples : modèle hiérarchique, réseaux, relationnel, objets…

Un SGBD est caractérisé par le modèle de données qu'il utilise. On exprime la création du schéma d'une base de données, c’est-à-dire la description de la structure des données, à l'aide d'un langage de définition des données (LDD). Les données sont manipulées à l'aide d'un langage de manipulation de données (LMD).



Modèle relationnel

Présentation du modèle

Modèle inventé par E. Codd en 1970.

Exemple de liste de tables dans une base de données :

```
+-------------------------+
| Tables_in_ChocolaterieDB
+-------------------------+
| Chocolat
| Chocolatier
| Client
| Commande
+-------------------------+
```

Exemple de table dans le modèle relationnel :

```
+---------------+----------------------+-------------------+
| idChocolatier | nom                  | ville
+---------------+----------------------+-------------------+
|             1 | Jeff De Bruges       | Bruges
|             2 | De Neuville          | Neuville St-Rémy
|             3 | Larnicol             | Melgven
|             4 | Jean-Paul Hevin      | Paris
|             5 | Patrick Roger        | Paris
|             6 | Jacques Genin        | Paris
|             7 | La Maison du chocolat | Bruges
|             8 | Michel Cluizel       | Damville
+---------------+----------------------+-------------------+
```

Description de cette table dans le modèle relationnel :

```
+---------------+-------------+------+-----+---------+----------------+
| Field         | Type        | Null | Key | Default | Extra
+---------------+-------------+------+-----+---------+----------------+
| idChocolatier | int(11)     | NO   | PRI | NULL    | auto_increment
| nom           | varchar(25) | NO   |     | NULL    |
| ville         | varchar(25) | NO   |     | NULL    |
+---------------+-------------+------+-----+---------+----------------+
```

UNE RELATION = UN TABLEAU A DEUX DIMENSIONS

UNE COLONNE = UN ATTRIBUT

EN-TÊTE DU TABLEAU = SCHEMA DE LA RELATION (DESCRIPTION DU TYPE)

UNE LIGNE = UN TUPLE

ENSEMBLE DES LIGNES = CONTENU DE LA RELATION (EXTENSION)

Définitions formelles :

Domaine : Ensemble de valeurs caractérisé par un nom

Ex: Entiers, réels, date, marque voiture, couleur, ...

Marque Voiture : {Renault, Peugeot, Citroen}

Couleur : { blanche, grise}

Produit cartésien d'un ensemble de domaines D1,D2,… Dn

noté D1 x D2 x … x Dn = ensemble des éléments $(v_1, v_2 \ldots, v_n)$ avec $v_1 \in D_1$, $v_2 \in D_2$, … , $v_n \in D_n$

Renault	*blanche*
Renault	*grise*
Peugeot	*blanche*
Peugeot	*grise*
Citroen	*blanche*
Citroen	*grise*

Relation

Soient D1, D2,..., Dn une liste de domaines, une relation est un sous-ensemble du produit cartésien D1 x *D2* x ... x *Dn.*

Tuple *(N-Uplet): un élément d 'une relation.*
$(v_1, v_2 \ldots, v_n)$

Exemple : (1435, Helfer, Paris)

Un tuple est unique dans une relation
L'ordre des tuples dans une relation n'a pas d'importance

Cardinalité d 'une relation = nombre de tuples

Attribut *: nom donné au rôle joué par un domaine dans une relation*

Exemple : Ville

Un attribut est unique par relation

L'ordre des attributs n'a pas d'importance.

Degré d'une relation = nombre d'attributs

Schéma d'une relation = nom de la relation suivi par la liste des attributs qui la composent et par la définition de leurs domaines R(A1 : D1, …, An : Dn) où R est le nom de la relation, Ai les attributs et Di les domaines associés.

Exemple, le schéma de la relation CHOCOLATIER est le suivant :

CHOCOLATIER(idChocolatier : Entiers, Nom : Caractères, Ville : Caractères)

Remarque : Les domaines peuvent être omis.

Remarque : le schéma définit la relation en intention alors que la table définit la relation en extension.

Règles d'intégrité structurelle

Clé

La clé d'une relation est un ensemble minimal d'attributs dont chaque valeur détermine un tuple unique dans toute extension de la relation.

Exemple :

l'attribut NoEmp peut être la clé de la relation EMPLOYE puisque tout employé possède un numéro unique et que deux employés ne peuvent pas avoir le même numéro

EMPLOYE (NoEmpl, Nom, Année, Nodept)

Contrainte référentielle

Une référence (ou clé étrangère) est un attribut (ou un groupe d 'attributs) dont les valeurs sont celles d'une clé d'une autre relation

Exemple

Employé (NoEmpl, Nom, Année, NoDept)

Département (NoDept, NomDépt, Budget)

Toute valeur de l'attribut NoDept dans la relation Employé doit se trouver dans la relation département.

Valeur nulle et clé

La valeur nulle (NULL) est une valeur introduite par convention dans une relation pour représenter une information inconnue ou inapplicable.

Ex: Employé (NoEmpl, Nom, NomMarital, Année, Adresse, Téléphone, Nodept)

Le téléphone non connu à une certaine date
(10, Durand, Faure, 1980, Paris, NULL, 75)

Le nom marital pour un employé
(23, Fergio, NULL, 1987, Marseille, 0234565412, 13)

Contrainte de relation : Toute relation doit posséder au moins une clé non nulle.

Schéma d'une base de données : schéma des relations et contraintes d'intégrité structurelle.

Le langage SQL (Structured Query Language)

Langage assertionnel permettant :
interrogation des données (requêtes)
manipulation des données (insertion, suppression, mise à jour)
définition des données et des contraintes sur celles-ci
définition des vues et d'index

Différentes normes ISO : SQL-86; SQL-89; SQL-92; SQL-99

Différents "dialectes" (Oracle, MySQL, …)

Création :

Le Langage de Définition de Donnés (LDD) permet la définition de la base de données.

Syntaxe pour la création de relations :

CREATE TABLE <nom_relation>

(<élément de relation> < , élément de relation>*

< , contrainte de relation>*
); >*]

<type de données> ::= VARCHAR <longueur> | INT | REAL | DATE …

Chaque relation est définie par un *nom de relation* et une liste d 'attributs

Chaque attribut est défini par un *nom d 'attribut* et un *type de données*

Une contrainte d 'attribut **concerne un seul attribut**

ex : valeur NULL impossible : NOT NULL
Attribut clé : PRIMARY KEY
Unicité de l'attribut : UNIQUE
Contrainte référentielle : REFERENCES <relation référencée>
[(<attribut référencé>)]
Valeur par défaut : DEFAULT <valeur>

Une contrainte de relation **concerne plusieurs attributs**

ex : Clé composée : PRIMARY KEY (nom_attribut [, nom_attribut]*)
Contrainte référentielle :
FOREIGN KEY (nom_attribut [, nom_attribut]*)
REFERENCES nom de relation [(nom_attribut [, nom_attribut]*)]

Exemples:

```
CREATE TABLE Fournisseur (
                    NumF INT NOT NULL,
...
                PRIMARY KEY (NumF) );

CREATE TABLE Piece (
                '               NumP INT NOT NULL,
                                NomP VARCHAR(30);
                ...
                PRIMARY KEY (NumP) );


CREATE TABLE Livraison  (
                                NumF INT NOT NULL,
                                NumP INT NOT NULL,
                                DateLiv DATE,
                                Qte INT,
                                PRIMARY KEY (NumF, NumP, DateLiv),
                FOREIGN KEY (NumF) REFERENCES Fournisseur(NumF),
                FOREIGN KEY (NumP) REFERENCES Piece(NumP) );
```

Mises à jour :

- Insertion :

INSERTION TUPLE à TUPLE

INSERT INTO <nom_relation> [(nom_attribut [, nom_attribut]*)]

VALUES (valeur [, valeur]*) ;

Exemple :

INSERT INTO Produit (NumP, NomP) VALUES (3, ‘Chaise’);

Les valeurs doivent être fournies dans l’ordre de déclaration des attributs de la liste ou, s’il n’y en n’a pas, du CREATE de la relation.

Si la liste d’attributs est incomplète, les attributs non spécifiés sont insérés avec des valeurs NULL

INSERTION D’UN ENSEMBLE DE TUPLES ‘CALCULES’ A PARTIR DE TUPLES DE LA BASE DE DONNEES

La clause VALUES est remplacée par un SELECT

Exemple :

```
INSERT INTO Fournisseurs-parisiens
        (SELECT *
                FROM Fournisseur
                WHERE VilleF = ‘Paris’);
```

- Suppression :

DROP TABLE nom_de_la_table

Ordre des suppressions : Il suffit de relire à l'envers le script de création de vos tables pour en déduire l'ordre de suppression à écrire dans le script de destruction de votre schéma.

Remarque pour la suite : les éléments qui utilisaient la table (vues, synonymes, fonctions et procédures) ne sont pas supprimés mais sont temporairement inopérants. Attention, une suppression ne peut pas être par la suite annulée.

DELETE FROM <nom_relation> [WHERE <condition de recherche>];

La clause WHERE permet de sélectionner les tuples à supprimer.

- Actualisation :

UPDATE nom_relation

SET nom_attribut = <expression de valeur>

[, nom_attribut = <expression de valeur>]*

[WHERE <condition de recherche>];

La clause WHERE permet de sélectionner les tuples à actualiser.

Ex : UPDATE Chocolatier

SET Ville = 'Paname'

WHERE Ville = 'Paris';

<expression de valeur> peut être : NULL, une constante ou une expression arithmétique contenant des attributs de la table à modifier.

En TP à ce semestre, nous utiliserons le SGBD libre MySQL (actuellement utilisé par Google, Facebook, Netflix…).

Prochains cours : au tableau !
Conseils :

- prendre des notes organisées (numérotez vos pages, noter la syntaxe de manière différente du reste du cours pour la retrouver facilement en TP par la suite),
- noter les explications orales et pas seulement le contenu du tableau,
- bien prendre les définitions mais aussi les exemples et les petits exercices proposés au long du cours.