Bases de Données Relationnelles

Partie 1 : Introduction &

SQL- Création des tables et insertion des données

Master 2 CGSI
CY Cergy Paris Université
Tianxiao.Liu@cyu.fr

Plan : séance 1

- Introduction au cours
  - Objectif, programme, organisation, évaluation…
- Système de Gestion de Base de Données (SGBD)
  - Motivation, fonctionnalités, architecture…
- Modèle relationnel
  - Table, colonne, clés…
- Langage SQL
  - Création et suppression des tables
  - Insertion des données

Objectif du cours

- Comprendre le fonctionnement d'un SGBD relationnel
- Maîtriser les notions de base du modèle relationnel
- **Maîtriser les opérations sur des données relationnelles par le langage SQL : mécanismes**
- Etre capable de concevoir / réaliser une mini base de données relationnelle répondant à un besoin

Programme du cours 1/2

- SGBD et modèle relationnel
  - Fonctionnalités et architecture
  - Table, colonne, clés primaires et étrangères
- SQL : manipulation des tables relationnelles
  - Création des tables et suppression des tables
- **SQL : manipulation des données**
  - Insertion des données
  - **Recherche : Sélection avec projection, prédicats et jointures (de différents types)**
  - **Tri, agrégats, regroupement et sous-requêtes**
  - Mise à jour des données
  - Suppression des données

Programme du cours 2/2

- Conception d'une base de données relationnelle
  - Analyse du problème
  - Récapitulation des informations pertinentes
  - Elaboration du schéma conceptuel
  - Conception et affinement du schéma logique
  - Population des données de tests
  - Création des requêtes de tests
  - **Création des requêtes paramétrées**

Organisation du cours

- CM-TD-TP
  - Séances 1-4 : Notions de base + SQL
  - Séances 5 : Conception de base de données + Projet

- Support de cours
  - En ligne : http://depinfo.u-cergy.fr/~tliu/bd.php
  - Slides de CM, sujets de TD
  - Corrections indicatives de TD
  - Sujet et consignes de projet

Evaluation 1/2

- Examen écrit (50 %)
  - Date : 20 octobre 2020 matin
  - Durée : 2 heures
  - **Documents de cours autorisés** sauf livres et appareils électroniques et ceux des autres

- Types d'exercices
  - Questions de compréhension du cours (3 pt)
  - Analyse des requêtes SQL (6 pt)
  - Ecriture des requêtes SQL (11 pt)

Evaluation 2/2

- Projet à réaliser (50%)
  - Conception et réalisation d'une mini base de données relationnelle
  - Travail en équipe de 3-4 étudiants

- Remise du projet
  - Date : avant l'examen écrit
  - Scripts SQL + Petit rapport (max. 6 pages)

Systèmes avant SGBD

- Constat des limitations des systèmes de fichiers
  - Manque de visibilité globale des données
  - Interrogation très difficile voire impossible
  - Pas de support de la gestion de confidentialité et de concurrence d'accès

- Premiers logiciels SGBD : Années 60
  - Hiérarchique : données organisées sous forme d'arborescence (DL1, IBM)
  - Données organisées sous forme de graphes (TOTAL, IBM)
  - Avantages : Vision globale des données, performance, persistance
  - Inconvénients : Interrogation des données difficile encore, confidentialité et concurrence d'accès insuffisantes

SGBD relationnel

- Evolution : **modèle relationnel**
  - Edgard Frank Codd, 1970
  - Basé sur la notion mathématique de relation
  - Possibilité d'une étude théorique

- Principaux SGBD relationnels du marché
  - ORACLE, MySQL, MS Access, DB2, SQL Server, PostgreSQL…

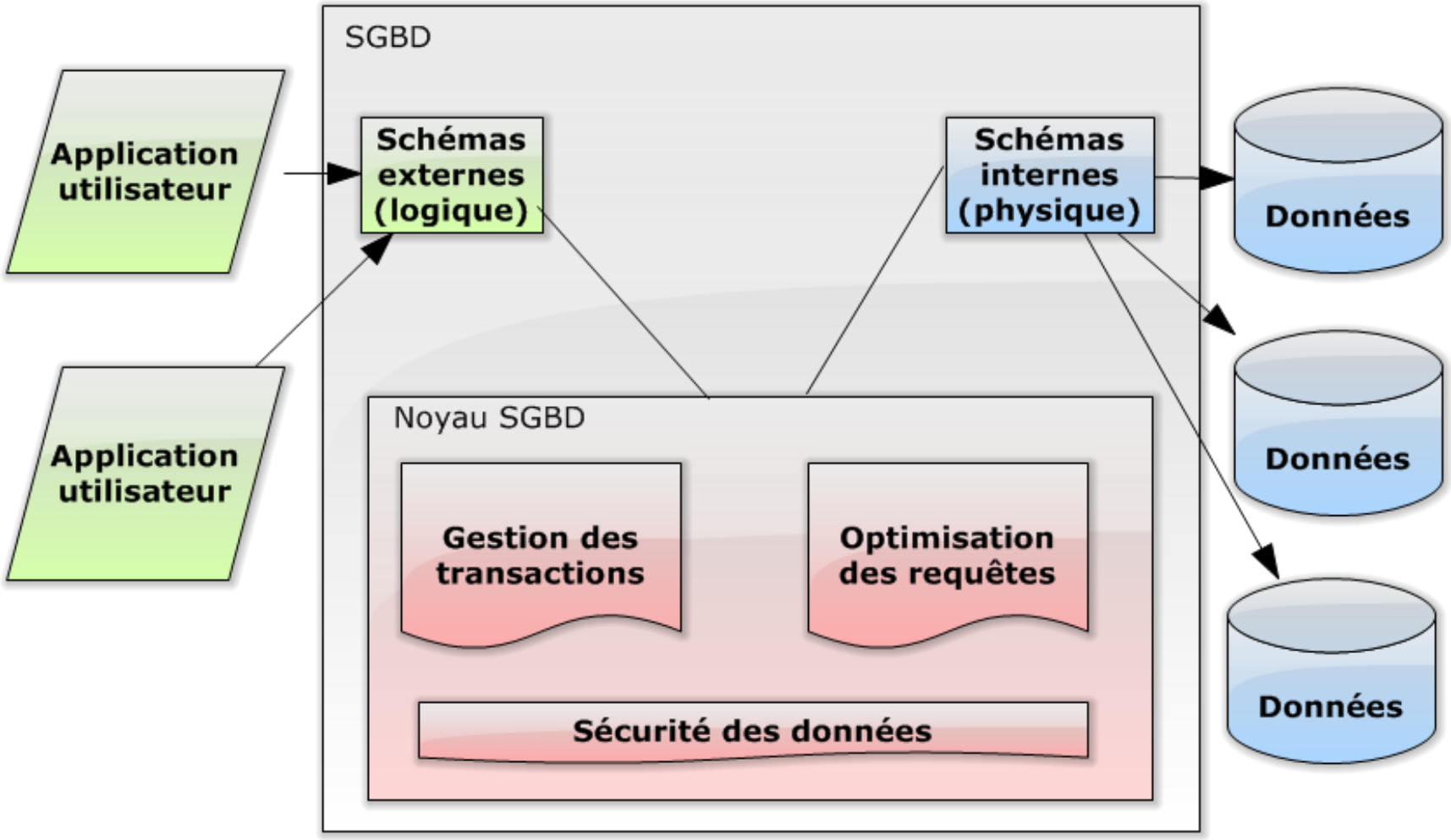
Fonctionnalités d'un SGBD

- Fonctionnalités principales
  - Manipulations des tables relationnelles
    - Schéma, création, mise à jour et suppression des tables
  - Opération ***CRUD*** des données
    - ***C****reate* : Enregistrement des données dans une table définie dans un schéma global logique
    - ***R****ead* : Recherche des données avec des critères
    - ***U****pdate* : Mise à jour des données existantes
    - ***D****elete* : Suppression des données

Fonctionnalités d'un SGBD

- Fonctionnalités "autour des données"
  - Gestion des transactions
    - Permet de gérer les accès concurrents aux mêmes données par plusieurs utilisateurs simultanément
  - Optimisation de requêtes
    - Permet de réduire le coût d'exécution des requêtes (surtout le temps de réponse)
  - Sécurité des données
    - Authentification et autorisation

Architecture d'un SGBD



Motivation du modèle relationnel

- Principe
  - Représenter les données du **même type** d'informations avec une **table** relationnelle
  - Prédéfinir les contraintes sur les données dans la table : type, unique, non nul…
  - Etablir des liens entre les tables pour avoir un **schéma global logique**
  - Diverses opérations spécifiques pour manipuler les données

Notions de base 1/2

- Table
  - Elle contient des données du même type de données
  - Composantes d'une table
    - Nom de la table (ex. adresse)
    - Une liste de colonnes
    - Des lignes (données)

id_adresse	numero	rue	ville	code_postal
1	2	rue Pasteur	Paris	75014
2	11	avenue des Champs Elysées	Paris	75008
3	6	boulevard du Port	Cergy	95000
4	3	avenue du Général de Gaulle	Versailles	78000
5	7	rue du Mont Valérien	Saint-Cloud	92210

Notions de base 2/2

- Colonne
  - Toutes les données sur la même colonne (de différentes lignes) sont du même type
- Clé primaire
  - Permet d'identifier une ligne de données
  - Valeur de la clé → unique pour chaque ligne
- Clé étrangère
  - Référence d'une valeur existante (souvent clé primaire) dans une autre table

Introduction au langage SQL

- SQL
  - Acronyme de *Structured Query Language*
- Un langage normalisé
  - Permet d'effectuer les opérations sur les données d'un SGBD relationnel
- On ne travaille que sur le schéma externe
- Attention
  - Il existe différents dialectes de SQL pour différents SGBD : ajouter des éléments hors de la norme

SQL : Création des tables 1/6

- Syntaxe

```
CREATE TABLE nom_table (
  nom_colonne1 TYPE CONTRAINTES… ,
  nom_colonne2 TYPE CONTRAINTES… ,
  … …
  nom_colonneN TYPE CONTRAINTES… ,
  d'autre contraintes …
);
```

- Attention
  - Pas de "," pour la dernière ligne entre parenthèses
  - N'oubliez pas le ";" à la fin → chaque requête SQL se termine par un ";"

SQL : Création des tables 2/6

- Règles d'écriture
  - Noms de tables : utiliser "_" pour séparer les mots
  - Ecrivez en majuscule les **mots réservés**
  - Pas d'accents sur les lettres utilisées dans les noms
  - Indentation pour la lisibilité
- Types de données
  - Nombre : `INT`
  - Numéro automatique incrémenté (utilisé pour définir les clés primaires) : `INT AUTO_INCREMENT`
  - Chaîne de caractères : `VARCHAR(taille)`
  - Nombre à virgule (point) flottante : `FLOAT`

SQL : Création des tables 3/6

- Types de données (suite)
  - date : `DATE` qui suit le format : YYYY-MM-DD
  - booléen : `BOOLEAN` représenté par `TINYINT(1)`
    - valeur `TRUE` pour vrai, valeur `FALSE` pour faux
  - clé étrangère (référence à une clé primaire de type numéro automatiquement incrémenté) : `INT`

- Attention
  - Ne pas utiliser les mots clés comme nom de colonne ou de table

SQL : Création des tables 4/6

- Contraintes
  - Présence obligatoire : `NOT NULL`
  - Unique (pas forcément la clé primaire) : `UNIQUE`
  - Clé primaire : `PRIMARY KEY (nom_colonne)`
  - Clé étrangère (écriture en une ligne):

```
FOREIGN KEY (nom_colonne) REFERENCES
  nom_table (nom_colonne)
```

SQL : Création des tables 5/6

- Un exemple complet

```
CREATE TABLE adresse(
  id_adresse INT AUTO_INCREMENT,
  numero VARCHAR(10) NOT NULL,
  rue VARCHAR(50) NOT NULL,
  ville VARCHAR(50) NOT NULL,
  code_postal VARCHAR(5) NOT NULL,
  PRIMARY KEY(id_adresse)
);
```

SQL : Création des tables 6/6

- Un autre exemple complet

```
CREATE TABLE client(
  id_client INT AUTO_INCREMENT,
  nom VARCHAR(50) NOT NULL,
  prenom VARCHAR(50) NOT NULL,
  date_naissance DATE NOT NULL,
  sexe BOOLEAN NOT NULL,
  id_adresse INT NOT NULL,
  PRIMARY KEY(id_client),
  FOREIGN KEY(id_adresse) REFERENCES
  adresse(id_adresse)
);
```

Insertion des données 1/3

- Syntaxe

```
INSERT INTO nom_table (colonne1, colonne2, …)
VALUES (valeur1, valeur2, …);
```

- Attention
  - Nombre de colonnes = Nombre de valeurs
  - Pas de colonnes spécifiées = toutes les colonnes
  - Cohérence entre types de colonnes et types de valeurs
  - On doit **ignorer** les colonnes de type `AUTO_INCREMENT` → clé premier → générée par le système

Insertion des données 2/3

- Règles d'écriture pour les valeurs
  - Valeur correspondant à une colonne `VARCHAR`
    - Entourée par les simples guillemets
    - Ex. `'Paul'` pour le prénom du client
    - Ex. `'75015'` pour le code postal de l'adresse
  - Valeur correspondant à une colonne de type `INT`
    - Ne mettez rien d'autre que la valeur elle-même
  - Valeur correspondant à une colonne de type `BOOLEAN`
    - `TRUE` ou `FALSE` Ex. `TRUE` pour indiquer masculin
  - Valeur correspondant à une colonne de type `DATE`
    - Entourée par les simples guillemets
    - Format de date : `'YYYY-MM-DD'`

Insertion des données *3/3*

- Exemples

```
INSERT INTO adresse(numero, rue, ville, code_postal)
VALUES('2', 'rue Pasteur', 'Paris', '75014');
```

  - La clé primaire id_adresse est générée. (La valeur commence par 1.)

```
INSERT INTO client(nom, prenom, date_naissance, sexe, id_adresse)
VALUES('Dupont', 'Paul', '1980-08-01', TRUE, 1);
```

  - Supposons que TRUE pour dire que le client est un homme
  - La clé primaire id_client est générée.
  - La clé étrangère id_adresse **référence** à la valeur 1 générée par la première requête.