Fondements des bases de données

Introduction

Marc Plantevit

marc.plantevit@liris.cnrs.fr

Objectif de l'enseignement :

Approfondir les connaissances du modèle relationnel et les fondements de la conception de bases de données :

- Contraintes et dépendances,
- Notions de systèmes d'inférences et de bases de règles,
- Processus de normalisation d'une base de données à partir des contraintes.
- Conception à l'aide d'un modèle conceptuel de données.

Pré-requis :

- Structure du modèle relationnel : attributs, relations, bases de données.
- Langages du relationnels : Algèbre et calcul relationnel, SQL
- Connaissance élémentaire du modèle E/A.
- Voir cours LIF04 d'E. Coquery (`http://liris.cnrs.fr/ecoquery/dokuwiki/doku.php?id=enseignement:lif4:start`) ainsi que celui de N. Lumineau (`http://www710.univ-lyon1.fr/~nluminea/index.php?ue=LIF4`).
- **Présupposés théoriques :** théorie des ensembles (graphes, arbres, treillis), théorie de la complexité et des automates (automates finis, machines de Turing, théorie de la calculabilité, théorie de la complexité), logique mathématique élémentaire.

Références

- S. Abiteboul, R. Hull et V. Vianu. *Fondements des bases de données.* Addison-Wesley, 1995.
- R. Ramakrishnam et J. Gehrke. *Database Management Systems*. 2003 (troisième édition).
- Matériel disponible en ligne allant avec la référence précédente : `http://pages.cs.wisc.edu/~dbbook/`.

Fonctionnement

- Introduction, rappels (2h)
- Contraintes et dépendances du modèle relationnel (6h)
- Conception de BD relationnelles (8h)
- Structures d'indexation (4h)
- Introduction au modèle XML (2h)

Modalités du contrôle continu intégral

- 3 contrôles partiels en séance de TD, chacun d'une durée de 45 minutes et valant 15% de la note finale.
- TP noté (25% de la note finale) : épreuve de 2h devant machine.
- Contrôle sur table à la fin du semestre : durée 1h30 pour 30% du total.

Plan

Principes

Base de données

Une grande quantité de données stockée dans un ordinateur.

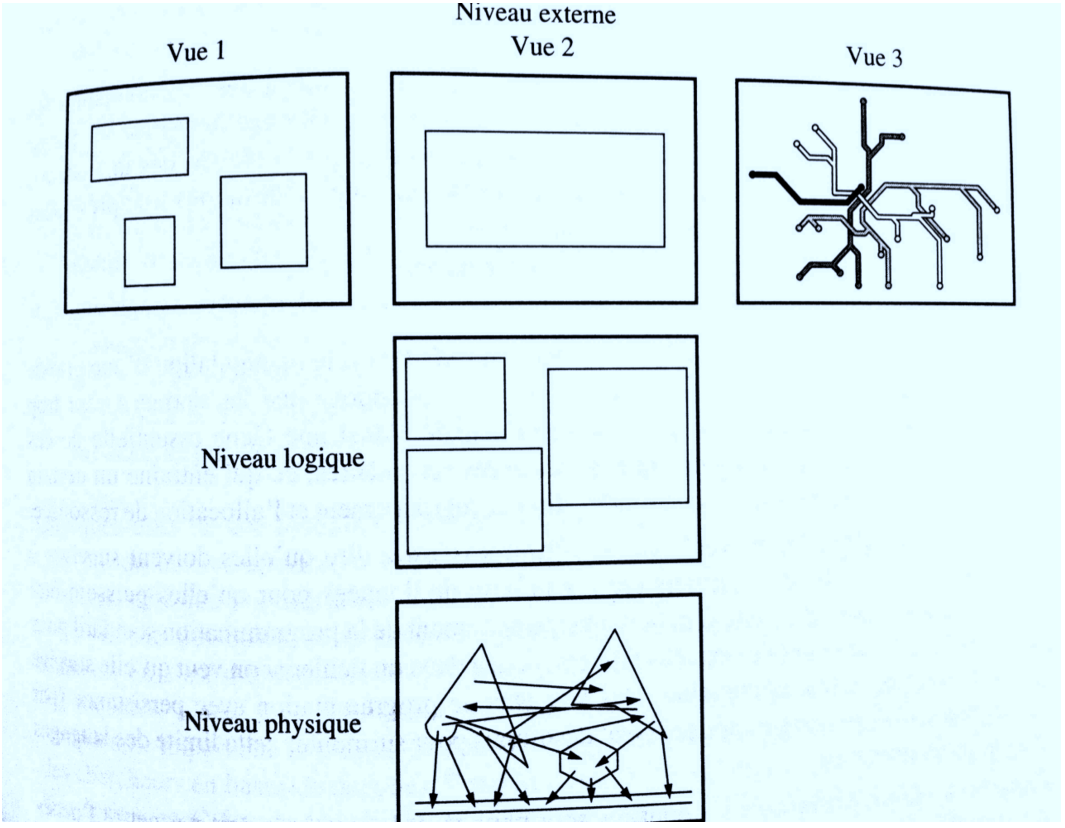
Qu'est ce qu'un SGBD ?

Un logiciel qui permet de stocker et de manipuler de grandes bases de données.

Architecture 3 niveaux

Détacher l'utilisateur de l'implémentation physique.

- physique,
- logique,
- externe.



Séparation logique / physique : principe d'indépendance des données

- Définition de *modèles de données logiques* : permettre à l'utilisateur de se concentrer uniquement sur une représentation logique des données sans se soucier de la représentation physique des données.
  - *langage de définition des données* (LDD), pour définir les aspects structurels des donneés.
  - *langage de manipulation des données* (LMD) pour y accéder et les mettre-à-jour.
- plusieurs aspects de l'implémentation physique peuvent être changés sans avoir à modifier la vision abstraite de la base de données.
- la distinction la plus importante qui existe entre les systèmes de fichiers et les systèmes de bases de données.

Séparation niveaux physique et externe

- permet des points de vue différent sur les bases de données qui sont adaptés à des besoins spécifiques.
- Les vues cachent ainsi les informations sans intérêt et restructure les données qu'elles retiennent.

Un compromis à trouver

Compromis à trouver entre le confort humain et une performance raisonnable.

- le système doit compiler les requêtes et les mises-à-jour destinées à la représentation logique dans les programmes « réels »

Fonctionnalités

- Manipulation de la mémoire secondaire ;
- Persistance ;
- Contrôle de concurrence ;
- Protection des données ;
- Interface homme-machine ;
- Distribution ;
- Compilation et optimisation.

Complexité et diversité

Applications : finances, personnel, inventaire, ventes, dossiers, biologie, etc.

Utilisateurs : programmeurs d'applications, sav, secrétaires, administrateurs de bases de données, geek, etc.

Modes d'accès : LMD linéaires et graphiques, interfaces graphiques, entrée des données, génération de rapports, etc.

Modèles logiques : les plus diffusées sont les modèles en réseau, hiérarchique, relationnel, orienté objet. Il existe des variations pour chacun de ces modèles, implémentées par divers distributeurs.

Modèles de données

Tout SGBD est conçu autour d'un *modèle de données* bien défini.
Comme dit précédemment, il s'agit de la combinaisons de 3 éléments :

- Une *structure*, qui correspond à l'organisation logique des données, la forme sous laquelle les utilisateurs vont les percevoir ou les représenter.
- Des *contraintes d'intégrité*, que l'on peut définir sur les données, afin d'en assurer l'intégrité et la cohérence avec le monde réel et les besoins des applications.
- Des langage de *manipulation des données*, pour les mises à jour et les interrogations.

- le modèle réseau
  - Structure : graphe orienté, les noeuds sont des enregistrements
  - Contraintes d'intégrité : un identifiant pour chaque noeud, un mécanisme de référence entre des noeuds
  - Manipulation : parcours de réseaux.
- le modèle hiérarchique
  - Structure : arborescente (forêts)
  - Contraintes d'intégrité : un identifiant pour chaque noeud, un mécanisme de référence entre des noeuds
  - Manipulation : navigation hiérarchique

- le modèle relationnel
  - Structure : ensembles de relations, qui sont des ensembles de tuples. Graphiquement, une relation est un tableau.
  - Contraintes d'intégrité : principalement les clés primaires (identifiants de tuples) et les clés étrangères (liens entre les relations)
  - Manipulation : algèbre relationnel, calcul relationnel, SQL.
- le modèle déductif
  - Structure : quasiment la même que le modèle relationnel
  - Contraintes d'intégrité : les mêmes que le modèle relationnel
  - Manipulation : langages logiques, le principal est *Datalog*. Contrairement aux langages du modèle relationnel, il admet la récursivité.

- le modèle Objet
  - Structure : logique objet, soit des classes, des objets, des attributs et des méthodes. Peut être vu comme un graphe orienté.
  - Contraintes d'intégrité : identifiant pour les objets, référence entre objets.
  - Manipulation : extensions de SQL comme OSQL ou OQL.
- le modèle Entité/Association
  - Structure : Entités (avec des attributs) et associations entre des entités.
  - Contraintes d'intégrité : identifiants, cardinalités sur les associations
  - Manipulation : aucun.

Plan

Notations

Les notations suivantes sont communément admises lorsque le contexte le permet.

- Un ensemble $\{A_1; A_2; ...; A_n\}$ sera noté par la concaténation de ses éléments : $A_1A_2...A_n$.
- Si X et Y sont deux ensemble, leur union $X \cup Y$ sera notée XY.
- Une séquence d'éléments, est notée $< A_1, ..., A_n >$. Lorsque le contexte est clair, les séquences et les ensembles seront notés de la même façon.

Exemple

- $\{A; B; C\}$ sera noté ABC
- Les notations ABC et BCA sont équivalentes puisque l'ordre n'a pas d'importance.
- L'ensemble $AABC$ n'existe pas, puisque l'élément A apparaît deux fois.
- Soit les ensemble $X = ABC$ et $Y = BD$, alors leur union $X \cup Y$ sera noté $XY = ABCD$.

Plan

Intuition

Étudiants	*NUM*	*NOM*	*PRENOM*	*AGE*	*FORMATION*
	28	Codd	Edgar	20	3
	32	Armstrong	William	20	4
	53	Fagin	Ronald	19	3
	107	Bunneman	Peter	18	3

Enseignants	*NUM*	*NOM*	*PRENOM*	*GRADE*
	5050	Tarjan	Robert	Pr.
	2123	De Marchi	Fabien	MCF
	3434	Papadimitriou	Spiros	Pr.
	1470	Bagan	Guillaume	CR

TABLE: Exemple de bases de données

- Soit $\mathcal{U}$, un ensemble infini dénombrable de *noms d'attributs* ou simplement *attributs*,
- $\mathcal{D}$ un ensemble infini dénombrable de *valeurs*.
- Soit $A \in \mathcal{U}$ un *attribut*, le *domaine* de A est un sous-ensemble de $\mathcal{D}$, noté $DOM(A)$.

Hypothèse d'unicité des noms :

Soient deux valeurs v_1 et v_2 des éléments de $\mathcal{D}$. On dit que v_1 et v_2 sont égales si et seulement si elles sont syntaxiquement identiques, c'est à dire qu'elles ont le même nom. En d'autres termes, lorsque deux attributs ont la même valeur, ils désignent la même chose.

schémas de relations et de bases de données

Un *schéma de relation* R est un ensemble fini d'attributs (donc $R \subseteq \mathcal{U}$).
Un *schéma de base de données* $\mathbf{R}$ est un ensemble fini de schémas de relation.

schémas de relations et de bases de données

Un *schéma de relation* R est un ensemble fini d'attributs (donc $R \subseteq \mathcal{U}$).
Un *schéma de base de données* $\mathbf{R}$ est un ensemble fini de schémas de relation.

Exemple

Une application gérant les cours dispensés au sein de l'UFR, regroupant des informations sur les étudiants, les formations, les enseignants, les modules, les UE, les salles de cours et les ressources matérielles (projecteurs, etc...).
La modélisation des étudiants pourrait se faire à l'aide du schéma de relation

$$ETUDIANTS = \{NUM; NOM; PRENOM; AGE; FORMATION\}.$$

Si on représente les autres informations dans d'autres schémas de relations, alors l'union de tous les schémas de relation obtenus constituera le schéma de notre base de données, qu'on pourra appeler *FACULTE*.

Première forme normale

Un schéma de relation R est en première forme normale (notée $1FN$) si, pour tout attribut $A \in R$, $DOM(A)$ est composé de valeurs atomiques.

La première forme normale est une condition fondamentale du modèle relationnel, garante de sa simplicité et efficacité, et l'hypothèse de schéma de relation universelle est toujours vérifiés, permettant de désigner de façon unique un attribut, sans aucune ambiguïté.
Un schéma de relation, ou de bases de données, est donc une boîte vide, avec une structure bien particulière, destinée à contenir des ensembles d'éléments possédant la même structure et donc sémantiquement équivalents.

tuple, Relation et Base de Données

Soit $R = A_1...A_n$ un schéma de relation. Un *tuple* sur R est un membre du produit cartésien $DOM(A_1) \times \ldots \times DOM(A_n)$. Une relation r sur R est un ensemble fini de tuples. Une base de données **d** sur un schéma de base de données $\mathbf{R} = \{R_1, ..., R_n\}$ est un ensemble de relations $\{r_1, ..., r_n\}$ définies sur les schéma de relation de **R**.

De façon informelle, on peut donc voir un tuple comme une séquence de valeurs, prises par les attributs du schéma de relation. Si t est un tuple défini sur un schéma de relation R, et X un sous-ensemble de R, on peut restreindre t à X en utilisant l'opération de projection, notée $t[X]$.

Exemple

Prenons la relation *Étudiants* : Si on nomme t_1 le premier tuple, alors $t_1[NOM] = Codd$, $t_1[NUM, AGE] = (28, 20)$.
L'ensemble de toutes les relations "peuplées" par des tuples constitue la base de données.

Plan

Plusieurs langages ont été définis pour l'interrogation et la manipulation des données dans le modèle relationnel. *Le résultat d'une requête est toujours une relation* : les langages diffèrent sur la façon de définir la relation en sortie. Ils sont de deux sortes :

- *les langages procéduraux* : AR,
- *déclaratif* : CRT, CRD, DATALOG(admettant la récursivité).

Le SQL correspond à l'implantation du calcul relationnel de tuple : c'est donc un langage déclaratif, possédant de nombreuses extensions pour faciliter l'usage et augmenter le pouvoir d'expression.

Algèbre Relationnel

Opérations de « base »

- Sélection ($\sigma_C(R)$) : retourne le sous-ensemble de tuples de la relation R vérifiant la condition C.
- Projection ($\pi_X(R)$) "supprime" les colonnes non désirées de la relation R.
- Produit cartésien ($R_1 \times R_2$) permet de combiner 2 relations.
- Différence ($R_1 - R_2$) retourne les tuples de R_1 qui ne sont pas dans R_2.
- Renommage ($\rho_{[\frac{x}{x'}]}(R)$)

- jointure ($\bowtie$), division, intersection : non essentiels mais (très) utile.

Calcul Relationnel à Variable Domaine (CRVD)

- Syntaxe :

$$\{x_1, \ldots, x_n \mid F\}$$

où F est une formule dont les variables libres sont $x_1, \ldots, x_n$ et composée à partir de :

  - De prédicats des relations de la base appliquées à des variables : $R(A_1 : y_1, \ldots, A_k : y_k)$
  - de comparaisons entre variables ou variables et constantes : $x \geq y$, $x = 123$, ...
  - De connecteurs logiques et quantificateurs : $\vee, \wedge, \Rightarrow, \forall, \exists$, ...

Calcul Relationnel à Variable Tuples (CRVT)

- Syntaxe :

$$\{x_1.A_1, \ldots, x_n.A_n \mid F\}$$

où F est une formule dont les variables libres sont $x_1, \ldots, x_n$ avec la possibilité d'avoir plusieurs fois la même variable x_i, composée de la manière suivante :

  - $R(x)$ où x est une variable représentant un tuple ;
  - de comparaisons : $x.A \bullet x'.A'$ ou $x.A \bullet c$ avec $\bullet \in \{<, >, \leq, \geq, =, \neq\}$ et c constante.
  - De connecteurs logiques et quantificateurs entre formules : $\vee, \wedge, \Rightarrow, \forall, \exists, \ldots$

Plan

Rappel SQL

à partir de Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke.

1. Les slides (en anglais) sont disponibles.
2. exemple jouet : Une base de données de marins qui réservent des bateaux.
3. Reservation : R[sid,bid,day]
4. Marins : S[sid,sname,rating,age]
5. Bateau ..

Requêtes SQL Basiques

```
SELECT [DISTINCT] target-list
FROM relation-list
WHERE qualification
```

- relation-list : Une liste de noms de relation (possibilité de mettre des variables).
- target-list : Une liste d'attributs de relations appartenant à relation-list
- qualification : Comparaisons (*Attr op const* ou *Attr1 op Attr2* où $op \in \{<, >, =, \leq, \geq, \neq\}$) combinées en utilisant AND, OR et NOT. DISTINCT est un mot-clé optionnel indiquant que le résultat ne doit pas contenir de duplications. Par défaut, les duplications ne sont pas éliminées !

Expressions et String

```
SELECT S.age, age1=S.age-5, 2*S.age AS age2
FROM Sailors S
WHERE S.sname LIKE 'B_%B'
```

Utilisation d'expressions arithmétiques ou de string matching.

- Trouver des triplets (âge des marins, et 2 champs décrits par des expressions) pour les marins dont les noms commencent et finissent par B et contiennent au moins 3 caractères.
- AS et = sont deux façons de nommer les champs retournés.
- LIKE est utilisé pour les expressions régulières ('_' pour au moins un caractère, '%' pour 0 ou plus de caractère/).

Union

Trouver les marins qui ont réservé un bateau rouge **ou** jaune.

```
SELECT S.sid
FROM Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid AND (B.color='red' OR
B.color='green')
```

```
SELECT S.sid
FROM Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid AND B.color='red'
UNION
SELECT S.sid
FROM Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid AND B.color='green'
```

Intersection

```
SELECT S.sid
FROM Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid AND B.color='red'
INTERSECT
SELECT S.sid
FROM Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid AND B.color='green'
```

Requêtes Imbriquées

Trouver le nom des marins qui ont réservé le bateau *#103*.

```
SELECT S.sname
FROM Sailors S
WHERE S.sid
IN
(SELECT R.sid
FROM Reserves R
WHERE R.bid=103)
```

```
SELECT S.sname
FROM Sailors S
WHERE EXISTS (SELECT * FROM Reserves R WHERE R.bid=103 AND
S.sid=R.sid
```

Un peu plus sur la comparaison d'ensembles

- IN, EXISTS, NOT IN, NOT EXISTS.
- Aussi op ANY, op ALL, op IN $<, >, \leq, \geq, =, \neq$
- Les marins dons la note est supérieure à celles de marins nommés Horatio.

```
 SELECT *
FROM Sailors S
WHERE S.rating > ANY (SELECT S2.rating FROM Sailors S2
WHERE S2.sname='Horatio')
```

Opérateurs d'agrégation

Une extension significative de l'AR.

```
COUNT (*), COUNT ( [DISTINCT] A), SUM ( [DISTINCT] A), AVG
( [DISTINCT] A), MAX (A), MIN (A)
SELECT COUNT (*) FROM Sailors S
SELECT AVG (S.age) FROM Sailors S WHERE S.rating=10
SELECT COUNT (DISTINCT S.rating) FROM Sailors S WHERE
S.sname='Bob'
SELECT AVG ( DISTINCT S.age) FROM Sailors S WHERE
S.rating=10
SELECT S.sname FROM Sailors S WHERE S.rating= (SELECT
MAX(S2.rating) FROM Sailors S2)
```

GROUP BY et HAVING

- ```
  SELECT [DISTINCT] target-list
  FROM relation-list
  WHERE qualification
  GROUP BY grouping-list
  HAVING group-qualification
  ```
- La target-list contient (i) des noms d'attributs (ii) des opérateurs d'agrégation (e.g., MIN(S.age)).
- La liste d'attributs (i) doit être un sous-ensemble de grouping-list. The attribute list (i) must be a subset of grouping-list.
- Trouver l'âge du plus jeune marin majeur (âge > 18) pour chaque niveau de notation ayant au moins 2 marins.
- ```
  SELECT S.rating, MIN(S.age)
  FROM Sailors S
  WHERE S.age>=18
  GROUP BY S.rating
  HAVING COUNT(*)>1
  ```

- Pour chaque bateau rouge, trouver le nombre de réservations reçues.
- ```
  SELECT B.bid, COUNT (*) AS scount
  FROM Sailors S, Boats B, Reserves R
  WHERE S.sid=R.sid AND R.bid=B.bid AND B.color='red'
  GROUP BY B.bid
  ```
  ```

Fin du premier cours.