University of Liège

Base de données

Info-0009

# Synthese BDD

Julien Gustin

February 1, 2021

# Contents

# 1 Chapitre 1 : Introduction

Une **Base de données** *BD* est l'ensemble de données conservées à long terme sur un ordinateur.

- **Modèle :** Concept utilisée pour structurer et définir des données.
- **Schéma :** Decription de l'organisation des données et de leurs type. Il ne varie pas au cours de l'utilisation de la BD.
- **Instance :** Contenu réel de la BD à un temps fixé.

## 1.1 Modèle entité-relation

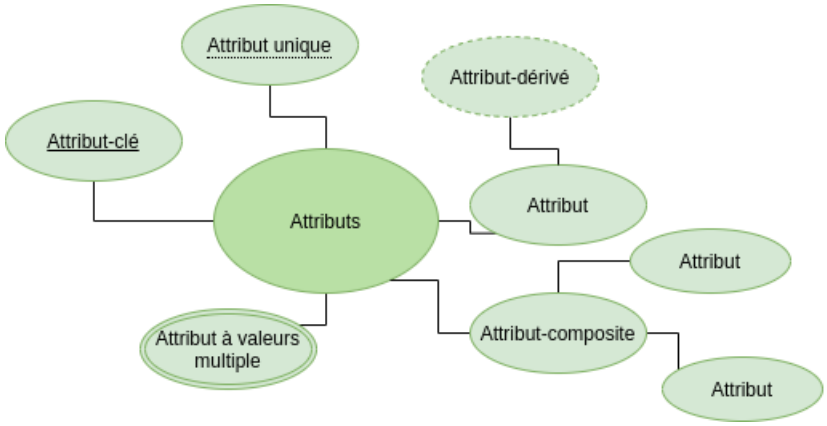

- **Entité** est un objet qui conserve de l'information (personne, voiture, entreprise, ...)
- **Relation** est le lient entre plusieurs entitées
- **Role** est le role de l'entité dans la relation ex : pere/fils
- **Cardinalité** (min, max)
  - min = 0, 1, si min = 0, participation totale des entités, sinon partielle.
  - max = 1, N

Soit k le nombres d'entitées

- k = 2 => binaire
- k = 3 => ternaire

## 1.2 Les attrbuts

Toutes informations conservées au sujet des entités d'un ensemble sont leurs attributs.



## 1.3 Les clés

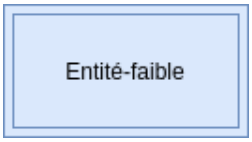
Une clé d'un ensemble d'entités est un ensemble (minimum) d'attributs qui identifient de façon unique une entité parmi cet ensemble.

### 1.3.1 Entités faible

Un ensemble d'entités ne disposant pas d'attribut constitutant une clé.

**La clé** d'un ensemble d'entités est constituée :

- D'attributs de l'ensembles d'entités, **et/ou**
- De roles joués par d'autre ensembles d'entités dans leur relation avec cet ensembles d'entités

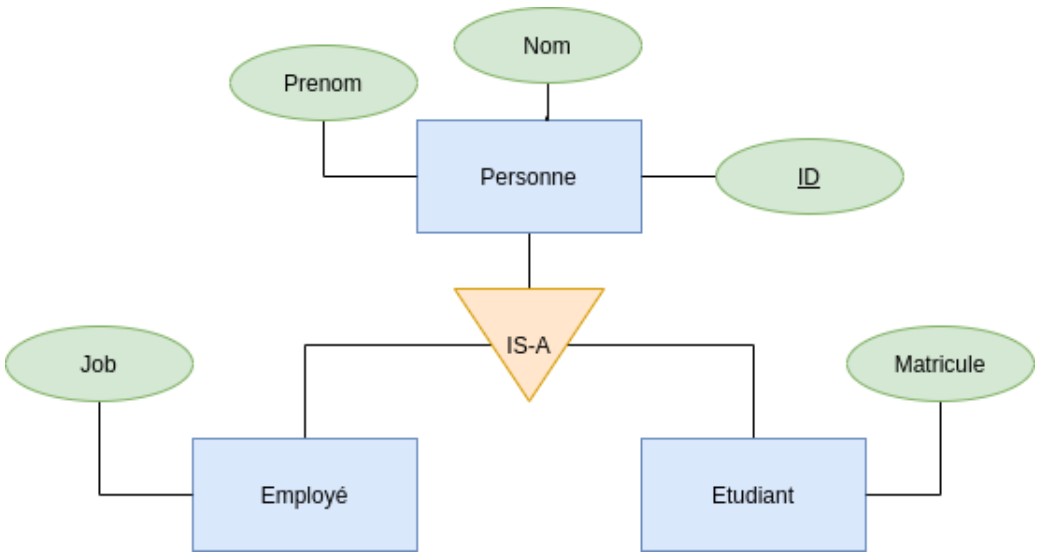



### 1.3.2 Attributs et clé de relations

Les relations peuvent avoir des attributs, exactement comme les entités.
Une clé d'une relations identifie de façon unique un tuple parmi tout ceux de la relation, elle est composée de :

- Roles joués par des ensembles d'entités dans cette relation **et/ou**
- D'attribut de la relation

## 1.4 Is-A




Employé et Etudiant hérite des attributs de Personne.
! Ne pas oublié cardinalité + role, si max = 1, on peut etre soit employé ou étudiants sinon on peut etre les deux.

# 2 Chapitre 2 : Modèle relationnel

Ici on utilise un seul type de structure pour représenter les données : **La relation** (table)

| A | B | C |
|---|---|---|
| $a_1$ | $a_1$ | $c_1$ |
| $a_1$ | $b_2$ | $c_1$ |
| $a_2$ | $b_2$ | $c_3$ |

Où A, B, C sont les attributs des relations et dont chaque ligne est un tuple.

Ex: Personne : Personne(<u>ID</u>, Nom, Prenom, Rue, Num)
(les attributs composites sont supprimés et remplacer par ses composites)

## 2.1 Les clés

- **Superclé** : Est un ensemble d'attributs qui identifie de manière unique un tuple de la relation. Càd qu'il ne peut y avoir dans la relation deux tuples distincts qui auraient les memes valeurs pour la super clé.
- **Clé** : Est une superclé minimal. Càd une superclé dont on ne peut enlever aucun attribut sans lui faire perdre sont statut de superclé.

Exemple :
Si on à $A \longrightarrow BCD$ (ou A est une superclé) alors si $AB \longrightarrow CD$. AB ne sera pas une clé mais une superclé.

## 2.2 Entités

### 2.2.1 Ensembles d'entité "normaux"

- Deviennent automatiquement des relations.
- Les attributs classiques (et les clés) sont conservées.
- Les attributs composant un attribut composite sont conservés mais pas l'attribut composite lui-meme.
- Les attributs dérivés ne sont pas conservés.
- Les attributs multi-évalués deviennent de nouvelles relations.

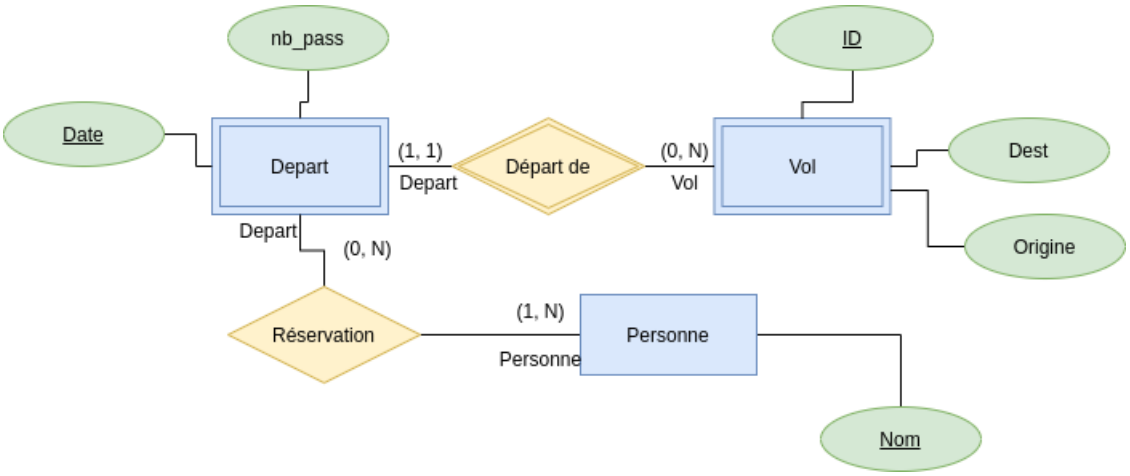
Si l'ensemble d'entités est relié à un ou plusieurs autre(s) ensemble(s) d'entités à travers une relation, et que sa cardinalité max = 1, alors la nouvelle relation comportera des clés étrangères faisant référence aux clés des autres ensembles d'entités.

### 2.2.2 Ensemble d'entités faibles

- Suivent les memes règles que les ensembles d'entités non faible.
- Leurs clé est composées de l'attribut-clé et de la clé de l'ensemble d'entité qui permet de les définirs.

## 2.3 Relations

Si une entité participant à la relation possède une cardinalité max = 1, cette relation n'est pas conservée. Sinon elle devient une relation dont la clé est l'union des clés de chaque ensemble d'entité qu'elle relie.
Exemple :




Vol(<u>ID</u>, Dest, Origin)
Depart(<u>Date</u>, <u>#ID</u>, nb_pass)
Reservation(<u>#Nom</u>, <u>#Date</u>, <u>#ID</u>)

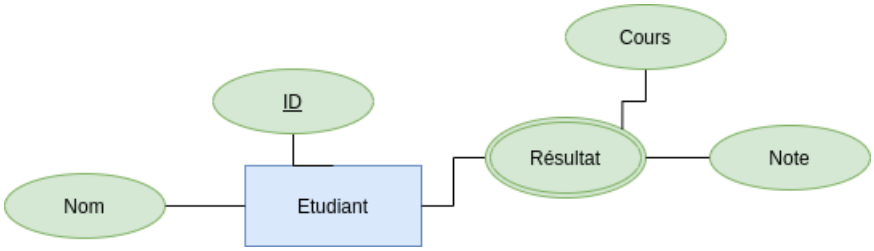
## 2.4 Cas particulier

### 2.4.1 Role (1, 1)

Soit $E_1$ et $E_2$ représenté par $T_1$ et $T_2$ et une relation R. Plutot que représenter R, on peut ajouter à $T_1$

1. Les attributs de la clé $T_2$
2. Les attributs de la relation R

### 2.4.2 Attributs multi-évalué




Etudiant(<u>ID</u>, Nom)
Resultat(<u>#ID</u>, <u>Cours</u>, Note)

### 2.4.3 IS-A

Les sous-entités hérite des attributs de l'entités mère.

## 2.5 Algèbre relationnelle

Soit les relations r du schema R
Soit les relations s du schema S

$r$ :

| $A$ | $B$ | $C$ |
|---|---|---|
| $a$ | $b$ | $c$ |
| $d$ | $a$ | $f$ |
| $c$ | $b$ | $d$ |

$s$ :

| $A$ | $B$ | $C$ |
|---|---|---|
| $b$ | $g$ | $a$ |
| $d$ | $a$ | $f$ |

$r \cup s$ :

| $A$ | $B$ | $C$ |
|---|---|---|
| $a$ | $b$ | $c$ |
| $d$ | $a$ | $f$ |
| $c$ | $b$ | $d$ |
| $b$ | $g$ | $a$ |

$r - s$ :

| $A$ | $B$ | $C$ |
|---|---|---|
| $a$ | $b$ | $c$ |
| $c$ | $b$ | $d$ |

$r \cap s$ :

| $A$ | $B$ | $C$ |
|---|---|---|
| $d$ | $a$ | $f$ |

$r$ :

| $A$ | $B$ | $C$ |
|---|---|---|
| $a$ | $b$ | $c$ |
| $d$ | $a$ | $f$ |
| $c$ | $b$ | $d$ |

$\pi_{\{B,A\}}(r)$ :

| $B$ | $A$ |
|---|---|
| $b$ | $a$ |
| $a$ | $d$ |
| $b$ | $c$ |

$r$ :

| $A$ | $B$ | $C$ |
|---|---|---|
| $a$ | $b$ | $c$ |
| $d$ | $a$ | $f$ |
| $c$ | $b$ | $d$ |

$\sigma_{B=b}(r)$ :

| $A$ | $B$ | $C$ |
|---|---|---|
| $a$ | $b$ | $c$ |
| $c$ | $b$ | $d$ |

$r$ :

| $A$ | $B$ |
|---|---|
| $a$ | $b$ |
| $d$ | $e$ |

$s$ :

| $C$ | $D$ |
|---|---|
| $c$ | $d$ |
| $f$ | $g$ |

$r \times s$ :

| $A$ | $B$ | $C$ | $D$ |
|---|---|---|---|
| $a$ | $b$ | $c$ | $d$ |
| $a$ | $b$ | $f$ | $g$ |
| $d$ | $e$ | $c$ | $d$ |
| $d$ | $e$ | $f$ | $g$ |

$r$ :

| $A$ | $B$ |
|---|---|
| $a$ | $b$ |
| $d$ | $e$ |

$s$ :

| $B$ | $C$ |
|---|---|
| $b$ | $f$ |
| $g$ | $h$ |

$r \bowtie s$ :

| $A$ | $B$ | $C$ |
|---|---|---|
| $a$ | $b$ | $f$ |

$r$ :

| $A$ | $B$ | $C$ |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

$s$ :

| $D$ | $E$ |
|---|---|
| 3 | 1 |
| 6 | 2 |

$r \underset{B<D}{\bowtie} s$ :

| $A$ | $B$ | $C$ | $D$ | $E$ |
|---|---|---|---|---|
| 1 | 2 | 3 | 3 | 1 |
| 1 | 2 | 3 | 6 | 2 |
| 4 | 5 | 6 | 6 | 2 |

$r$ :

| $A$ | $B$ | $C$ | $D$ |
|---|---|---|---|
| $a$ | $b$ | $c$ | $d$ |
| $a$ | $b$ | $e$ | $f$ |
| $g$ | $h$ | $c$ | $d$ |
| $i$ | $j$ | $k$ | $\ell$ |

$s$ :

| $C$ | $D$ |
|---|---|
| $c$ | $d$ |
| $e$ | $f$ |

$r \div s$ :

| $A$ | $B$ |
|---|---|
| $a$ | $b$ |

$r$ :

| $B$ | $C$ | $D$ |
|---|---|---|
| $b$ | $c$ | $d$ |
| $j$ | $k$ | $\ell$ |

$\delta_{A \leftarrow B}(r)$ :

| $A$ | $C$ | $D$ |
|---|---|---|
| $b$ | $c$ | $d$ |
| $j$ | $k$ | $\ell$ |

# 3 Chapitre 3 : Dépendances fonctionnelles

Soit un schéma de relation R($A_1$, ..., $A_n$) et soit $X, Y \subseteq \{A_1, ..., A_n\}$

- Les ensembles X et Y définissent une dépendance fonctionnelle, notée $X \longrightarrow Y$ *X détermine Y.*
- Une relation r de schéma R satisfait une dépendance fonctionnelle $X \longrightarrow Y$ si $\forall$ tuples $t_1, t_2 \in r$ si $t_1[x] = t_2[x]$ alors $t_1[y] = t_2[y]$

Exemple :
Client(Nom, Adresse, Solde_du) $\Longleftrightarrow$ Nom $\longrightarrow Adresse, Solde\_du$

## 3.1 Definition

Soit F et G deux ensembles de dépendances :

- Une dépendance $X \longrightarrow Y$ est impliquée logiquement par F ($F \models X \longrightarrow Y$) si pour toute relation qui satisfait F satisfait aussi $X \longrightarrow Y$.
- F et G sont équivalents ($F \equiv G$) si toute dépendance de G est impliquée logiquement par F et vice-versa.
- La fermeture de F est l'ensemble $F^+$ des dépendances logiquement impliquées par F, càd. $F^+ = \{X \longrightarrow Y | F \models X \longrightarrow Y\}$
- La fermeture $X^+$ à partir de X par rapport à l'ensemble des dépendances F est l'ensemble des $A \in U$ t.q. $F \models X \longrightarrow A$

$$F^+ = \{X \longrightarrow Y | F \models X \longrightarrow Y\} = \{X \longrightarrow Y | X \subseteq U \text{ et } Y \subseteq X^+\}$$

## 3.2 Propriétés

- Si $Y \subseteq X$, alors $X \longrightarrow Y$ (Réflexivité)
- Si $X \longrightarrow Y$ et $Z \subseteq U$, alors $XZ \longrightarrow YZ$ (Augmentation)
- Si $X \longrightarrow Y$ et $Y \longrightarrow Z$, alors $X \longrightarrow Z$ (Transitivité)
- Si $X \longrightarrow Y$ et $X \longrightarrow Z$, alors $X \longrightarrow YZ$ (Union)
- SI $X \longrightarrow Y$, alors $X \longrightarrow Z$ si $Z \subseteq Y$ (Décomposition)

## 3.3 Calcul de fermeture

**Algorithme (calcul de $X^+$:**

Données : X, U, F

1. $X^0 = X$
2. $X^{i+1} = X^i \bigcup \{A | \exists Z : Y \longrightarrow Z \in F \text{ et } A \in Z \text{ et } Y \subseteq X^i\}$
3. Si $X^{i+1} = X^i$, L'algorithme s'arrete.

## 3.4 Notion de clé

Soit F un ensemble de dépendances

- Un ensemble d'attribut X est une **clé** d'une relation de schéma R par rapport à F, si X est un ensemble min. t.q. $F \models X \longrightarrow R$
- Un ensemble d'attribut X est une **super-clé** d'une relation de schéma R par rapport à F, si X est un ensemble (non nécessairement min.) t.q. $F \models X \longrightarrow R$

## 3.5 Formes Normales

La forme normales permet d'éviter la duplication d'information.

### 3.5.1 BCNF

Un schéma de relation est en **BCNF** si $\forall$ dépendances non-triviale $X \longrightarrow A$, alors X est une superclé

**Décomposition sans perte** Soit un schéma R et p = $(R_1, ..., R_k)$ une décomposition de R t.q. : $R = R_1 \cup ... \cup R_k$ on à $m_p(r) = \pi_{R_1} \bowtie ... \bowtie \pi_{R_k}(r)$

Critère :

- Une décomposition $p(R_1, R_2)$ est sans perte par rapport à r, si $\pi_{R_1} \bowtie \pi_{R_2}(r) = r$
- Une décomposition $p(R_1, R_2)$ est sans perte par rapport ) F si: $ou \begin{cases} R_1 \cap R_2 \longrightarrow R_1 - R_2 \in F^+ \\ R_1 \cap R_2 \longrightarrow R_2 - R_1 \in F^+ \end{cases}$

**Algorithme de décomposition en BCNF (sans perte)**

1. Si R n'est pas en BCNF, soit une dépendance non-triviale $X \longrightarrow A$ de $F^+$, où X n'est pas une super-clé
2. On découpe R en $\begin{cases} R_1 = R - A \\ R_2 = XA \end{cases}$
3. On applique l'algo à $\begin{cases} R_1, \pi_{R_1}(F) \\ R_2, \pi_{R_2}(F) \end{cases}$

Puisque les relations deviennet de plus en plus petite, la décomposition finira par s'arreter.

### 3.5.2 3 FN

**Definition :**
Premier: attribut faisant partie d'une clé.
Non premier: ne font pas partie d'une clé.

Un schéma de relation est en 3FN si $\forall$ dépendances non-triviales : $X \longrightarrow A$ où A est non premier alors X est une super-clé.

**Pour les exercices :** Les dépendances problématique en BCNF seront accepté en 3FN si les attributs de la partie de droite sont premier.

### 3.5.3 2 FN

Exclut les dépendances non-triviales $X \longrightarrow A$, où A est non-premier et où X est un sous-ensemble propre d'une clé.

**Pour les exercices :** Est-ce que la partie de gauche des dépendances problématique en 3FN sont des sous ensemble de clé? Si non => 2FN.

### 3.5.4 1 FN

Attributs à valeurs atomique (toujours vrais).

### 3.5.5 4 FN

Si pour toute dépendance $X \rightarrow\rightarrow Y$

- Soit $Y \subseteq X$
- Soit XY = R
- Soit X est une super-clé de R

### 3.5.6 5 FN

Relation r de schéma R par rapport à un ensemble de dépendances fonctionnelles et joint F

- Soit elle est triviale
- Soit chaque $R_i$ est une super-clé de R

### 3.6 Dependances à valeurs multiple (DVM)

On à une dépendance à valeurs multiple entre X et Y ($X, Y \subseteq R$) si les valeurs possibles de Y sont éterminées par X et ce indépendamment du contenu du reste de la relation : on note $X \rightarrow\rightarrow Y$ (X multi-détemine Y)

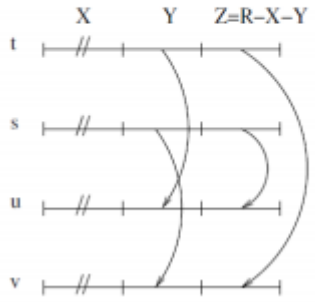
Exemple: R(Cours, Jours, Etudiant)

| Cours | Jour | Etudiant |
|---|---|---|
| #1 | Mardi | M. Lejeune |
| #1 | Jeudi | M. Lejeune |
| #1 | Mardi | J. Gustin |
| #1 | Jeudi | J. Gustin |

Exemple: R(Cours, Jours, Etudiant)
$Cours \rightarrow\rightarrow Jour$
$Cours \rightarrow\rightarrow Etudiant$

Une relation r de schéma R satisfait une dépendance à valeur multiple $X \rightarrow\rightarrow Y$ si $\forall$ paire de tuples t, s $\in r$ t.q. t[x] = s[x]. Il existe deux tuples, u, v $\in$ r t.q.

1. $u[X] = v[X] = t[X] = s[X]$
2. $u[Y] = t[Y]$ et $u[R–X–Y] = s[R–X–Y]$
3. $v[Y] = s[Y]$ et $v[R–X–Y] = t[R–X–Y]$




#### 3.6.1 Propriété des DVM

- Si $Y \subseteq X$ alors $X \rightarrow\rightarrow Y$ (Réflexivité)
- Si $X \rightarrow\rightarrow Y$ alors $X \rightarrow\rightarrow (R - XY)$ (Complémentation)
- Si $X \longrightarrow Y$ alors $X \rightarrow\rightarrow Y$ (Reproduction)
- Si $X \rightarrow\rightarrow Y$ et $X \rightarrow\rightarrow Z$ alors $X \rightarrow\rightarrow Y \cap Z$ (Intersection)
- Si $X \rightarrow\rightarrow Y$ et $X \rightarrow\rightarrow Z$ alors $X \rightarrow\rightarrow Y \backslash Z$ (Différence)
- Si $X \rightarrow\rightarrow Y$ et $X \rightarrow\rightarrow Z$, alors $X \rightarrow\rightarrow YZ$ (union)

# 4 Chapitre 4 : Langages d'interrogations

## 4.1 Ensemble / Multi-ensemble

Un **multi-ensemble** est une collection d'éléments pour laquelle on tient compte de la multiplicités. Le multi-ensembles {1, 1, 2, 3} est différent du multi-ensemble {1, 2, 3}.
Considérer les relations comme des multi-ensembles permet d'accélèrer certaines opérations.
Conversion multi-ensemble -> ensemble : (suppression des tuples redondant ($\Delta$(r))

$r$ :

| $A$ | $B$ | $C$ |
|---|---|---|
| $a$ | $b$ | $c$ |
| $d$ | $a$ | $f$ |
| $d$ | $a$ | $f$ |

$\Delta(r)$ :

| $A$ | $B$ | $C$ |
|---|---|---|
| $a$ | $b$ | $c$ |
| $d$ | $a$ | $f$ |

## 4.2 Opérateur calculant des valeurs agrégées

$r$ :

| $A$ | $B$ | $C$ |
|---|---|---|
| 1 | $b$ | $c$ |
| 3 | $a$ | $f$ |
| 3 | $a$ | $f$ |
| 2 | $b$ | $f$ |

$\texttt{SUM}(A) = 9$
$\texttt{AVG}(A) = 2,25$
$\texttt{MIN}(A) = 1$
$\texttt{COUNT}(A) = 4$

## 4.3 Opérateur de groupement

Notation : $\gamma_L(r)$ où L est une liste d'éléments qui sont :

$r$ :

| ID_ET | ID_COURS | COTE |
|---|---|---|
| 1 | *INFO0009* | 18 |
| 3 | *INFO0009* | 8 |
| 3 | *INFO0016* | 7 |
| 2 | *INFO0016* | 17 |

$\gamma_{ID\_COURS,\texttt{AVG}(COTE)\rightarrow M\_COURS}(r)$ :

| ID_COURS | M_COURS |
|---|---|
| *INFO0009* | 13 |
| *INFO0016* | 12 |

## 4.4 Opérateurs de tri

Notation: $\tau_L$(r)

Si $\tau_{ID,Cote}$(r) La machine tri d'abord par cote puis par ID.

### 4.5 Opérateur de joint externe

Notation : $r \overset{\circ}{\bowtie} s$

Calcule un joint sans perdre les tuples pour lesquels il n'y à pas de tuples en concordance sur les attributs commun.

$r_1$ :

| $A$ | $B$ | $C$ |
|---|---|---|
| $a$ | $b$ | $c$ |
| $d$ | $a$ | $f$ |
| $d$ | $a$ | $f$ |

$r_2$ :

| $C$ | $D$ | $E$ |
|---|---|---|
| $c$ | $d$ | $e$ |
| $c$ | $d$ | $e$ |

$r_1 \overset{\circ}{\bowtie} r_2$ :

| $A$ | $B$ | $C$ | $D$ | $E$ |
|---|---|---|---|---|
| $a$ | $b$ | $c$ | $d$ | $e$ |
| $a$ | $b$ | $c$ | $d$ | $e$ |
| $d$ | $a$ | $f$ | $\perp$ | $\perp$ |
| $d$ | $a$ | $f$ | $\perp$ | $\perp$ |

### 4.6 SQL

**Suppression** :

**delete from** $r$
**where** $\psi$

**Insertion** :

**insert into** $r$
**values** $(v_1, \ldots, v_k)$

**Modification** :

**update** $r$
**set** $A_1 = \mathcal{E}_1, \ldots, A_k = \mathcal{E}_k$
**where** $\psi$

Listes non exhaustif de requètes SQL :

- **SELECT** (distincts) (* = tout) (AVG) (Count) ...
- **FROM**
- **WHERE** (and) (in) (or) (Exists) (Not exists) (is null) ...
- **UNION**
- **INTERSECT**
- **EXCEPT**
- **CROSS JOIN** (produit cartésien)
- **NATURAL JOIN** (joint naturel)
- **GROUP BY**
- **LIKE / NOT LIKE**
- **ORDER BY** (asc) (desc)

# 5 Chapitre 5 : Mise en oeuvre et utilisation

## 5.1 Contrainte d'intégrité

- **Locale** Primary-key : Concerne que la relation définie
- **Référence** Foreign-key : Implique plus d'une relation

## 5.2 Locale

- **Not Null** : Impose qu'un attribut ne puisse pas etre sans valeurs.
- **Default** : Spécifie, une valeur par défaut.
- **Primary-key** : $A_1, ..., A_n$ : spécifie un ensemble d'attributs constituant une clé, et plus précisement un clé que l'on à choisi comme la clé préférentielle, clé primaire.
- **Unique** $A_1, ..., A_n$ : specifie un ensemble d'attribut constituant une clé (autre que la clé primaire).

## 5.3 Etrangères

Un tuple dans une relation qui fait référence à un tuple d'une autre relation doit faire référence à un tuple existant dans cette relation.

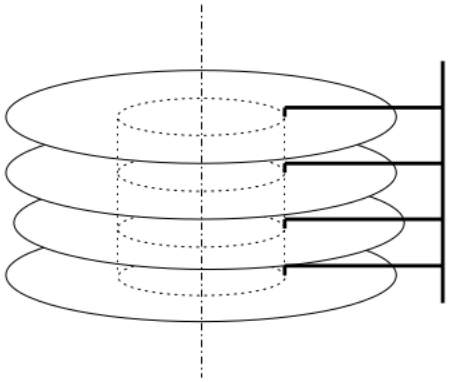
## 5.4 Vues

Une relation dérivée d'autre relation enregistrée dans la BDD.

# 6 Chapitre 6 : L'implémentation du modèle relationnel

## 6.1 Disques dur (HDD)

- Un disque comporte un ensemble de sufrace
- Sur chaque surface on trouve un ensemble de pistes (track) concentrique
- Chaque piste est divisée en secteurs *Petit groupe de données adressable et transférable sur un disque*
- Un cylindre est l'ensemble des pistes se trouvant à une position identique sur les différentes surface.



## 6.2 Les performances d'un disque

- **Seek-times** : Temps de positionnement de la tête de lecture du disque sur le bon cylindres
- **Latency** : Temps d'attente pour que le bon secteur passe en dessous de la tête de lecture *(10ms)*

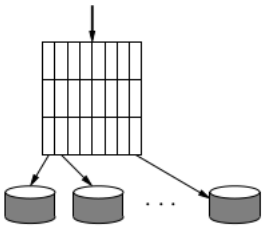
## 6.3 Raid *Redundant Array of Inexpensive Disks*

Pour obtenir de plus grosse capacités et de meilleures performances, on utilise des ensembles de disques qui sont gérés comme une seul disque virtuel : un RAID.

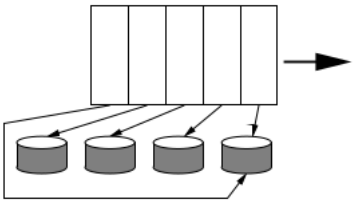
### 6.3.1 RAID 0 - Striping

Consiste à distribuer les données entre les disques du RAID, en vue de paralléliser les accès *(Plusieurs disques utilisé tous en meme temps)*. Le striping peut etre combinée avec l'écriture miroir ou avec les codes de correction d'erreur.

- **Bit level** : Creer des groupes de 8 disques et à répartir les bits de chaque octet entre ces 8 disques.

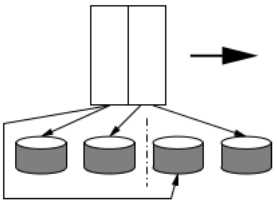


- **Block level** : Correspond à utiliser $n$ disques et à écrire le bloc $i$ sur le disque *(i mod n) +1*



### 6.3.2 RAID 1 - Mirroring

Consiste à partitionner les disques en deux, et à maitnenir une copie complète des données. L'écriture se fait symétriquement sur les deux.



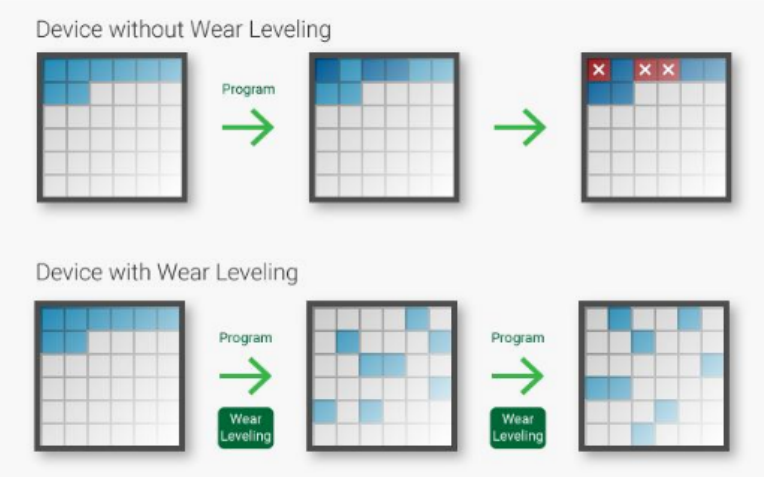
### 6.3.3 RAID 2-6

Prévoit une redondance par l'utilisation de codes de correction d'ereur.

## 6.4 SSD

| Positif + | Négatif - |
|---|---|
| - plus rapide | - les cellules se dégradent |
| - moins de bruit | - plus cher |
| - plus robustre | - moins de capacités |

Afin d'éviter une dégradation du disque, il existe le **"Wear leveling"** qui va répartir uniformément la charge d'écriture dans toutes les cellules.




Il existe d'autre type de stockages tel que

- **DAS** (direct attached storage) stockage local.
- **SAN** (storage area network) stockage reseau (bloc).
- **NAS** (network attached storage) stockage reseau (fichier).

## 6.5 Les fichiers

Les relations sont implémentées dans des fichiers en utilisant le système de gestion de fichier d'un OS. Ces fichiers sont vus comme des séquences d'octets destinée à etre lue et écrite séquentiellement. Pour implémenter une BD il faut des fichiers organisés sous la forme d'un ensemble de tuples ou d'enregistrement et des techniques permettant de retrouver rapidement un enregistrement à partir de sa clé.

## 6.6 Fichiers ISAM *Indexed sequential acces method*

Est constitué d'une séquence de blocs pour lesquels sont conservés les enregistrements. L'utilisation d'index permet une recherche rapide d'un bloc.

Les index en sql sont spécifiés au moment ou la tables est crée et on peut choisir si il sera réalisé à l'aide d'un B-tree ou une table Hash.

### 6.6.1 B-Trees *arbre équilibré optimisé*

L'équilibre est maintenu en le réorganisant périodiquement mais aussi en laissant varier le nombre de successeurs de chaque noeud dans certaine limites.

**Implémentation optimale** : Taille bloc = taille secteur

- recherche sans index : $\mathcal{O}(n)$
- recherche avec index : $\mathcal{O}(log(n))$

### 6.6.2 Hash-tables

Souvent utilisé pour des relations temportaire conservées en mémoires. Ca consiste en un enregistrement répartis en "bacs" dans un tableau. (chainage)

Le nombre d'enregistrement moyen par bac $< 1$

Pour ceci il nous faut une bonne fonction de hashage qui calculera le hash d'un enregistrment.

- Temps d'enregistrement $\mathcal{O}(1)$

## 6.7 Critère de performances *temps de calcul*

- $n_r$ : nombres de tuples de r
- V(X, r) : nombres de tuples distincts qui apparaissent dans r pour les attributs X (le nombres d'occurence de tuplesayant les meme valeurs pour les attributs X est donc $\frac{n_r}{V(X,r)}$
- $s_r$ : tailles des tuples

## 6.8 Complexité du joint $r_1 \bowtie r_2$

### 6.8.1 Joint sans index

**Méthode naive :** $\mathcal{O}(n_{r1} \times n_{r2})$
Examiner toutes les paires de tuples et comparer les valeurs des attributs en commun

**Tri et fusion :** $\mathcal{O}(n_{r1} \log n_{r1} + n_{r2} \log n_{r2})$
Trier les deux relations par rapport à leurs attributs en commun. Fusionner les résultats.

### 6.8.2 Joint utilisant une table "hash

**hash :** création $= \mathcal{O}(n_{r1} + n_{r2})$, ensuite $\mathcal{O}(1)$
On construit une table "hash" pour $r_2$ sur les attributs communs X. On parcours $r_1$ et pour chacun de ses tuples (et sa valeur pour X), on va chercher dans la table "hash" les tuples de $r_2$ qui ont la meme valeur pour X.

### 6.8.3 Joint un présence d'un index

**B-tree :** $\mathcal{O}(n_{r1}(1 + \log n_{r2})$
Supposons que $r_2$ possède un index pour des attributs communs X. On parcourt $r_1$ et pour chaque tuple de $r_1$ (et sa valeur pour X) on va chercher dans $r_2$ les tuples qui ont la meme valeur pour X.

## 6.9 Principes généraux de l'optimisation

- Sur base de différentes strat. estimer le cout et le minimiser.
- Effectuer les séléctions et projection dès que possible.
- Rassembler des stats sur le contenu de la BD pour estimer les couts.

Exemple :

*Sélection et joint.*
Soit un attribut $A \in R$.

$$\sigma_{A=a}(r \bowtie s) \;=\; (\sigma_{A=a} r) \bowtie s$$

*Utilité* : Réduire la taille des relations avant le calcul d'un joint.

# 7 Chapitre 6.b : Les transactions

Les transactions permettent d'éviter les conflits en cas d'utilisation de la BD par plusieurs personnes en meme temps.

1. L'intération avec la BD se fait exclusivement par l'intermédiare de transaction qui en maintiennet la cohérence.
2. Garantis l'atomicité, si une transaction est exécutée, elle doit l'etre entièrement. En cas de parallélisme entre transactions. Chacune doit etre exécuter de façon en apparence indivisible.

TRANSACTION :

- Atomicity
- Consistency
- Isolation
- Durability

**Un Ordonnancement** d'un ensemble de transactions $\{T_1, ..., T_K\}$ est un ordre d'exécution des opérations $a_i^j$ des transactions $T_i$. Il est séquentiel lorsqu'il existe une permutation $\pi$ de {1, ..., k} t.q. l'ordonnancement est $T_{\pi(1)}, ..., T_{\pi(k)}$. Et est séquentialisable s'il exise un ordonnancement séquentiel de $\{T_1, ..., T_k\}$ qui "donne le meme résultat".

## 7.1 Les verrous

- Lock(D, mode) où mode = {r, w}
- Unlock (D)

☐ Chaque donnée est verrouiller avant sa première utilisation et déverrouillée après sa dernière utilisation.

☐ Aucune opération de verrouillages ne peut suivre une opération de déverrouillage de la meme transaction.

☐ 2 Phases : Verrouillage et Deverrouillage

## 7.2 Verrous explicites

- Verrous sur les tables
  - **LOCK TABLE <table> READ** Verrouille la table en lecture.
  - **LOCK TABLE <table> WRITE** Verrouille la table en écriture.
  - **UNLOCK TABLES** Enlève les verrous de toutes les tables.
- Verrous sur les tuples
  - **SELECT ... LOCK IN SHARE MODE** permet la lecture et bloque les autres sessions s'ils essayent d'écrire. (plus un verrou en lecture)
  - **SELECT ... FOR UPDATE** idem, mais un second appel est cette fois bloquant. (plus un verrou en écriture)
  - Déblocage avec **COMMIT** (mise à jour dans la BD) ou **ROLLBACK** (annule toute modification)

Commence toujours par **START TRANSACTION**

## 7.3 Niveaux d'isolations

Permet d'éffectuer des transactions avec certaines propriétés sans s'inquièter de l'utilisation explicite de verrous.

4 MODES

1. **READ UNCOMMITTED** pas de protection
2. **READ COMMITTED** lecture uniquement de données validées
3. **REAPEATABLE READ** lecture identique dans toute transaction (au sein d'une meme transaction affichera toujours les memes résultats, cependant pas de verrous en insertion)
4. **SERIALIZABLE** l'insertion sera bloquante, donc plus de problème d'écriture / lecture

DESACTIVATION DE LA VALIDATION AUTO :

- set autocommit = 0

# 8 Chapitre 7 : Bases de données déductibles

L'idée est de coupler une BD à un ensmble de règles logiques qui permettent d'en déduire de l'information. Elle est constitué de :

- Ensemble de prédicats extensionnels, dont l'extension est conservée explicitement dans la BD.
- Ensemble de prédicats intentionnels, dont l'extension est défini par des règles déductives.

EXEMPLE :

**Base de données extensionnelle :** Prédicat *parent* à 2 arguments (ou relation *parent* à 2 attributs)

| *parent* | |
|---|---|
| *anna* | *bob* |
| *bob* | *chris* |
| *bob* | *dan* |
| *dan* | *eric* |

**Base de données intentionnelle :** Prédicat *grandparent* à 2 arguments (ou relation *grandparent* à 2 attributs)

*grandparent(X,Y)* ← *parent(X,Z)* AND *parent (Z,Y)*

## 8.1 Restriction

1. Les prédicats du membres de gauche sont uniquements des prédicats dérivés (Intentionnels)
2. **Condition de sureté :**
   - Toute variable utilisée dans une règle doit apparaitre dans au moins un atome dont le prédicat représente un relation et qui n'est pas précédé d'une négation.

EXEMPLE :

- **Regle non sure :** $q(X,Y) \leftarrow p(X,Z)$ AND NOT $r(X,Y,Z)$ $AND$ $X \geq Y$
  - $X \geq Y$ n'est pas un prédicats, nous voyons que Y n'apparait jamais dans un prédicat non précédé d'un NOT.
- **Regle sure :** $q(X) \leftarrow q(X)$ $AND$ $X \geq 0$
  - X apparait dans un prédicat qui n'est pas précédé d'un NOT donc c'est bon.

**Doit fonctionner de manière récusif :**

- cas de base
- cas récursif

# 9 Chapitre 8 : BD Orientées Objet

Utilisé notamment pour des données plus complexes t.q. des images, multimédia etc.

- Spécifie la structure d'objet complexe.
- Spécifie les opérations à appliquer à ces objets.

## 9.1 Définition

- $\{D_i\}$ est un ensemble fini de domaine, D = $\cup_i D_i$.
  - $D_{int}$ : L'ensemble des entiers
  - $D_{pays}$ : L'ensemble des pays
  - ...
- **A** : un ensmeble dénombrable d'attributs.
  - NOMBRE, ETUDIANTS, PAYS, ...
- **ID** : un ensemble dénombrable d'identificateur.
  - OBJ1, OBJ2, ...

## 9.2 L'ensemble des valeurs d'objets

### 9.2.1 Valeurs de base

Ce sont :

- **nil** : utilisé pour réprésenter une valeur non définie.
- les éléments $d \in D$.

### 9.2.2 Valeurs-ensembles

Ce sont les sous-ensembles de ID.

### 9.2.3 Valeurs-tuples

Ce sont les fonctions (partielles) de A vers ID.

## 9.3 L'ensemble des objets

Un **objet** est une paire (id, v) où

- $id \in ID$ est l'identificateur de l'objet.
- $v \in V$ est la valeur de l'objet.

Il existe sous 3 formes possibles:

- Objets de base :
  - (id, v) où v est une valeur de base et le type $D_i$, avec $v \in D_i$. Le type de nil est NIL
  - Exemple : (NOM, $D_{string}$)
- Objets-ensembles :
  - (id, v) où v est une valeur-ensembles et de type $2^T$ avec T qui est l'union des types des objets (id', v') t.q. $id' \in v$.
  - Exemple : (ENFANTS, $T_{enfants}$) ou le type $T_{enfants} = 2^{D_{string}}$
- Objet tuple :
  - (id, v) où v est une valeur-tuple et de type $\{(a_i, T_i)$ t.q. $v(a_i)$ est définie, $T_i$ est le type de l'objet (id', v') pour lequel id' = v($a_i$)
  - Exemple : $T_{point} = \{(LATITUDE, D_{lat}), (LONGITUDE, D_{long})\}$

## 9.4 Manipulation des objets

Lie une classe avec ses méthodes
→ Seules une méthodes associé à un objet peuvent être utilisées pour le manipuler (Encapsulation)

## 9.5 Les classes

Ce sont les ensembles d'objet du meme type. Comme en OOP (héritage = IS-A). Une autre classse qui hérite des attributs et méthodes, elle est plus spécialisée.

## 9.6 Modèle Relationnel Objet

Permet de combiner les deux, pour avoir les avantages des deux points : SQL - Objet

# 10 Chapitre 8.b XML

C'est un langage d'annotation de textes permettant d'y ajouter des informations.

Le **XML** est très utile pour exporter de l'information d'une BD et l'incorporer dans une autre.

```
<?XML version="1.0" encoding="UTF-8 ?> <!--declare le type de document -->
<Note> <!-- Ouvrir Tag, qui est un objet de type tuple {a, sujet, texte} -->
    <a> Latour </a> <!-- data -->
    <sujet> Rappel </sujet> <!-- data -->
    <texte> Ne pas oublier l'exam de BD! </texte> <!-- data -->
</note> <!-- ferme le tag -->
```

### 10.0.1 Definitions de types de documents

Data type définition (DTD). La définition du type de document est mentionné dans l'entete.

```
<?XML version="1.0" encoding="UTF-8 ?>
<! Doctype note [
    <! Element note (a, de, sujet, texte)> <!-- defintion du tuple note -->

    <!-- Defintion des 4 attributs -->

    <! Element a (#PCDATA)>
    ...
    <! ELEMENT texte (#PCDATA)>
]>
<note>
...
</note>
```

- Si Well-formed, utilisable dans tout les navigateurs
- On peut ajouter une précision (Répétition)
  - ? 0 ou 1 fois
  - * 0 ou + fois
  - + 1 ou + fois
  - EXEMPLE : <!ELEMENT note (message+)>

# 11 Chapitre 9 : L'intégration des données

## 11.1 Les entrepots de données

Permettent de regrouper des données dans un entrepot pour pouvoir en faire des analyses statistiques.

Va consolider les données de la BD pour pouvoir faire ces opérations.

### 11.1.1 OLTP : *On line transaction processing*

Sont de petites transaction consistant en une recherche d'information et souvent une mise à jour.

### 11.1.2 OLAP : *On line analytical processing*

Grosse transaction (moins fréquentes) qui vont impliquer une grande partie des données pour réaliser des calculs statistiques.

## 11.2 L'extraction

Processus sur lequel les données opérationnelles sont transférés vers l'entrepot.

- vue matérialisée
- mise à jour incrémentale

## 11.3 Organisation : Schéma en étoile

- **Les faits**, accumulation de données reprendant des faits simples. (Exemple : chiffres de ventes)
- **Les données "dimensionnelles**, Des données qui sont de faibles taille et ne sont pas amener à changer au cours du temps.

## 11.4 Extraire des informations d'un entrepot

## 11.5 ROLAP *relationnal on-line analytical processing*

- Utilise SQL pour extraire des informations, mais coute chers en ressource.

## 11.6 MOLAP *multidimensionnal on-line analytical processing*

- Voir la BD comme un cube à n dimension, une dimension par attribut dimensionnel et les éléments du cube sont des valeurs des attributs indépendants.

# 12 Chapitre 10.a : NoSql *Not only SQL*

Permet de travailler avec des données plus volumineuses.

- **Données organisées en agrégats** au lieux de stocker des données dans différentes tables on fusionne tout en une seule ligne disponible tout de suite
- **Un agrégat** est un ensembles des données que l'on consulte habituellment ensemble, par exemple : l'information concernant un client et ses commandes.
- On accède aux agrégats à partir d'une clé (clé, valeur)

Ainsi la structure de données la plus approprié pour ceci est la **Hash table**

- (clé, valeurs) sont réparties sur différentes machines d'une grappe. (sharding)
- Les memes données peuvent apparaitre plusieurs fois (rapidité d'accès et fiabilité)

Mais que ce passe t-il en cas de mise à jour d'une information? Transaction BASE:

- **Basically Available** (disponiblité assurée)
- **Soft-state** (l'état du système peut évoluer)
- **Eventually consistent** (la cohérence est atteinte à terme)

## 12.1 Différence NoSQL - Relationnel

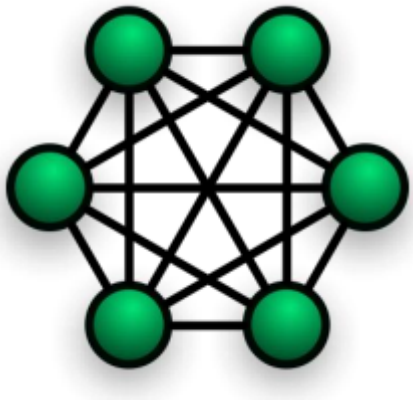
1. Dans le relationnel les données doivent etre recherchées dans plusieurs tables. → Jonction entre toutes les tables.
2. Transaction ACID très couteuses à garantir dans un contexte de réplication. Il faut synchroniser les copies. → Pas le cas avec NoSQL.
3. Les garenties fortes offertes par les transactions du relationnel ne sont pas toujours nécessaires. (exemple : vue yt)
4. NoSQL privilégie un temps de réponse court pour tous les utilisateurs par rapport à une cohérence parfaite.
5. Utiliser des données pas (ou moins) structurées a ses avantages.

# 13 Chapitre 10.b BlockChain

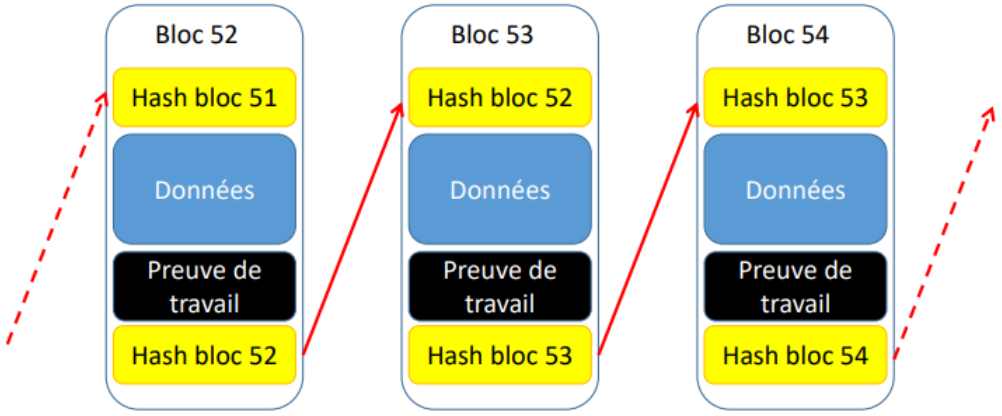
## 13.1 Principe

C'est une base de données (ressemblant à une NoSQL), c'est une BD distribuée.

1. Pérenne
    - Les données restent de manière permanente, ce qui peut aussi poser un problème de taille de données? (on ne peut pas supprimé des données, juste ajouté)
2. Infalsifiable
    - Ou, du moins, très difficile à falsifier
3. Distribué
    - On parle plutot de réseau maillé (chaque noeud est relié à tout les autres)



La base de données est ce qu'on appel une BlockChain, car c'est tout simplement une chaîne de bloc.




- **Hash** : Les hash permettent de rendre la falsification difficile, le hash d'un bloc prends en compte le hash du bloc suivant
- **Données** :
- **Preuve de travail** :

Le hash est le résultat d'une fonction de hashage (SHA-256) qui est le meilleur algortihme connu, (aucune collisions connus à ce jour).

## 13.2 Difficulté

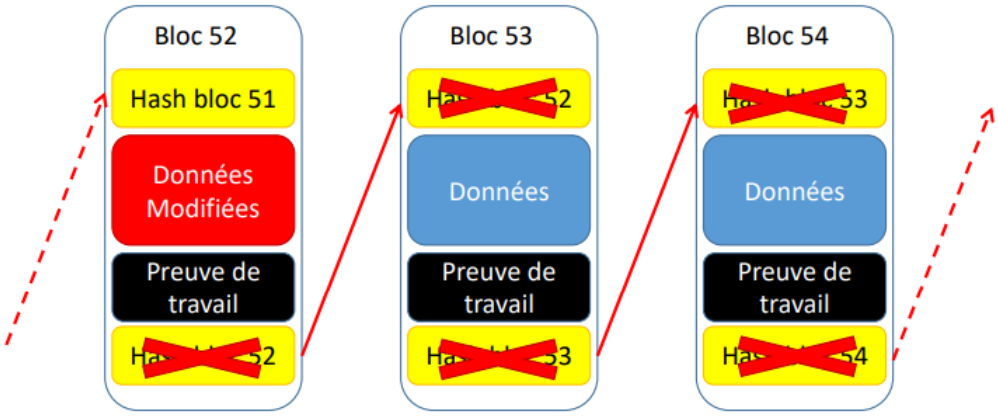
- Chaque personne possède une version de la blockchain.
- N'importe qui peut rajouter un bloc.
- On ajoute la notion de difficulté : un bloc peut etre ajouté que si son hash est inférieur à une certaine valeur (cible).
  - C'est en modifiant la preuve de travail que l'ont peut atteindre un hash suffisament faible.
  - Il faut un grand nombre d'essais pour celà

Exemple d'ajout d'un bloc :

1. Envois de la requète à tout le monde (mineur?).
2. Calcul d'un hash inférieur à la cible.
3. Une fois un hash trouvé, diffusion de celui-ci et ajout du bloc.

## 13.3 Sécurité

Admettons que je veuille modifier les données du bloc 52 :
→ re-calcul du hash et ainsi de ceux qui suivent
→ le dernier bloc aura donc plus de risque de se faire falsfier.




## 13.4 Cryptomonnaies

Ce sont des blockchains stockant des information monétaires.
Cryptomonnaies ≠ Monnaie virtuelle

### 13.4.1 Limitation des duplicatas

- Difficulté de hashage adapté, pour qu'il faut environ 10 minutes pour trouver un nouveau bloc.
- Duplicatas toujours possibles, mais peu fréquents.
- Miner du bitcoin = calcul de hash.
- Si je met en production un gros cluster, j'apporte plus de capacité de calcul mondialement et le temps moyen sera inférieur à 10 minutes.
  - Solution : Modification de la difficulté de hashage tous les 2016 blocs ajoutés.

### 13.4.2 Minage

Lorsque que quelqu'un trouve une preuve de travail satisfaisante, il reçoit 12,5 bitcoins. Et ainsi un nouveau bloc pourra etre rajouté.

- Très rare, car un hash doit commencer par 72 zeros. (en moyenne il faut calculer la fonction de hashage $2^{72}$ fois)
  - Solution : groupement de mineurs pour augmenter la probabilité et partager les récompenses.
- Est-ce intéressant ?
  - en groupe, ou avec fermes de bitcoins
  - tout les 210000 blocs la récompense est divisée en deux

### 13.4.3 Cryptographie

- Chaque utilisateur possède deux clés cryptographiques
  - Clé privée
  - Clé publique
- La transaction "A donne X bitcoins à B" est signée par A
- Tout utilisateur peut dès lors utiliser la clé publique de A pour vérifier si le message n'a pas été altéré

### 13.4.4 Problèmes

- Prends énormement de places sur le disque dur (il faut stocker toute la blockchain).
- Niveau vie privée c'est ok, car l'argent ne transite que d'un compte à l'autre, mais on ne sait pas à qui appartient le compte.
- Le nombre de bitcoins à miner est fini (32 millions).