Conception des bases de données
La normalisation

Objectifs du cours

- Normalisation des relations
- Première forme normale
- Dépendance fonctionnelle
- Deuxième forme normale
- Troisième forme normale
- Conclusion

Normalisation des relations

- Il existe différents modèles relationnels pour un même problème
- Parmi ces possibilités, quelles sont les plus à mêmes à éviter les problèmes de cohérence lors d'opérations de mise à jour ?
- Ces relations « bien faites » sont dites « normalisées »
- Il existe une échelle de nuances dans la normalisation

Besoin de normaliser

- Ex : Un parc de machines est représenté par la relation unique suivante : Parc(num, type, caracTech, lieu, ...)
- Contrainte : des machines de même type ont les mêmes caractéristiques
- Redondance : plusieurs couples (type, caracTech) peuvent être présents.

num	type	caracTech	lieu
12	G200	PPro 200MhZ	Bureau 308
24	DInspiron7000	PII 350Mhz	Bureau 294
17	IBM Aptiva	486 DX2 50 Mhz	Bureau 304
33	G200	PPro 200MhZ	Bureau 227
...	...	...	...

Problème de mise àjour

- Anomalie d'insertion. Impossible de mémoriser les caractéristiques d'un type de machine n'existant pas dans le parc
- Anomalie de suppression. Supprimer une machine unique représentante de son type supprime aussi toutes les infos dont on disposait
- Anomalie de modification. Toute modification de caractéristique d'un type de machine doit être répercutée sur toutes les autres machines

Risques d'incohérence en cas de M.A.J

Principe

- La théorie de la normalisation repose sur l'analyse des dépendances entre attributs.
- Elle propose des méthodes systématiques pour décomposer les relations incriminées.
- Il doit toujours être possible de reconstituer la relation originelle par jointure.
- Ces méthodes conduisent à des résultats "normaux".
- Certains AGL produisent directement les tables normalisées.
- Attention à vouloir normaliser en toutes circonstances : dans certaines situations, on est amené à dénormaliser.

Première forme normale

- Une relation est en première forme normale si et seulement si :
  - tous les attributs ont une valeur atomique
  - tous les attributs contiennent des valeurs non répétitives
  - tous les attributs sont constants dans le temps

Client(numclient, nom, age, adresse, tel) →

Client(numclient, nom, date_nais, adresse, code_postal, ville, tel)

Dépendance fonctionnelle

- Soit $R(\Delta)$ une relation, Δ ses attributs et X et Y des groupes d'attributs de R, il existe une dépendance fonctionnelle entre X et Y (on dit aussi que X détermine Y et l'on note $X \rightarrow Y$) si dans la relation R chaque valeur de X détermine une et une seule valeur de Y

Exemple de dépendance fonctionnel e

$$
\begin{aligned}
A &\rightarrow B \\
B, C &\rightarrow D \\
D &\rightarrow E \\
A, C &\rightarrow D \\
A, C &\rightarrow E
\end{aligned}
$$

A	B	C	D	E
a1	b1	c1	d3	e2
a1	b1	c3	d4	e3
a2	b2	c4	d2	e1
a3	b1	c1	d3	e2
a2	b2	c4	d2	e1

- Une dépendance fonctionnelle est une assertion sur toutes les extensions possibles de la relation et non pas uniquement sur les valeurs actuelles

Propriétés des dépendances fonctionneles

- Afin de manipuler les *DF* il existe 3 règles qui permettent de déduire de nouvelles dépendances fonctionnelles et de permettre l'élaboration de preuves, les axiomes d'Armstrong :
  - *La réflexivité :* si X est un ensemble d'attributs tels que $Y \subseteq X$ alors on a
    $X \rightarrow Y$ (et comme dépendance triviale $X \rightarrow X$).
  - *L'augmentation :* si $X \rightarrow Y$ et W est un ensemble d'attributs alors $W, X \rightarrow W, Y$ (W, X est une notation simplifiée de $W \cup X$).
  - *La transitivité :* si $X \rightarrow Y$ et $Y \rightarrow Z$ alors $X \rightarrow Z$

- Ces règles sont correctes (ne déduisent que des DF valides), et complètes(permettent de déduire toute dépendance valide)

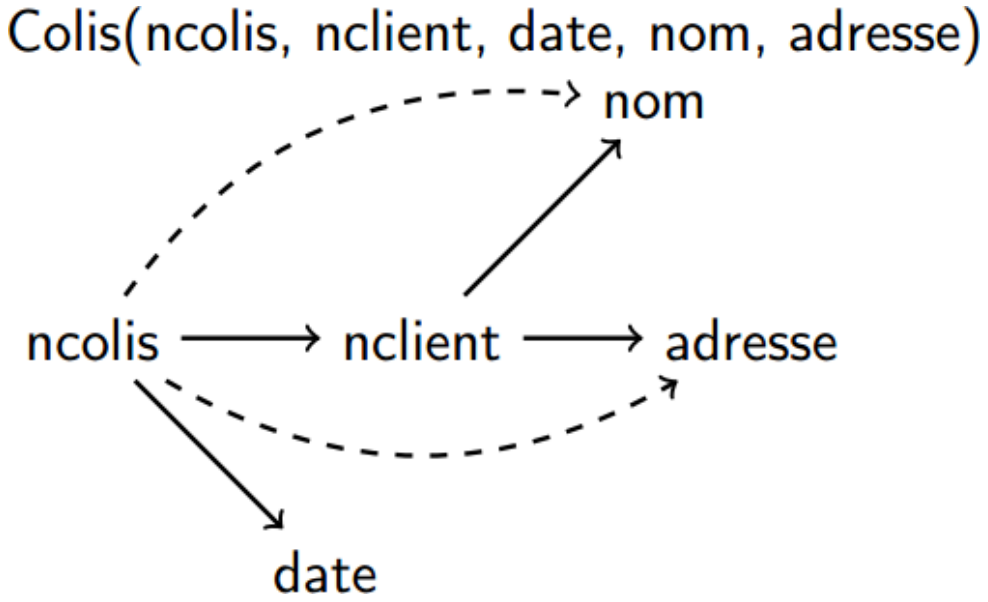
Exemple d'application

$$\mathsf{DF} = \begin{cases} A & \rightarrow B \\ B, C & \rightarrow D \\ D & \rightarrow E \\ A, C & \rightarrow D \\ A, C & \rightarrow E \end{cases}$$

- Démontrer à partir de l'ensemble de *DF* précédent que $A,\ D \rightarrow B,\ E$.
  - Par augmentation de $A \rightarrow B$ avec D on obtient $A,\ D \rightarrow B,\ D$
  - Par augmentation de $D \rightarrow E$ avec B on obtient $B,\ D \rightarrow B,\ E$
  - Par transitivité des deux précédentes on obtient $A,\ D \rightarrow B,\ E$

Fermeture et couverture minimale

- *DF* dérivée = *DF* construite à partir d'autres via les axiomes d'Armstrong
- Fermeture de *F* (un ensemble de *DF*)= ensemble des *DF* qui peuvent être dérivée à partir de *F*
- Couverture minimale de *F* = ensemble minimal de *DF* à partir desquels on peut dériver *F*



Deuxième forme normale

- Une relation est en deuxième forme normale si et seulement si :
  - elle est en en première forme normale
  - tous les attributs non-clés ne dépendent pas d'une partie de la clé primaire mais bien de la totalité de la clé primaire

Commande (Numcli, CodeArticle, Date, QtéCommandée, Désignation)

→

Commande (Numcli, CodeArticle, Date, QtéCommandée)

Articles(CodeArticle, Désignation)

Troisième forme normale

- Une relation est en troisième forme normale si et seulement si :
  - elle est en en deuxième forme normale
  - tous les attributs n'appartenant pas à la clé ne dépendent pas d'un attribut non-clé (la dépendance fonctionnelle est directe)

Commande(NuméroCommande, #CodeClient, Nom client, #RefArticle)

→

Commande(NuméroCommande, #CodeClient, #RefArticle)

Clients(CodeClient, Nom client)

5) principes pour une BDDR performante:

1) **AUCUNE REDONDANCE** : sinon cela augmente le volume de la base et multiplie les mises à jour par autant de données redondées, donc augmente la durée du traitement, donc dégrade les performances.
2) **PAS D'ANOMALIE TRANSACTIONNELLE** : la modification d'une valeur sémantique doit se traduire par la modification d'une seule donnée dans une seule table. Sinon il faudra mettre à jour de nombreuses lignes et c'est pénalisant : durée de la requête plus longue.
3) **LES DONNÉES DOIVENT ÊTRE ATOMIQUES** : une colonne de table ne doit contenir qu'une seule information. En revanche, une vue peut contenir des données concaténées, agrégées, calculées… Une donnée non atomique se traduit par des requêtes alambiquées (SUBSTRING par exemple), OU, multiples critères dans la clause WHERE et souvent par l'incapacité à utiliser les index. En sus cela empêche une évolution aisée de la structure de la base.

4) **LES COLONNES D'UNE TABLE DOIVENT CORRESPONDRE À DES ATTRIBUTS PROPRE À L'ENTITÉ QUE LA TABLE MODÉLISÉE** : un numéro de téléphone, un mail ou une adresse ne sont pas des attributs propre à une personne, mais relatif à une table de téléphone, de mail ou d'adresse associé à la personne. En créant des tables artificiellement obèses (nombreuses colonnes), les requêtes sont plus complexes, l'accès aux données se fait la plupart du temps par balayage de la table (aucune utilisation d'index), la durée de verrouillage est très notablement allongée, car c'est toujours la même table qui est impactée par les mises à jour.

5) **LES CLEFS PRIMAIRE DES TABLES DOIVENT ÊTRE CONCISES, ASSÉMANTIQUE ET INVARIABLE** : la meilleure clef primaire est un auto-incrément de type INT ou BIGINT. Car courte (4 ou 8 octets, c'est-à-dire la longueur du mot du processeur en 32 ou 64 bits), assémantique (aucune signification de la données dont la valeur est arbitraire) et invariables (la clef ne sera jamais mise à jour). Avec une telle clef, et si les FOREIGN KEY sont correctement indexées, peu importe le nombre des jointures, car contrairement à ce qu'écrivent de nombreux benêts, ce n'est pas la jointure qui pose des problèmes de performance, mais l'encombrement de la RAM avec des tables obèses, et l'impossibilité d'exploiter les index avec des tables ne respectant pas ces principes de modélisation…