5. La clé primaire des tables

Introduction

On l'a vu, la clé primaire identifie chaque objet et doit donc posséder une valeur unique dans chaque table. Si nécessaire, plusieurs attributs peuvent être groupés pour former une clé primaire unique.

Clés primaires composées

En pratique, mieux vaut **éviter** dans la mesure du possible de définir des **clés primaires composées**. Leur utilisation pose plus de problèmes qu'elle n'apporte d'avantages, notamment lors du développement des programmes. En outre, si une telle clé doit servir de clé étrangère dans un autre type d'objet, il faudra faire figurer tous les attributs de la clé composée dans cet autre type d'objet. Mieux vaut donc créer un nouvel attribut sans signification pour les utilisateurs (un numéro d'objet).

Exemple:

Il s'agit ici d'enregistrer le nombre d'heures travaillées par un employé chaque mois.

heures travaillées (*no employé*, année, mois, heures)

Que choisir comme clé primaire? Clairement, le numéro d'employé ne suffit pas puisqu'il y aura un enregistrement (objet) pour chaque employé chaque mois. On pourrait proposer la solution suivante:

heures travaillées (*no employé*, année, mois, heures)

qui est parfaitement correcte. Mais, pour les raisons citées, on préférera la solution suivante:

heures travaillées (no enregistrement, *no employé*, année, mois, heures).

Clés primaires "parlantes", clés primaires neutres

Il faut éviter de choisir comme clé primaire un attribut dont la valeur contient des informations complexes, un attribut dit "parlant". C'est attribut est évidemment un candidat à la clé primaire, mais ne constitue pas un bon choix parce que sa valeur est susceptible de **changer**. Mieux vaut créer pour la clé primaire un nouvel attribut sans signification pour les utilisateurs (un numéro d'objet).

Exemple 1: le sempiternel problème des codes d'articles

Dans une certaine entreprise, les articles sont numérotés comme suit:

XX-NNNN

Le premier chiffre désigne la catégorie d'article et le second la sous-catégorie. Par exemple, l'article 37-1235 appartient à la catégorie 3 sous catégorie 7.

En outre, les catégories 1 à 7 dénotent des articles fabriqués en interne, alors que les catégories 8 et 9 désignent des articles de revente.

De tels systèmes de désignation (numériques ou avec des lettres, cela n'y change rien), malheureusement très courants sous une forme ou une autre, ne sont déjà pas judicieux en termes de gestion (que faire si une catégorie a plus de dix sous-catégories, ou si un article précédemment acheté est subitement fabriqué?). Mais ils posent aussi de gros problèmes au niveau informatique, principalement parce leurs valeurs sont susceptibles de changer.

Le code d'article utilisé dans l'entreprise en question ne constitue donc pas un bon choix comme clé primaire. On le conservera comme attribut, mais on créera pour servir de clé primaire un nouvel attribut *code_article_neutre*. Ce nouveau code article n'aura aucune signification pour les utilisateurs qui ne le verront peut être même pas apparaître sur leurs écrans ou listes. Il sera géré par le système: les articles seront automatiquement numérotés de 1 à n au fur et à mesure de leur introduction dans la base.

Exemple 2: l'ancien numéro AVS suisse

L'ancien numéro AVS (no de sécurité sociale suisse) que recevait chaque personne était composé de quatre parties: un code de 3 chiffres correspondant au nom de famille, l'année de naissance sous forme de 2 chiffres, un groupe de 3 chiffres indiquant sexe et jour de naissance et un suffixe servant à distinguer les doublons, à indiquer la provenance nationale et à fournir un chiffre de contrôle. On ne pouvait faire plus compliqué!

Exemple: 707.40.408.214

707: codification du nom de famille; ce code change lorsque les femmes se marient
40: année de naissance
4: trimestre de naissance; chez les femmes, les trimestres sont numérotés de 5 à 8
08: jour de naissance dans le trimestre (l'exemple spécifie un homme né le 8.10.40)
2: une autre personne possède déjà la même clé
1: nationalité suisse
4: chiffre de contrôle.

Ce système n'a cessé de poser d'immenses problèmes aux administrations (notamment du fait que les femmes changeaient de numéro lors du mariage). Ceci et le fait qu'il soit inutilisable en informatique a obligé récemment la Confédération à le changer et à assigner de nouveaux numéros neutres et permanents.

Notre choix pour les clés primaires

L'attribut idéal pour une clé primaire ne contient donc dans le cas idéal **aucune** information gérée par les utilisateurs. On dit que c'est une **clé neutre** ou non-parlante. La valeur de cette clé ne sert à rien d'autre qu'à identifier l'objet dans la base et dans les programmes. L'avantage essentiel est que cette valeur ne change jamais et ne comporte pas de problèmes de limites de principe.

Nous présentons souvent ici une clé primaire neutre sous la forme: *#-clé*. Par exemple *#-personne*, prononcé "numéro personne" ou "code personne".

Exemple/exercice d'attribution de clés

Dans cet exercice, c'est le même que l'exemple présenté précédemment, il s'agit d'enregistrer les heures consacrées par des collaborateurs d'une société à différents projets pour des clients de l'industrie automobile (évidemment dans le but de facturer ces heures).

Nous présentons d'abord la façon de ne **pas** faire, solution typique de ce qui était courant par le passé lorsqu'on était sévèrement lié par la place disponible pour stocker les informations (par exemple sur des cartes perforées), mais également typique de ce que ferait un amateur avec Excel ou Access.

Employé	**Projet**	**Heures**
Jean Dupont	Renault	20
Fritz Durant	Volvo	25
Eve Meier	Ford	18
Pierre Muller	Volvo	7
Eve Meier	Renault	10

La solution précédente risque de poser de gros problèmes. Que fera-t-on si Eve Meier se marie et devient Eve Durant, ou si un deuxième Pierre Muller est engagé, ou si l'un des clients change de raison sociale.

En outre une telle solution ne fonctionne évidemment que pour un seul mois, la période de décompte dans cette entreprise. Le mois suivant, il faudra ouvrir une nouvelle table.

Le problème vient évidemment du fait que ni l'employé ni le client ne sont déterminés par des clés univoques.

La solution idéale, style base de données relationnelle, comprend non pas une, mais **trois** tables: *employé, client, heures*.

Employé (#-employé, nom, prénom, …)

Client (#-client, raison sociale, adresse, …)

Heures (#-rapport, *#-employé, #-client,* année-mois, heures)

Nous avons ici défini une clé neutre pour la table *heures*, bien que le regroupement des attributs *#-employé, #-client, année-mois* soit également univoque (comme nous l'avons vu précédemment) et constitue donc un candidat à la clé primaire.

Employé		
#-employé	**Nom**	**Prénom**
1001	Dupont	Jean
1002	Durant	Fritz
1003	Meier	Eve
1004	Muller	Pierre

Client		
#-client	**Raison sociale**	**Adresse**
17	Ford	Detroit
18	Renault	Paris
19	Volvo	Göteborg
20	VW	Wolfsburg

Heures				
#-rapport	**#-employé**	**#-client**	**Année-mois**	**Heures**
81001	1001	18	200905	20
81002	1002	19	200906	25
81003	1003	17	200905	18
81004	1004	19	200905	7
81005	1003	18	200906	10

Évidemment, trois tables, c'est plus complexe à gérer qu'une seule table! Mais quels grands avantages cela nous procure.

En premier lieu, les employés et les clients sont gérés séparément, avec tous les attributs nécessaires, pas seulement ceux requis pour la facturation des heures. Ces tables peuvent également servir à d'autres utilisations. Les informations au sujet de ces entités ne sont stockées qu'à un seul endroit. Si une valeur change, par exemple la raison sociale d'un client, il ne faut la changer qu'en un endroit.

Le recours à des clés primaires neutres permet également d'identifier de façon univoque chaque objet enregistré dans le système, même si deux d'entre eux ont certaines valeurs d'attributs identiques.

Par contre, le recours à des clés neutres comporte aussi quelques désavantages. Pour comprendre les informations dans la table *heures*, nous devons sans cesse nous référer aux tables *employé* et *client*. Pour remplir la table *heures*, nous devons connaître les numéros d'employé et de client.

Heureusement, les outils à notre disposition nous facilitent la vie et nous encouragent donc à adopter la solution plus correcte.

Excel, mais surtout Access et tous les autres véritables SGBD nous permettent de créer des rapports (on les appelle des requêtes) qui présentent des valeurs tirées de plusieurs tables en effectuant automatiquement la jointure (liaison) entre tables par l'intermédiaire des clés. Il devient alors facile dans l'exemple ci-dessus de produire un rapport des heures par employé, par client ou par mois.

La multiplication du nombre de tables dans le modèle relationnel et la nécessité de réunir des données tirées de plusieurs tables était une des principales réticences exprimées au sujet de ce modèle par rapport aux "anciennes façons de procéder". C'est trop compliqué et trop lent, disait-on. Aujourd'hui, ces critiques sont sans fondement dans la majorité des situations, même si les problèmes de performance restent quelques fois critiques et nous obligent à quelques fois à adopter des solutions pas totalement pures.