Modèle relationnel

Domaine (théorique)

Le domaine est l'ensemble des valeurs possible dans une base de donnée. Par exemple:

- Ensemble des entiers : ${1, 2, 3, 4, 5, ...}$
- Ensemble des entiers allant de 1 à 4 : ${1, 2, 3, 4}$
- Ensemble des noms : ${Robert, Jean, Thomas, Marc, Jules, Simone}$
- ...

En pratique, ce domaine est défini par les types et les contraintes.

Produit cartésien (théorique)

Le produit cartésien est la "mise en relation" de plusieurs domaines. Par exemple :

Produit cartésien des noms et des entiers allant de 1 à 4 donne 24 combinaisons possibles. Mais seul 6 seront utilisée (une par nom).

Une relation

Une relation est un sous-ensemble nommé d'un *produit cartésien*. Dans l'exemple précédent, cela donnera par exemple : (toutes les occurences sont des Personnes qui sont donc dans la table "Personnes")

Nom	Age
Robert	2
Jean	1
Thomas	2
Marc	4
Jules	3
Simone	1

Une relation est représentée par une table. Une table (ou relation) est consituée de :

- Une colonne représente un attribut, une propriété. Par exemple: "Age"
- Une ligne (tuple) représente une occurence et est unique. Par exemple "Simone - 1" est un tuple

La cardinalité d'une relation est le nombre de tuples qui la compose, et le degré de la relation est le nombre d'attributs.

Schéma d'une relation

Le schéma d'une relation (ou d'une table) consiste de :

- Du nom de la relation, exemple, « *Personnes* »
- De la liste des attributs. Pour chaque attribut est définis:
  - Un nom, exemple, « *Nom* »
  - Un type, exemple, une chaine de caractère de 25 caractères maximum « *varchar(25)* »
  - Potentiellement des contraintes, par exemple, ne peut pas être nulle « *NOT NULL* »

En SQL voici un exemple de schéma:

```
CREATE TABLE Personnes (
    id INT NOT NULL,
    nom VARCHAR(25) NOT NULL,
    age INT NOT NULL,
    PRIMARY KEY (id)
);
```

Pour juger de la qualité d'un schéma il faut s'assurer que le schéma :

- Ne va pas créer de redondances
- Ne va pas créer d'anomalies
  - Anomalie de mise à jour, une mise à jour d'une information n'est pas répercutée sur toutes le occurences de l'information
  - Anomalie de suppression, la suppression d'un tuple entraine une perte d'information pertinente

Types de clés

- Les clés **candidates** est un sous-ensemble d'attributs tel que
  - Doivent être uniques (contraite d'unicité), 2 tuples ne peuvent pas avoir la même valeur pour cet ensemble d'attributs
  - Les attributs présent doivent être absolument nécessaire (contraite d'irreductibilité)

Exemples de clé candidates : Matricule, nom + prénom + addresse, etc.

- La clé **primaire** est un attribut choisi parmis les clés candidates choisi pour identifier un tuple. Dans la représentation une clé primaire sera soulignée.

Exemple de clé primaire : Matricule

- Les clés **étrangères** sont des références à des clés candidates (souvent la clé primaire), d'une autre table ou de la même table, soit de faire des liens entre des tuples.
  - Elle doivent faire référence à une clé primaire qui existe déjà
  - Si une clé étrangère existe pour une clé primaire on ne peut supprimer la clé primaire sans supprimer la clé étrangère. (ces deux règles sontla contraite d'intégrité référentielle)

Exemple de clé étrangère : Mention d'un matricule dans une autre table

- Les **surclés** sont des ensembles qui ne respectent pas la contraite d'irreductibilité des clés candidates. Soit une clé candidates (+ d'autres attributs) = surclés.

Exemples de surclé : matricule, matricule + nom

Les contraintes

- La contrainte d'unicité (`UNIQUE`) signifie que qu'il ne peut pas y avoir 2 fois la même valeur pour cet attribut dans la table
- La contrainte `NOT NULL` signifie qu'il faut y avoir une valeur pour cet attribut, il signifie que l'attribut est obligatoire

Types primitifs des attributs

> Les types peuvent être différents d'un SGBD à un autre. Par exemple dans Oracle, le type `boolean` n'existe pas il faut donc créer une colone `integer` qui a un domaine de $[0,1]$.

- Type `integer`, seulement les nombres entiers
- Type `numeric(precision[, scale])` admet des nombres réels, `precision` indique le nombre de chiffre et `scale` indique le nombre de chiffres décimaux (après la virgule)
- Type `char(longeur)` indique une chaine de caractère qui prends un espace fixe dans la base de donnée.
- Type `varchar(longueur)` indique la même chose que le précédent, sauf que l'espace est variable
- Type `date` indique une date (jour, mois, année)

- Type `timestamp` indique une date et une heure
- Type `boolean` admet les valeurs "vrai" ou "faux" (1 ou 0)

Les opérateurs relationnels (algèbre relationnelle)

- La sélection (`SELECT ... WHERE ...`) qui permet d'obtenir une nouvelle relation de même schéma qui satisfairont une condition donnée.
- La projection (`SELECT ...`) permet d'obtenir, par une sélection, seulement certains attributs.
- L'union de deux relations permet de combiner deux tables (ayant le même schéma) en une seule liste de tuple.
- L'intersection est comme l'union, elle combine deux table, sauf que contrairement à l'union, elle supprimme les doublons de tuples.
- La jointure

La normalisation (les formes normales)

En base de données relationnelles on défini différentes formes normales, elles permettent d'éviter la redondance des données et d'utiliser le SGBD à son plein potentiel.

La première forme normale (1FN) défini qu'il ne peut pas y avoir des tuples en doublon dans une table et que les attributs ne doivent pas être décomposable (par exemple "nom_prenom" est décomposable en "nom" et "prenom"), on dit alors que les attributs sont "atomiques"

La deuxième forme normale (2FN) défini qu'aucun attribut ne peut dépendre d'uniquement une partie de la clé primaire. Par exemple:

Employé
nom
prenom
nom_departement
description_departement

Si on imagine une table "employé" qui est identifié par la clé primaire "nom + prenom + nom_departement". On peut voir que "description_departement" dépends d'une partie de la clé car il est directement lié à "nom_departement". Par conséquent cela n'est pas une 2e forme normale.

Pour que celle ci soit en une deuxième forme normale on peut soit, définir un seul attribut comme clé primaire (exemple, "id_employé") ou mettre description_departement dans une autre table et utiliser nom_departement comme clé primaire de celle ci:

Employé
nom
prenom
nom_departement

Departement
nom_departement
description_departement

Pour qu'une table soit en 3e forme normale (3FN) il faut qu'aucun attribut ne dépende d'autres attributs (autre que la clé primaire), imaginons l'exemple précédent sauf que **nom_departement** ne fait plus partie de la clé primaire :

Employé
nom
prenom
nom_departement
description_departement

Ici description_departement dépends de nom_département, par conséquent cette table n'est pas en 3e forme normale. On peut séparer de nouveau en deux tables comme vu précédemment pour régler ce problème.

Pour résumer les formes normales, pour tester une table on doit se poser ces questions :

- Les attributs sont-ils atomiques (non décomposables) ? Si oui -> 1FN
- Si 1FN, est ce qu'aucun attribut ne dépends d'une partie de la clé primaire / il n'y qu'un seul attribut comme clé primaire ? Si oui -> 2FN
- SI 2FN, est ce qu'aucun attribut ne dépends d'autres attribut que la clé primaire ? Si oui -> 3FN

Revision #1
Created 26 April 2023 19:01:26 by SnowCode
Updated 26 April 2023 19:01:26 by SnowCode