Chapitre 4 : Modélisation logique de données

Intro : A la suite d'un bon MCD càd conforme aux règles et principes données on va pouvoir le transformer pour une structure relationnelle d'une bonne base de données.

La modélisation logique des données, il s'agit de construire une structure de base de données. On l'appelle MLD (càd modélisation logique de données). Il s'agit de construire une structure de base de données relationnelle si bien que certains ne parle pas de MLD mais de MLR soit modèle logique relationnelle.

Comment s'obtient un MLD ? En transformant le MCD construit selon certaines règles. Dans cette partie, on n'enrichit plus le contenu du MCD au contraire.

I/ La modélisation logique des données : Illustrations

Le représentation sous forme de ligne n'est pas forcément complète d'où la représentation graphique ou une flèche s'ajoute au MCD. Cette flèche part de la clé étrangère et aboutit à la clé primaire (sa base)

On a l'entrée et la sortie. Comment on s'y prend? Grâce à une transformation. On va utiliser des fragment de MCD ou on va avoir des règles de transformations qui va nous faire aboutir à une structure de base de données. Les règles de transformations devraient être appliqué selon un certains ordre.

Principes de transformations :

On part d'un modèle conceptuel de données exprimé avec le formalisme entité-association qui respecte les règles données dans le chapitre 3.

La structure obtenus st une structure relationnelle dans lequel les tables sont au moins en 2ei forme normale

NB : Une base de données de qualité est une base de données dans laquelle les tables sont en troisième forme normale

1er règle : Celle qui va nous permettre de transformer les entités types en table

- Toutes entité-types devient une table
- Les propriétés du types entités devient des champs

2e règle : Transformation d'association-types

Association binaire où il existe (1,1) d'un côté

Du coté (1,1) on met l'identifiant du côté d'a coté et référençant la table d'à coté il s'agit de clé étrangère

Association binaire (0,1) d'un coté

Binaire qui serait n des deux côtés : Création d'une nouvelle table qui a pour clé primaire les deux clés et une clé sur la propriété de l'association

Association ternaire ou plus : création d'une nouvelle table

{Scénario d'interaction et modélisation des besoins nouvelle orientation➔ modéliser les objectifs avec des exemples }

Règle de transformation de spécialisation :

- Plusieurs solutions sont possibles (voir les deux flèches bleue comme deux solutions distinct

Dans la première solution on a fait une table pour personne et des types entités spécialisés (étudiant et enseignant). Attention, dans étudiant nous avons encore noss donc le numéro de sécurité social. Pourquoi? Noss : un étudiant est une personne donc c'est ce qui va nous permettre de récupérer son nom et son âge (idem pour enseignant). Dans cette solution, peu importe les contraintes qu'on a au-dessus.

Dans la deuxième solution, on va réaliser la table étudiante mais on a fait hériter toutes les propriétés du générique. Idem pour enseignant, et dans personne on aura des personnes qui ne seront ni étudiant et enseignant par exemple le personnel administratif (les flèches dans le slides sont une erreur il faudrait les enlever)

Dans la troisième solution : une seule table en mettant tout dedans. Remarque : cette table contient des valeurs nuls par exemple les enseignants n'auront pas de niveau. Avec notre boite à outils base de données relationnelle il est possible de créer des vues qui nous donnera l'illusion de travailler que sur des étudiants ou des enseignants

Dans la quatrième solution : on va ne faire des tables uniquement pour les spécialisés. On va reporter les propriétés générique et les spécifiques. Quand il n'y a pas totalité cette solution n'est pas possible. En BDD relationnelle il est possible d'avoir deux tables avec la même clé primaire qui sont issus justement de ce "soucis"

Cas particulier de la spécialisation (slide 19): Lorsque nous avons présenter dans le MCD la structure de spécialisation les sous-type pouvait ne pas avoir d'identifiant (car ils héritent). Là on est dans le cas où dans la structure de spécialisation, les sous-types ont des id spécifique. Attention, il ne faut pas oublier le noss dans étudiant et enseignant qui nous permettent de rebondir sur personne afin de récupérer son nom et son âge.

Slide 20 : Cas de l'historique de propriétés → le H : signifie qu'on veut avoir une trace des anciennes valeurs que ce soit pour le salaire dans le 1er MCD ou l'ensemble des tables pour le 2^{e} MCD

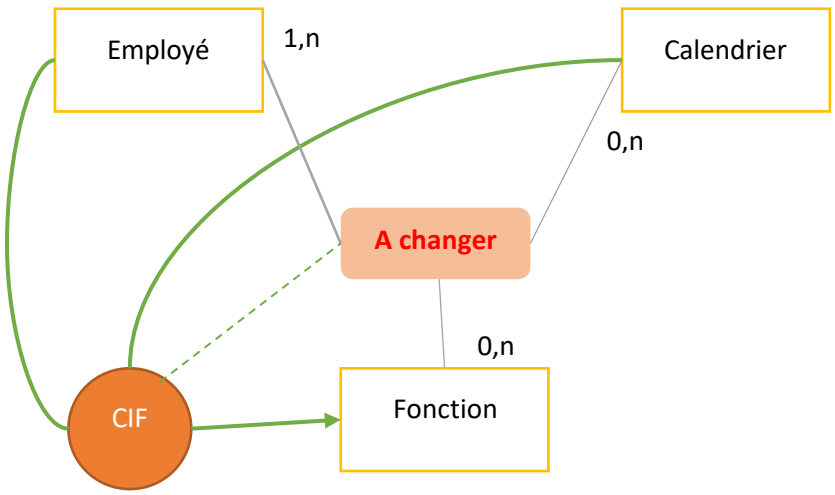
Slide 22 : On a plusieurs forme d'expression des contraintes, la procédurale (càd écriture des contraintes dans les programmes par ex. triggers) et la déclarative (déclaration clé primaire et secondaire)

Slide 23 : Il existe des contraintes contrôlées immédiatement et automatiquement (ex. primary key) , les contraintes d'intégrité peuvent également différée en fin de transaction mais de manière générale c'est immédiat.

Lorsqu'on a une base de données qui existe et fonctionne, il est possible d'ajouter à cette base de données des contraintes. Ça peut poser des problèmes, par ex. le fait d'ajouter des contraintes peuvent poser pb car les données contenus dans les table peuvent ne pas vérifier la contrainte. A force de rajouter des contraintes on va avoir potentiellement des conflits, on a encore des problèmes qui ne sont pas encore résolu. Avant de rajouter des contraintes il faut prendre beaucoup de précaution. Nous on souhaite exprimer dans notre structure de base de données les contraintes du MCD. Quelles sont les outils à nos disposition?

Slide 24 : Les contraintes intra-relation on connait la : clause primary key clause not null ,clause default, clause unique, clause check. Toutes ces contraintes sont issues du create table, idem que la clause de foreign key

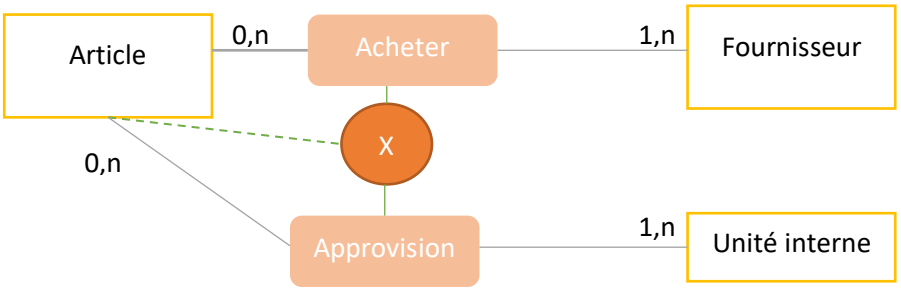
Slide 26 : Tous ça va pouvoir être exprimée en termes de table et des contraintes



Create table Employé
Create Table Fonction
Create table Calendrier
Create Histofonction(noemp… réf employé, nofonct…référence fonc…. Date…réf calendrier) Primary key (noemp, date)

Slide 28 : Assertion, ici nous sommes dans la structure d'une base de données, on peut créer des tables, des domaines, des vues ainsi que des assertions. Pour comprendre l'assertion, il faut prendre les tables ici présentes. Ici ARTICLE(art_num,art_pv), LIGNE_CMD(…lcd_pu, art_pv). Il n'existe pas dans la base de données des articles dont le prix est inférieur a 75% du prix du catalogue. Voilà ce que signifie l'assertion. Les assertions sont à l'extérieur des tables au même niveau que les creates tables)

Transformation en MLD (couleurs différentes pour les différentes étapes):



Create table ARTICLE(réf-art)

Create table FOURNISSEUR(réf-four)

Create UNITE-INTERNE(réf-unité)

Create table ACHETER(réf-art…. Référence article ,réf-four……référence fournisseur) Primary key (réf-art, réf-four)

Create Table APPROVISIONNER(réf-art…référence article, réf-unité…référence unité-interne) Primary key(réf-art, réf-unité)

Assertion : il n'existe pas dans la table acheté d'article qui est dans la table approvisionner

CREATE ASSERTION CX
CHECK not exists(SELECT * FROM acheter WHERE réf-art in(SELECT REF-art, FROM approvisionner)
UNION (SELECT * FROM approvisionner WHERE réf-art in(SELECT réf-art FROM acheter);

Slide 29 : Introduction triggers : Imaginez que dans tes MCD tu as un champs calculé comme ceci :
DEPARTEMENT(nodep,budep,nb_emp)
EMPLOYE(noemp, nomemp…, nodep)

Remarque : nb-emp est un champ calculé, un trigger est laissé la mise à jour et aller réparer la mise à jour si elle produit une incohérence en termes de contrainte. Avec cet exemple on va introduire la notion de trigger, c'est cette notion qu'on va pouvoir insérer dans sur un schéma de BD. Il est également en dehors des tables au même niveau que les assertions, table etc… Il va s'exécuter lorsqu'un certains évènement au niveau de la base de données. Les événements sont des mis à jour (insert, update ou delete). Après l'événement, il faut faire quelque chose, faire est donc une mise à jour de la base de données pour respecter la cohérence
Rappel Syntaxe UPDATE : update<table> set salaire = salaire *20% Where <condition>

Lorsqu'on va écrire un trigger, on va lui donner un nom

Create Trigger AJOUT-EMPLOYE
ON INSERT EMPLOYE E
DO UPDATE DEPARTEMENT D set nb_emp=nb_emp+1, where (condition si on est sur oracle)
nodep=new.nodep (le new est un mécanisme syntaxique pour faire référence ce qu'on a dans l'événement).

Dans SQL SERVER la syntaxe aurait été différente mais l'idée reste la même. La syntaxe aurait été. SQL SERVER part du principe que dans un trigger où il y a insert, l'employé sera mis dans une table données inserted. La condition : WHERE nodep in SELECT nodep FROM insterted)

Slide 33 : La démarche générale est composée de trois étapes. La première étape est la transformation c'est ce qu'on a vu dans la première partie du cours. On a appliqué toutes les règles de transformations du MCD en MLD sans oublié les contraintes. Le résultat obtenus c'est ce qu'on appelle le MLD BRUT, le résultat obtenu est déjà de bonne qualité mais il peut avoir quelques faiblesses, c'est la raison pour laquelle il est en deuxième forme normale. Les formes normales permettent de caractériser la qualité d'une table. Au niveau 2, il reste de la redondance, car au niveau MCD on le fait sans se soucier de la redondance qu'on va conséquemment retrouver dans le MLD brut. On ne va donc pas s'arreter la, il faut avoir une collection de table en troisième forme normale. Il s'agit de la phase de normalisation qui consiste à enlever de la redondance. Une fois qu'on a fini ça, la base de données semble être de bonne qualité. Va-t-on s'arrêter la ? Non, on va chercher à optimiser la base de données qui peut se traduire par rajouter de la redondance. Afin de gérer la redondance afin de MIEUX la gérer, car il est nécessaire d'en être consciente. Enlever de la redondance fait perdre en performance, il s'agit d'un compromis que nous devons faire.

Exemple de normalisation : Après la transformation, on va supprimer des champs redondant

➔ En rouge, normalisation, en bleu des exemples de l'optimisation

EMPLOYE(noemp, noss, noemp, ~~**nobur**~~,notel,~~**nodep**~~,noproj)
DEPARTEMENT(nodep,budgetdep,nodep,**nbemp + trigger**)
BUREAU(nobu,surface,nodep)
TELEPHONE(nodep,typetem,nobur)

Dans la phase optimisation va consister à modifier la table :

- On va lors rajouter de la redondance selon les cas pour avoir plus de performance
- Ajouter un champ calculé sur les tables (par ex. si tous les matins on a besoin d'une requête qui calcule le nombre d'employé dans un département, il est possible d'en ajouter. Mais il ne faudra pas oublier d'ajouter des triggers)
- Si on a deux tables que l'ensemble des requêtes portes sur une partie, on va pouvoir la couper soit horizontalement ou verticalement)

L'optimisation c'est tout ça et bien plus, tout cela se fait en étudiant les requêtes utilisées dans la base de données