OP4

BD & Web

Résumé de cours

Partie I : Bases de données

M. Hascoët
mountaz@lirmm.fr
Université Montpellier II
Place Eugène Bataillon
34095 Montpellier Cedex 05

Chapitre 1 : Modèle Entité-Association

1. Introduction

Le modèle entité-association est le principal modèle permettant la modélisation conceptuelle d'une base de données relationnelle. Ce modèle peut-être utilisé seul ou dans le cadre d'une méthodologie complète d'analyse et de conception de système d'information.

Dans ce cours, nous n'aborderons pas les problèmes de méthodologie comme ils peuvent être abordés dans des méthodologies complètes comme Merise, par exemple. Nous nous concentrerons sur l'essentiel de la phase de conception: l'élaboration d'un modèle conceptuel de données s'appuyant sur le modèle entité-association.

Le modèle entité-association a une double origine. En effet, il est apparu à la même période aux Etats-Unis avec Chen (1976) et en France, où il a été proposé par Tardieu (1975). Ce modèle a plus tard été intégré dans la méthode MERISE et a ainsi profité de son développement.

Dans ce chapitre nous nous intéressons aux principales caractéristiques du modèle entité-association au travers des notions :

- d'entités (et classes d'entités)
- de propriétés (aussi appelées attributs)
- d'associations (ou classes d'association)
- de cardinalités

2. Entités et occurrences d'entités

Définition : On appelle entité toute classe d' **« objets concrets ou abstraits »** ayant une **existence propre** et présentant un **intérêt particulier** pour les informations à gérer.

Définition 2: On appelle occurrence tout élément de cette classe.

3. Associations et occurrences d'associations

Définition: une association (ou classe d'association) représente un lien sémantique de même nature reliant les occurrences de plusieurs entités et présentant un intérêt pour le système d'informations.

Définition: une occurrence d'association est un n-uplet représentant le lien entre les occurrences des entités.

Arité d'une association:
Définition: le nombre d'entités reliées entre elles par une association constitue l'arité de l'association. L'*arité* est aussi appelée *dimension* ou *ordre* de l'association.

Les associations d'arité 2 sont appelées associations binaires, les associations d'arité 3 sont appelées associations ternaires. Plus généralement, les associations d'arité n ou n >3 sont appelées association n-aire.

4. Attribut ou propriété

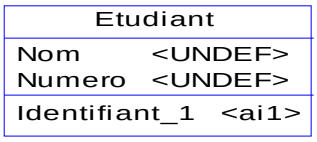
Définition : On appelle attribut ou propriété toute donnée élémentaire ou « atomique » ayant un sens.
Les attributs prennent des *valeurs* aussi appelées *occurrences* de l'attribut.

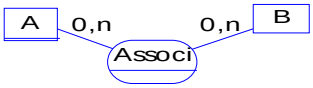
5. Représentations graphiques :

Le modèle entité-association est souvent exprimé sous forme graphique. Il existe plusieurs systèmes de conventions graphiques.

Nous adopterons le suivant :

- Les entités sont représentées par des rectangles
- Les attributs sont inscrits à l'intérieur des rectangles
- Les types associations sont représentés par des ovales et des liens vers les entités qui sont reliées par l'association
- Les cardinalités (cf paragraphe suivant) sont inscrites sur les liens





6. Cardinalités :

Soit association L reliant une entité A à une entité B. La cardinalité de l'association est définie par deux couples de valeur (a,a') et (b,b'). Ces couples ont pour but de représenter l'intervalle dans lequel se trouve le cardinal de l'image par L des occurrences de A dans B et réciproquement. Ainsi, les couples (a,a') et (b,b') peuvent être interprétés de la manière suivante :

- a (respectivement b) est le nombre *minimum* d'images qu'une entité de type A peut avoir dans B (respectivement une entité de type B dans A),
- a' (respectivement b') est le nombre *maximum* d'images qu'une entité de type A peut avoir dans B (respectivement une entité de type B dans A).

a	a'	Nb entités
0		Certaines entités de type A ne sont associées à aucune entité de type B
1		Toute entité de type A est associée à au moins une entité de type B (a au moins une image dans B)
	1	Toute entité de type A peut être associée au plus à une entité de type B
	n	Toute entité de type A peut être associée à n entités de type B

NB : La notion de cardinalité n'est pas définie de la même manière dans le modèle Américain et dans le modèle Européen. Dans le premier, il n'existe que la notion de cardinalité maximale. Ainsi, le couple (a,a') défini ci-dessus serait remplacé par a' dans certains modèles et la cardinalité de l'association dans ces modèles serait définie par le seul couple (a',b') (au lieu de (a,**a'**) , (b,**b'**) dans le modèle Merise). Dans ce cours, nous adopterons la définition Merise de la cardinalité.

7. Clé d'entité

Définition : Une clé d'entité est un attribut ou ensemble d'attributs qui caractérise de manière unique les occurrences de l'entité.

NB : il n'est pas rare qu'une entité n'ait pas de clé « naturelle », on ajoute alors parfois un attribut (code, numéro, identifiant) qui n'a pas d'autre intérêt que celui de faire office de clé.

8. Remarques diverses :

a. Remarques sur le choix d'association binaire versus association n-aire :

Soit A, B, C trois entités et R1, R2, R3 trois associations. Lorsque A, B, C sont liées on peut souvent choisir de représenter cette « liaison » de manières différentes. Par exemple on peut choisir de définir trois associations R1(A,B), R2(B,C), R3(A,C), ou bien d'en définir une seule R4(A,B,C). Comment choisir entre ces différentes solutions?

b. Associations ternaires versus associations à attributs

Règle 1 : un attribut peut être placé dans une association lorsqu'il dépend fonctionnellement de toutes les entités liées par l'association.

c. Le problème de la représentation du temps dans les modèles entité-association

Vocabulaire :

On appelle *point de vue diachronique* un modèle qui représente une action se déroulant dans le temps. Par opposition on appelle le *point de vue synchronique* une vision atemporelle que le modèle représente généralement.

Règle 2 : le plus souvent pour passer d'une vision synchronique à une vision diachronique on transforme une association n-aire en une association n+1-aire avec l'introduction d'une entité temporelle

9. Bibliographie

ACSIOME (1989). Modélisation dans la conception des systèmes d'information. Ed. Masson

Chapitre 2 : Introduction au modèle relationnel

1. Définitions de base

Univers

Un univers est un ensemble fini U dont les éléments sont appelés attributs.
Notation U = { A:domA, B:domB, ...}

Domaine

Un domaine est un ensemble fini des valeurs atomiques possibles pour un attribut
Une valeur atomique est une valeur indivisible du point de vue du modèle de données
Un domaine peut être défini en extension en donnant la liste des valeurs composantes ou en compréhension en définissant une propriété caractéristique du domaine.

Schéma de relation

Etant donné un univers U, un schéma de relation est un sous-ensemble R de U noté R (A1:D1, A2:D2, …An:Dn).
Chaque attribut Ai est le nom d'un « rôle » joué par le domaine Di dans le schéma de relation R.
Un schéma de relation R est utilisé pour décrire une relation.

Schéma de base de données

On appelle schéma de base de données relationnelles un ensemble de schémas de relations.

Relation

Une relation r sur le schéma R= {A1:D1, A2:D2, ..., An:Dn} notée r(R) est UN ENSEMBLE de n-uplets (tuple).
Chaque n-uplet ei est une liste ordonnée de n valeurs ei = <v1, v2, ...,vn> où chaque vi est une valeur du domaine Di de l'attribut Ai ou une valeur nulle spéciale représentant l'absence d'information.
Remarque : une relation peut être vue comme l'extension du schéma de relation, elle varie au cours du temps alors que le schéma est en principe invariable.

2. Passage du modèle entité association (EA) au modèle relationnel

Afin de passer d'un modèle EA à un modèle relationnel on peut utiliser les trois règles suivantes :

1) Pour chaque entité E du modèle EA on crée un schéma de relation R dans le modèle relationnel. R contient tous les attributs de E.
2) Si une entité est prise dans une association avec une cardinalité (0,1) ou (1,1) alors on ajoute dans le schéma de relation R correspondant à E, une clé de chacune des autres entités participant à l'association ainsi que tous les attributs de l'association.
3) Si dans une association A il n'existe pas d'entité E pour laquelle la cardinalité de l'association soit (0,1) ou (1,1), on crée alors un nouveau schéma de relation R correspondant à A. Le schéma R contient une clé de chaque entité participant à l'association ainsi que tous les attributs de l'association

Chapitre 3 : SQL (Structured Query Language)

1. Introduction

Conçu dans les années 70 par E. Codd, IBM. C'est le plus répandu des langages d 'accès aux bases de données.
Comporte

- langage de définition des données
  - o création de schémas de relations
- langage de manipulation des données

2. Le langage de manipulation des données

" SELECT ... FROM ... WHERE " :
La clause SELECT exprime le résultat attendu sous la forme d'une liste d'attributs auxquels il est possible d'appliquer différents opérateurs et fonctions
La clause FROM liste les relations utilisées pour évaluer les requêtes
La clause WHERE qui est facultative énonce une condition que doivent respecter les enregistrements (ou n-uplets) sélectionnés. En SQL il est possible d'exprimer des requêtes correspondant à des opérations diverses telles que la sélection, projection, union, produit cartésien ou jointure. Il est également possible d'utiliser SQL pour effectuer des tris ou des regroupement ainsi que des calculs.

Nous commençons par présenter les opérations de base :

Sélection

La sélection prend en entrée une relation R1 définie sur un schéma SR et une condition de sélection C. La sélection produit en sortie une nouvelle relation R2 de même schéma SR ayant comme n-uplets ceux de R1 satisfaisant à l'expression de sélection C.

Formulation SQL:
select *
from R
where C

Projection

La projection prend en argument une relation R et un schéma de relation $X \subseteq R$ et renvoie une relation $\pi x(R)$ définie par : $\pi x(R) = \{t/X \mid t \in R\}$ où t/X est la restriction de t sur X, ie le n-uplet composé des valeurs correspondant aux attributs de X uniquement.

Formulation SQL:
Select X
from R

Exemples :
Quels sont les codes des livres parus en 2002 ?
Quel est le nom de l'étudiant donc le numéro est 34J8UHKLI ?
Qui a écrit le livre dont le code est 2-634-26555-56-V ?

Union

L'union prend en argument 2 relations R et S sur le même schéma RS et renvoie une relation $R \cup S$ telle que
$R \cup S = \{t \mid t \in R \text{ ou } t \in S\}$

Formulation SQL:
SELECT *
FROM R UNION SELECT * FROM S
Implanté dans MySQL à partir de MySQL 4.0.0
Remarque : select … union [ALL | DISTINCT] select … [union [[ALL | DISTINCT] select … union]
Sans le mot ALL on a par défaut DISTINCT (ie pas de doublons, ie un ensemble, un vrai)
On NE peut PAS mélanger ALL et DISTINCT dans la même requête.

Exemple :
Quels sont les clients endettés ?
(Ayant un solde < 0 ou un emprunt)

SELECT *
FROM EMPRUNT UNION SELECT * FROM COMPTE where solde <0

Différence

Cette opération prend en argument 2 relations R et S sur le même schéma RS et renvoie une relation R\S telle que $R \setminus S = \{t \mid t \in R \text{ et } t \notin S\}$

Formulation SQL:
SELECT *
FROM R
WHERE not exists (SELECT * FROM S WHERE R.C1 = S.C1 and … R.Ci = S.Ci)

Exemple :
Quels sont les clients NON endettés de l'agence « Arceaux »
SELECT *
FROM COMPTE
WHERE solde > 0 AND codeClient not in EMPRUNT

Produit

Cette opération prend en argument 2 relations R et S sur les schémas respectifs SR et SS et renvoie une relation R x S sur le schéma de relation $SR \cup SS$ telle que :

$$RxS = \{t \mid t/SR \in R \text{ et } t/SS \in S\}$$

Formulation SQL :
SELECT * FROM R, S

Jointure

Cette opération prend en argument 2 relations R et S sur les schémas respectifs SR et SS. Elle renvoie une relation $R \bowtie S$ sur le schéma de relation $SR \cup SS$ telle que

$$R \bowtie S = \{t \mid t/SR \in R \text{ et } t/SS \in SS\}$$

Formulation SQL:
SELECT *
FROM R,S
WHERE R.C1 = S.C1 and … R.Ci = S.Ci avec Ci attributs communs à SR et SS

Exemples :
1. Où habitent les clients dont s'occupe Mr Martin?

```
SELECT nomClient,rue,ville
FROM CLIENT,SUIVI_CLIENT
WHERE
        nomConseiller = "Martin" AND
        SUIVI_CLIENT.codeClient = CLIENT.codeClient
```

2. Quels sont les noms des clients de l'agence « Centre » qui ont à la fois un compte et un emprunt

```
Select nomClient
From EMPRUNT, COMPTE
Where
        nomA = "Centre" AND
        E.codeClient = C.codeClient
```

Intersection

L'intersection prend en argument 2 relations R et S sur le même schéma SR et renvoie une relation $R \cap S$ sur le schéma de relation $SR \cap SS$ telle que

$$R \cap S = \{t \mid t \in R \text{ et } t \in S\}$$

Formulation SQL:
SELECT *
FROM R,S
WHERE R.C1 = S.C1 and … R.Cj = S.Cj pour tout Cj attribut de SR

Exemple :
Quels sont les clients qui ont un compte dans toutes les agences de Montpellier ?

```
SELECT nomC
FROM COMPTE
WHERE nomA = ALL (SELECT nomA FROM AGENCE where villeA = "Montpellier")
```

Cas général :

Plus généralement une requête SQL s'écrit

```
SELECT [DISTINCT] { * | <nom_de_relation>,<nom_d'attribut> | <nom_d'attribut>
FROM <nom_de_relation>
[WHERE <condition>]
[GROUP BY <nom_d'attribut> [, ...] [HAVING <condition>] ]
```

[ORDER BY {<nom_d'attribut> | <numéro_de_colonne>} [{ASC | DESC}] [, ...]

Exemples :
1. Classer les noms des clients par ville
Select *
From CLIENT
Group by ville

2. Donner les noms des agences par ordre alphabétique
Select nomA from AGENCE
order by nomA

Les fonctions agrégats

Fonctions d'agrégats:
COUNT, SUM, MIN, MAX, AVG
Ce sont des fonctions qui permettent de faire des calculs et qui s'utilisent dans les clauses select et having

Exemples :
1. Donner la somme de tous les emprunts
Select sum(montant)
From EMPRUNT

2. Donner le nombre de clients par ville
Select ville, count(*)
From CLIENT
Group by ville

3. Donner le montant de l'emprunt max
Select max(montant)
From EMPRUNT

3. Langage de définition des données

Création de table

```
CREATE TABLE nom_table(nom_colonne1 TYPE OPTION, nom_colonne2 TYPE OPTION,
… , nom_colonneN TYPE OPTION);
```

OPTION peut prendre les valeurs:
AUTO_INCREMENT : un numéro unique est attribué et incrémenté à chaque création;
NOT NULL : le champ doit être obligatoirement rempli ;
NULL : les valeurs peuvent être omises ;
PRIMARY KEY : la colonne est une clé primaire ;
UNSIGNED : pour les types numériques entiers, indique que l'entier est non négatif ;
ZEROFILL : pour les types numériques entiers et réels, sert à compléter avec des 0 l'affichage en fonction du nombre maximal de chiffres affichés ;
BINARY : pour les types chaînes CHAR et VARCHAR, indique que les valeurs des colonnes sont sensibles à la casse pour les opérations de tri, comparaison ;

DEFAULT "valeur" : valeur par défaut du champ.
TYPE est à choisir parmi:

Entiers

Les entiers peuvent être signés (+/-) ou non signés (uniquement positifs) ce qui se traduit par UNSIGNED :
TINYINT[(taille)] 1 octet (i.e. [-127,128] si signé et [0,255] pour un non signé) ;
SMALLINT[(taille)] 2 octets (ie de [-32767,32768] ou [0,65536]) ;
INT[(taille)]4 octets (ie de [-2147483647, 2147483648] ou [0, 4294967296]) ;
INTEGER[(taille)] synonyme de INT ;
BIGINT[(taille)]8 octets ;
NB :taille détermine la taille maximale du nombre de chiffres affichés en précisant taille. L'attribut ZEROFILL est utilisé pour compléter avec des 0 les valeurs jusqu'à la taille maximale d'affichage.

Réels

REAL[(taille,nbd)]synonyme de double :
DOUBLE[(taille,nbd)] nombre à virgule 8 octets ;
FLOAT[(taille,nbd)] nombre à virgule 4 octets;
DECIMAL(taille,nbd) nb stocké comme une chaîne (un caractère par chiffre) ;
NUMERIC(taille,nbd) synonyme de décimal.
NB : pour préciser la taille maximale d'affichage (taille) ainsi que la position de la virgule (nbd).

Chaînes de caractère et texte

CHAR(n) une chaîne de longueur n fixe ;
VARCHAR(n) une chaîne de longueur variable et d'au plus n caractères, la taille réellementréservée dépend de la taille de la chaine ;
TEXT chaîne texte ;

Enumerations

ENUM(value1,value2,value3,...) la valeur de la colonne doit être une des valeurs de l'énumération ;

Dates et temps

DATE une date (l'affichage sera au format 'YYYY-MM-DD') ;
YEAR une année (l'affichage sera au format 'YYYY') ;
TIME une heure (l'affichage sera au format 'HH:MM:SS') ;
TIMESTAMP une durée (l'affichage est au format 'YYYYMMDDHHMMSS', 'YYMMDDHHMMSS', 'YYYYMMDD', ou 'YYMMDD') ;
DATETIME une date et heure (l'affichage sera au format 'YYYY-MM-DD HH:MM:SS').

Suppression de table

```
DROP TABLE [IF EXISTS] nom_table1, nom_table2, ...
```

NB : L'expression IF EXISTS évite qu'il y ait une erreur si la table n'existe pas. Il est possible de supprimer une table ou plusieurs, en utilisant une liste de noms de tables séparés par des virgules.

Modification de table

```
ALTER TABLE nom_table alteration
```

Valeurs pour `alteration`	Commentaire
`ADD COLUMN <definition colonne>`	Ajouter une colonne
`ALTER COLUMN <definition colonne>`	Modifier une colonne
`DROP COLUMN <nom colonne>`	Suppression d'une colonne

Insertion de n-uplets (enregistrements)

```
INSERT INTO nom_table (nomcol1, nomcol2, ...) VALUES ('val1', val2, ...)
```

Suppression de n-uplets

```
DELETE FROM nom_table WHERE expression
```

Modification de n-uplets

```
UPDATE nom_table SET col1='val1', col2='val2',... WHERE expression
```

Chapitre 4 : Dépendances fonctionnelles et formes normales

Dépendances fonctionnelles

Soient R un schéma de relation, X,Y des ensembles d 'attributs de R.
on dit que la relation r de schéma R vérifie la dépendance fonctionnelle (DF) X-> Y si lorsque deux n-uplets ont même valeur sur les attributs de X alors ils ont même valeur sur ceux de Y.
ie « les valeurs de Y sont fonction de celles de X »

Implication

Soit F un ensemble de DF. On dit que la DF X->Y est impliquée par F si tte relation r de schéma R qui vérifie F vérifie X->Y.
On note alors F |= X->Y

Axiomes d'Armstrong

a) Réfléxivité
si $Y \subseteq X$ alors X->Y est vérifiée. (DF triviale). en particulier X->X
b) Augmentation
$\forall Z \subseteq R$ si X->Y est vérifiée alors XZ -> YZ aussi.
$\forall Z \subseteq R$, X-> Y |= XZ ->YZ
c) Transitivité
Si X->Y et Y -> Z sont vérifiées alors X--> Z aussi.
Soit {X-> Y, Y -> Z} |= X ->Z

Fermeture

On appelle fermeture de F noté F+, l 'ensemble de toutes les DF impliquées par F.

Théorème

Si X-> Y se déduit de F en appliquant les règles de Armstrong alors $F \in F+$
réciproquement toute DF $G \in F+$ peut se déduire en utilisant les règles de Armstrong

Autres règles de Armstrong

Elles sont construites à partir des axiomes pour faciliter les démonstrations
d) Additivité :
X->Y, X->Z |= X->YZ (NB: YZ = YUZ)
e) Pseudo-transitivité
{X -> Y, WY -> Z} |= XW -> Z
f) Décomposition
X-> Y |= X->Z si Z est inclus dans Y

Couverture minimale de F

Objectif: trouver un ensemble minimal de DF représentant "la même information" qu'un ensemble de DF donné F mais sans redondance.

Définition
F,G, 2 ensembles de DF. G est appelé couverture minimale de F ssi
(1) G+ = F+
(2) tt membre droit d 'une DF est réduit à un seul attribut

(3) pour AUCUNE df X->A de G on a G - {X->A} |= G
(4) pour aucune df (X->A) de G on n 'a G |= G -{X-A} ∪ Y->A avec Y⊂ X

4. Clés

Définition

Soit R(A1,...An) un schéma de relation, et F l'ensemble de DF associées à R. On dit que X ⊂ R est une clé de R ssi l 'UNE des conditions ci-dessous est vérifiée :

- X->R appartient à F+
- F |= X -> R
- toute relation r sur R qui satisfait F vérifie X->R

Clé minimale

Si de plus pour aucun sous-ensemble Y ⊂ X, on a Y -> R dans F+ alors X est appelée Clé Minimale

Propriétés utiles sur les clés

P1: tout attribut qui ne figure pas dans le membre droit d'une df non triviale de F doit appartenir à toute clé K de R
P2: soit X l'ensemble des attributs de R qui ne figurent pas dans le membre droit d'une df non triviale de F. Si X est une clé alors R possède une clé minimale unique (et c'est X).
P3: un schéma muni d'une seule DF possède une clé unique.

5. Formes Normales

3ième Forme Normale, notée 3FN

Soit un schéma de relation R, et F un ensemble de DF sur R. Un schéma de relation est dit en troisième forme normale, si tout attribut A qui n'appartient à aucune des clés minimales du schéma, ne dépend QUE des clés du schéma et des ensembles d'attributs qui le contiennent (DF triviales dans ce cas).

Forme Normale de Boyce-Codd, notée FNBC

Un schéma de relation est dit en FNBC si TOUT attribut du schéma ne dépend que des clés minimales du schéma et des ensembles d'attributs qui le contiennent (dépendances triviales).