Chapitre 3

LE MODELE RELATIONNEL

Le modèle relationnel a été inventé en 1960 et a fait l'objet de très nombreuses recherches qui ont débouché sur la réalisation et commercialisation de SGBDs relationnels. C'est le modèle le plus utilisé par les SGBDs actuellement disponibles sur le marché.

C'est un modèle de données plus simple que celui de l'entité association, ce qui explique son succès tant sur le plan théorique (théorie de la normalisation, définition de langages de manipulation de données), que sur celui des réalisations. Mais cette simplicité est aussi sa faiblesse principale: c'est un outil trop pauvre sémantiquement pour pouvoir bien représenter la complexité du monde réel. Ce pourquoi d'autres modèles de type entité association ou orientés objets ont été ensuite proposés.

Ce chapitre présente rapidement les concepts de base du modèle relationnel.

1 Définitions

Les objets et associations du monde réel sont représentés par un concept unique: la relation. Les relations sont des tableaux à deux dimensions, souvent appelés tables.

Exemples

Etudiant

N°Etud	Nom	Prénom	Age
136	Rochat	Jean	19
253	Aubry	Annie	20
101	Duval	André	20
147	Rochat	Marc	21

Cet exemple représente une relation (ou table) décrivant les étudiants. Etudiant est le nom de la relation. Les entêtes des colonnes, N°Etud, Nom, Prénom et Age, sont les **attributs** de la relation. Chaque ligne de la table correspond à une occurrence. Par exemple <253, Aubry, Annie, 20> constitue un n-uplet ou **tuple**, qui décrit une occurrence, à savoir l'étudiante Annie Aubry. Le modèle relationnel interdit d'avoir des doubles: deux tuples d'une même relation ne peuvent pas être identiques. Enfin, il est usuel de souligner l'attribut, ou les attributs, qui constituent l'identifiant de la relation; pour la relation Etudiant c'est l'attribut N°Etudiant.

Cours

Nom C	Horaire	Prof
Algo	Lundi 10-12	Duval
Système	Mardi 16-17	Malin

Suit

N° Etud	NomC
253	Algo
136	Système
253	Système

On remarque que l'identifiant de la relation "Suit" (qui traduit un type d'association) est composé des identifiants des deux relations précédentes. Cette relation Suit exprime le lien entre un étudiant, désigné par son numéro, et un cours, désigné par son nom. Si on avait utilisé le modèle entité association, les relations Etudiant et Cours auraient été modélisées par des types d'entité, Etudiant et Cours, alors que la relation Suit aurait été modélisée par un type d'association reliant ces types d'entité.

Notion de domaine

Un domaine est un ensemble de valeurs que peut prendre un attribut; c'est le domaine définition d'un ou plusieurs attributs.
Exemple de domaines:

Dnom : chaînes de caractères de longueur maximale 30
Dnum : entiers compris entre 0 et 99999
Dcouleur : {"bleu", "vert", "jaune"}
Dâge : entiers compris entre 16 et 65

Les SGBD usuels offrent des domaines prédéfinis standard, comme INTEGER, CHAR, VARCHAR, DATE…

Relations

Une relation est définie par :

- son nom
- liste de couples décrivant ses attributs (nom d'attribut : domaine)
- son (ses) identifiant(s)
- sa définition (phrase en français)

Les trois premières informations: nom de la relation, liste des couples (attribut : domaine) et identifiant(s) constituent le **schéma** de la relation.
Exemple : schéma de la relation Etudiant :

Etudiant (N° Etud : Dnum, Nom : Dnom, Prénom : Dnom, Age : Dâge)

La **population** d'une relation est constituée de l'ensemble des tuples de la relation. C'est un ensemble au sens mathématique du terme: il n'y a donc ni doubles, ni ordre (les nouveaux tuples sont rajoutés à la fin de la relation).
On appelle **schéma d'une base de données relationnelle** l'ensemble des schémas de ses relations.
On appelle base de données relationnelle, l'ensemble des populations de toutes ses relations.

2 Règles de modélisation

Les attributs sont tous simples et monovalués: toute valeur prise par un attribut pour un tuple est atomique (non décomposable par le SGBD) et unique. Les notions d'attribut multivalué ou complexe n'existent pas dans le modèle relationnel.

Cas d'un attribut facultatif du modèle entité association

Certains SGBDs relationnels gèrent une valeur spéciale, appelée valeur nulle et notée "?". Cette valeur peut être prise par tout attribut (sauf si dans le schéma on a spécifié le contraire); elle signifie alors que le tuple n'a pas de valeur pour cet attribut. Par exemple, un étudiant, Marc Dumont, de numéro 123, dont on ne connaîtrait pas l'âge, serait représenté par le tuple :

< 123, Dumont, Marc, ? > .

Une telle solution implique un traitement particulier de cette valeur nulle dans les langages de manipulation des données, notamment l'emploi d'une logique à trois valeurs : Vrai, Faux et Inconnu. En général il est bon d'éviter autant que possible l'emploi de valeurs nulles. Pour cela on peut utiliser, quand la valeur n'est pas renseignée, une valeur par défaut, par exemple 0 pour un nombre. SQL permet de définir une valeur par défaut pour un attribut grâce à la clause DEFAULT.
A l'opposé, les attributs obligatoires sont traduits en SQL lors de la création de la table par la clause NOT NULL.

Cas d'un attribut complexe du modèle entité association

Soit par exemple, un attribut complexe en entité association, Adresse, composé des attributs: n°bâtiment, nom-rue, ville, code-postal. Si on veut ajouter cet attribut dans la relation Etudiant, il y a plusieurs solutions.

Solution 1: On ajoute l'attribut Adresse qui a pour domaine les chaînes de caractères. Dans ce cas, l'utilisateur ne peut pas poser de questions sur la ville, il devra lire l'adresse et chercher dans la chaîne de caractères le nom de la ville lui-même, ou par programme.

Solution 2: On ajoute les attributs suivants : n°bâtiment, nom-rue, ville, code-postal à la relation. Le SGBD connaît le détail de l'adresse, mais ne connaît pas la notion globale d'adresse.

La solution est choisie en fonction du type d'utilisation de l'attribut.

Cas d'un attribut multivalué du modèle entité association

Par exemple, on veut mémoriser les différents prénoms des étudiants (et non pas uniquement leur premier prénom).

Solution 1: Avoir dans la relation Etudiant plusieurs attributs : Prénom1, Prénom2,...
Problèmes : combien mettre d'attributs Prénom ? Comment poser des questions sur cet attribut (on est obligé de poser autant de questions que d'attributs déclarés !).
C'est une mauvaise solution à ne pas utiliser.

Solution 2: On ne garde que N°Etud, Nom, Age pour la relation Etudiant, et on crée une relation supplémentaire pour l'attribut multivalué, EtudPrénoms :

EtudPrénoms

N°Etud	Prénom
136	Jean
136	Marie
136	André
253	Annie
253	Claudette
101	André
147	Marc
147	Antoine

Cette solution n'a pas les inconvénients de la solution précédente. Elle est valable pour tous les attributs multivalués, qu'ils soient simples ou complexes.

Si l'on veut, plutôt que l'ensemble des prénoms, la liste ordonnée des prénoms on décrira la relation:
EtudPrénoms2 (N°Etud, N°Prénom, Prénom)
ce qui correspond en entité association à un attribut multivalué de type liste.

3 Identifiant d'une relation

Définition : L'identifiant (aussi appelé clé) d'une relation est un ensemble minimum d'attributs de la relation, tel qu'il n'existe pas deux tuples ayant même valeur pour cet identifiant.
Autre définition équivalente: C'est un ensemble minimum d'attributs tel que tous les autres attributs en dépendent fonctionnellement (voir le chapitre sur la normalisation et les dépendances fonctionnelles).

Un identifiant peut-être composé d'un ou plusieurs attributs. Une relation peut avoir un ou plusieurs identifiants. Une relation étant un ensemble de tuples sans double, elle a toujours un identifiant qui,

dans le cas le plus défavorable, est composé de tous les attributs de la relation. Par convention, l'attribut (ou les attributs) constituant l' (ou un des) identifiant(s) est souligné.

Exemples
Etudiant (N°Etud, nom, prénom, age): Il n'y a pas deux étudiants qui ont le même numéro.
La relation ci-dessus, EtudPrénoms2, possède deux identifiants qui sont (N°Etud+N°Prénom) et (N°Etud+Prénom).

On ne peut pas avoir de valeur inconnue (notée "?") pour un attribut qui fait partie d'un identifiant. Par exemple pour la relation Etudiant, on renseigne obligatoirement le numéro d'étudiant. Si l'on pouvait entrer dans la relation deux tuples sans N°Etud, alors il existerait deux étudiants avec la même valeur pour l'identifiant (valeur inconnue), ce qui est impossible d'après la définition de l'identifiant. On a donc la règle suivante:

Règle: tous les attributs de tout identifiant doivent toujours avoir une valeur connue.

Exemple
Insérer dans Etudiant: < ?, Rochat, Jean, 19> est une instruction que le SGBD va refuser.

SQL offre deux concepts pour traduire la notion d'identifiant: la clé primaire (appelée "primary key") et les clés candidates (déclarée par la clause "unique"), qui se différencient lors de la déclaration des clés externes (voir le paragraphe suivant). Si la relation n'a qu'un seul identifiant, ce sera en SQL la clé primaire de la table. Si la relation a plusieurs identifiants, il faut choisir pour clé primaire "le meilleur": celui dont on est sûr que sa valeur ne sera jamais nulle ni modifiée. Les autres identifiants seront déclarés avec la clause UNIQUE.

4 Identifiant externe

Certains attributs référencent des tuples d'une autre relation (ou parfois de la même); c'est-à-dire que leur valeur est toujours égale à celle de l'identifiant d'un tuple existant dans l'autre relation. On les appelle identifiants externes (ou clés externes ou clés étrangères). Par exemple, la relation Suit (NomC, N°Etud) possède un identifiant (NomC+N°Etud), et deux identifiants externes : NomC et N°Etud. En effet, NomC "référence" un Cours, c'est à dire que si une valeur NomC existe dans Suit, alors il doit nécessairement exister un cours de ce nom là dans la relation Cours. De même, N°Etud "référence" un Etudiant.

Le schéma d'une relation comprend donc, en plus de la définition du nom de la relation et de ses attributs et domaines associés, la définition de son (ses) identifiant, et celle de ses identifiants externes, s'il en existe.

Les identifiants externes sont déclarés de la façon suivante:

Suit (N°Etud : Dnum, NomC : Dnom)
Identifiants externes: N°Etud **référence un** Etudiant
NomC **référence un** Cours
Dans le cas où la relation référencée possède plusieurs identifiants la clause précise alors lequel, comme ci-dessous:
N°Etud **référence un** Etudiant•N°Etud

Définition :
Soient deux relations R1(X, Y) et R2 (V, W), où X, Y, V, W, désignent des attributs ou des ensembles d'attributs, et où X est un identifiant de R1, on dit que W est un identifiant externe sur R1 (ou que W référence R1) si pour tout tuple de R2, la valeur prise par W est nécessairement la

valeur de X pour un tuple existant de R1. Autrement dit, à tout instant, l'ensemble des valeurs prises par W est compris dans l'ensemble des valeurs prises par X.

Une fois déclaré l'identifiant externe W de R2 sur R1, le SGBD vérifie automatiquement :

- toutes les insertions de tuples dans R2: il vérifie que la valeur de W existe dans R1. Si ce n'est pas le cas l'insertion dans R2 est refusée;
- toutes les modifications de W: il vérifie que la nouvelle valeur de W existe dans R1. Si ce n'est pas le cas la modification est refusée;
- toutes les suppressions de tuples de R1: il vérifie qu'il n'existe pas de tuple dans R2 référençant ce tuple de R1 à supprimer. S'il en existe, selon le SGBD soit la suppression est refusée, soit la suppression est propagée: les tuples de R2 qui référençaient cette valeur de X sont eux aussi supprimés.

Le SGBD assure ainsi, l'**intégrité de référence** (ou intégrité référentielle) de la base de données.

En SQL, la clause pour déclarer une clé externe sur la clé primaire d'une table est: REFERENCES nom-table; celle pour déclarer une clé externe sur une clé candidate d'une table est: REFERENCES nom-table (nom-attribut-clé-candidate).

5 Contraintes d'intégrité

Quelque soit le modèle de données (entité association, relationnel ou autre) il existe toujours des règles du monde réel qui ne peuvent pas être exprimées par les concepts du modèle. Certaines de ces règles restreignent les valeurs que peuvent prendre les données de la base. Elles sont appelées contraintes d'intégrité. Les SGBD relationnels offrent plusieurs mécanismes pour exprimer ces contraintes d'intégrité (en plus de l'intégrité référentielle des identifiants externes). Ce sont :

- attribut obligatoire: clause NOT NULL
- identifiant: clauses PRIMARY KEY et UNIQUE
- domaine de valeurs particulier d'un attribut: clause CHECK qui permet de vérifier que la valeur de l'attribut satisfait une condition
- contrainte d'intégrité quelconque: TRIGGER qui permet de spécifier que lors de toute insertion (ou suppression ou modification) d'un tuple dans telle relation telle condition doit être satisfaite sinon telle action doit être entreprise automatiquement par le SGBD, comme par exemple refuser l'insertion ou envoyer un message d'alerte.

6 Conclusion

Un schéma relationnel se compose:
pour chaque relation de:
- nom de la relation
- définition
- attributs + domaines
- identifiant(s)
- éventuellement identifiant(s) externe(s)
+ contraintes d'intégrité associées à cette relation.

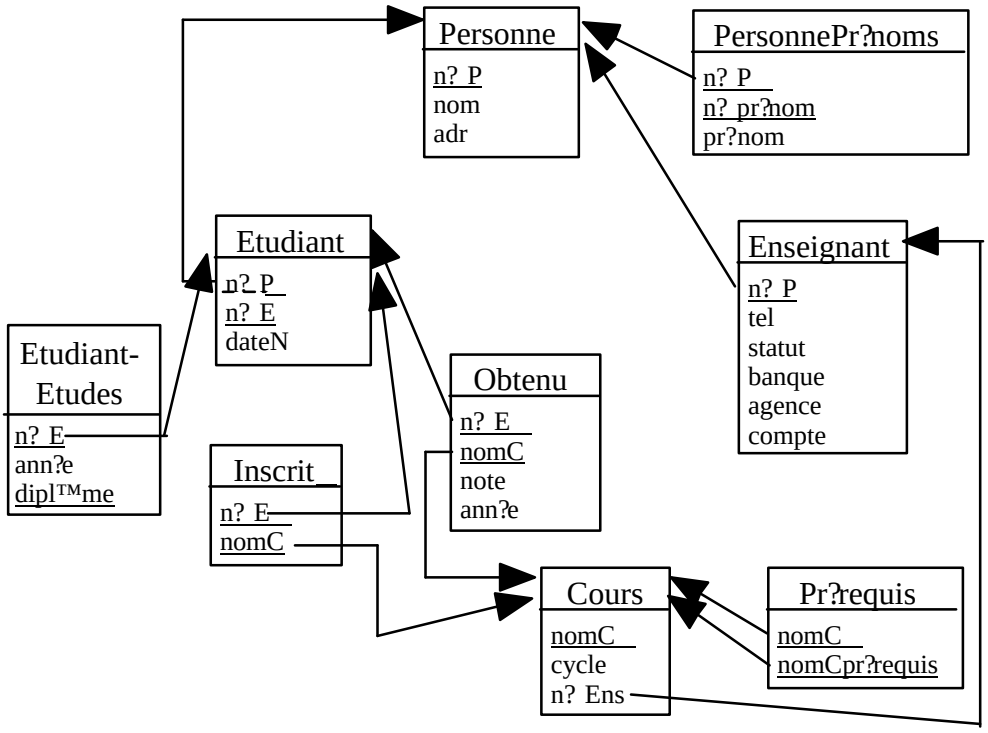
et des autres contraintes d'intégrité qui portent sur plusieurs relations.

Un exemple de schéma de base de données relationnelles pour une entreprise de formation permanente est donné dans les pages suivantes.

Exemple : la base de données "FormaPerm"

La même base de données qui a été modélisée dans le chapitre sur le modèle entité association est maintenant modélisée en relationnel. Un diagramme relationnel est présenté, puis le schéma complet avec les définitions des tables ainsi que les contraintes d'intégrité, et enfin la population d'une mini base de données qui servira pour les langages de manipulation.

Diagramme relationnel



Sur le diagramme, les flèches représentent les contraintes d'intégrité référentielle. Chaque flèche part d'un identifiant externe et pointe la table référencée.
On remarquera que le diagramme fournit moins d'informations que le schéma textuel, notamment il est difficile de différencier graphiquement un identifiant composé de plusieurs attributs de plusieurs identifiants simples.

Schéma relationnel formel

Domaines

Dnom : chaînes de caractères de longueur inférieure à *30*
Dch100 : chaînes de caractères de longueur inférieure à *100*
Dannée : [*1970* : *1990*]
Dnote : [*0.0* : *10.0*]
Ddate : [*1* :*31*] / [*1* :*12*] / [*1920* :*2100*]

Relation *Personne*

Attributs: *n°P* : entier sans nul
nom : *Dnom* sans nul
adr : *Dch100* sans nul
Identifiant: (*n°P*)
Définition: *tout étudiant et tout enseignant de l'institut.*

Relation *PersonnePrénoms*

Attributs: *n°P* : entier sans nul
prénom : *Dnom* sans nul
n°prénom : entier sans nul
Identifiants: (*n°P + prénom*) , (*n°P +n°prénom*)
Identifiant externe : *n°P* référence une *Personne*
Définition: *prénoms des personnes*

Relation *Etudiant*

Attributs: *n°P* : entier sans nul
n°E : entier sans nul
dateN : *Ddate* sans nul
Identifiants: (*n°E*)
(*n°P*)
Identifiant externe : *n°P* référence une *Personne*
Définition: *tout individu qui est actuellement inscrit à l'institut, ou qui a déjà passé avec succès un des cours de l'institut*

Relation *EtudiantEtudes*

Attributs: *n°E* : entier sans nul
année : *Dannée* sans nul
diplôme : *Dnom* sans nul
Identifiant: (*n°E + diplôme*)
Identifiant externe : *n°E* référence un *Etudiant.n°E*
Définition: *études antérieures des étudiants*

Relation *Enseignant*

Attributs: *n°P* : entier sans nul
tel: entier sans nul
statut : *Dnom* sans nul
banque : *Dnom* sans nul
agence : *Dnom* sans nul

compte : entier sans nul
Identifiant: (*n°P*)
Identifiant externe : *n°P* référence une *Personne*
Définition: *tout individu assurant actuellement un ou plusieurs cours à l'institut*

Relation *Cours*

Attributs: *nomC* : *Dnom* sans nul
cycle : entier sans nul
n°Ens : entier sans nul

Identifiant: (*nomC*)

Identifiant externe : *n°Ens* référence un *Enseignant*

Définition: *tout cours offert par l'institut*

Relation *Obtenu*

Attributs: *n°E* : entier sans nul
nomC : *Dnom* sans nul
note : *Dnote* sans nul
année : *Dannée* sans nul

Identifiant: (*n°E + nomC*)

Identifiants externes : *n°E* référence un *Etudiant.n°E*
nomC référence un *Cours*

Définition: *l'étudiant n°E a réussi le cours nomC telle année et a obtenu telle note*

Relation *Inscrit*

Attributs: *n°E* : entier sans nul
nomC : *Dnom* sans nul

Identifiant: (*n°E + nomC*)

Identifiants externes : *n°E* référence un *Etudiant.n°E*
nomC référence un *Cours*

Définition: *actuellement, l'étudiant n°E est inscrit au cours nomC*

Relation *Prérequis*

Attributs: *nomC* : *Dnom* sans nul
nomCprérequis : *Dnom* sans nul

Identifiant: (*nomC + nomCprérequis*)

Identifiants externes : *nomC* référence un *Cours*
nomCprérequis référence un *Cours*

Définition: *le cours nomCprérequis est un prérequis pour le cours nomC*

Contrainte d'intégrité: *dans tout tuple, nomCprérequis doit être différent de nomC*

Contraintes d'intégrité:

Pour tout tuple de Prérequis <nomC, nomCprérequis>, le cycle de nomCprérequis dans Cours doit être inférieur ou égal à celui de nomC.

Pour tout tuple de Etudiant: année_en_cours - dateN = 18

Pour tout tuple de EtudiantEtudes <n°E , année, diplome>, soit dateN la date de naissance de l'étudiant dans la relation Etudiant, alors: dateN < année

Pour tout tuple de Obtenu <n°E, nomC, note, année>, soit dateN la date de naissance de l'étudiant dans la relation Etudiant, alors: dateN < année

Pour tout tuple de Inscrit, <n°E, nomC>, le n°E doit exister dans Obtenu associé à tous les cours qui existent dans Prérequis associés à nomC.

NB Les mots en italique sont définis par l'utilisateur, les autres sont des clauses du langage de définition de schéma.

Schéma relationnel SQL

```
CREATE TABLE Personne
                ( n°P  INTEGER NOT NULL ,
                nom VARCHAR(30)  NOT NULL ,
                adr VARCHAR(100)  NOT NULL ,
   PRIMARY KEY (n°P )  )

CREATE TABLE PersonnePrénoms
                ( n°P  INTEGER NOT NULL ,
                prénom VARCHAR(30)  NOT NULL ,
                n°prénom  INTEGER NOT NULL ,
   PRIMARY KEY (n°P , n°prénom ) ,
   UNIQUE (n°P , prénom ) ,
   FOREIGN KEY (n°P)  REFERENCES Personne  ON DELETE CASCADE )

CREATE TABLE Etudiant
                ( n°P   INTEGER NOT NULL ,
                n°E   INTEGER NOT NULL ,
                dateN  DATE  NOT NULL ,
  PRIMARY KEY (n°E ) ,
  UNIQUE (n°P ) ,
  FOREIGN KEY  (n°P)  REFERENCES Personne )

CREATE TABLE EtudiantEtudes
                ( n°E   INTEGER NOT NULL ,
                année  INTEGER  NOT NULL ,
                diplôme  VARCHAR(30)   NOT NULL ,
   PRIMARY KEY (n°E , diplôme ) ,
   FOREIGN KEY  (n°E)  REFERENCES Etudiant )

CREATE TABLE Enseignant
                ( n°P   INTEGER NOT NULL ,
                tel   INTEGER  NOT NULL ,
                statut  VARCHAR(30)  NOT NULL ,
                banque  VARCHAR(30)  NOT NULL ,
                agence   VARCHAR(30)  NOT NULL ,
                compte  INTEGER  NOT NULL ,
   PRIMARY KEY (n°P ) ,
   FOREIGN KEY  (n°P)  REFERENCES Personne )

CREATE TABLE Cours
                ( nomC  VARCHAR(30)  NOT NULL ,
                cycle  INTEGER NOT NULL ,
                n°Ens  INTEGER NOT NULL ,
   PRIMARY KEY (nomC ) ,
   FOREIGN KEY  (n°Ens)  REFERENCES Enseignant )

CREATE TABLE Obtenu
                ( n°E   INTEGER NOT NULL ,
                nomC  VARCHAR(30)  NOT NULL ,
```

```
                note  NUMERIC(2,1)  NOT NULL ,
                année  INTEGER  NOT NULL ,
PRIMARY KEY (n°E , nomC ) ,
FOREIGN KEY (n°E)  REFERENCES Etudiant.n°E ,
FOREIGN KEY (nomC)  REFERENCES Cours )
```

CREATE TABLE Inscrit
(n°E INTEGER NOT NULL ,
nomC VARCHAR(30) NOT NULL ,
PRIMARY KEY (n°E , nomC) ,
FOREIGN KEY (n°E) REFERENCES Etudiant.n°E ,
FOREIGN KEY (nomC) REFERENCES Cours)

CREATE TABLE Prérequis
(nomC VARCHAR(30) NOT NULL ,
nomCprérequis VARCHAR(30) NOT NULL ,
PRIMARY KEY (nomC , nomCprérequis) ,
FOREIGN KEY (nomC) REFERENCES Cours ,
FOREIGN KEY (nomCprérequis) REFERENCES Cours)

NB Les contraintes d'intégrité seront implémentées en SQL par des clauses CHECK et des TRIGGER.

Base de données relationnelle partielle de FormaPerm

Personne

n°P	nom	adr
1111	Rochat	Lausanne
6666	Walter	Lausanne
5555	Bernard	Yverdon
2222	Rochat	Renens
3333	Muller	Morges

Etudiant

n°P	n°E	dateN
6666	22	3/3/78
1111	36	1/1/79
5555	44	2/2/78

Enseignant

n°P	tel	statut
2222	80111111	prof
3333	80222222	assistant

Cours

nomC	cycle	n°Ens
système	2	2222
BD	2	2222
SI	2	2222
algo	1	3333
C	1	3333

Obtenu

n°E	nomC	note	année
22	algo	10	1998
22	C	9	1998
44	algo	9	1999

Inscrit

n°E	nomC
36	algo
36	C
22	système
22	SI
44	C
44	BD

Prérequis

nomC	nomCprérequis
système	algo
système	C
BD	algo
SI	algo
SI	C