Université Claude Bernard Lyon 1

Année 2016-17
Responsable : Nicolas Lumineau

U.E. BDW1
Bases de Données et programmation WEB

Partie "Modélisation"

Conception de Base de données

Dans ce fascicule, vous trouverez différents rappels autour du schéma conceptuel Entité-Association (EA), de sa réalisation à sa transformation en un modèle logique relationnel. Vous trouverrez également quelques rappels sur les commandes SQL permettant l'implémentation du modèle logique relationnel dans un Système de Gestion de Bases de Données (SGBD) relationnel. Des exercices applicatifs vous sont proposés avec leur corrigé. Pour finir, vous trouverez à la fin du document le premier devoir à rendre.

Pourquoi ? Comment ?

A propos du schéma Entité/Association

Pourquoi construire un schéma Entité-Association ?

La création d'une base de données nécessite généralement une phase d'observation et de compréhension du métier qui concerne les données que l'on souhaite stocker. Un modèle conceptuel de type schéma Entité-Association est un outil pour modéliser les données à partir de l'observation d'un environnement, un méier... Il permet d'identifier les différents éléments qui composent l'environnement ainsi que les relations qui les lient entre eux. C'est une étape intermédiaire nécessaire pour ne pas attaquer directement la création de la base de données avec tous les risques d'erreurs possibles du fait du manque de recul para rapport à l'ensemble des éléments constituant l'environnement.

Comment construire un schéma Entité-Association ?

Un schéma entité-association (EA) est composé, comme son nom l'indique, d'entités et d'associations. Il existe différents formalismes pour représenter un schéma EA (cf CHEN en 1976, MERISE...) Dans le formalisme qui nous intéresse, une entité de nom E1 ayant pour attribut d'entité A1, B1, C1 où A1 identifie sans ambiguïté E1 est représentée sur la Figure 1.a. Une association R1 ayant pour attribut d'association X1 et qui permet d'associer les deux entités E1 et E2 est représentée sur la Figure 1.b. Sur la Figure 1.b , c1 et c2 représentent les cardinalités de l'association. Les valeurs possibles de c1 et c2 sont : "1..1", "0..1", "1..N", "0..N", "M..N". Le chiffre de gauche représente le nombre minimum de liens possibles et le chiffre de droite représente le nombre maximum de liens possibles. En fonction des combinaisons de valeurs de c1 et c2, l'appellation de l'association changera. On parlera d'association *one-to-one*, *one-to-many* ou *many-to-many*. Il existe aussi des associations particulières : les associations ternaires, les associations récursives et les liens d'héritage. A noter qu'un type particulier d'entité existe pour représenter des éléments qui dépendent fortement d'autres éléments : on parlera d'entité faible. Toutes ces notions sont détaillées dans les questions suivantes.

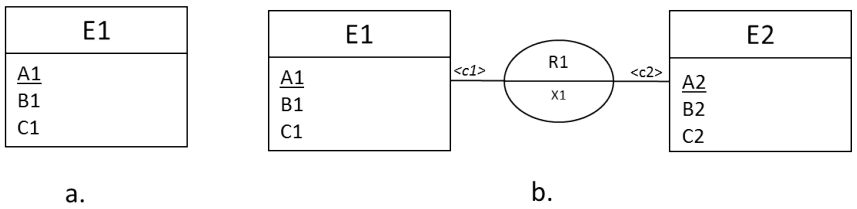
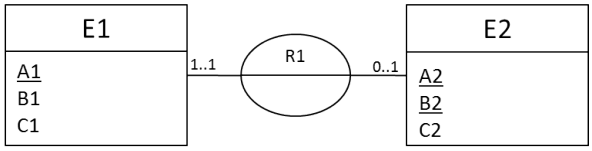


FIGURE 1 – a. une entité avec ses attributs et son identifiant. b. Une association reliant deux entités, avec un attribut d'association et des cardinalités.

Comment représente-t-on une association *one-to-one* ?

L'association *one-to-one* correspond au cas où les cardinalités c1 et c2 introduites précédemment sont "0..1" ou "1..1". L'association *one-to-one* peut se représenter de la manière suivante :



Cela signifie que :

— l'Entité E1 est liée exactement une fois à l'entité E2 par R1.

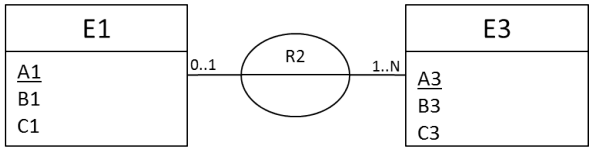
— l'Entité E2 est liée au plus une fois à l'entité E1 par R1.

Pourquoi a-t-on besoin des associations *one-to-one* ?

Une association *one-to-one* est utile pour exprimer qu'au plus un lien peut être établi entre deux éléments. Par exemple, le président d'un pays ne peut diriger qu'un seul pays et un pays ne peut être diriger que par un seul président. Une association *one-to-one* sera nécessaire pour relier l'entité Président à l'entité Pays. La cardinalité "0..1" ou "1..1" s'appliquera de chaque côté de l'association Préside.

Comment représente-t-on une association *one-to-many* ?

L'association *one-to-many* correspond au cas où les cardinalités c1 et c2 introduites précédemment sont "0..1" ou "1..1" pour c1 et "0..N" ou "1..N" pour c2 (ou inversement c1 en "0..N" ou "1..N" et c2 en "0..1" ou "1..1"). L'association *one-to-many* peut se représenter de la manière suivante :



Cela signifie que :

— l'Entité E1 est liée au plus une fois à l'entité E3 par R2.

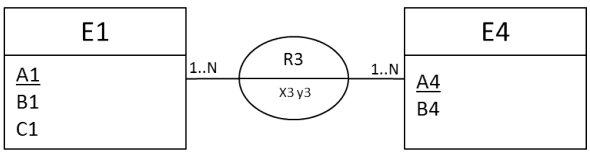
— l'Entité E3 est liée au moins une fois à l'entité E1 par R2.

Pourquoi a-t'on besoin des associations *one-to-many* ?

Une association *one-to-many* est utile pour exprimer qu'un élément peut être en lien avec plusieurs autres éléments de la même entité. Par exemple, le gérant d'un magasin peut gérer plusieurs magasins et un magasin est géré par un seul gérant. Une association *one-to-many* sera nécessaire pour relier l'entité Gérant à l'entité Magasin. La cardinalité "0..N" ou "1..N" s'appliquera à l'association Gère du côté de l'entité Gérant et la cardinalité "1..1" s'appliquera à l'association Gère du côté de l'entité Magasin.

Comment représente-t-on une association *many-to-many* ?

L'association *many-to-many* correspond au cas où les cardinalités c1 et c2 introduites précédemment sont "0..N", "1..N" ou "M..N". L'association *many-to-many* peut se représenter de la manière suivante :



Cela signifie que :

— l'Entité E1 est liée au moins une fois à l'entité E3 par R3.

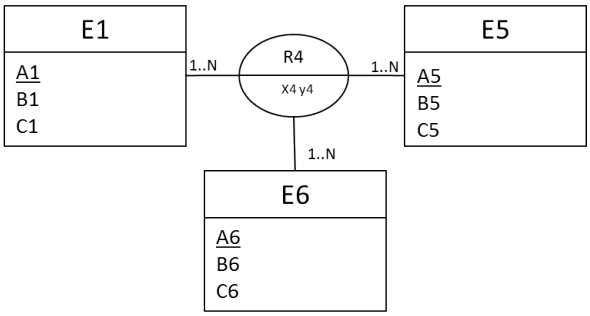
— l'Entité E3 est liée au moins une fois à l'entité E1 par R3.

Pourquoi a-t'on besoin des associations *many-to-many* ?

Une association *many-to-many* est utile pour exprimer que plusieurs élements d'une même entité peuvent être en lien avec plusieurs autres éléments d'une autre entité. Par exemple, un écrivain peut écrire plusieurs livres et un livre peut être écrit par plusieurs écrivains (co-auteurs). Une association *many-to-many* sera nécessaire pour relier l'entité Ecrivain à l'entité Livre. La cardinalité "0..N" ou "1..N" s'appliquera de chaque côté de l'association Ecrire.

Comment représente-t'on une association ternaire ?

L'association ternaire est une association qui a la particularité de relier 3 entités. L'association ternaire peut se représenter de la manière suivante :



Cela signifie que :

— l'Entité E1 est liée au moins une fois au couple (E5,E6) par R4.

— l'Entité E5 est liée au moins une fois au couple (E1,E6) par R4.

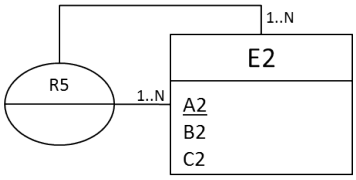
— l'Entité E6 est liée au moins une fois au couple (E1,E5) par R4.

Pourquoi a-t-on besoin des associations ternaire ?

Une association ternaire est utile pour exprimer le fait que le lien entre deux éléments ne peut se faire qu'en considérant un troisième élément. Par exemple, un client qui dispose de comptes bancaires dans différentes banques. Le lien entre le client et son compte ne peut se faire sans considérer la banque qui gère le compte. Une association ternaire sera nécessaire pour relier les entités Client, Compte et Banque.

Comment représente-t'on une association récursive ?

L'association récursive correspond au cas où les deux entités reliées par une association sont la même entité. L'association récursive peut se représenter de la manière suivante :



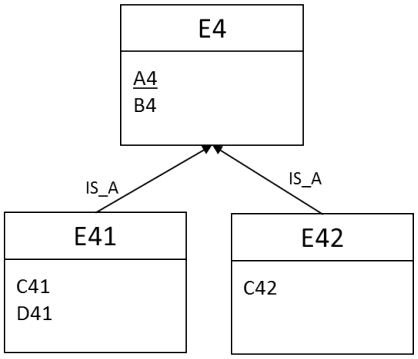
Pourquoi a-t-on besoin des associations récursives ?

Une association récursive est utile pour exprimer que des éléments représentés par une même entité sont en lien. Par exemple, un film est un remake d'un autre film. Une association récursive sera nécessaire pour relier l'entité Film à elle-même.

Comment représente-t'on un lien d'héritage (lien IS_A) entre des entités ?

L'héritage entre entités permet de regrouper au sein d'une entité mère des attributs qui sont communs à différentes entités filles. Un lien d'héritage peut se représenter de la manière suivante :

Cela signifie que :



— l'Entité E4 est l'entité mère qui factorise les attributs communs aux entités E41 et E42

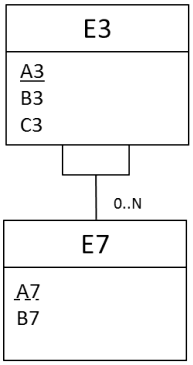
— les entités E41 et E42 sont les entités filles qui contiennent les attributs qui leur sont spécifiques.

Pourquoi a-t-on besoin des liens d'héritage ?

Les liens d'héritage sont utiles pour factoriser des attributs communs à plusieurs entités. Par exemple, une personne dont on connait le numéro de sécurité social, le nom, le prénom et l'adresse mail peut être i) un étudiant pour qui on stocke en plus l'année de première inscription ou ii) un enseignant pour qui on stocke la matière enseignée. Pour cela, on crée une entité Personne qui est l'entité mère et les entités Etudiant et Enseignant qui sont les entités filles. Les entités filles sont reliées à l'entité mère respectivement par des liens IS_A. L'entité Personne contient les attributs 'numSecu', 'nom', 'prenom' et 'email', et elle est identifié par l'attribut numSecu. L'entité Etudiant contient l'attribut 'premiereInscription'. L'entité Enseignant contient l'attribut 'matière'.

Comment représente-t-on une entité faible ?

L'entité faible est une enité qui dépend fortement d'une autre entité. L'entité faible ne peut exister sans une autre entité. De même, l'entité faible ne peut être identifiée de manière non ambigüe sans considérer l'autre entité à laquelle elle est liée L'entité faible peut se représenter de la manière suivante :



Cela signifie que :

— l'Entité E7 est une entité faible et qu'elle ne peut exister sans l'entité E3

— l'Entité E3 peut être liée plus d'une fois à l'entité E7

— l'attribut A7 (souligné par un trait en pointillé) ne suffit pas à identifier l'entité E7.

Pourquoi a-t-on besoin des entités faibles ?

Une enité faible est utile pour exprimer le lien fort entre deux entités. Un lien fort qui fait qu'une entité (l'entité faible) ne peut exister et ne peut être identifiée sans l'existance de l'autre. Par exemple, une salle est une entité faible associée à un bâtiment, car la salle ne peut exister sans le bâtiment et un numéro de salle identifie la salle de manière relative au bâtiment où elle se trouve.

A propos du passage du schéma Entité/Association au modèle relationnel

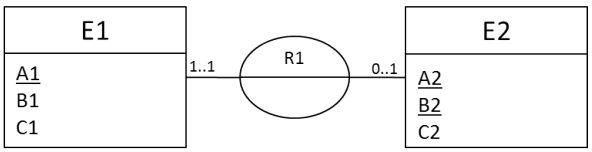
Comment convertir un schéma Entité-Association en relations ?

La transformation du schéma Entité-Association s'effectue en appliquant les règles suivantes :

1. Chaque entité devient une relation
2. Chaque identifiant d'entité devient clé primaire de la relation
   - pour les entités faibles, la clé primaire est composée de l'identifiant de l'entité faible combinée à l'identifiant de l'entité à laquelle elle est rattachée.
   - pour les entités filles reliées à une entité mère par un lien IS_A, elles héritent de l'identifiant de l'entité mère qui devient leur clé primaire.
3. Pour chaque association *one-to-one*, l'identifiant de la première entité devient clé étrangère dans la seconde entité. (Il est possible également d'ajouter l'identifiant de la seconde entité comme clé étrangère dans la première entité). Les attributs d'association sont ajoutés à côté de la clé étrangère.
4. Pour chaque association *one-to-many*, l'identifiant de l'entité à laquelle la cardinalité "0..N" ou "1..N" ou "M..N" est associée devient clé étrangère de l'autre entité (*i.e.* celle à laquelle la cardinalité "0..1" ou "1..1" est associée). Les attributs d'association sont ajoutés à côté de la clé étrangère.
5. Pour chaque association *many-to-many*, une nouvelle relation est créée. La clé primaire de cette relation est composée des identifiants des entités reliées. Les attributs d'association sont ajoutés à cette nouvelle relation.
6. Pour chaque association ternaire, (idem que pour les *many-to-many*) une nouvelle relation est créée. La clé primaire de cette relation est composée des identifiants des 3 entités reliées. Les attributs d'association sont ajoutés à cette nouvelle relation.
7. Pour chaque association récurssive, on applique la règle précédente associée qu'il s'agisse d'une *one-to-one*, *one-to-many* ou *many-to-many*. Il sera cependant nécessaire de faire un renommage pour enlever toute ambiguité sur les identifiants.

Comment transformer une association *one-to-one* ?

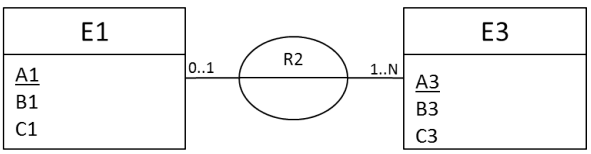
Le cas suivant dans le schéma entité-association :



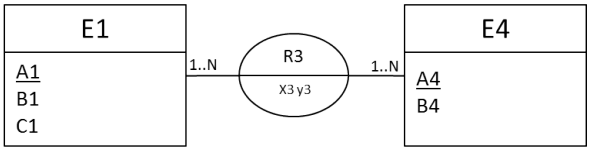
se transforme en : *E1(A1, B1, C1, #A2)*
E2(A2, B2, C2)
ou *E2(A2, B2, C2, #A1)*

Comment transformer une association *one-to-many* ?

Le cas suivant dans le schéma entité-association :



se transforme en : *E1(A1, B1, C1, #A3)*
E3(A3, B3, C3)

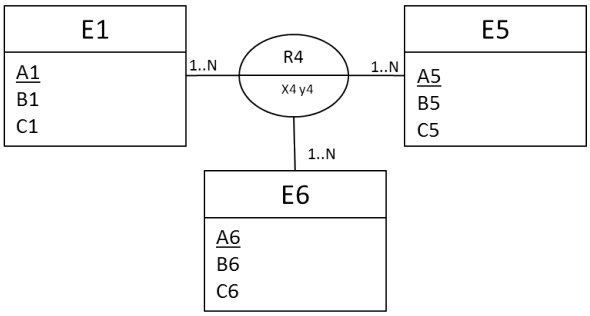


Comment transformer une association *many-to-many* ?

Le cas suivant dans le schéma entité-association :
se transforme en : *E1(A1, B1, C1)*
E4(A4, B4)
R3(#A1, #A4, X3, Y3)

Comment transformer une association ternaire ?

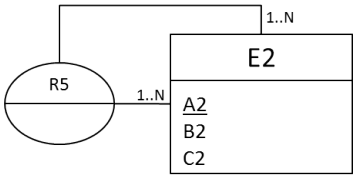
Le cas suivant dans le schéma entité-association :



se transforme en : *E1(A1, B1, C1)*
E5(A5, B5, C5)
E6(A6, B6, C6)
R4(#A1, #A5, #A6, X4, Y4)

Comment transformer une association récursive ?

Le cas suivant dans le schéma entité-association :

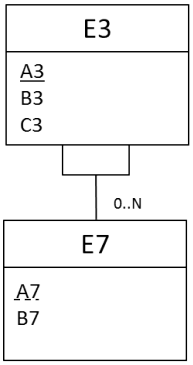


se transforme en : *E2(A2, B2, C2)*
R5(#A2, #A2')

Si l'association récursive était une *one-to-one*, on aurait eu : *E2(A2, B2, C2, #A2')*

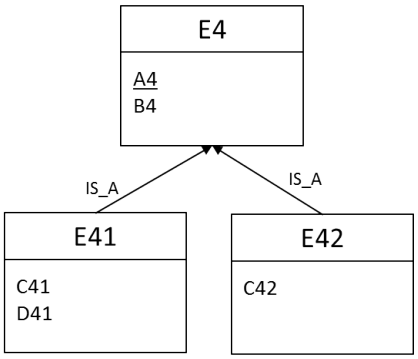
Comment transformer une entité faible ?

Le cas suivant dans le schéma entité-association :
se transforme en : *E3(A3, B3, C3)*
E7(#A3,A7, B7)



Comment transformer un lien d'héritage (lien IS_A) ?

Le cas suivant dans le schéma entité-association :



se transforme en : *E4(A4, B4)*
E41(A4, C41, D41)
E42(A4, C42)

A propos des commandes SQL pour la définition des données

Le langage SQL permet non seulement de manipuler les données mais aussi de les définir.

Remarque : Les instructions ci-dessous sont valables pour les bases de données de type MySQL.

Comment créer/modifier/supprimer une table ?

— La commande SQL pour créer la table E1 avec les attributs A1, B1, C1 et A2 respectivement de type entier, chaine de caractères, date et entier, est :

```
CREATE TABLE E1(A1 number(5), B1 varchar(50), C1 date, A2 number(5));
```

— Les commandes SQL pour modifier la structure de la table E1 sont [1] :

  — pour ajouter un attribut D1 de type chaine de caractères :

  ```
  ALTER TABLE E1 ADD D1 varchar(50);
  ```

  — pour renommer l'attribut D1 en D1' :

  ```
  ALTER TABLE E1 CHANGE D1 D1' varchar(50);
  ```

  — pour supprimer l'attribut D1 :

  ```
  ALTER TABLE E1 DROP D1;
  ```

— La commande SQL pour supprimer la table E1 est :

```
DROP TABLE E1;
```

1. pour plus d'information se référencer à la spécification du SGBD.

Comment créer/supprimer des contraintes d'intégrités ?

La création de contraintes d'intégrité (clé primaire, clé étrangère, unicité) constitue un modification du schéma de la base, il est donc nécessaire de passer par la commande **ALTER TABLE...**. Ainsi :

— pour l'ajout de la clé primaire A1 dans la table E1 :

```
ALTER TABLE E1 ADD CONSTRAINT pk_E1 PRIMARY KEY (A1);
```

NB : Par convention les noms des contraintes de clé primaire commence par *pk_*.

— pour l'ajout de la clé clé étrangère A2 qui référence E2 dans la table E1 :

```
ALTER TABLE E1 ADD CONSTRAINT fk_E1 FOREIGN KEY (A2) REFERENCES E2(A2);
```

NB : Par convention les noms des contraintes de clé étrangère commence par *fk_*.

— pour l'ajout d'une contrainte d'unicité sur l'attribut B1 dans la table E1 :

```
ALTER TABLE E1 ADD CONSTRAINT uk_B1E1 UNIQUE (B1);
```

NB : Par convention les noms des contraintes d'unicité commence par *uk_*.

— pour supprimer les contraintes précédemment créée :

```
ALTER TABLE E1 DROP CONSTRAINT pk_E1;
ALTER TABLE E1 DROP CONSTRAINT fk_E1_E2;
ALTER TABLE E1 DROP CONSTRAINT uk_B1E1;
```

NB : d'où l'intérêt d'avoir donné un nom aux contraintes.

Comment créer/modifier/supprimer un tuple (niveau instance) ?

— Pour créer le tuple (1, "TOTO", NULL, 24), il est nécessaire de l'insérer dans la table E1.

```
INSERT INTO E1 (A1, B1, C1, A2) VALUES (1, "TOTO", NULL, 24);
```

— Pour modifier le tuple (1, "TOTO", NULL, 24) en (1, "TITI", "2016-06-06", 24), il est nécessaire de mettre à jour la table E1.

```
UPDATE E1 SET B1="TITI", C1="2016-06-06" WHERE A1=1;
```

— Pour supprimer le tuple (1, "TITI", "2016-06-06", 24), il est nécessaire de l'effacer de la table E1.

```
DELETE FROM E1 WHERE A1=1;
```

Exercices applicatifs

Exercice 1 : Rétro-ingénierie

On souhaite modéliser et interroger une base de données sur le cinéma. Le schéma de la base de données relationnelle est le suivant :

Acteur(idAc, nom, prenom, dNaissance, dDeces, #idAg, depuis)
Agent(idAg, nom, prenom, telephone)
Realisateur(idRe, nom, prenom, dNaissance)
Personnage(nom, prenom, age, sexe)
Film(idF, titre, anneeSortie, duree, #idRe)
JoueRole(#idF, #idAc, #nom, #prenom, #age)
EstEnLien(#idF1, #idF2, typeLien)
Cinema(idC, nom, ville, adresse)
Salle(#idC, numero, capacite, climatise, 3Dpossible)
EstProjete(#idF, #idC, #numero, dateP, heureP, 3D, nbBillets)
Producteur(idP, nom, pays)
Produit(#idP, #idF, invest)

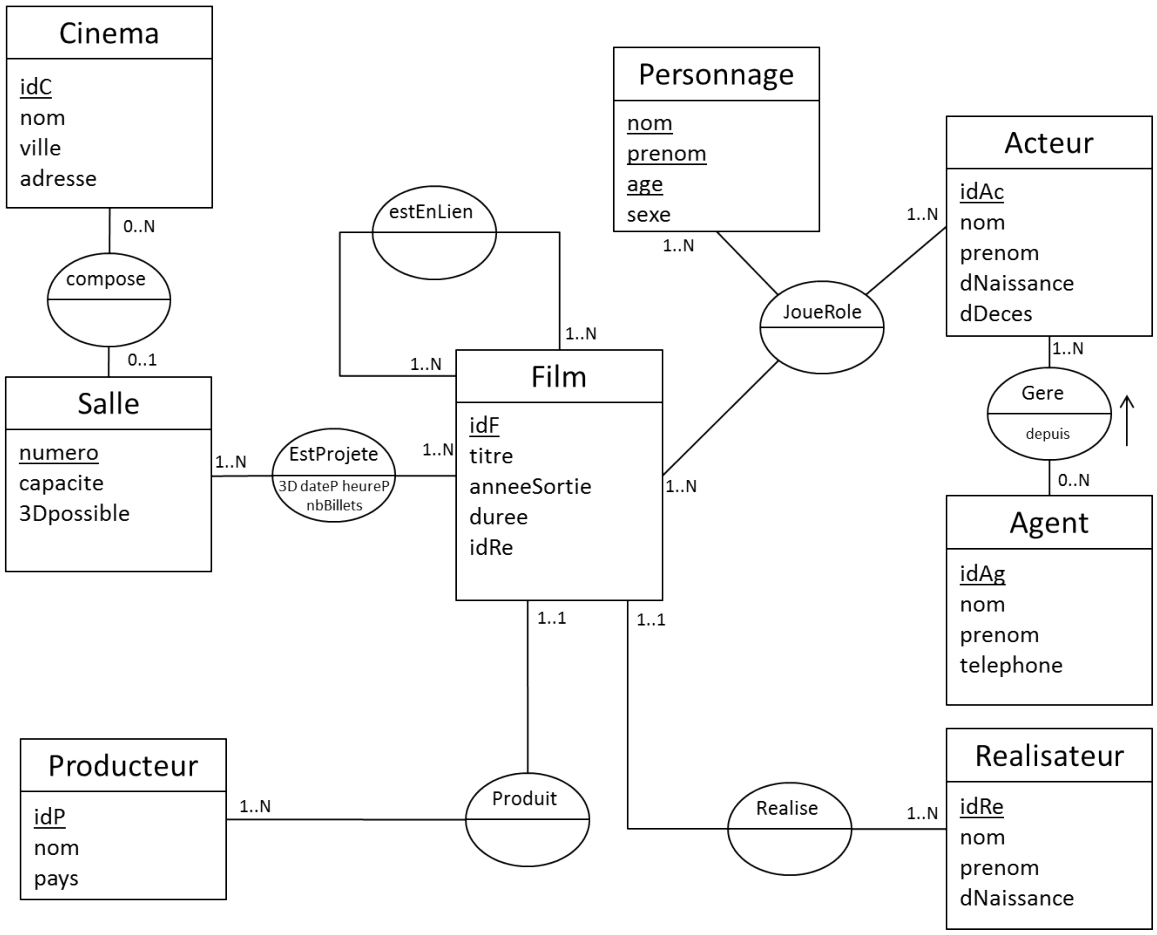
Un **acteur** est identifié par un entier 'idAc' et on dispose de son nom, prénom, date de naissance et de sa date de décès s'il est décédé. On dispose également de l'entier 'idAg' qui identifie son agent. L'attribut 'depuis' indique depuis quand 'idAg' est l'agent de 'idAc'. Un **agent** est identifié par un entier 'idAg' et on dispose de son nom, prénom et numéro de téléphone. Un **réalisateur** est identifié par un entier 'idRe' et on dispose de son nom, prénom et date de naissance. Deux artistes (acteur ou réalisateur) différents ne peuvent pas avoir à la fois le même nom, prénom et date de naissance. Un **personnage** est identifié par son nom, son prénom, sa classe d'âge (enfant, jeune, adulte, sénior) et son sexe ('H', 'F'). Un **film** est identifié par un entier 'idF', on dispose de son titre, de son année de sortie, de sa durée exprimée en minutes et de l'identifiant de son réalisateur. Dans **JoueRole**, on précise que l'acteur 'idAc' joue le personnage ('nom','prenom','age') dans le film 'idF'. Dans **EstEnLien**, on précise que le film 'idF1' est lié au film 'idF2'. Ce lien peut être de type ('typeLien') : suite, parodie, remake, reboot, prequel. Un **cinéma** est identifié par un entier 'idC' et on stocke son nom, la ville où il se trouve et son adresse. Une **salle** est identifiée par le couple (identifiant de cinéma,numéro de salle) et on stocke sa capacité en termes de nombre de places. On stocke également le fait que la salle est climatisée ou pas ('oui' ou 'non') et que l'on puisse projeter des films en 3D ou pas ('oui' ou 'non'). Dans **EstProjete**, on précise que le film 'idF' est projeté dans la salle ('idC','numero') à la date 'dateP' et à l'horaire 'heureP'. On précise également si la projection a été faite en 3D ('oui' ou 'non') et le nombre de billets vendus pour cette projection. Un **producteur** est identifié par un entier 'idP' et on stocke son nom et le code du pays où est implanté son siège. Dans **Produit**, on précise que le producteur 'idP' a produit le film 'idF' avec un investissement de 'invest' euros.

Exemple de tuples :

- Pour Acteur : (123, 'DUPOTAGER', 'Jean', '1972-06-19', NULL, 456, 2004) ;
- Pour Agent : (456, 'DIPOURCENT', 'Ella', '555-555')
- Pour Realisateur : (68, 'TARTARINO', 'Justin', '1963-03-27') ;
- Pour Personnage : ('BOND', 'James', 'adulte', 'H') ;
- Pour Film : (14, 'Brice de Nice', 2005, 98, 145) ;
- Pour JoueRole : (10, 45, 'SKYWALKER', 'Anakin', 'enfant') ;
- Pour EstEnLien : (103, 11, 'remake') ; /* le film 103 est un remake du film 11 */
- Pour Cinema : (85, 'ZOLA', 'Villeurbanne', '117 Cours Emile Zola') ;
- Pour Salle : (75, 1, 120, 'oui', 'non') ;

— Pour EstProjete : (257, 74, 4, '2015-06-02', '20 :25', 'oui', 120) ;
— Pour Producteur : (5, 'EuropaCorp', 'FR') ;
— Pour Produit : (3, 14, 5 340 000) ;

Le concepteur de la base de données explique que pour concevoir la base de données CINEMA, il a construit le schéma entité association représenté sur la figure suivante :



1. Des incohérences apparaissent entre ce schéma Entité/Association et le modèle relationnel fourni précédemment. Indiquez ci-dessous 6 erreurs dans le schéma entité/association et la modification qu'il serait nécessaire d'appliquer pour rendre le schéma Entité/Association cohérent avec le modèle relationnel (qui lui est correct).

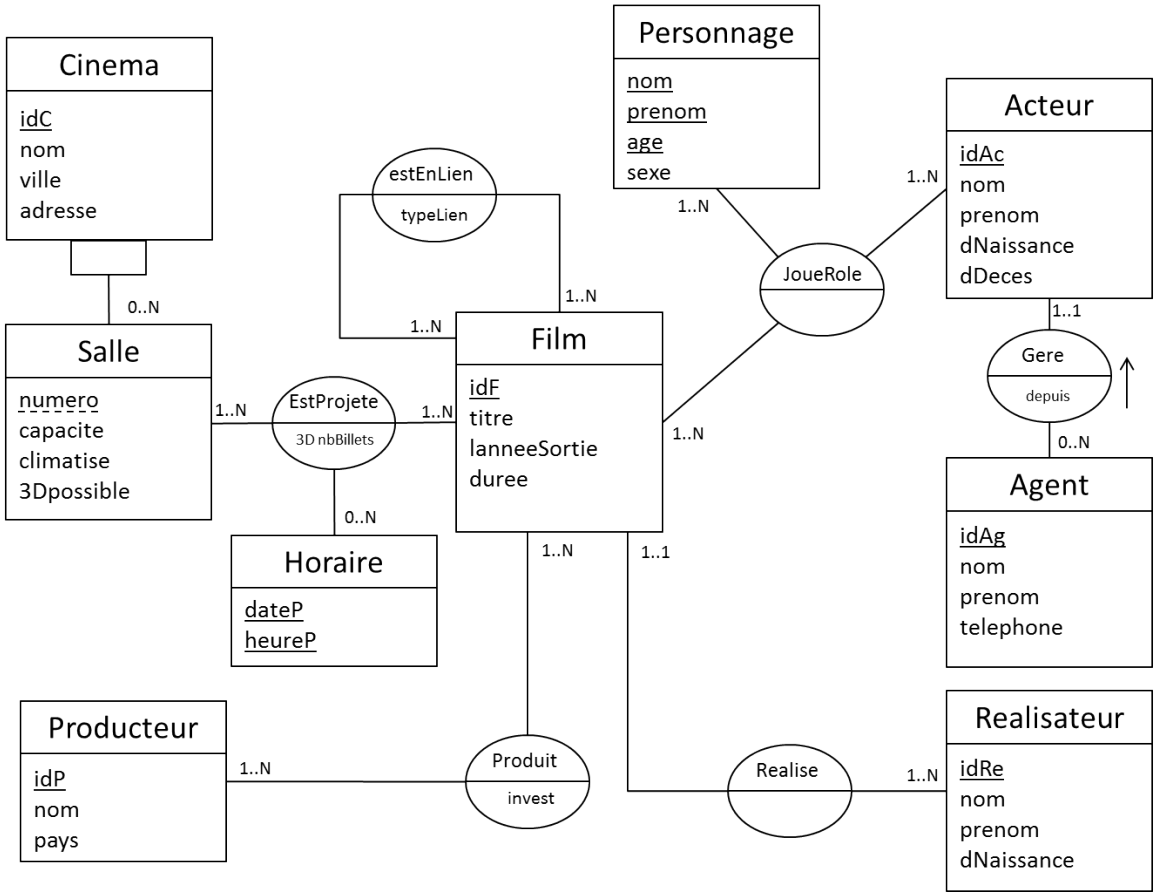
2. Donner les commandes SQL permettant de construire la base de données sur le cinéma.

Corrections

Question 1

- il manque l'attribut climatise dans l'entité Salle.Ajouer l'attribut 'climatise' dans l'entité 'salle'
- l'association entre Salle et Cinema n'est pas correcte. Il faudrait que Salle soit une entité faible rattachée à Cinema
- Il manque un attribut d'association 'typeLien' dans l'association 'EstEnLien'
- L'association 'Produit' entre Producteur et Film devrait être une many-to-many et non une one-to-many
- L'association 'Gere' entre Acteur et Agent devrait être une one-to-many et non une many-to-many
- L'association 'EstProjete' entre Salle et Film devrait être une ternaire relié à une entité Horaire ayant pour identifiant le couple (date, heure)
- L'attribut 'idRe' n'a rien à faire dans l'entité Film.

Le schéma E/A cohérent serait alors :



Question 2

```
CREATE TABLE Agent(idAg int(5), nom varchar(50), prenom varchar(50),
telephone varchar(10) );
ALTER TABLE Agent ADD CONSTRAINT pk_agent PRIMARY KEY (idAg);

CREATE TABLE Acteur(idAc int(5), nom varchar(50), prenom varchar(50),
dNaissance date(), dDeces date(), idAg int(5), depuis date() );
ALTER TABLE Acteur ADD CONSTRAINT pk_acteur PRIMARY KEY (idAc);
ALTER TABLE Acteur ADD CONSTRAINT fk_acteur_agent FOREIGN KEY
(idAg) REFERENCES Agent(idAg);

CREATE TABLE Realisateur(idRe int(5), nom varchar(50), prenom varchar(50),
```

```
dNaissance date() );

CREATE TABLE Personnage(nom varchar(50), prenom varchar(50), age varchar(50),
sexe char(1) );
ALTER TABLE Personnage ADD CONSTRAINT pk_perso PRIMARY KEY (nom, prenom, age);

CREATE TABLE Film(idF int(5), titre varchar(50), anneeSortie int(4),
duree int(3), idRe int(5) );
ALTER TABLE Film ADD CONSTRAINT pk_film PRIMARY KEY (idF);
ALTER TABLE Film ADD CONSTRAINT fk_film_realisateur FOREIGN KEY
(idRe) REFERENCES Realisateur(idRe);


CREATE TABLE JoueRole(idF int(5), idAc int(5), nom varchar(50),
 prenom varchar(50), age varchar(50) );

ALTER TABLE JoueRole ADD CONSTRAINT pk_joueRole
PRIMARY KEY (idF,idAc,nom, prenom, age);
ALTER TABLE JoueRole ADD CONSTRAINT fk_joueR_film FOREIGN KEY
(idF) REFERENCES Film(idF);
ALTER TABLE JoueRole ADD CONSTRAINT fk_joueR_acteur FOREIGN KEY
(idAc) REFERENCES Acteur(idAc);
ALTER TABLE JoueRole ADD CONSTRAINT fk_joueR_perso FOREIGN KEY
(nom, prenom, age) REFERENCES Personnage(nom, prenom, age);

CREATE TABLE EstEnLien(idF1 int(5), idF2 int(5), typeLien varchar(50) );
ALTER TABLE EstEnLien ADD CONSTRAINT pk_estenlien PRIMARY KEY (idF1, idF2);
ALTER TABLE EstEnLien ADD CONSTRAINT fk_estenlien_film1 FOREIGN KEY
(idF1) REFERENCES Film(idF);
ALTER TABLE EstEnLien ADD CONSTRAINT fk_estenlien_film2 FOREIGN KEY
(idF2) REFERENCES Film(idF);


CREATE TABLE Cinema(idC int(5), nom varchar(50), ville varchar(50),
 adresse varchar(50) );
ALTER TABLE Cinema ADD CONSTRAINT pk_cinema PRIMARY KEY (idC);

CREATE TABLE Salle(idC int(5), numero int(2) , capacite int(3), climatise char(3),
3Dpossible char(3) );
ALTER TABLE Salle ADD CONSTRAINT pk_salle PRIMARY KEY (idC, numero);
ALTER TABLE Salle ADD CONSTRAINT fk_salle_cinema FOREIGN KEY
(idC) REFERENCES Cinema(idC);

CREATE TABLE EstProjete(idF int(5), idC int(5), numero int(2), dateP date(),
heureP timestamp(), 3D char(3), nbBillets int(3) );
ALTER TABLE EstProjete ADD CONSTRAINT pk_estprojete PRIMARY KEY (idF, idC, numero, da
ALTER TABLE EstProjete ADD CONSTRAINT fk_estprojete_film FOREIGN KEY
(idF) REFERENCES Film(idF);
ALTER TABLE EstProjete ADD CONSTRAINT fk_estprojete_salle FOREIGN KEY
(idC, numero) REFERENCES Salle(idC, numero);

CREATE TABLE Producteur(idP int(5), nom varchar(50), pays varchar(50));
ALTER TABLE Producteur ADD CONSTRAINT pk_producteur PRIMARY KEY (idP);

CREATE TABLE Produit(idP int(5), idF int(5), invest int(10));
ALTER TABLE Producteur ADD CONSTRAINT pk_produit PRIMARY KEY (idP,idF);
```

```
ALTER TABLE Produit ADD CONSTRAINT fk_produit_producteur FOREIGN KEY
(idP) REFERENCES Producteur(idP);
ALTER TABLE Produit ADD CONSTRAINT fk_produit_film FOREIGN KEY
(idF) REFERENCES Film(idF);
```