Introduction aux Bases de données

Cours 1 : Introduction–Conception

UFR 919 – Licence informatique

LU2IN009

Informations pratiques

Responsable de l'UE :

- Camelia Constantin

Chargés de cours :

- Mercredi 8h45 : Mohamed-Amine Baazizi
- Jeudi 14h00 : Camelia Constantin

Informations supplémentaires :

Moodle : Introduction aux bases de données relationnelles - S2

Bibliographie conseillée

Notes de cours

- S. Gancarski. Introduction aux bases de données. UPMC, Paris 6, janvier 2003 – lien sur le site de l'UE

Livres en anglais

- R. Ramakrishnan and J. Gehrke. Database Management Systems 3e édition, McGraw Hill, 2002 -http://pages.cs.wisc.edu/~dbbook/ (*Disponible bib. MIR et MIE)*
- A. Silberschatz, HF. Korth and S. Sudarshan. Database System concepts 6e édition, McGraw Hill, 2011-http://db-book.com/

Livres en français :

- S. Abiteboul, R. Hull, V. Vianu, Les fondements des bases de données, Vuibert (*Disponible bib. MIR et MIE)*
- G. Gardarin. Bases de données - objet et relationnel. Eyrolles. (*Disponible bib.* L1-L2 scientifique *et MIE)*

Aperçu sur les bases de données

Qu'est ce qu'une base de données ?

- Collection de données structurées suivant la réalité modélisée

Où trouve-t-on des bases de données ? Exemple:

- web: sites marchands, réseaux sociaux, ...
- finance: applications financières, gestion de comptes, ...
- économie : e-commerce (amazon), services de ventes/achats, ...
- industrie: gestion de centrales nucléaires, chaînes de production, ...
- transports: réservation de billets, gestion de trains/avions, ...
- science: données d'expérimentation,...
- services publiques: impôts, police, open-data, …

Quels sont les types de Bases de Données ?

- Les BD relationnelles (prédominantes : données de gestion)
- Les BD objet, XML, JSON, RDF (orientées web, données techniques)

SGBD* vs système de fichiers

Accès aux données
- écrire un programme dédié à chaque tache

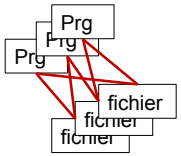
Redondance des données
- la même donnée stockée dans plusieurs fichiers

Cohérence des données
- difficulté d'exprimer et de garantir des contraintes d'intégrité

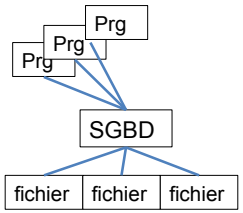
Performance d'accès
- les données souvent volumineuses, plusieurs usagers

Concurrence d'accès
- interaction entre plusieurs programmes

Sécurité et protection des données
- données de sensibilités différentes

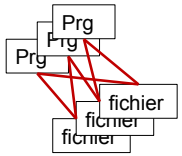


(*) Système de Gestion de Bases de Données
Database Management System (DBMS)

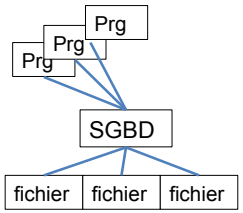


SGBD* vs système de fichiers

SGBD :
- Contient les BDs ainsi que leur description et contraintes (méta-données stockes dans le catalogue du SGBD)
- Centralisation de l'information, représentation de relations complexes entre les données, permet de trouver et de mettre à jour efficacement des données reliées
- Contrôle de la redondance des données
- Contrôle des niveaux d'accès aux données (sécurité des données)
- Stockage persistant pour les données des applications, structures de stockage et interrogation efficace des données
- Abstraction des données, isolation entre les données et les traitements
- Plusieurs vues des mêmes données, interfaces utilisateur multiples
- Accès simultané aux données centralisées par plusieurs applications, contrôle de concurrence, partage de données
- Définition de contraintes d'intégrité et automatisation de leur maintenance (triggers)

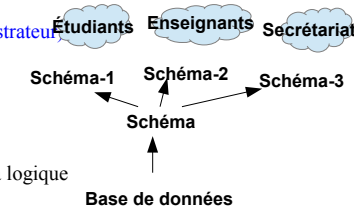


(*) Système de Gestion de Bases de Données
Database Management System (DBMS)



L'architecture à 3 niveaux

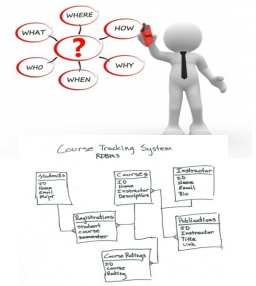
Niveaux d'abstraction
- Niveau des vues (accessible à l'utilisateur)
  Quelles données peut-on voir → schéma externe
- Niveau logique (accessible au concepteur/programmeur)
  Quelles données sont stockées → schéma logique
- Niveau physique (accessible au concepteur et à l'administrateur)
  Comment les données sont stockées → schéma physique



Indépendance entre les niveaux
- Indépendance physique des données
  le changement du schéma physique n'affecte pas le schéma logique
- Indépendance logique
  le changement du schéma logique n'affecte pas le schéma externe

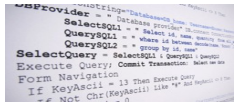
Comment construire une base de données

1 Analyse des besoins
observer le monde réel
identifier les informations pertinentes

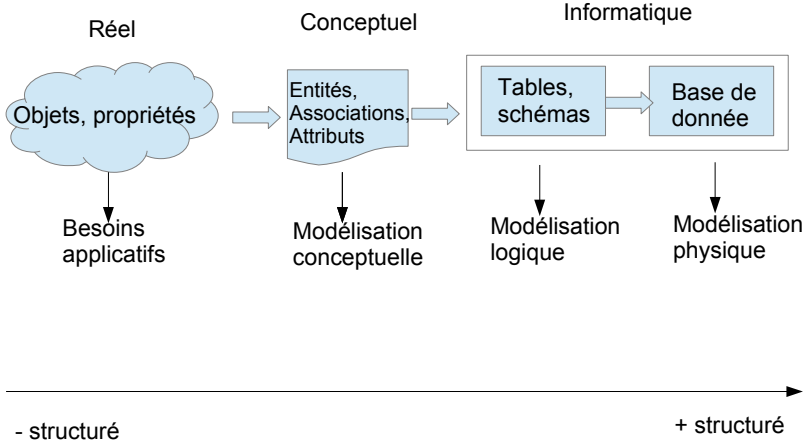


2 Modélisation des données
formaliser les besoins

3 Implantation des données et des procédures d'interrogation et de mise à jour
BD proprement parlé



Construction d'une base de données



Quels outils ?

1. Analyse des besoins :
   - discussion 'informelle' qui découle sur une documentation technique
2. Modélisation des données :
   - traduction des besoins en des concepts de base : entités et lien entre elles (modèle Entite-Association)
3. Implantation des données :
   - langages compréhensibles par la machine
   - SQL (*Structured Query Language)* est le standard pour les données relationnelles

Modèles de Données

Modèle entité-association

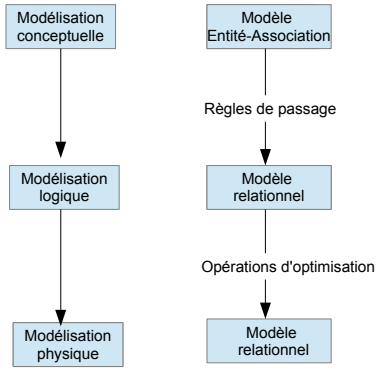
- Description haut niveau des données
- entité et les liens ente elles (associations)

⇨concepteur, client

Modèles relationnel

- Représentation logique des données
- Concept de tables

⇨ concepteur, développeur



Langages pour les BD

Langage déclaratif (se concentrer sur la logique de l'application, pas sur l'accès aux données)

- Quelles données retourner, supprimer, modifier?
- Le système génère un programme « optimisé » pour réaliser la tâche demandée
- Langage de définition des données (*Data Definition Language*)
  - Création des tables
  - Définition des contraintes
- Langage de manipulation des données (*Data Manipulation Language*)
  - Interrogation des données
  - Insertions, modification et suppression données

But du cours

Introduction aux Bases de données relationnelles sous un angle applicatif

- Modélisation et représentation des données
- Implantation et manipulation des données
- Programmation en lien avec les données

Aperçu des cours liés

- LUIN3009 : mise en œuvre et optimisation dans les systèmes BD
- MLBDA (M1) : modèles et langages pour les BD
- BDR (M1) : BD réparties
- BLDE (M2) : Big Data

Plan du cours

- Conception d'un schéma de Bases de Données

  Modèle Entité-Association
- Interrogation des données

  Langage de requêtes : Calcul relationnel et SQL
- Saisie, modification et cohérence des données

  Langage de Définition de Données (DDL)

  Langage de Manipulation de Données (DML)
- Manipulation complexe des données

  Programmation en PL/SQL, Déclencheurs (triggers)

Conception d'un schéma Entité-Association

Démarche

Analyse des besoins

- Discussion **informelle** avec les futurs utilisateurs
- Identifier les objets du monde réel et des liens entre eux
- Identifier les opérations sur ces objets et les éventuelles évolutions

→ Document technique décrivant les données de l'application

Etablissement du schéma conceptuel

- Langage de **haut niveau** (ex. le modèle entité-association)
- Décider des données devant être stockées, de leur propriétés et des relations entre elles
- Définir les contraintes à respecter

→ Schéma dans un langage de haut niveau (Entité-Association)

Etude de cas : BD d'une université

Les besoins :

- Gérer les inscriptions des étudiants à des modules
- Gérer l'affectation des tuteurs à des étudiants
- Gérer le planning des salles

Les objets à modéliser :

- Les étudiants Les modules
- Les tuteurs
- Les salles

Les liens entre les objets (scénario) :

- Les étudiants s'inscrivent à un ou plusieurs modules pour une année universitaire
- Le cours d'un module a lieu dans une salle donnée ; il débute à une heure connue et se déroule pendant une durée connue.

Le modèle Entité-Association

- Proposé par Peter Chen en 1976
- **Principe** : Transcrire les besoins en terme de *classes* d'*entités* et de *classes* d'*associations*
- Les *entités* = les objets du monde réel
- Une classe d'entités = ensemble d'entités possédant les mêmes propriétés
- Les *associations* = les relations liant les entités
- Une classe associations = ensemble d'associations reliant des entités de la même classe
- Les *attributs* = les propriétés qui renseignent certaines informations sur une entité ou une associatio

Entité

- Tout objet du monde réel pertinent pour l'application, peut être concret ou abstrait.
- Exemple d'entités concrètes :
  - Le médecin "Anne DUPONT" est une entité
  - La salle 24-34/208 est une entité
  - L'enseignement "2I009" est une entité
- Exemple d'entités abstraites (qui ne correspondent pas à des objets physiques) :
  - Le virement n° XXX ayant eu lieu le 20/01/19 est une entité
  - Le compte en banque d'un client est une entité
  - Un contrat d'assurance est une entité

Classe d'entités

- Permet de décrire un ensemble d'entités de même type (ayant les mêmes caractéristiques)
- Exemple de classes d'entités :
  - La classe *Étudiant* décrit l'ensemble des étudiants de l'université (tout étudiant a un nom, prénom, adresse, matricule)
  - La classe *Module* décrit toutes les modules de l'université (tout module a un code, un intitulé et un niveau)
  - La classe *Médecin* décrit tous les médecins de l'application (tout médecin a un matricule et un nom)
  - etc

Attribut

- Modélise une propriété/caractéristique d'une classe d'entités
- Possède un nom et un domaine de valeurs (=ensemble de valeurs permises)
- Est *atomique* (ne peut pas être multivalué)
- Un attribut prend une seule valeur à la fois pour chaque entité
- Exemple d'attributs :
  - Le nom, le prénom, l'adresse et le matricule sont des attributs de la classe *Étudiant*
  - Le nom est une chaîne de 1 à 20 caractères et l'attribut correspondant est Nom.
  - Le matricule est un nombre entier avec 4 chiffres
  - L'attribut prénom *ne peut pas contenir plusieurs chaînes de caractères* (e.g "Jean", "Louis") mais une seule chaîne de caractères (e.g "Jean, Louis")
  - Le nom de l'étudiant de matricule 1234 (=une entité) ne peut pas être à la fois "Dupont" et "Martin"

Identifiants

- Un sous ensemble d'attributs d'une classe d'entités
- Permet de distinguer les entités de la même classe (deux entités de la même classe ne peuvent pas avoir le même identifiant)
- Toute classe d'entités doit avoir au moins un identifiant (si plusieurs identifiants potentiels → en choisir un seul)
- Un identifiant peut être :
  - Naturel (construit à partir des attributs de la classe)
  - Artificiel (rajouté aux attributs de la classe, lorsque les attributs de la classe ne permettent pas de définir un identifiant)

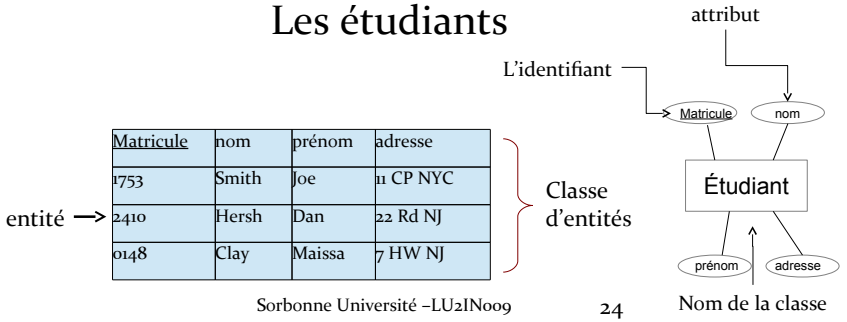
Exemples d'identifiants

- Le matricule de l'étudiant permet de distinguer l'ensemble d'étudiants de l'université → peut être identifiant de la classe *Étudiant*
- Pour une salle d'enseignement, son numéro (e.g 207) ne permet pas de la distinguer des autres salles de l'université (il peut y avoir une autre salle 207), on a besoin aussi de connaître son emplacement (e.g 24-34), donc le couple (numéro, emplacement) (e.g (207, 24-34)) sera l'identifiant de la classe *Salle*
- Si chaque étudiant a un nom différent des autres étudiants ainsi qu'un matricule différent des autres étudiants, choisir un des deux (e.g matricule) comme identifiant de la classe *Étudiant*
- Si plusieurs enseignants peuvent avoir le même nom et le même prénom (le couple nom, prénom ne peut pas jouer le rôle d'identifiant), alors ajouter un attribut artificiel id à la classe *Enseignant* qui sera désigné comme identifiant.

Etude de cas : BD d'une université

Classe d'entités

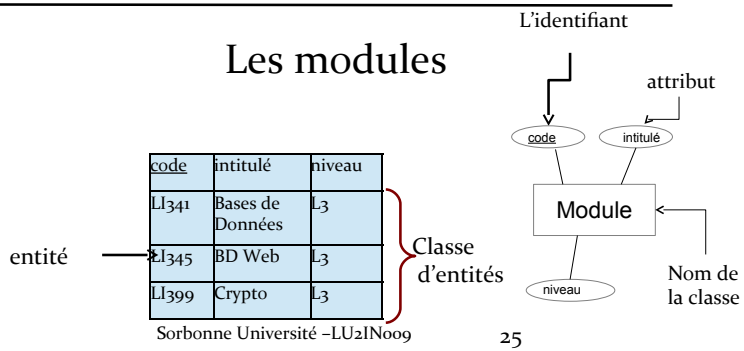
ensemble d'entités possédant les mêmes propriétés



Etude de cas : BD d'une université

Classe d'entités

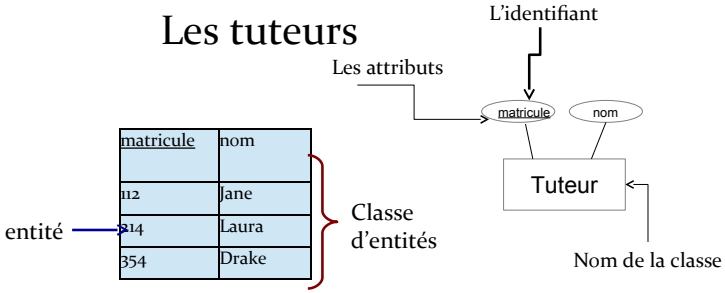
ensemble d'entités possédant les mêmes propriétés



Etude de cas : BD d'une université

Classe d'entités

ensemble d'entités possédant les mêmes propriétés

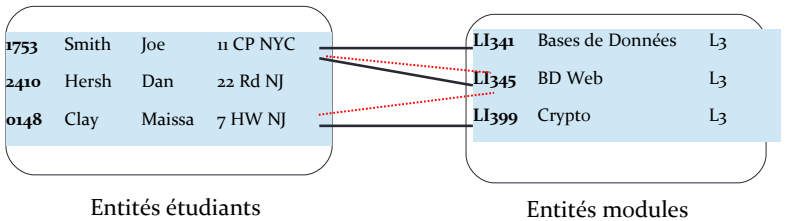


Etude de cas : BD d'une université

Association

- Lien entre entités (relation entre deux ensembles)
- Il est possible d'avoir plusieurs associations entre les mêmes entités avec des sémantiques différentes

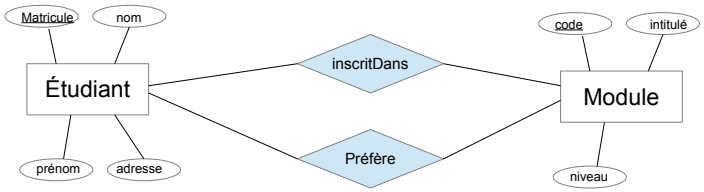
« Etudiant *inscritDans* Module » et « Etudiant prefère Module »



Classe d'associations

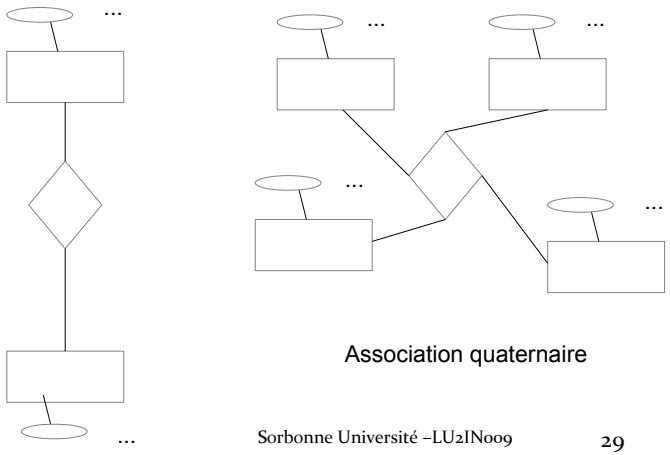
- L'ensemble d'associations de même type (qui relient les mêmes ensembles d'entités et qui ont la même sémantique)
- Désignée généralement par un verbe

- Exemple (représentation graphique) :



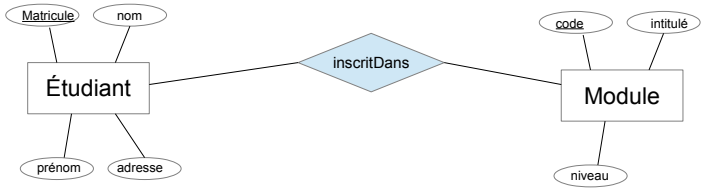
Types de classes d'association

Associations binaires (relient deux entités)
Associations N-aires (relient plus de deux entités)



Association quaternaire

Exemple d'associations appartenant à une classe



1753	Smith	Joe	11 CP NYC
2410	Hersh	Dan	22 Rd NJ
0148	Clay	Maissa	7 HW NJ

1753, LI341
1753, LI345
0148, LI399

LI341	Bases de Données	L3
LI345	BD Web	L3
LI399	Crypto	L3

Entités étudiants

Entités modules

L'identifiant d'une association

Une association est identifiée au moyen des identifiants des entités qu'elle met en relation [7] on ne peut pas associer les mêmes entités plusieurs fois avec des associations de même sémantique
Exemple: l'étudiant 1753 ne peut pas s'inscrire plusieurs fois au module LI341

1753	Smith	Joe	11 CP NYC
2410	Hersh	Dan	22 Rd NJ
0148	Clay	Maissa	7 HW NJ

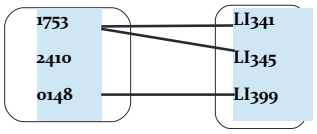
1753, LI341
1753, LI345
0148, LI399

LI341	Bases de Données	L3
LI345	BD Web	L3
LI399	Crypto	L3

Entités étudiants

sera notée

Entités modules



étudiants modules

ou

étudiants	modules
1753	LI341
1753	LI345
O148	LI399

Etude de cas : BD d'une université

Attributs d'une association

- Les identifiants des entités qu'elle relie, plus éventuellement d'autres attributs

Exemple: Etudiant *inscritDans* Module pour une année

Les attributs d'associations

1753	Smith	Joe	11 CP NYC
2410	Hersh	Dan	22 Rd NJ
0148	Clay	Maissa	7 HW NJ

1753, LI341, 2012
1753, LI345, 2012
0148, LI399, 2013

LI341	Bases de Données	L3
LI345	BD Web	L3
LI399	Crypto	L3

Les étudiants

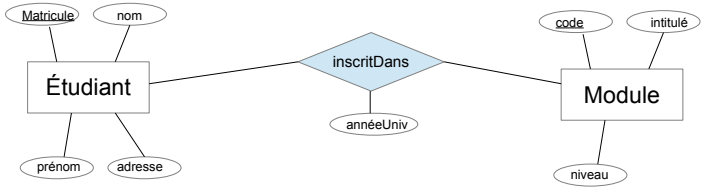
Les modules

Attributs d'une classe d'associations

Identifiant: L'ensemble des identifiants des classes d'entités qu'elle relie

Autres attributs: l'ensemble des valeurs des attributs des associations de cette classe

Exemple: Etudiant *inscritDans* Module pour une année



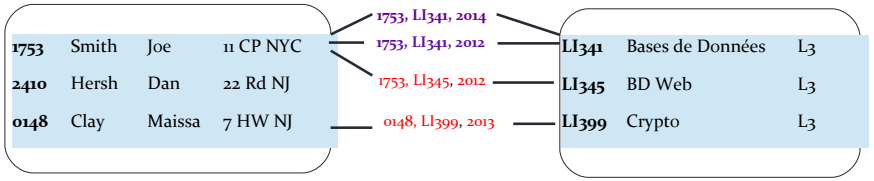
L'identifiant de la classe *inscritDans* est {Matricule, code}

Identifiant d'une classe d'associations

On souhaite qu'un étudiant puisse s'inscrire à un module plusieurs fois, à des années différentes

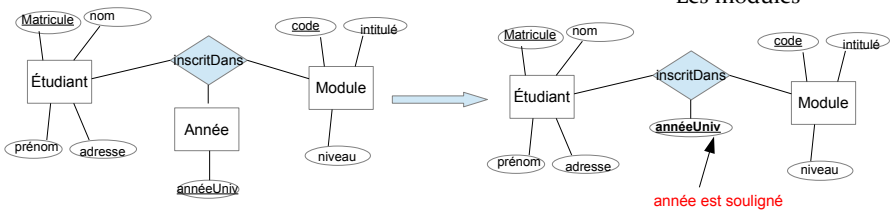
→ Problème : l'identifiant d'une association est composé des identifiants des entités reliées

→ Solution : intégrer l'année dans l'identifiant de l'association



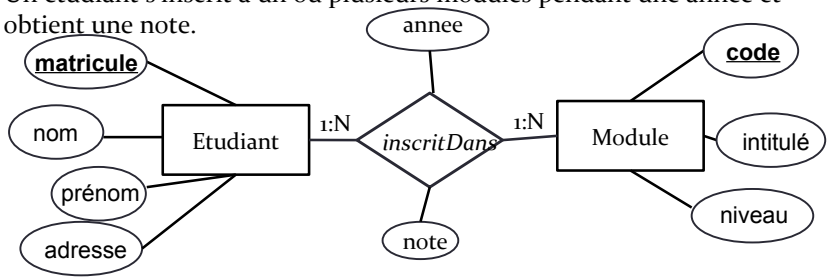
Les étudiants

Les modules



Attribut d'une classe d'association

Un étudiant s'inscrit à un ou plusieurs modules pendant une année et obtient une note.



etudiants	modules	Annee	note
1753	LI341	2012	9
0148	LI345	2012	13

Identifiant d'une classe d'associations

Un étudiant s'inscrit à un ou plusieurs modules pendant une année ou plusieurs années et obtient à chaque inscription une note.

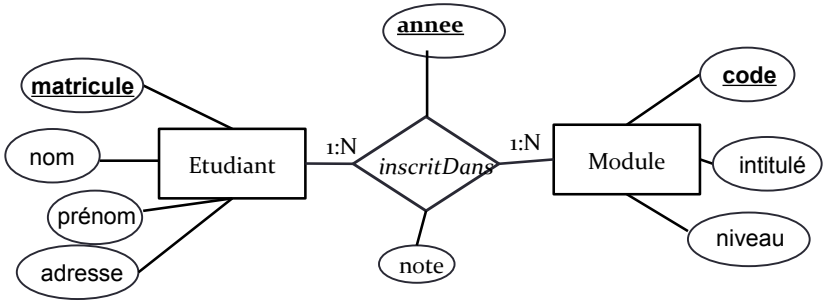
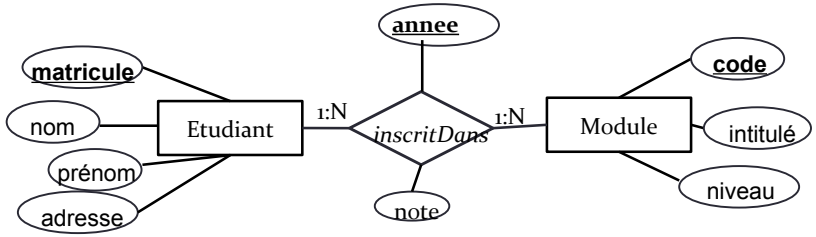


Illustration sur une instance



etudiants	modules	Annee	note
1753	LI341	2012	9
1753	LI341	2013	10.5
0148	LI345	2012	13

Simplification des appellations

Omission du terme « classe »

- Entité sous-entend classe d'entités
- Association sous entend classe d'associations
- Au niveau E/A, on se préoccupe seulement des classes, pas des instances particulières

La cardinalité d'une association

Nombre d'entités liées par les associations

Intervalle de valeurs [m:n]

Description	Entités	cardinalité
Un module n'ouvre qui s'il y a au moins un étudiant d'inscrit	Module Etudiant	1:N
Un module a lieu dans la même salle	Module Salle	1:1

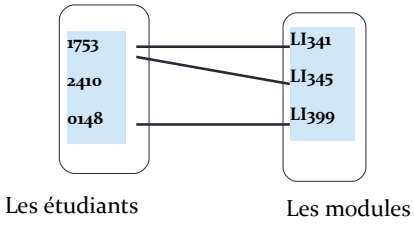
Etude de cas : BD d'une université

Les cardinalités d'associations

- Etudiant *inscritDans* Module
  - Un étudiant DOIT s'inscrire dans au moins <u>un</u> module
  - Un module n'ouvre que s'il y a au moins <u>un</u> étudiant

---

L'association « Etudiant *inscritDans* Module »



etudiants	modules
1753	LI341
1753	LI345
2401	
0148	LI399

Quel est le problème?

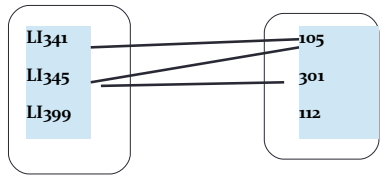
Etude de cas : BD d'une université

Les cardinalités d'associations

- Module *aLieuDans* Salle
  - Un module a lieu dans une et une seule salle
  - Une salle peut être utilisée pour plusieurs modules ou rester inoccupée

---

L'association « Module *aLieuDans* Salle »



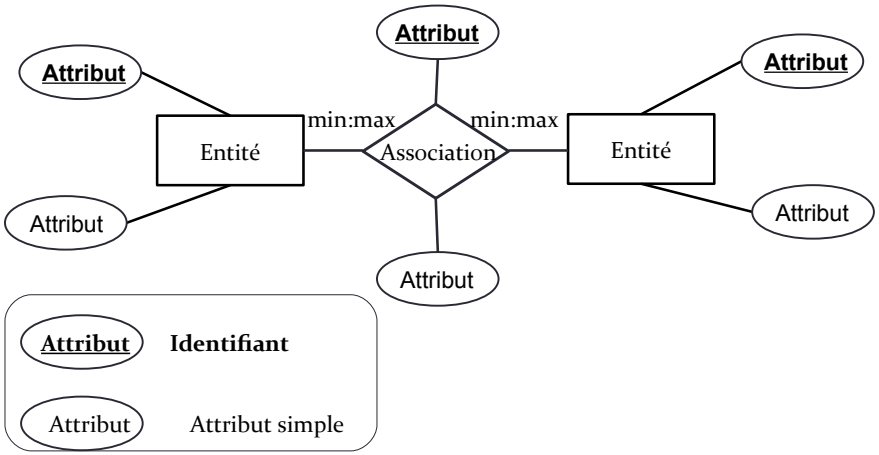
Les modules

Les salles

modules	salles
LI341	105
LI345	105
LI399	301
	112

Quel est le problème?

Modèle E-A : Notation graphique



Etude de cas : BD d'une université

Un étudiant DOIT s'inscrire à au moins un module	?
Un module n'ouvre que s'il y a au moins un étudiant d'inscrit	?

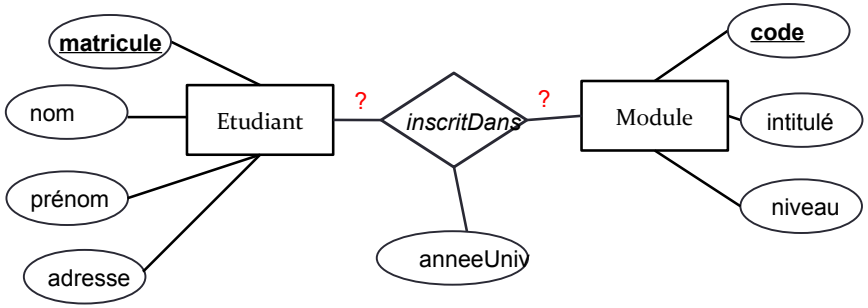
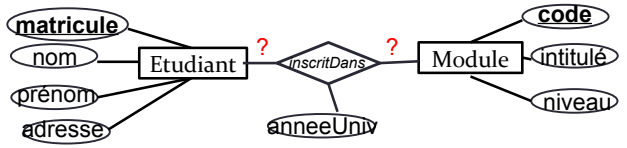
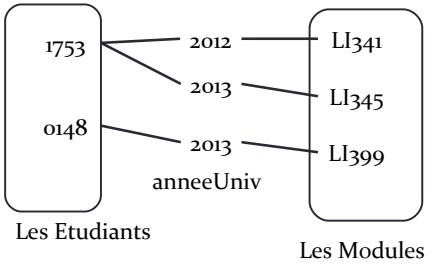


Illustration sur une instance





Les Etudiants

Les Modules

etudiants	modules	anneeUniv
1753	LI341	2012
1753	LI345	2013
0148	LI399	2013

Etude de cas : BD d'une faculté

Un module a lieu dans une salle et une seule	?
Un salle peut être utilisée pour plusieurs modules ou rester inoccupée	?

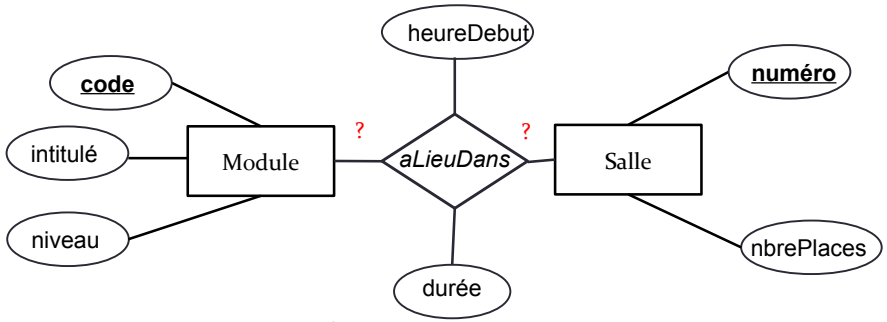
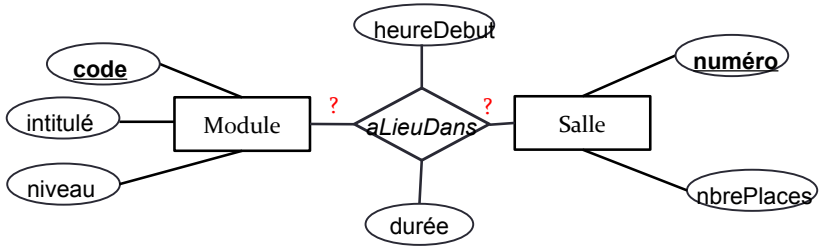


Illustration sur une instance



modules	salles	heureDebut	duree
LI341	105	830	120
LI345	105	1030	90
	214		

Association ternaire

- Un étudiant doit s'inscrire dans au moins un module à une année donnée.
- Il se voit affecté un un tuteur, un tuteur doit être affecté à au moins un étudiant.
- Un module doit avoir au moins un étudiant.

Remarque: la cardinalité de l'association d'un étudiant avec un tuteur doit être la même que la cardinalité de l'ssociation d'un étudiant avec un module

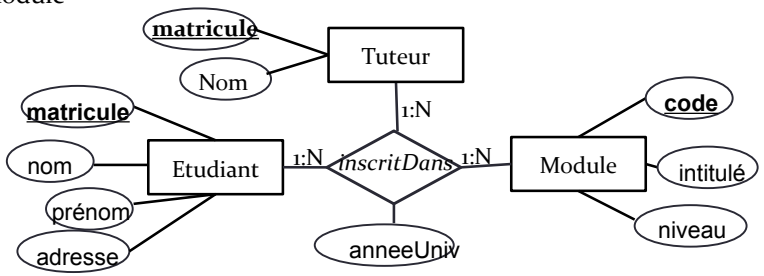
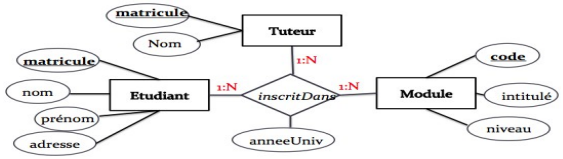


Illustration sur une instance



etudiants	modules	tuteurs	anneeUniv
1753	LI341	112	2012
1753	LI345	354	2102
0148	LI399	214	2013

3 *Est-il toujours possible d'exprimer une association n-aires avec des associations binaires?*

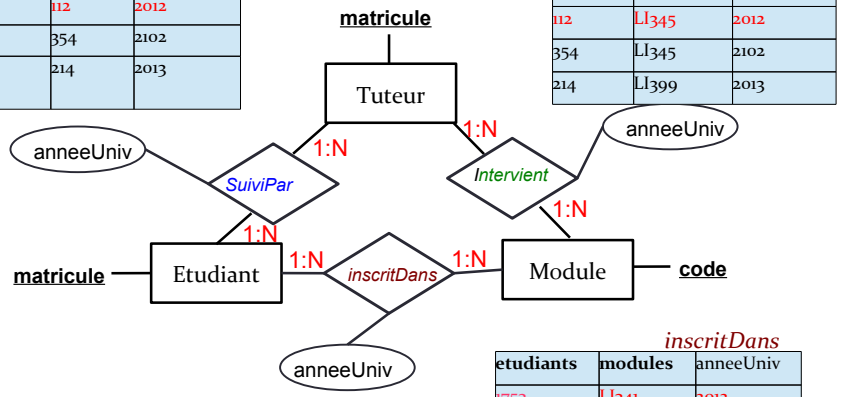
Trois associations binaires

SuiviPar

etudiants	tuteurs	anneeUniv
1753	112	2012
1753	354	2102
0148	214	2013

Intervient

tuteurs	modules	anneeUniv
112	LI345	2012
354	LI345	2102
214	LI399	2013



inscritDans

etudiants	modules	anneeUniv
1753	LI341	2012
1753	LI345	2102
0148	LI399	2013

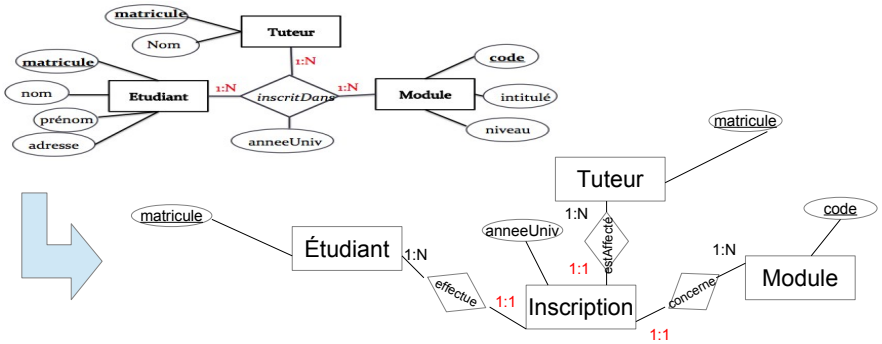
- Un étudiant peut être inscrit dans un module et être suivi par un tuteur qui n'intervient pas dans le module !
- On peut aussi enlever l'inscription d'un étudiant à un module alors que le tuteur existe toujours !

Règle de transformation d'une association n-aire en entité

Pour une association A entre les entités E1, …, En

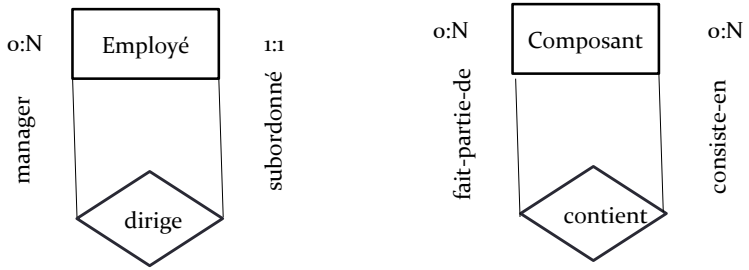
- Construire une entité E à partir des attributs de A hormis son identifiant
- Attribuer un identifiant à E (artificiel si aucun sous-ensemble de E ne peut être choisi comme identifiant
- Créer entre chaque entité Ei et E une association Ai de cardinalité 1:1

Exemple :



Association réflexive

- Une entité est associée à elle même
- On distingue deux *rôles*
- Les cardinalités peuvent être distinctes



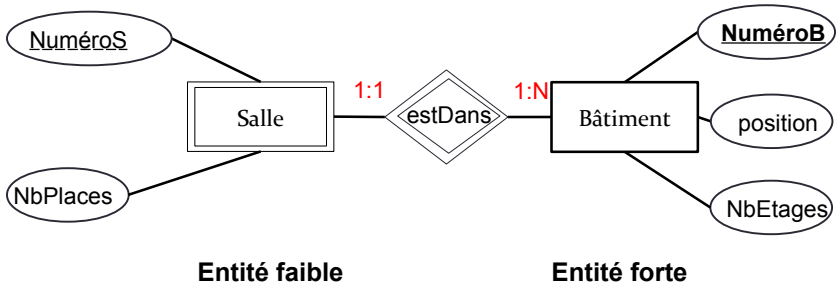
Entités faibles

- Entités ne possédant pas assez d'attributs leur permettant d'être identifiées
- Elles sont identifiées relativement à une autre entité appelée forte

Exemples :

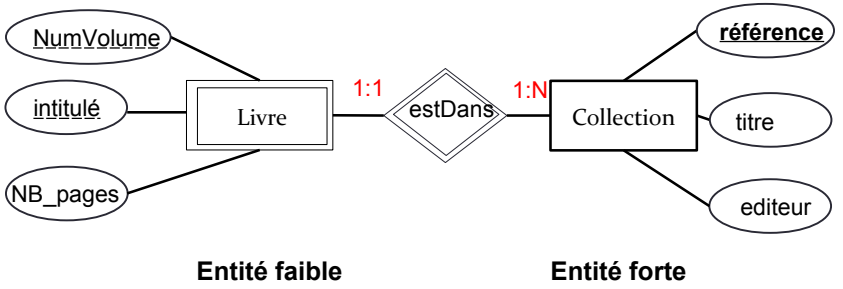
- Entité Section définie par rapport au Livre qui la contient
- Entité Livre définie par rapport à une entité Collection
- Entité Salle définie par raport à un entité Bâtiment
- Entité Bâtiment définie par rapport à une entité Campus

Entités faibles (exemple)



- Les entités faible possèdent des attributs discriminants dont les valeurs sont uniques (e.g *NuméroS)* dans le contexte de l'entité forte
- L'attribut discriminant est toujours souligné en pointillé
- Cardinalité **1:1** *implicite*
- Pas d'attribut pour l'association

Entités faibles (autre exemple)



Choix de conception

Analyse des besoins produit une spécification peu précise

→ plusieurs choix de conception possibles

Questions fréquentes :

1. Un objet du monde réel peut-il être modélise par une entité ou par un attribut ?
2. Un objet du monde réel peut-il être modélise par une entité ou par une association ?
3. Un attribut décrit-il une association ou une entité?

Choix de conception:

Entité ou attribut ?

Question. Pour renseigner l'adresse d'un étudiant

1) rajouter un attribut adresse à l'entité étudiant ? Ou
2) introduire une nouvelle entité, adresse, ayant comme attributs numéro, voie, code postal ?

Réponse. Décision relativement facile à prendre si l'on connaît l'application et son évolution.

Entité ou association ?

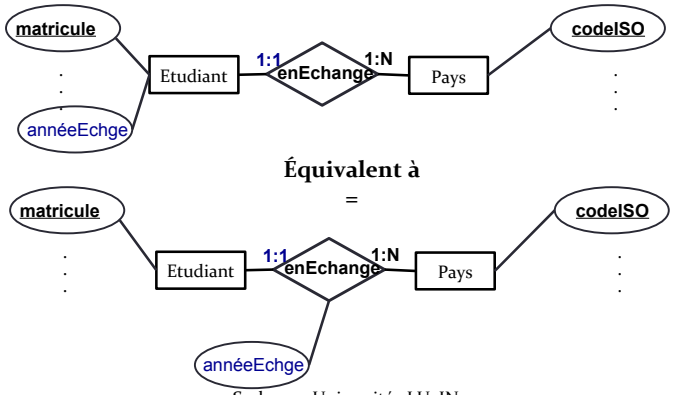
Règle générale : toute action impliquant deux entités donne lieu à une association.

Exemple: (le cours d'un) module a lieu dans une salle → association *ALieuDans*

Choix de conception

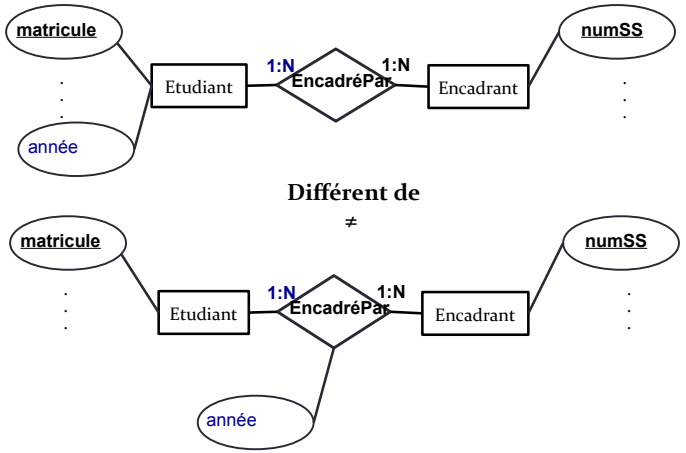
Attribut d'une association ou d'une entité? Le choix dépend des cardinalités

- Cardinalité **1 à plusieurs** : les deux alternatives sont équivalentes puisque l'une des deux entités participe une seule fois dans l'association
- Cardinalité **plusieurs-à-plusieurs** : la sémantique diffère selon le cas où l'attribut est au niveau de l'entité ou de l'association



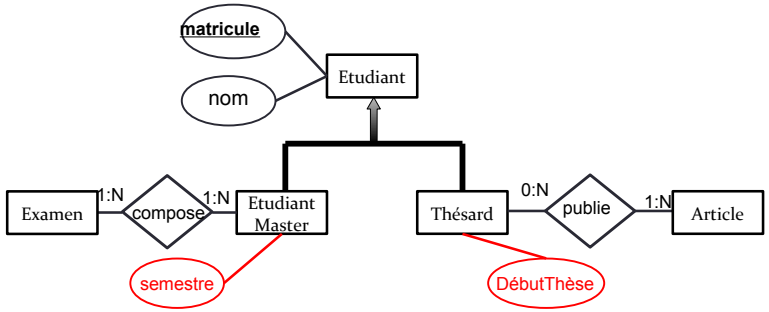
Attribut d'association ou d'entité?

Cardinalité plusieurs-à-plusieurs



Spécialisation

- Utile lorsque les objets à modéliser partagent certaines propriétés et possèdent d'autres priorités propre à eux
- **Principe** : créer une entité avec les propriétés en commun dont vont *hériter* des propriétés plus spécifiques
- Exemple : il peut y avoir deux types d'étudiants
  - *Etudiants en master* passent des examens
  - *Etudiants en thèse* publient des articles scientifiques



Conclusion

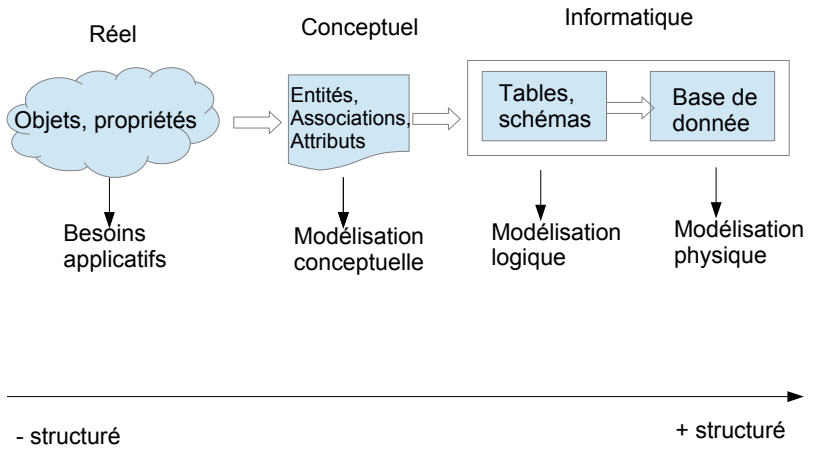
- L'intérêt des bases de données
  - Méthodologie pour la conception et la structuration de données
  - Différents niveaux d'abstraction qui permettent l'interopérabilité entre les systèmes
- Plusieurs étapes pour créer une base de données
  - Analyse de besoins
  - Modélisation des données
  - Création des données
- Modélisation des données
  - Transcription de la réalité vers le modèle Entité-Association
  - Plusieurs alternatives
- Suite: passage du modèle Entité-Association vers le modèle relationnel

Introduction aux Bases de données

Cours 2 : Modèle relationnel-Passage E/A vers le modèle relationnel

LU2IN009

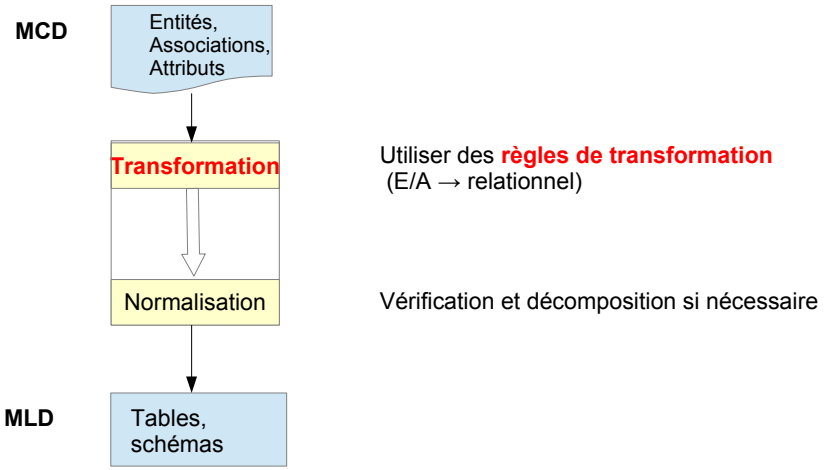
Rappel: construction d'une BD



Rappel : construction d'une BD

- **Modèle conceptuel des données (MCD)**: description de l'application dans un langage de haut niveau (Entité-Association) qui ne tient pas compte du SGBD
- **Modèle logique des données (MLD)**: description des données dans un formalisme compatible avec un SGBD (schemas, tables, colonnes, clés primaires et étrangères)
- **Modèle physique des données (MPD):** implémentation du modèle logique dans le SGBD (affiner le MLD en un schéma pour un SGBD spécifique), utilisation de SQL (create TABLE..), types des attributs, index, dénormalisation

Transformation MCD ⁷ MLD



Le modèle relationnel

Basé sur la définition et la manipulation de **relations**:

- Données: organisées dans des relations **(perçues par l'utilisateur comme tables)**
- *Table (relation)* = **ensemble** de **n-uplets** avec mêmes attributs, représentée sous la forme d'un tableau à deux dimensions:
  - Chaque *colonne* correspond à un *attribut* A_i
  - Chaque *ligne* (tuple, n-uplet) est une séquence de n valeurs atomiques (v_1, .., v_n) où chaque v_i est la valeur (nombre, chaine de caractères, date, ...) d'un attribut A_i ou **NULL** (absence de valeur).

Relation et attribut

La table (relation) **Etudiants** = ensemble de lignes (tuples ou n-uplets)

*{1753, Smith, Joe, 11 CP NYC}, {2410, Hersh, Dan, 22 Rd NJ}, {0148, Clay, Maissa, **NULL**}*

Les attributs ou colonnes

Les lignes, n-uplets ou tuples

matricule	**nom**	**prénom**	**adresse**
1753	Smith	Joe	11 CP NYC
2410	Hersh	Dan	22 Rd NJ
0148	Clay	Maissa	**NULL**

Valeur de l'attribut adresse non spécifiée (pas connue)

Attributs

Attribut : un nom qui décrit une propriété
- *Exemple :* le propriétés matricule, nom, prénom, adresse d'un étudiant

Domaine d'un attribut :
- l'ensemble des *valeurs atomiques* de l'attribut
- *Exemple :* **matricule** ∈{'1753', '2410', '0148'}, **adresse** est une chaîne de 20 caractères

Valeur NULL : l'absence temporaire de valeur (inconnu) ou l'inapplicabilité d'une valeur pour un attribut dans un tuple

Les attributs ou colonnes

Les lignes, n-uplets ou tuples

matricule	**nom**	**prénom**	**adresse**
1753	Smith	Joe	11 CP NYC
2410	Hersh	Dan	22 Rd NJ
0148	Clay	Maissa	**NULL**

Sous-ensemble du domaine de valeurs de prénom

Clés

- Plus petit sous-ensemble d'attributs qui identifient chaque ligne de ***manière unique***. [7] il n'existe pas deux lignes avec les mêmes valeurs pour l'ensemble de ces attributs
- *Exemple:* l'attribut **matricule** pour la relation Étudiant

Chaque ligne a une valeur de matricule différente

matricule	**nom**	**prénom**	**dateNaiss**	**adresse**
1753	Smith	Joe	1992-01-12	11 CP NYC
9832	Smith	Dan	1989-04-03	22 Rd NJ
4755	Smith	Joe	1994-11-29	7 HW NJ
6842	Roy	Ian	1992-05-18	NULL

- Est-ce que les ensembles suivants peuvent être des clés?
  - *{nom, prénom}*
  - *{nom, prénom, dateNaiss}*

Clé primaire et clé candidate

- Chaque relation doit posséder *au moins une clé*
- Une relation a *au moins une clé candidate (chacun des attributs est renseigné, pas de valeurs NULL)*
  - 3 on choisit *une seule* comme *clé primaire*
- *Exemple:* matricule est clé primaire, *{nom, prenom, dateNaiss}* est clé candidate

Chaque ligne a une valeur de matricule et de l'ensemble *{nom, prénom, dateNaiss}* différente

matricule	nom	prénom	dateNaiss	adresse
4753	Smith	Joe	1992-01-12	11 CP NYC
9832	Smith	Dan	1989-04-03	22 Rd NJ
4755	Smith	Joe	1994-11-29	7 HW NJ
6842	Roy	Ian	1992-05-18	NULL

- *Notation:* la clé primaire est soulignée, les clés candidates sont mentionnées en langage naturel

Clé primaire et clé candidate

- *Contraintes de l'application:*
  - Chaque module doit avoir un code différent
  - Il n'existe pas deux modules avec le même intitulé pour un niveau donné
  - Un enseignant ne peut pas être responsable de plus d'un module par niveau

La table **Modules**

code	intitulé	niveau	responsable
2I009	Bases de Données	L2	Smith
MI005	Bases de Données	M1	Roy
3I004	Programmation	L3	Smith

- Quelles sont les clés candidates?
- Quelle est la clé primaire?
- Les tuples suivants peuvent-ils exister dans la table **Modules**?
  - *{'2I009', 'BD Web', 'L3', 'Roy'}*
  - *{'MI006', 'Crypto', 'M1', 'Roy'}*
  - *{'3I009', 'Bases de Données', 'L2', 'Roy'}*

Clés étrangères

Clé étrangère:

- *sous-ensemble d'attributs* dont les valeurs proviennent des clés candidates de la même table ou d'autre table
- mécanisme de référencement des n-uplets

clé primaire — clés étrangères — clé primaire

code	intitulé	niveau	SalleC1	SalleC2
LI341	Bases de Données	L3	105	302
LI345	BD Web	L3	105	105
LI399	Crypto	L3	302	NULL

Table **Modules**

numéro	nbPlaces
105	102
302	46
112	10

Table **Salles**

- Le nom d'une clé étrangère n'est pas nécessairement le même que celui de la clé référencée

Clés étrangères : autre exemple

- La table **Module** contient *deux clés étrangères*:
  - salle : fait référence à l'attribut numéro de la table **Salle**
  - suit: fait référence à l'attribut code de la table **Module**

clé primaire — clés étrangères — clé primaire

Table **Module**

code	intitulé	niveau	suit	Salle
LI341	Bases de Données	L3	NULL	105
LI345	BD Web	L3	LI341	105
LI399	Crypto	L3	L345	302

Table **Salles**

numéro	nbPlaces
105	102
302	46
112	10

- Valeurs permises pour une clé étrangère:
  - Valeurs *déjà existantes* des clés candidates ou *NULL* (inconnu)

Clés étrangères : exemple complet

Salles

numéro	nbPlaces
105	102
302	46
112	10

Modules

code	intitulé	niveau	suit	salle	resp
LI341	Bases de Données	L3	NULL	105	479
LI345	BD Web	L3	LI341	105	845
LI399	Crypto	L3	LI345	302	214

Enseignants

matricule	nom
479	Josh
845	Lana
214	Liz

Etudiants

matricule	..
1753	
0148	

Inscriptions

etud	module	anneU
1753	LI341	2012
1753	LI345	2013
0148	LI399	2012

annee
2012
2013

- Des clés étrangères peuvent composer la clé candidate/primaire d'une relation:
  - *Exemple:* la clé {etud, module} de la table **Inscriptions**

Schéma relationnel : aperçu

Schéma d'une relation:

- Nom de la relation + Liste de ses attributs avec leur domaines (nombre, chaîne de caractères, date…) + Clés des tables et contraintes d'intégrité (cf. partie III)

Instance d'une relation:

- Ensemble des n-uplets de la table

Schéma d'une base de données:

- Ensemble des schémas des relations qui la composent

1753	Smith	Joe	11 CP NYC
2410	Hersh	Dan	22 Rd NJ
0148	Clay	Maissa	7 HW NJ

Clé primaire

matricule	nom	prénom	adresse
numérique	caractères	caractères	caractères

Instance Etudiants={3 n-uplets} **Schéma Etudiants**

Notations

- Schéma d'une BD = ensemble des schémas de relation
  S* = *{R1, R2, …,Rn} où **Ri** est un schéma de relation

- Schéma de relation = ensemble des attributs avec leurs domaines respectifs et les contraintes
  R(*A1:D1, A2:D2, …, Am:Dm)*, A1 est clé primaire: relation d'arité m

- Exemple:

  Etudiants(matricule :*Number*, nom : *Varchar*, prenom : *Varchar*, adresse : *Varchar*)

  Modules(code : *Number*, intitule : *Varchar*, niveau : *Varchar*, salle : *Number*)

  Salles(numero : *Number*, capacite : *Number*)

  Schéma de la Base de Données: {Etudiants, Modules, Salle}

  ★ *Number*=numérique, *Varchar=chaîne de caractères de longueur variable*

Simplification des notations

- *Convention de notation:*
  - Clé primaire : soulignement
  - Clés étrangères : astérisque et désignation de la table référencée
  - On omet les domaines des attributs

- *Exemple:*

  Etudiants(matricule, nom, prenom, adresse, collaborateur*)
  ⟶ collaborateur fait référence à (la clé primaire de) Etudiants

  Modules(code, intitule, niveau, salle*)
  ⟶ salle fait référence à Salles

  Salles(numero, capacite, précédente *, suivante*)
  ⟶ précédente et suivante font chacune référence à Salles

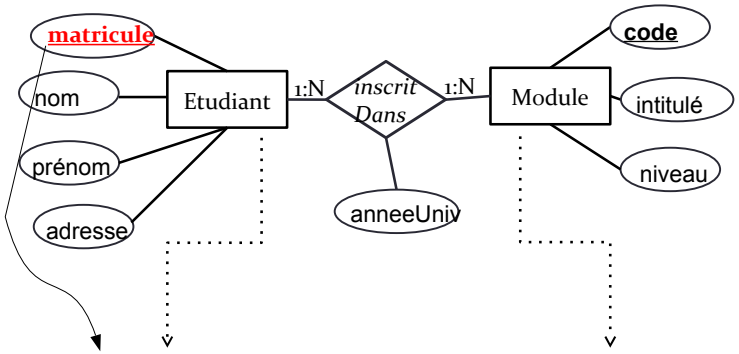
Avantages du modèle relationnel

- Proche de la réalité et simple
  - La plupart des entités du monde réel partagent les mêmes attributs
  - Familiarité des utilisateurs avec les tableaux
- Repose sur des fondements solides
  - Théorie des Ensembles
  - Logique du Premier Ordre
- Doté de langages de requêtes puissant
  - Algèbre relationnelle, Calcul des Prédicats
  - SQL (Structured Query Language)

Traduction E/A – modèle relationnel

- Règles de transformation des Entités:
  - Une entité *devient* une relation
  - Les attributs d'une entité *deviennent* les attributs de la relation
  - Tout ensemble d'attributs identifiant une entité *devient* la clé primaire de la relation
  - Tout ensemble d'attributs susceptibles de jouer le rôle d'identifiant d'entité *devient* clé candidate de la relation

Exemple



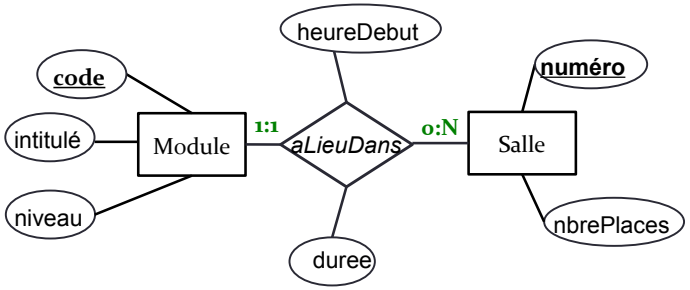
Etudiant(matricule, nom, prenom, adresse) **Module**(code, intitulé, niveau)

{nom, prénom} est second identifiant de l'entité Étudiant ⁷ {nom, prénom} est clé candidate dans la relation **Etudiant**

Règles de transformation d'une association

- Traitement différent en fonction des cardinalités:
- **Cas 1** : Association n-aire avec cardinalités (x,y) et au moins une cardinalité (1,1), où (x, y) peut être: (0,n), (1,n), (0,1)
  - La table obtenue pour l'entité correspondante à la cardinalité (1,1) contiendra aussi les attributs de l'assciation ⁷ si plusieurs cardinalités (1,1), modifier seulement la table correspondante à l'une de ces entités
- **Cas 2** : Association n-aire avec cardinalités (x, n) (x:{0,1})
  - Créer une table ayant comme attributs tous les attributs de l'association et comme identifiants les identifiants de l'association
- **Cas 3** : Associations n-aire avec cardinalités (x, y) et (0,1), où (x, y) ≠ (1,1) , (x, y) peut être {(0, n), (1, n), (0,1)}. Transformations possibles:
  - Similaire à celle pour le Cas 1 ⁷ problème: valeurs NULL possibles
  - Similaire à celle pour le Cas 2 ⁷ préférable car élimination des valeurs NULL

Cas1: Association n-aire (x, y) et (1,1)



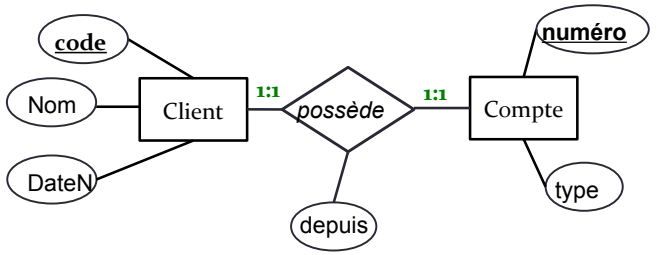
Rappel **:** attributs de l'association {code, numéro, heureDebut, durée}

Module(code, intitule, niveau, numéroSalle*, heureDebut, duree)
- NuméroSalle référence numéro de la table Salle

Salle(numéro, nbrePlaces)

Cas1: Association n-aire (x, y) et (1,1)

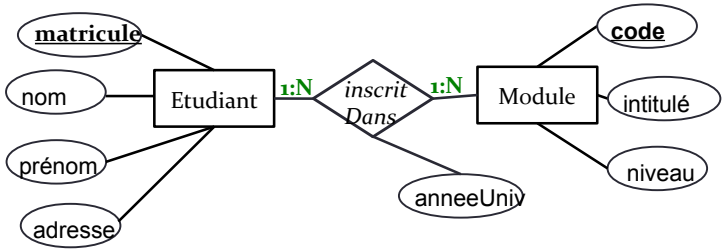
- Si plusieurs cardinalités 1:1, modifier la table de seulement une des relations correspondantes



Compte(numéro, type)

Client(code, Nom, DateN, NumeroCompte*, depuis)
- NumeroCompte référence numéro de la table Compte
  - *La table Compte peut être enlevée si l'entité Compte n'est associé à aucune autre entité, tous ses attributs seront stockés dans la relation Client*

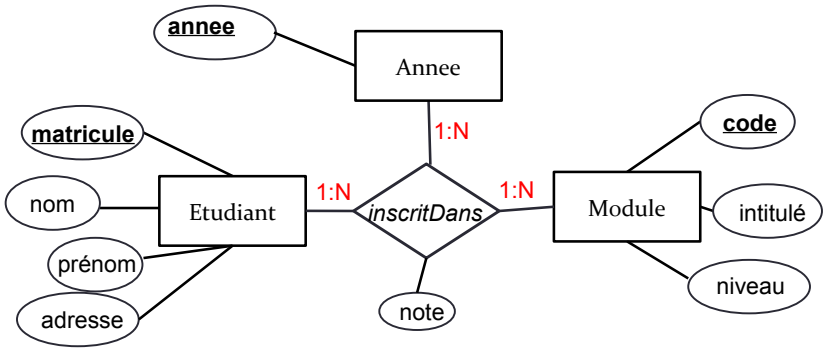
Cas 2 : Association n-aire (x, n)



Etudiant(matricule, nom, prenom, adresse) **Module**(code, intitulé, niveau)

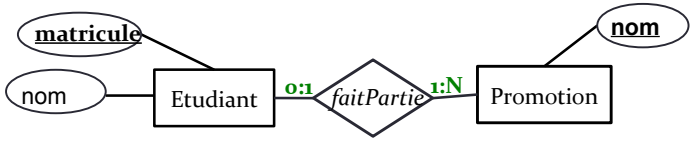
Inscriptions(matricule*, code*, anneeUniv)

Exercice : traduction d'une association ternaire (x,n)



Cas 3 : Association n-aire (x, y) et (0,1)

- Première possibilité : même transformation que pour le Cas 1



Etudiant(matricule, … , promo*) **Promotion**(nom)

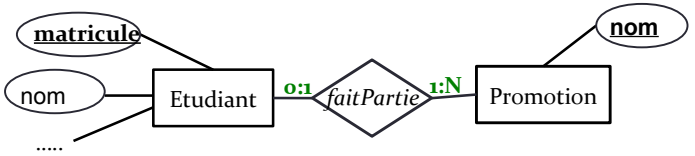
- promo clé étrangère fait référence à nom de Promotion

matricule	**nom**	**….**	**promo**
1753	Smith		M.Curie
2410	Hersh		NULL

nom
M. Curie
J. Fourrier

- *Problème* : possibilité d'avoir des valeurs NULL

Cas 3 : Association n-aire (x, y) et (0,1)



Etudiant(matricule, … , ~~promo*~~) **Promotion**(nom)

Etud-Promo(matEtu*, promo*)

matricule	**nom**	**….**
1753	Smith	
2410	Hersh	

matEtu	**promo**
1753	M. Curie

nom
M. Curie
J. Fourrier

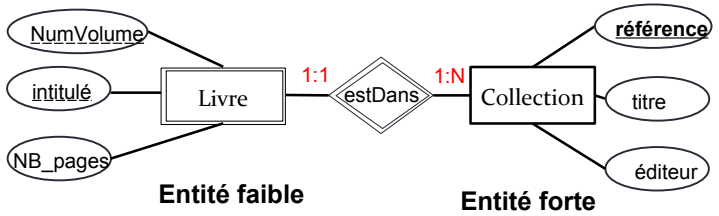
- *Avantage* : stocker uniquement les paires (étudiant, promo) qui existent
  - (évite les valeurs NULL)

Règle de transformation des entités faibles

- Transformation des entités faibles:
  - Créer une relation pour l'entité faible
  - Clé de la relation correspondante à une entité faible = concaténation de l'identifiant de l'entité faible et celui de l'entité dont elle dépend
    - les attributs constituant l'identifiant de l'entité forte contituent une clé étrangère (qui fait aussi partie de la clé primaire)

- Transformation similaire à celle pour le Cas 1, en incluant en plus la clé étrangère dans la clé primaire

Transformation des entités faibles



Collection (référence, titre, éditeur)

Livre (référence*, numVolume, intitulé, NB_pages)
- référence est une clé étrangère qui fait référence à la clé primaire de Collection

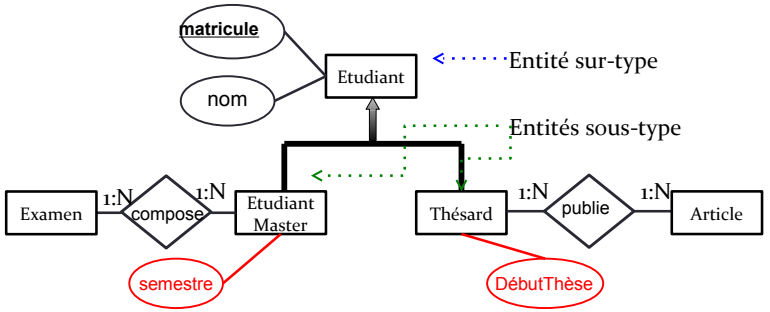
Clé primaire Livre:{référence, numVolume, intitulé}

Règle de transformation de la spécialisation

- Transformation de la spécialisation:
  - Créer une relation (table) pour chaque entité sous-type, les attributs de la relation sont ceux de l'entité sous-type
  - *Clé de la relation*: l'identifiant de l'entité sur-type

- **Cas particulier,** si l'entité sur-type est abstraite (il n'existe pas d'instance de cette entité dans l'application):
  - supprimer la table correspondante à l'entité sur-type
  - rajouter tous ses attributs dans toutes les tables correspondantes aux entités sous-type

Exemple

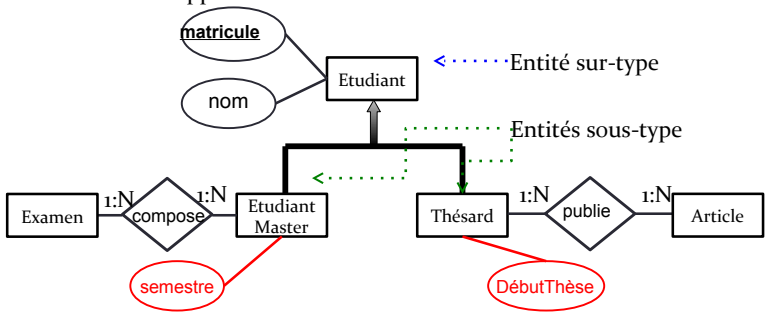


Etudiant(matricule, nom)
EtudiantMaster(matricule*, semestre)
Thesard(matricule*, debutThese)

Dans **EtudiantMaster** et **Thesard** matricule est clé primaire et étrangère (référence matricule de la table **Etudiant**) en même temps!

Exemple

- Il n'existe pas d'étudiants autres que les étudiants inscrits en master ou en thèse : [7] on supprime la table **Etudiant**



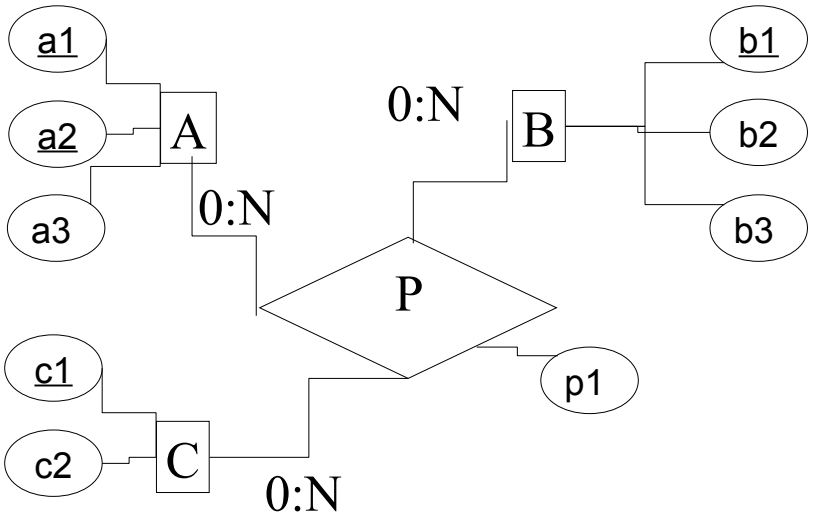
~~**Etudiant**(matricule, nom)~~

EtudiantMaster(matricule, nom, semestre)

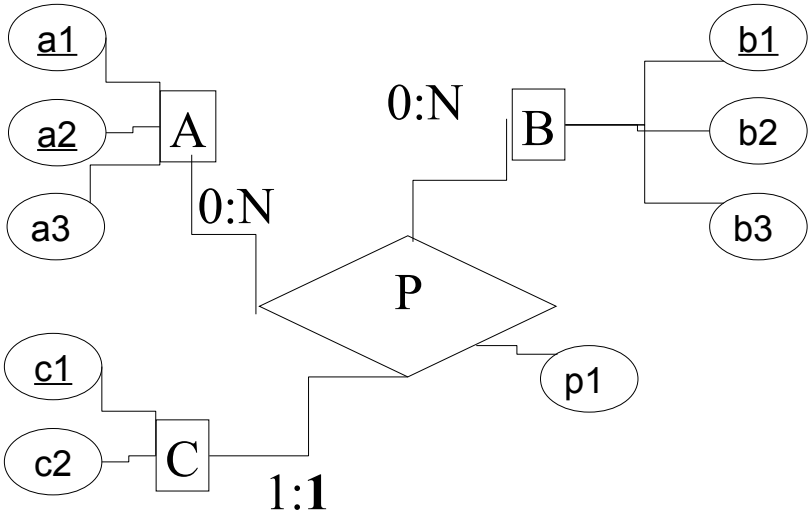
Thesard(matricule, nom, debutThese)

Dans **EtudiantMaster** et **Thesard** matricule est la clé primaire, elle n'est pas clé étrangère.

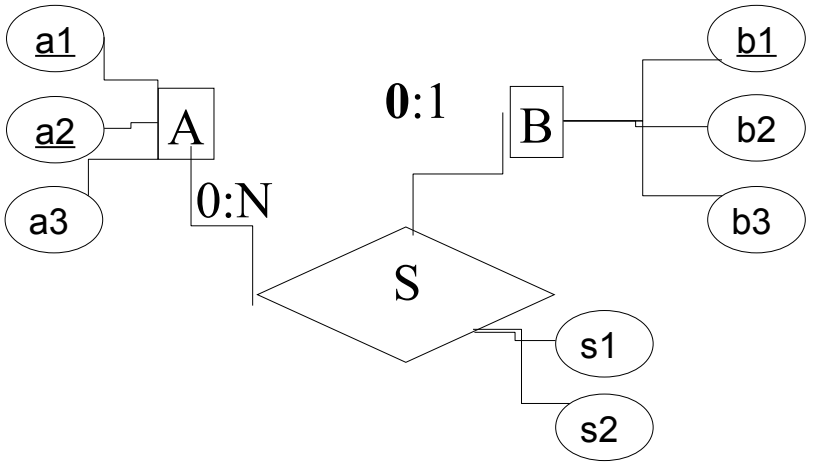
Exercices de traduction : 1



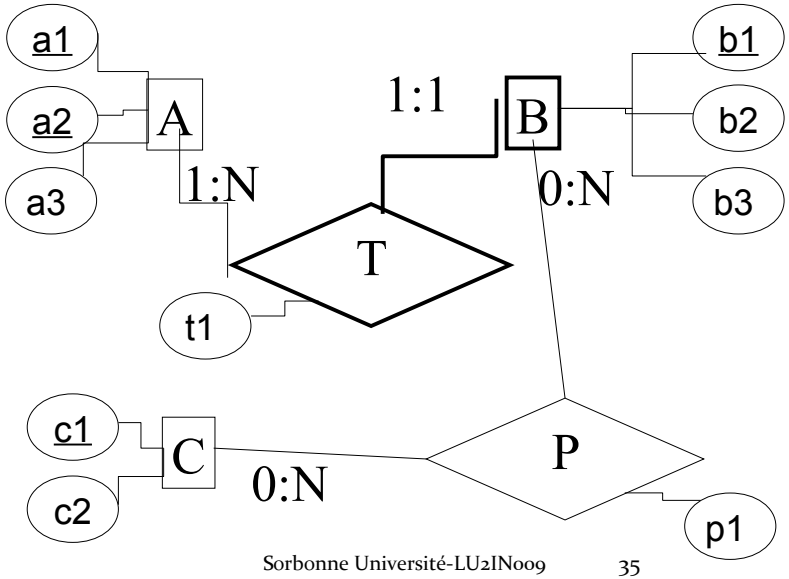
Exercices de traduction : 2



Exercices de traduction : 3



Exercices de traduction : 4



Exercice de rétro-ingénierie : 1

Soit le schéma relationnel suivant. En déduire un schéma /EA correspondant.

- ARTICLE (Aid)
- CATEGORIE (Cid)
- REFERENCE (Aid*, Cid*)
- PERSONNE (Pid, NomP, Prenom, Email)
- ArticleLangue(Aid, Langue, Titre, Contenu)
- CatégorieLangue(Cid, Langue, NomCat)
- ECRIT (Pid*, Aid*, Langue, Titre)

Exercice de rétro-ingénierie : 2

Soit le schéma relationnel suivant. En déduire un schéma EA correspondant.

SPORTIF (SID, Nom, DateNaiss, Manager)

MANAGER (MID, Tarif, Experience)

CLUB (CID, DateCreation, Budget, Division)

CONTRAT (SID, CID, SAISON-DEBUT, NBSais, Salaire, Augmentation)

RENCONTRE (LOCAL, VISITEUR, SAISON, Vainqueur)

Corrigé

- Traduction EA → relationnel
  - A(a1,a2 a3) commun aux 4 exercices
  - Ex1 : immédiat
  - Ex2 : B(b1,b2,b3) P(a1*,a2*,b1*,p1) C(c1,c2, a1*,a2*,b1*,p1)
  - ou bien A et B idem, C(c1, c2) rajouter à P c1*
  - Ex3 : B(b1,b2,b3) S(a1*,a2*,b1*, s1, s2)
  - Ex4 : B(b1, a1*,a2*,b2, b3, t1), C(c1,c2), P(b1*, a1*,a2*,c1*,p1)

Introduction aux Bases de données
Cours 3 : Calcul Relationnel

LU2IN009

Interrogation des données

Algèbre relationnelle (voir 3I009)

- Langage procédural : comment calculer les données
- Fondé sur la théorie des ensembles

Calcul relationnel des n-uplets

- Langage déclaratif : qu'est ce qu'on veut calculer
- Fondé sur la logique des prédicats

SQL (Structured Query Language)

- Langage basé sur le calcul relationnel
- Comporte en plus des fonctions d'agrégation
- Implante également les opérateurs de l'algèbre

Rappel logique 1er ordre

- appelé également logique des prédicats
- Système formel utilisé en Maths, Philo, Info.
- Enrichit la logique propositionnelle avec notion de variables et de quantificateurs

Logique propositionnelle

si Jean est malade
alors il ne sort pas
Jean est malade
conclusion : Jean ne sort pas

si p alors non q
p
_____
non q

Logique des prédicats

si un homme est malade
alors il ne sort pas
Jean est malade
conclusion : Jean ne sort pas

si P(x) alors non Q(x)
P(Jean)
_____
non Q(Jean)

Calcul Relationnel des n-uplets : aperçu

Langage fondé sur la logique des prédicats

- Tuple avec n attributs → prédicat n-aire
- Valeur atomique → constante

Etudiants

	matricule	**nom**	**prénom**	**dateNaiss**	**adresse**
Tuples	1753	Smith	Joe	1992-01-12	11 CP NYC
	9832	Smith	Dan	1989-04-03	22 Rd NJ
	….				

Prédicats

Etudiants('1753','smith', 'Joe', '1992-01-12', '11 CP NYC')
Etudiants('9832','smith', 'Dan', ' 1989-04-03', '22 Rd NJ')
…

Calcul Relationnel des n-uplets : aperçu

Langage fondé sur la logique des prédicats

- Formule logique pour exprimer une condition devant être respectée par les n-uplets à retourner

Requête : retourner les étudiants ayant pour nom 'Smith'
Condition : Etudiants(x) ∧ x.nom='Smith'

Etudiants

matricule	nom	prénom	dateNaiss	adresse
1753	Smith	Joe	1992-01-12	11 CP NYC
9832	Smith	Dan	1989-04-03	22 Rd NJ
….				

Tuples

Prédicats

Etudiants('1753','smith', 'Joe', '1992-01-12', '11 CP NYC')
Etudiants('9832','smith', 'Dan', ' 1989-04-03', '22 Rd NJ')
…

Calcul Relationnel des n-uplets : aperçu

Langage fondé sur la logique des prédicats

- Formule logique pour exprimer une condition devant être respectée par les n-uplets à retourner

Requête : retourner les étudiants ayant pour nom 'Smith'
Condition : Etudiants(x) ∧ x.nom='Smith'

Etudiants

Réponse

matricule	nom	prénom	dateNaiss	adresse
1753	Smith	Joe	1992-01-12	11 CP NYC
9832	Smith	Dan	1989-04-03	22 Rd NJ
….				

Tuples

Prédicats

Etudiants('1753','smith', 'Joe', '1992-01-12', '11 CP NYC')
Etudiants('9832','smith', 'Dan', ' 1989-04-03', '22 Rd NJ')
…

Calcul Relationnel des n-uplets : aperçu

Requête : retourner les étudiants inscrits en LI341
Condition : Etudiants(x) ∧ Inscriptions(y) ∧x.matricule=y.matricule ∧ y.code='LI341'

Etudiants

Réponse

matricule	nom	prénom	dateNaiss	adresse
1753	Smith	Joe	1992-01-12	11 CP NYC
9832	Smith	Dan	1989-04-03	22 Rd NJ
….				

Inscriptions

Matricule*	Code*
1753	LI341
1753	LI345
9832	LI341

Calcul Relationnel des n-uplets : aperçu

Requête : retourner les étudiants inscrits en Bases de Données ('BD')
Condition : Etudiants(x) ∧ Inscriptions(y) ∧x.matricule=y.matricule ∧ Modules(z) ∧ y.code=z.code∧z.intitule='BD'

Etudiants

matricule	nom	prénom	…
1753	Smith	Joe	
9832	Smith	Dan	
….			

Inscriptions

Matricule*	Code*
1753	LI341
1753	LI345
9832	LI341

Modules

code	intitulé	niveau
LI341	BD	L3
LI345	Web	L3
LI399	Crypto	L3

Autres requêtes :

- retourner les étudiants inscrits dans un même module que 'Jack'?
- retourner les étudiants inscrits dans aucun module?
- retourner les étudiants inscrits dans tous les modules?
- ….

Calcul Relationnel des n-uplets : syntaxe

Forme générale des requêtes

{ expression | Condition }

Format résultat

Formule logique du 1er ordre

Expression	Signification
v, w, …z	tous les attributs plusieurs tables
$v.a_1$, $v.a_2$, …$v.a_m$	certains attributs même table
$v.a_1..v.a_m$,$w.b_1..w.b_p$ ..	certains attributs plusieurs tables

Calcul Relationnel des n-uplets : syntaxe

Forme générale des requêtes

{ expression | Condition }

Format résultat

Formule logique du 1er ordre

- **R(v)**
- **v.A op w.B où op∈{=,≠,>,<,≤,≥}**
- **v.A op 'val' valeur atomique**
- **$C_1 \wedge C_2$, $C_1 \vee C_2$, ¬C, C1➔C2**
- **∃v (C) , ∀w (C) v et w variables liées**

Les variables libres apparaissent dans *expression* et *condition*, les variables liées uniquement dans *condition* { expression(v1,v1,…,vn) | condition(v1,v1,…,vn, vn+1, …, vm)}

Calcul Relationnel des n-uplets : syntaxe

- Priorité des connecteurs logiques
  - Négation ¬
  - Conjonction ∧
  - Disjonction ∨
  - Implication ➔
- Utilisation des parenthèses pour lever l'ambiguité

Calcul Relationnel des n-uplets : sémantique

Forme générale des requêtes

{ expression | Condition }

Format résultat

Formule logique du 1er ordre

Retourner les n-uplets spécifiés par expression tel que Condition est vérifiée

Calcul Relationnel des n-uplets : sémantique

Forme condition	Nom	Sémantique
R(v) ou v∈R	Liaison de variables	Vraie lorsque la table **R** contient des n-uplets
1. v.A op w.B **2. v.A op 'val'**	Expressions de comparaisons	1. Vraie lorsque l'attribut **A** du n-uplet **v** est égal/différent de/…l'attribut **B** du n-uplet **w** 2. Vraie lorsque l'attribut **A** du n-uplet **v** est égal/différent de/… **val**
1. $C_1 \wedge C_2$ **2. $C_1 \vee C_2$** **3. $\neg C$**	Expressions booléennes	1. Vraie lorsque les deux conditions vraies 2. Vraie lorsque l'une des deux conditions vraie 3. Vraie lorsque **C** est fausse
∃v (C)	Quantificateur existentiel	vrai si **C** est vrai pour au moins un n-uplet **v**
∀w (C)	Quantificateur universel	vrai si **C** est vrai pour tout les n-uplets **w**

Calcul Relationnel des n-uplets : exemples

Etudiants(matricule, nom, prenom, adresse, collaborateur*)
Inscriptions(matricule*, code*)
Modules(code, intitule, niveau, salle*)
Salles(numero, capacite, précédent *, suivante*)

1. Tous les étudiants
2. Le prénom des étudiants ayant pour nom de famille Smith
3. Le matricule des étudiants inscrits dans le module 'BD' ainsi que le code de ce module
4. a) Les intitulés des modules où 'Jack' est inscrit
   b) Les étudiants inscrits dans un même module que 'Jack' ainsi que le code de ce module

Sélection

Etudiants(matricule, nom, prenom, adresse, collaborateur*)
Inscriptions(matricule*, code*)
Modules(code, intitule, niveau, salle*)
Salles(numero, capacite, précédent *, suivante*)

1. Tous les étudiants

Sélection

Etudiants(matricule, nom, prenom, adresse, collaborateur*)
Inscriptions(matricule*, code*)
Modules(code, intitule, niveau, salle*)
Salles(numero, capacite, précédent *, suivante*)

2. Le prénom des étudiants ayant pour nom de famille Smith :

Jointure

Etudiants(matricule, nom, prenom, adresse, collaborateur*)
Inscriptions(matricule*, code*)
Modules(code, intitule, niveau, salle*)
Salles(numero, capacite, précédent *, suivante*)

3. Le matricule des étudiants inscrits dans le module ‘BD’ ainsi que le code de ce module :

Illustration évaluation requête 3

Etudiants

e →

matricule	nom	prénom	…
1753	Smith	Joe	
9832	Smith	Dan	
….			

Inscriptions

i →

Matricule*	Code*
1753	LI341
1753	LI345
9832	LI341

Modules

m →

code	intitulé	niveau
LI341	BD	L3
LI345	Web	L3
LI399	Crypto	L3

Jointure

Etudiants(matricule, nom, prenom, adresse, collaborateur*)
Inscriptions(matricule*, code*)
Modules(code, intitule, niveau, salle*)
Salles(numero, capacite, précédent *, suivante*)

4. a) Les intitulés des modules où ‘Jack’ est inscrit :

Evaluation requête 4-a

Etudiants

j →

matricule	nom	prénom	…
1753	Smith	Joe	
9832	Smith	Jack	
….			

Inscriptions

i →

Matricule*	Code*
1753	LI341
1753	LI345
9832	LI341

Modules

m →

code	intitulé	niveau
LI341	BD	L3
LI345	Web	L3
LI399	Crypto	L3

Jointure

Etudiants(matricule, nom, prenom, adresse, collaborateur*)
Inscriptions(matricule*, code*)
Modules(code, intitule, niveau, salle*)
Salles(numero, capacite, précédent *, suivante*)

4. b) Les étudiants inscrits dans un même module que 'Jack' ainsi que l'intitulé de ces modules :

Evaluation requête 4-b

Etudiants

	matricule	nom	prénom	…
e →	1753	Smith	Joe	
j →	9832	Smith	Jack	
	….			

Inscriptions

	Matricule*	Code*
i1 →	1753	LI341
	1753	LI345
i2 →	9832	LI341

Modules

	code	intitulé	niveau
m →	LI341	BD	L3
	LI345	Web	L3
	LI399	Crypto	L3

Différence

Etudiants(matricule, nom, prenom, adresse, collaborateur*)
Inscriptions(matricule*, code*)
Modules(code, intitule, niveau, salle*)
Salles(numero, capacite, précédent *, suivante*)

5. a) Les étudiants qui ne sont pas inscrits au module 'LI345'

Evaluation requête 5-a)

Etudiants

	matricule	nom	prénom	…
e →	1753	Smith	Joe	
	9832	Smith	Jack	
	….			

Inscriptions

	Matricule*	Code*
i →	1753	LI341
	1753	LI345
	9832	LI341

Différence

Etudiants(matricule, nom, prenom, adresse, collaborateur*)
Inscriptions(matricule*, code*)
Modules(code, intitule, niveau, salle*)
Salles(numero, capacite, précédent *, suivante*)

5. b) Les étudiants qui ne sont inscrits dans aucun module

Evaluation requête 5-b)

Etudiants

e →

matricule	nom	prénom	…
1753	Smith	Joe	
9832	Smith	Jack	
….			

Inscriptions

i →

Matricule*	Code*
1753	LI341
1753	LI345
9832	LI341

Modules

code	intitulé	niveau
LI341	BD	L3
LI345	Web	L3
LI399	Crypto	L3

Division

Etudiants(matricule, nom, prenom, adresse, collaborateur*)
Inscriptions(matricule*, code*)
Modules(code, intitule, niveau, salle*)
Salles(numero, capacite, précédent *, suivante*)

6. Les étudiants inscrits dans tous les modules

Evaluation requête 6

Etudiants

e →

matricule	nom	prénom	…
1753	Smith	Joe	
9832	Smith	Jack	
….			

Inscriptions

i →

Matricule*	Code*
1753	LI341
1753	LI345
9832	LI341

Modules

m →

code	intitulé	niveau
LI341	BD	L3
LI345	Web	L3

Jointures

Etudiants(matricule, nom, prenom, adresse, collaborateur*)
Inscriptions(matricule*, code*)
Modules(code, intitule, niveau, salle*)
Salles(numero, capacite, précédent *, suivante*)

7. Les étudiants inscrits dans au moins deux modules

Evaluation requête 7

Etudiants

e →

matricule	nom	prénom	…
1753	Smith	Joe	
9832	Smith	Jack	
….			

Inscriptions

i1 →
i2 →

Matricule*	Code*
1753	LI341
1753	LI345
9832	LI341

Modules

code	intitulé	niveau
LI341	BD	L3
LI345	Web	L3
LI399	Crypto	L3

Jointure

Etudiants(matricule, nom, prenom, adresse, collaborateur*)
Inscriptions(matricule*, code*)
Modules(code, intitule, niveau, salle*)
Salles(numero, capacite, précédent *, suivante*)

8. Les étudiants inscrits à au moins deux modules de niveau L3

Evaluation requête 8

Etudiants

e →

matricule	nom	prénom	…
1753	Smith	Joe	
9832	Smith	Jack	
….			

Inscriptions

i1 →
i2 →

Matricule*	Code*
1753	LI341
1753	M009
9832	LI341
1753	LI345

Modules

m1 →
m2 →

code	intitulé	niveau
LI341	BD	L3
LI345	Web	L3
LI399	Crypto	L3
M009	BD avancées	M1

Jointure

Etudiants(matricule, nom, prenom, adresse, collaborateur*)
Inscriptions(matricule*, code*)
Modules(code, intitule, niveau, salle*)
Salles(numero, capacite, précédent *, suivante*)

9. a) Les étudiants inscrits à au moins deux modules de niveau L3 ou de niveau M1

Evaluation requête 9-a)

Etudiants

e →

matricule	nom	prénom	…
1753	Smith	Joe	
9832	Smith	Jack	
….			

Inscriptions

	Matricule*	Code*
i1 →	1753	LI341
	1753	M009
	9832	LI341
i2 →	1753	LI345

Modules

	code	intitulé	niveau
m1 →	LI341	BD	L3
m2 →	LI345	Web	L3
	LI399	Crypto	L3
	M009	BD avancées	M1

Jointure

Etudiants(matricule, nom, prenom, adresse, collaborateur*)
Inscriptions(matricule*, code*)
Modules(code, intitule, niveau, salle*)
Salles(numero, capacite, précédent *, suivante*)

9. b) Les étudiants qui sont soit inscrits à au moins deux modules de niveau L3 soit à un module de niveau M1

Evaluation requête 9-b)

Etudiants

e →

matricule	nom	prénom	…
1753	Smith	Joe	
9832	Smith	Jack	
….			

Inscriptions

	Matricule*	Code*
i1 →	1753	LI341
i →	1753	M009
	9832	LI341
i2 →	1753	LI345

Modules

	code	intitulé	niveau
m1 →	LI341	BD	L3
m2 →	LI345	Web	L3
	LI399	Crypto	L3
m →	M009	BD avancées	M1

Cas particuliers

- Requêtes sans réponse
  - { e | Etudiants(e) ∧ Modules (e) } : deux tables ne peuvent avoir exactement le même n-uplet (schéma différents)
  - {e |Etudiants(e) ∧ e.salaire>100} : salaire n'est pas attribut de Etudiants
- Requêtes avec réponse infinie → requête pas sûre
  - {e | ¬ Etudiants(e) } : chercher partout sauf dans la table Etudiants
  - {e | ∀ x Etudiants(x) ∧ … } : tous les n-uplets du monde doivent être dans Etudiant
  - {e | e.att>5} : e instanciée un nombre infini car pas liée à une table

Les requêtes sur une BD **doivent** être sûres !!!

Requêtes sûres

- Bonnes pratiques
  - Avec ∀, il faut toujours un ⇒ qui suit
  - Forme équivalente qu'on retrouve dans SQL:
    { t | Table(t) …. ∃ v ∈TableBis……¬ ∃ v ∈TableTer … }

Conclusion

- Présentation d'un langage de requête fondé sur la Logique du Premier Ordre
  - Les prédicats sont des tables, les constantes sont les valeurs atomiques, les variables sont liées aux n-uplets
  - Exprimer une requête = spécifier une condition logique → langage déclaratif
  - Autre variante : les variables liées aux attributs (calcul du domaine)
  - Avantage calcul des n-uplets : traduction quasi-directe vers SQL
- Prochain cours : SQL

Introduction aux Bases de données
Cours 4 : Introduction à SQL

UFR 919 – Licence

2e année

Plan

- Introduction
- Sélection et projection
- Tri des résultats
- Opérations ensemblistes
- Fonctions numériques, de caractères et de dates

SQL : Structured Query Language

- Langage d'interrogation pour les BD relationnelles
- Développé chez IBM (1970-80)
- Devenu une norme (ANSI/ISO) en 1986
- Malgré ça, implantations différentes selon SGBD
- Langage déclaratif (basé sur calcul relationnel de tuple)

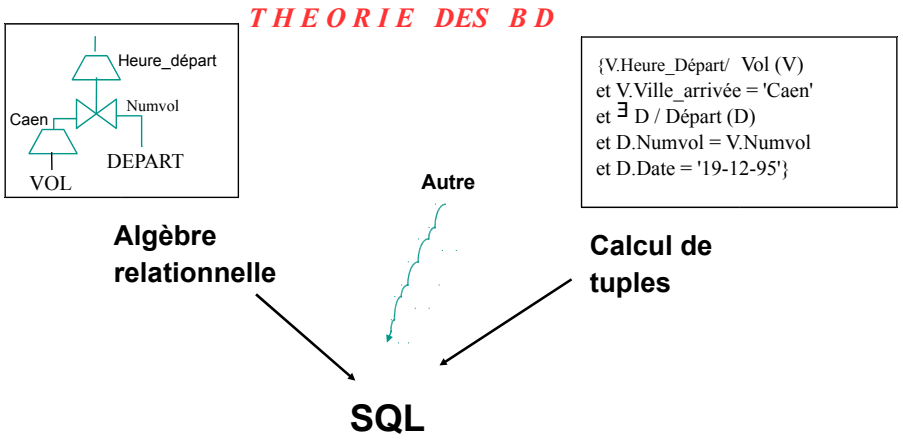
L'évolution des sta ndards SQL

- Début : SQL86
- SQL89 ou SQL1
- SQL92 ou SQL2
- SQL99 ou SQL3 (ajout récursivité, triggers, fonctions OO, types binaires,...)
- SQL2003 (ajout manipulation XML, auto-incrément...)
- SQL2008 (ajout des fonctions de fenêtrage, limite du nombre de résultats, ...)

Principaux rôles de SQL

1) Définir et modifier le schéma d'une BD
2) Manipuler les données (ajout, suppression, modification)
3) Interroger les données

D'où vient SQL ?



Syntaxe simplifée de SQL

SELECT liste-colonnes
FROM *Table*
[WHERE condition] ;

Retourne

- Les attributs de liste-colonnes
- Des enregistrements de la table *Table*
- qui vérifient condition

  La clause where est facultative mais très utile

Exemples

Schéma de la BD

Emp (Eno, Ename, Title, City)
Project(Pno, Pname, Budget, City)
Pay(Title, Salary)
Works(Eno, Pno, Resp, Dur)

Noms de tous les employés

Noms et budgets des projets

Remarques 1/2

- La clause from déclare les variables (calcul)
  - § Par défaut nom de la relation : from R, S
  - § on peut renommer : from R v1, S v2...
- Pour retourner toutes les colonnes
  - § Select *
- Sémantique « multi-ensembliste »:
  - § Possibilité d'avoir des doublons
    (parce que les éliminer coûte cher, parce qu'on peut vouloir les compter,...)
  - § Les éliminer avec le mot clé **distinct**
    select DISTINCT

Exemples

Schéma de la BD

Emp (Eno, Ename, Title, City)
Project(Pno, Pname, Budget, City)
Pay(Title, Salary)
Works(Eno, Pno, Resp, Dur)

Toutes les informations sur les employés

Toutes les villes où vivent des employés

L'ensemble des villes où vivent des employés

Remarques 2/2

- Possibilité d'exprimer des opérations arithmétiques
  - § (att1+att2, att*1.5, etc)
- Possibilité de retourner des chaines entre ' '
- Possibilité de préfixer les attributs par le nom de la table ou une variable
  - § Lever les ambiguïtés de noms d'attributs
- Possibilité de renommer une colonne dans le SELECT avec le mot-clé AS
  - § Lisibilité des résultats

Exemples

Schéma de la BD

Emp (Eno, Ename, Title, City)
Project(Pno, Pname, Budget, City)
Pay(Title, Salary)
Works(Eno, Pno, Resp, Dur)

Salaires mensuel par titre(considérer que Salary est pour un an)

Toutes les villes où vivent des employés

Noms et budgets des projets

Exemple

PROJ

PNO	PNAME	BUDGET
P1	Instrumentation	150000
P2	Database Develop.	135000
P3	CAD/CAM	250000
P4	Maintenance	310000
P5	CAD/CAM	500000

SELECT PNO, BUDGET
FROM PROJ :

PNO	BUDGET
P1	150000
P2	135000
P3	250000
P4	310000
P5	500000

SELECT PNAME FROM PROJ :

PNAME
Database Develop.
Instrumentation
CAD/CAM
Maintenance
CAD/CAM

SELECT DISTINCT PNAME
FROM PROJ :

PNAME
Maintenance
CAD/CAM
Database Develop.
Instrumentation

WHERE : Prédicats

Prédicats simples :

- Expression1 θ Expression2
  - où Expression1 peut être un attribut ou une expression arithmétique impliquant des attributs, $\theta \in \{<, >, =, <=, >=, <>\}$ et Expression2 une expression ou une valeur de domaine
- Exemples :
  - R.Name = 'J. Doe'
  - (S.Age + 30) >= 65
  - R.A = S.B

Prédicats composés :

- prédicats simples combinés avec les connecteurs logiques AND, OR, NOT

Exemple de sélection

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng.
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

SELECT * FROM EMP
WHERE TITLE = 'Elect. Eng.' ;

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E6	L. Chu	Elect. Eng.

Requêtes avec prédicats

Emp (Eno, Ename, Title, City)
Pay(Title, Salary)
Project(Pno, Pname, Budget, City)
Works(Eno, Pno, Resp, Dur)

Professions qui gagnent plus de 50 000 € par an ?

Numéros des managers d'un projet pendant plus de 17 mois?

IN, BETWEEN, LIKE

- Appartenance à un ensemble de valeurs :

  Att IN (Const1, Const2, ...)
- Appartenance à un intervalle de valeurs :

  Att BETWEEN Constante1 AND Constante2
- Ressemblance à un motif :

  Att LIKE 'MOTIF'

§ où MOTIF combine des chaînes et des joker
- % pour une chaîne quelconque (y compris vide)
- _ pour un caractère quelconque et un seul

Requêtes avec prédicats (2)

Emp (Eno, Ename, Title, City) **Project**(Pno, Pname, Budget, City)
Pay(Title, Salary) **Works**(Eno, Pno, Resp, Dur)

Nom des projets de Paris, Lyon ou Nantes ?

Comment le faire sans IN ?

Nom des projets ayant un budget compris entre 5M et 10M euros?

Comment le faire sans BETWEEN ?

Requêtes avec prédicats (3)

Emp (Eno, Ename, Title, City) **Project**(Pno, Pname, Budget, City)
Pay(Title, Salary) **Works**(Eno, Pno, Resp, Dur)

Nom des employés commençant pas C?

Nom des employés dont le 2ème numéro est un 5?

Nom des employés habitant une ville composé de 2 mots (ex : Chatenay Malabry)?

Valeurs nulles

- La valeur de certains attributs peut
  - ne pas être connue (ex. : année de construction du Louvre)
  - ou ne pas avoir de sens (ex. : nom de jeune fille pour un homme)

on parle alors de valeurs nulles (mot-clé NULL)

- NULL n'est pas une valeur mais une absence de valeur! Les opérations ou les comparaisons ne peuvent lui être appliqué
- Toute opération (+,-,/,*,substr, to_char,...) appliquée à NULL donne NULL
- Toute comparaison avec NULL donne ni vrai ni faux, mais INCONNU

Notions de sémantique Tri-Valuée abordées en cours 8 : compléments SQL

Syntaxe du tri

SELECT liste-colonnes
FROM nomtable
WHERE condition
ORDER BY liste-colonnes ;

- Dans la clause ORDER BY, on peut avoir des :
  - des noms de colonnes
  - des expressions avec noms de colonnes
  - des numéros de position des colonnes dans la clause SELECT.
- On précise le sens : ASC (par défaut) ou DESC
- Les valeurs nulles sont à la fin par ordre croissant, au début par ordre décroissant.

Exemple de tri

Emp (Eno, Ename, Title, City) **Project**(Pno, Pname, Budget, City)
Pay(Title, Salary) **Works**(Eno, Pno, Resp, Dur)

Noms, budgets et villes des projets de budget supérieur à 250 000 euros, en ordonnant le résultat par ordre décroissant de budget puis par nom par ordre alphanumérique croissant ?

Exemple tri lexicographique

NUMPROJ	NOMPROJ	BUDGET
1	aa	20
2	bb	100
3	ab	30
4	aa	200

select * from project order by nomproj , budget desc;

NUMPROJ	NOMPROJ	BUDGET
4	aa	200
1	aa	20
3	ab	30
2	bb	100

Exemple de tri (2)

Emp (Eno, Ename, Title, City) **Project**(Pno, Pname, Budget, City)
Pay(Title, Salary) **Works**(Eno, Pno, Resp, Dur)

Noms, budgets TTC (TVA 20%) et villes des projets, en ordonnant le résultat par ordre décroissant de budget TTC ?

Noms, budgets et villes des projets en ordonnant le résultat par ordre décroissant de budget TTC ?

Opérations ensemblistes

On peut réaliser des opérations ensemblistes sur les clauses SELECT.

3 opérations ensemblistes

UNION	union de deux ensembles
INTERSECT	intersection de deux ensembles
MINUS	différence de deux ensembles (norme : EXCEPT)

Principe

Pour les opérations ensemblistes :

- Pas de lien entre les objets sélectionnés dans les 2 requêtes
- Même schéma dans les SELECT des deux requêtes : c'est à dire même nombre d'attributs et chacun du même type (par forcément le même nom)
- Le schéma en sortie correspond au schéma de la première requête
- Par défaut, les opérations ensemblistes éliminent les doublons (ensemble). Pour garder les doublons (multi-ensemble), il faut ajouter ALL après l'opérateur : UNION ALL, EXCEPT ALL, INTERSECT ALL

UNION

Emp (Eno, Ename, Title, City) **Project**(Pno, Pname, Budget, Town)
Pay(Title, Salary) **Works**(Eno, Pno, Resp, Dur)

Noms des villes où habitent des employés ou où sont localisés des projets?

INTERSECTION

Emp (Eno, Ename, Title, City) **Project**(Pno, Pname, Budget, Town)
Pay(Title, Salary) **Works**(Eno, Pno, Resp, Dur)

Noms des villes où habitent des employés et où sont localisés des projets?

DIFFERENCE

Emp (Eno, Ename, Title, City) **Project**(Pno, Pname, Budget, Town)
Pay(Title, Salary) **Works**(Eno, Pno, Resp, Dur)

Noms des villes où habitent des employés mais où n'est localisé aucun projet?

Fonctions

De nombreuses fonctions existent pour :

- Manipuler les dates
- Manipuler les chaînes de caractères
- Manipuler les chiffres

Cependant, bien qu'une norme existe, elles diffèrent souvent d'un SGBD à l'autre...

Quelques exemples suivent.

Les fonctions de date

Fonction(s)	Desc	Norme	Access	MySQL	Sql Srv	Oracle
Current_date	Date courante	O	N	O	N	N
Current_time	Heure courante	O	N	O	N	N
Getdate	Heure et date courante	N	N	N	O	N
Now	Heure et date courante	N	O	O	O	N
Sysdate	Date et heure courante	N	N	O	N	O
Day/month/year	Sélectionne le jour/mois/an	N	O	O	O	N
To_char(f1,f2)	Conversion de date ou numérique en string	N	N	N	N	O

Les fonctions sur chaînes de caractères

Fonction	Desc	Norme	Access	MySQL	Sql Srv	Oracle
Lower/Upper	Mise en minuscules/majusc	O	N	O	O	O
Substring	Extraction sous-chaîne	O	N	O	N	N
Substr	Extraction sous-chaîne	N	N	N	N	O
Position	Position d'une chaîne dans une autre	O	N	O	N	N
Locate	Position d'une chaîne dans une autre	N	O	O	O	O

Les fonctions numériques

Fonction	Desc	Norme	Access	MySQL	Sql Srv	Oracle
Abs	Valeur absolue	N	O	O	O	O
Ceiling	Valeur approchée haute	N	O	O	O	N
Ceil	Valeur approchée haute	N	N	N	N	O
Floor	Valeur approchée basse	N	O	O	O	O
Cos, sin, tan, exp, log,mod, power, sqrt	Opérations diverses	N	O	O	O	O

Fonctions sur dates (1)

Les dates ont un format de stockage optimisé et un format d'affichage/saisie par défaut qui dépend de la configuration du SGBD

- On peut afficher une date (attribut date) au format voulu :

  to_char(att_Date, 'masque')

- Ou saisir une date sans connaître le format de saisie par défaut :

  to_date('chaîne', 'masque')

A noter qu'on peut soustraire deux dates pour obtenir une durée, ou additionner une durée à une date pour obtenir une autre date

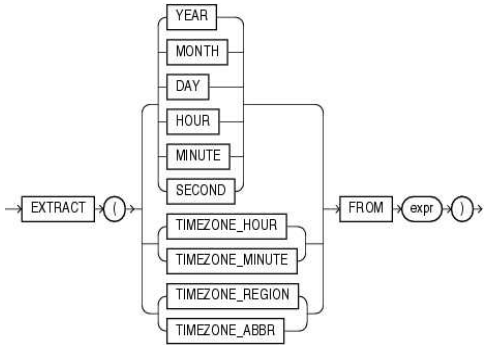
Fonctions sur dates (2)

Le masque se compose de :

- YEAR, YY, YYYY pour l'année en toutes lettres, sur 2 chiffres ou sur 4
- MONTH, MON, MM pour le mois en entier, abrégé, sur 2 chiffres
- DAY, DDD, DD, D pour le jour en lettre, le jour de l'année, du mois ou de la semaine
- HH, HH24, MI, SS pour heure sur 12H, sur 24H, les minutes, les secondes
- - ; / , _ , etc pour les séparateurs

Fonctions sur dates (3)

- La fonction Extract permet d'extraire un élément d'une date



SELECT EXTRACT(YEAR FROM DATE '1998-03-07') FROM DUAL;

EXTRACT(YEARFROMDATE'1998-03-07')

1998

Exemple

```
SELECT last_name, employee_id, hire_  date
   FROM employees
   WHERE EXTRACT(YEAR FROM
   TO_DATE(hire_date, 'DD-MON-RR')) > 1998
   ORDER BY hire_date;

LAST_NAME                 EMPLOYEE_ID HIRE_DATE
------------------------- ----------- ---------
Landry                            127 14-JAN-99
Lorentz                           107 07-FEB-99
Cabrio                            187 07-FEB-99
```


Exemples sur Oracle (1/3)

```
SELECT  TO_CHAR
     (SYSDATE, 'MM-DD-YYYY HH24:MI:SS')
 FROM DUAL;

SQL> SQL>   2    3
TO_CHAR(SYSDATE,'MM-DD-
YYYYHH24:MI:SS')
-------------------------------------------------------
02-19-2015 12:50:46
1 ligne sélectionnée
```


Exemples sur Oracle (2/3)

```
SELECT TO_DATE ('02-19-2015' )
FROM DUAL;

SQL> SQL>   2    3        ('02-19-2015' )
    *
ERREUR à la ligne 2 :
ORA-01843: ce n'est pas un mois valide
```


Exemples sur Oracle (3/3)

```
SELECT TO_DATE
     ('02-19-2015','MM-DD-YYYY' )
     FROM DUAL;

SQL> SQL>   2    3
TO_DATE('0
----------
19/02/2015

1 ligne sélectionnée.
```

Exemples de fonctions sur dates

Emp (Eno, Ename, Title, City)
Pay(Title, Salary)
Project(Pno, Pname, Budget, City, StartingDate)
Works(Eno, Pno, Resp, Dur)

Nom des projets ayant commencé avant le 1er janvier 2015?

```
Attention !! avec StartingDate < '01/01/2015' comparaison de
   chaines de caractères (ordre alphanumérique) et non de dates
```

Afficher tous les projets commencé ce mois ?

Trouver le jour (lundi, mardi, etc) de votre naissance ?

Exemple Oracle

Jour de votre date de naissance (1 juillet 1992)

```
SELECT TO_CHAR(TO_DATE('01-07-1992','DD-
MM-YYYY'), 'DAY-DD-MM-YYYY')
    FROM DUAL;

SQL> SQL>  2
TO_CHAR(TO_DATE('01-07-1992','DD-MM-
YYYY'),'DAY-DD-MM-YYY
----------------------------------------------------
MERCREDI-01-07-1992
1 ligne sélectionnée.
```

Introduction aux Bases de données

Cours 5 : Requêtes SQL sur plusieurs tables

UFR 919 – Licence

2e année

Rappel : Requêtes simples

SELECT liste-colonnes
FROM Table
WHERE Condition;

Sélectionne

- les colonnes précisées dans le SELECT
- provenant de la table précisée dans le FROM
- dont les lignes vérifient les conditions précisées dans le WHERE.

Requêtes complexes : produit cartésien

SELECT Atts
FROM T1, T2 …, Tn
WHERE Condition ;

Atts *liste d'attributs* ∈Att(T1)*Att(T2)*…*Att(Tn)
Condition *contient des sous-conditions de la forme*
- Ti.a θTi.b ou
- Ti.a θ cste

où a *et* b *sont des attributs de* Ti, cste *une constante et* i=1..n

- Schéma = concaténation des attributs des Ti restreints à Atts
- Résultat = TOUTES les combinaisons des n-uplets de T1, …,Tn vérifiant Condition

Exemple

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	**M. Smith**	**Analyst**
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	**B. Casey**	**Analyst**
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	**J. Jones**	**Analyst**

PAY

TITLE	SALARY
Elect. Eng.	55000
Analyst	70000
Mech. Eng.	45000
Programmer	60000

SELECT **ENO, ENAME, EMP.TITLE, SALARY**
FROM EMP, PAY
WHERE **EMP.TITLE='Analyst'**

ENO	ENAME	EMP.TITLE	SALARY
E2	M. Smith	Analyst	55000
E2	M. Smith	Analyst	70000
E2	M. Smith	Analyst	45000
E2	M. Smith	Analyst	60000
E5	B. Casey	Analyst	55000
E5	B. Casey	Analyst	70000
E5	B. Casey	Analyst	45000
E5	B. Casey	Analyst	60000
E8	J. Jones	Analyst	55000
E8	J. Jones	Analyst	70000
E8	J. Jones	Analyst	45000
E8	J. Jones	Analyst	60000

Requêtes complexes : jointure

SELECT liste-colonnes
FROM T1, …, Tn
WHERE Condition;

Condition *contient des sous-conditions de la forme*
Ti.a θ Tj.b (condition de jointure) *ou* Ti.a θ cste

- Schéma avant la projection = concaténation des attributs des différentes tables Ti
- Résultat avant la projection = TOUTES les combinaisons des lignes de T1, …,Ti dont on ne garde que les lignes vérifiant Condition

Exemple

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	**M. Smith**	**Analyst**
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	**B. Casey**	**Analyst**
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	**J. Jones**	**Analyst**

PAY

TITLE	SALARY
Elect. Eng.	55000
Analyst	70000
Mech. Eng.	45000
Programmer	60000

SELECT **ENO, ENAME, EMP.TITLE, SALARY**
FROM EMP, PAY
WHERE **EMP.TITLE=PAY.TITLE**
And PAY.TITLE='Analyst';

EXERCICE : TROUVER LE RESULTAT

Jointure : remarques

- Un tuple de la table *T1* apparait dans le résultat de le jointure s'il joint **avec au moins 1** tuple de *T2*
- Très souvent, les jointures permettent de « traverser » les associations du schéma E/A
- Si un attribut de même nom dans les 2 tables alors nécessité de préfixer l'attribut par le nom de la table (ou par son renommage si la table est renommée)
- Si oubli de la condition de jointure... produit cartésien ! Donc si k tables dans le from, en général au minimum k-1 conditions de jointure dans le WHERE

Exemples

Emp (Eno, Ename, Title, City) **Project**(Pno, Pname, Budget, City)
Pay(Title, Salary) **Works**(Eno, Pno, Resp, Dur)

Noms et salaire des employés ?

Noms et titres des employés qui travaillent dans un projet pendant plus de 17 mois?

Exemples

Emp (Eno, Ename, Title, City) **Project**(Pno, Pname, Budget, City)
Pay(Title, Salary) **Works**(Eno, Pno, Resp, Dur)

Numéros et noms des projets dans lesquels a travaillé l'employé 10?

Noms et titres des employés qui travaillent dans un projet à Paris ?

Exemples

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	M. Smith	Analyst
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Analyst
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Analyst

PAY

TITLE	SALARY
Elect. Eng.	55000
Analyst	70000
Mech. Eng.	45000
Programmer	60000

```
SELECT ENO, ENAME, EMP.TITE, SALARY
FROM EMP, PAY
WHERE EMP.TITLE=PAY.TITLE
```

ENO	ENAME	EMP.TITLE	SALARY
E1	J. Doe	Elect. Eng.	55000
E2	M. Smith	Analyst	70000
E3	A. Lee	Mech. Eng.	45000
E4	J. Miller	Programmer	60000
E5	B. Casey	Analyst	70000
E6	L. Chu	Elect. Eng.	55000
E7	R. Davis	Mech. Eng.	45000
E8	J. Jones	Analyst	70000

Jointure : cas particulier

Emp (Eno, Ename, Title, City) **Project**(Pno, Pname, Budget, City)
Pay(Title, Salary) **Works**(Eno, Pno, Resp, Dur)

Nom des employés originaires de la même ville?

EMP :

Eno	Ename	Title	City
10	Pierre	Analyst	London
20	Lucy	Engineer	Paris
30	Yifan	RH	Munich
40	Anne	RH	London
50	Ryadh	Analyst	Paris

Résultat :

E1.Ename	E2.Ename
Pierre	Pierre
Pierre	Anne
Lucy	Lucy
Lucy	Ryadh
Yifan	Yifan
Anne	Anne
Anne	Pierre
Ryadh	Ryadh
Ryadh	Lucy

Cas particulier : l'auto-jointure

Auto-jointure : jointure impliquant 2 fois la même table

```
SELECT liste-colonnes
FROM  T1 var1, T1 var2
WHERE conditions;
```

- Renommage des tables dans le FROM obligatoire
- Tous les attributs préfixés par var1 et var2 car présents dans les 2 tables T1 et T2 respectivement
- Attention car un enregistrement dans T1 apparait aussi dans T2 (et donc peut joindre avec lui-même suivant le prédicat de jointure)

Auto-jointure : exemples

Comment améliorer ? 1ère idée

Nom des employés originaires de la même ville?

EMP :

Eno	Ename	Title	City
10	Pierre	Analyst	London
20	Lucy	Engineer	Paris
30	Yifan	RH	Munich
40	Anne	RH	London
50	Ryadh	Analyst	Paris

Résultat :

E1.Ename	E2.Ename
Pierre	Anne
Lucy	Ryadh
Anne	Pierre
Ryadh	Lucy

Auto-jointure : exemples

Comment améliorer ? 2ème idée

Nom des employés originaires de la même ville?

EMP :

Eno	Ename	Title	City
10	Pierre	Analyst	London
20	Lucy	Engineer	Paris
30	Yifan	RH	Munich
40	Anne	RH	London
50	Ryadh	Analyst	Paris

Résultat :

E1.Ename	E2.Ename
Lucy	Ryadh
Anne	Pierre

SQL : Requêtes imbriquées

Requête imbriquée dans la clause WHERE d'une requête externe:

```
SELECT ...
FROM   ...
WHERE  [Opérande] Opérateur (SELECT ...
                             FROM ...
                             WHERE ...)
```

3 imbrications possibles :

- **(A$_1$,...A$_n$) IN <sous-req>** exprime inclusion ensembliste
- **EXISTS <sous-req>** exprime condition d'existence
- **(A$_1$,...A$_n$) <comp> [ALL|ANY] <sous-req>** exprime comparaison avec quantificateur (ANY par défaut)

Opérateur IN

```
SELECT ...
FROM ...
WHERE (A1,...,An) [NOT] IN (SELECT B1,...,Bn
                            FROM   ...
                            WHERE  ...)
```

Sémantique : la condition est vraie si le n-uplet désigné par (A_1,..., A_n) de la requête *externe* appartient (n'appartient pas) au résultat de la requête *interne*.

Exemple avec IN

Emp (Eno, Ename, Title, City)
Pay(Title, Salary)

Project(Pno, Pname, Budget, City)
Works(Eno, Pno, Resp, Dur)

Noms des employés qui habitent dans des villes où il y a des [n'y a pas de] projets de budget inférieur à 50?

Imbrications avec IN : remarques

- La requête interne est effectuée dans un premier temps et son résultat est stocké (temporairement) en RAM (ou sur disque si trop gros!)
- La requête interne n'utilise pas la (les) table(s) de la requête externe
- Le IN étant ensembliste, les nuplets retournés par la requête interne doivent être du même format que le nuplet avant le IN (même nombre d'attributs et de même domaine, par forcément de même nom)
- Le SELECT de la requête externe ne peut retourner que des attributs appartenant aux tables placées dans son FROM

opérateur EXISTS

Q

```
SELECT …
FROM …
WHERE
```

Q'

```
[NOT] EXISTS (SELECT *
              FROM …
              WHERE P)
```

- *Sémantique opérationnelle :*
  - pour chaque n-uplet *x* de la requête externe Q, exécuter la requête interne Q'; s'il existe au moins un [s'il n'existe aucun] n-uplet y dans le résultat de la requête interne, alors sélectionner *x*.
- *Sémantique dénotationelle :*
  - $\{ x... \mid Q(x) \wedge [\neg]\exists y\ (Q'(y)) \}$

Imbrications avec EXISTS

- Les deux requêtes sont *corrélées* : la condition *P* dans la requête interne Q' exprime une jointure entre les tables de Q' et les tables de la requête externe Q.
  => requête interne exécutée pour chaque nuplet de la requête externe
- Seule l'existence d'un résultat importe d'où le * dans le SELECT de la requête interne
- Le SELECT de la requête externe ne peut retourner que des attributs appartenant aux tables placées dans son FROM

Exemples avec EXISTS 1/3

Emp (Eno, Ename, Title, City) **Project**(Pno, Pname, Budget, City)
Pay(Title, Salary) **Works**(Eno, Pno, Resp, Dur)

Noms des employés qui habitent dans une ville où il y a un projet?

Pour chaque employé on va vérifier si un projet est localisé dans sa ville.

Si oui, EXISTS vaut vrai et on retourne le nom de l'employé

Exemples avec EXISTS 2/3

Emp (Eno, Ename, Title, City) **Project**(Pno, Pname, Budget, City)
Pay(Title, Salary) **Works**(Eno, Pno, Resp, Dur)

Noms des employés qui habitent dans une ville sans projet?

Impossible à faire avec condition dans le WHERE

Exercice : exprimer cette requête à l'aide d'un *NOT IN*

Exemples avec EXISTS 3/3

Emp (Eno, Ename, Title, City) **Project**(Pno, Pname, Budget, City)
Pay(Title, Salary) **Works**(Eno, Pno, Resp, Dur)

Noms des projets ayant le plus grand budget?

Jointures et doublons

Soient les deux instances de relations suivantes

Eno	Ename	Title
10	Pierre	Analyst
20	Lucy	Engineer
30	Ryadh	Analyst

EMP

Title	Sal
Analyst	40K
Engineer	30K
Director	60K

Pay

R1 : select Pay.Sal from Emp, Pay where Emp.Title=Pay.Title;

R2 : select Sal from Pay where Title in (select Title from Emp);

R3 : select Sal from Pay p where exists (select * from Emp e where p.Title = e.Title) ;

Quelle(s) requête(s) retourne(nt) des doublons ? Expliquer.

Jointures et doublons

Soient les deux instances de relations suivantes

Eno	Ename	Title
10	Pierre	Analyst
20	Lucy	Engineer
30	Ryadh	Analyst

EMP

Title	Sal	increase
Analyst	40K	2 %
Engineer	30K	2 %
Director	60K	5 %

Pay

R'1 : select Pay.increase from Emp, Pay where Emp.Title=Pay.Title;

R'2 : select increase from Pay where Title in (select Title from Emp);

R'3 : select increase from Pay p where exists (select * from Emp e where p.Title = e.Title) ;

Est-ce que le distinct permet de rendre les requêtes équivalentes ?

Jointures et doublons

Soient les deux instances de relations suivantes

Eno	Ename	Title
10	Pierre	Analyst
20	Lucy	Engineer
30	Ryadh	Analyst

EMP

Title	Sal	increase
Analyst	40K	2 %
Engineer	30K	2 %
Director	60K	5 %

Pay

R'1 : select Pay.increase from Emp, Pay where Emp.Title=Pay.Title;

R'2 : select increase from Pay where Title in (select Title from Emp);

R'3 : select increase from Pay p where exists (select * from Emp e where p.Title = e.Title) ;

Est-ce que le distinct permet de rendre les requêtes équivalentes ?

Exercice

Soit le schéma suivant

Etudiants(matricule, nom, prenom, adresse, collaborateur*)

Inscriptions(matricule*, code*)

Modules(code, intitule, niveau, salle*)

Salles(numero, capacite, précédent *, suivante*)

Exprimer en SQL les requêtes suivantes :

1- Le matricule des étudiants inscrits dans le module 'BD' avec code module

2- Les intitulés des modules où 'Jack' est inscrit

3- Le noms et prénom des étudiants inscrits dans un même module que 'Jack'

4- Les étudiants qui ne sont pas inscrits au module '2I009'

5- Les étudiants inscrits dans au moins deux modules de L3

6- Les étudiants inscrits à au moins deux modules de L3 ou deux modules de M1