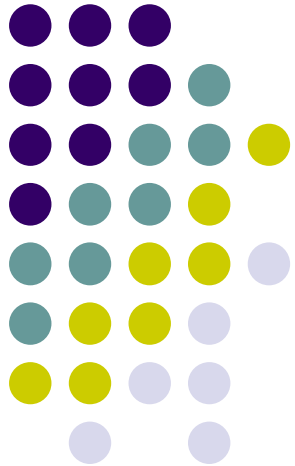
Bases de Données

Meltem Öztürk



- Introduction
- Modèle Entité/Association
- Modèle relationnel
- SQL

Introduction

Base de données:

- collection d'informations ou de données qui existent sur une ***longue période de temps*** et qui décrivent les **activités** d'une ou plusieurs organisations

- ensemble de données ***modélisant les objets d'une partie du monde réel*** et servant de support à une **application informatique**

- un gros ensemble d'informations ***structurées mémorisées*** sur un support permanent

Introduction

Exemples de BD:

- BD d'université (données sur les étudiants, les enseignements, les salles, etc.)
- BD de compagnie aérienne (données sur les clients, les vols, les réservations, etc.)
- BD bancaire (données sur les clients, les comptes, les transactions, etc.)

Introduction

SGBD

Systèmes de Gestion de Bases de Données (*DataBase Management Systems - DBMS*)

ensemble de logiciels systèmes permettant aux utilisateurs de faire des ***applications*** (insérer, modifier, et rechercher) efficacement des données spécifiques dans ***une grande masse d'informations*** (pouvant atteindre plusieurs milliards d'octets) ***partagée par de multiples utilisateurs***

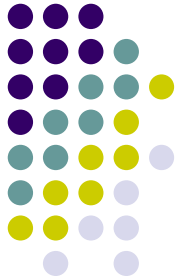
SGBD vs gestionnaire de fichier

- Banque : 500 Go de données

  (employés, départements, clients, services: retrait/crédit d'un compte, ajout d'un compte, solde d'un compte, relevé d'un compte, ...)
  - Accès concurrentiels aux données par +ieurs employés
  - Recherches rapides
  - Modifications des données sans incohérence
  - Restriction de l'accés de certaines données à certaines personnes

Problèmes en absence de SGBD

- Programmes d ’application écrits directement au-dessus du système de gestion de fichier
  - redondance (coût de stockage et d ’accès)
  - incohérence (ex: changement d ’adresse)
  - difficulté d ’accès (requêtes non prévues dans les programmes)
  - isolation des données (nouveau programme qui cherche des données dans des fichiers variés de différents formats)
  - anomalie due à la concurrence (deux opérations en même temps)
  - manque de sécurité
  - gestion de l’intégrité (obéir à des contraintes)



Introduction

Principales fonctionnalités d'un SGBD:

- **Contrôle de la redondance d'information**
- **Partage des données**
- **Gestion des autorisations d'accès**
- **Vérifications des contraintes d'intégrité**
  Contraintes structurelles (un employé a un seul chef), contraintes dynamiques (un salaire ne peut diminuer), etc.
- **Sécurité et reprise sur panne**
  Transactions, journalisation (version de sauvegarde+journal des mouvements)

Introduction

Différents langages d ’un SGBD :

- ***LDD*** : Langage de Définition de Données,
  - construire un schéma pour décrire la structure, incluant les contraintes
- ***LMD*** : Langage de Manipulation de Données
  - appliquer les opérations aux données (retrouver et mettre à jour les données)

Introduction

Modèle de données ***(décrire les données, les relations entre elles, leur sémantique, les contraintes d'intégrité, etc.):***

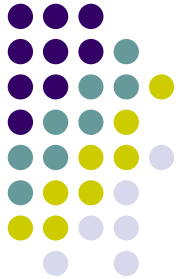
- Modèle conceptuel (entité/association)
  - Plus lisibles (graphiques)
  - Entité, association, attribut, identificateur, etc.

- Modèle logique (logique relationnel)
  - Plus facilement implantable
  - Relation, attribut, domaine, clé, n-uplet, etc.

Modèle Entité/Association

- Le modèle E/A
  - sert à concevoir une base de données.
  - est important pour le développement d'une application viable
  - est basé sur une représentation graphique
  - est simple et suffisamment puissant pour représenter des structures reationnelles
- Mais il a des insuffisances comme de ne proposer que des structures (pas de manipulations ou représentation des contraintes, etc.), de mener certains ambiguités pour des schémas complexes.

Exemple utilisé dans cette partie



- Bases de données sur les films
  - Films, années de production
  - Acteurs, metteurs en scène
  - Cinémas, scéances
  - Internautes, etc..

commençons par voir les difficultés qui apparaissent si on voit notre comme un simple fichier de texte ...

Exemple de base de données

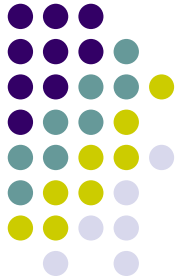
titre	année	nomMES	prénomMES	annéeNaiss
Alien	1979	Scott	Ridley	1943
Vertigo	1958	Hitchcock	Alfred	1899
Psychose	1960	Hitchcock	Alfred	1899
Kagemusha	1980	Kurosawa	Akira	1910
Volte-face	1997	Woo	John	1946
Pulp Fiction	1995	Tarantino	Quentin	1963
Titanic	1997	Cameron	James	1954
Sacrifice	1986	Tarkovski	Andrei	1932

Exemple de base de données

Critiques sur notre exemple: il est possible de représenter la même information +ieurs fois

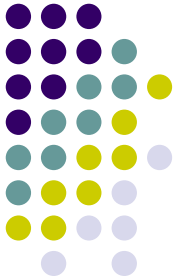
- Anomalie lors d'une insertion :
  - Insertion du même film plusieurs fois, et même avec différentes données!
  - Comment distinguer un film d'un autre?
- Anomalie lors d'une modification
  - Si on modifie la date de naissance de Hitchcock?
- Anomalie lors d'une destruction
  - Si on supprime un film, on supprime toutes données associées, y compris celles du réalisateur

Bonne méthode:



- Représenter individuellement les films et les réalisateurs : une action sur l'un n'entraîne pas systématiquement une action sur l'autre
- Définir une méthode d 'identification d 'un film ou d 'un réalisateur : permet d 'assurer que la même information est représentée une seule fois
- Préserver le lien entre les films et les réalisateurs : pour savoir quel metteur en scène a réalisé quel film

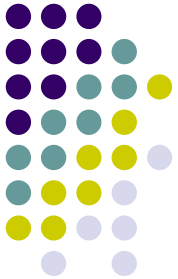
Changeons notre modèle



- Points 1 + Point 2 :
  - création de deux tables séparées (films et réalisateurs)
  - trouver une solution pour identifier d'une manière unique un film (son titre?) et un metteur en scène (les numérotés?).

Alors finis les redondances

Changeons notre modèle: Point 1+2



titre	année
Alien	1979
Vertigo	1958
Psychose	1960
Kagemusha	1980
Volte-face	1997
Pulp Fiction	1995
Titanic	1997
Sacrifice	1986

La table des films

id	nomMES	prénomMES	annéeNaiss
1	Scott	Ridley	1943
2	Hitchcock	Alfred	1899
3	Kurosawa	Akira	1910
4	Woo	John	1946
5	Tarantino	Quentin	1963
6	Cameron	James	1954
7	Tarkovski	Andrei	1932

La table des réalisateurs

Changeons notre modèle : point 3

titre	année	*idMES*
Alien	1979	1
Vertigo	1958	2
Psychose	1960	2
Kagemusha	1980	3
Volte-face	1997	4
Pulp Fiction	1995	5
Titanic	1997	6
Sacrifice	1986	7

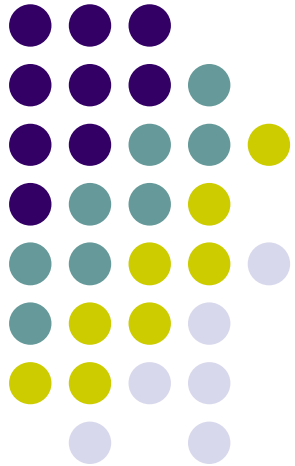
La table des films

id	nomMES	prénomMES	annéeNaiss
1	Scott	Ridley	1943
2	Hitchcock	Alfred	1899
3	Kurosawa	Akira	1910
4	Woo	John	1946
5	Tarantino	Quentin	1963
6	Cameron	James	1954
7	Tarkovski	Andrei	1932

La table des réalisateurs

Bases de Données

Meltem Öztürk



- Introduction
- Modèle Entité/Association
- Modèle relationnel
- SQL

Type d ’entité

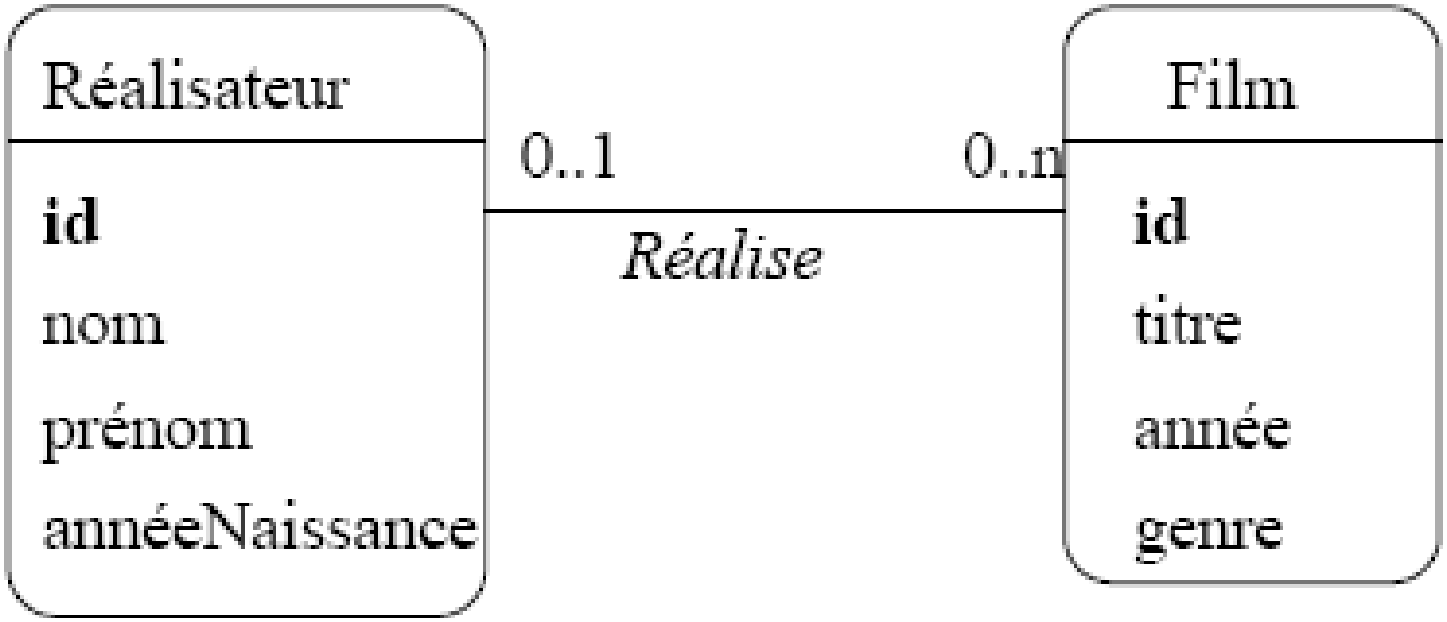


FIG. 3.4 – *Représentation de l’association.*

Schéma + complet de notre exemple

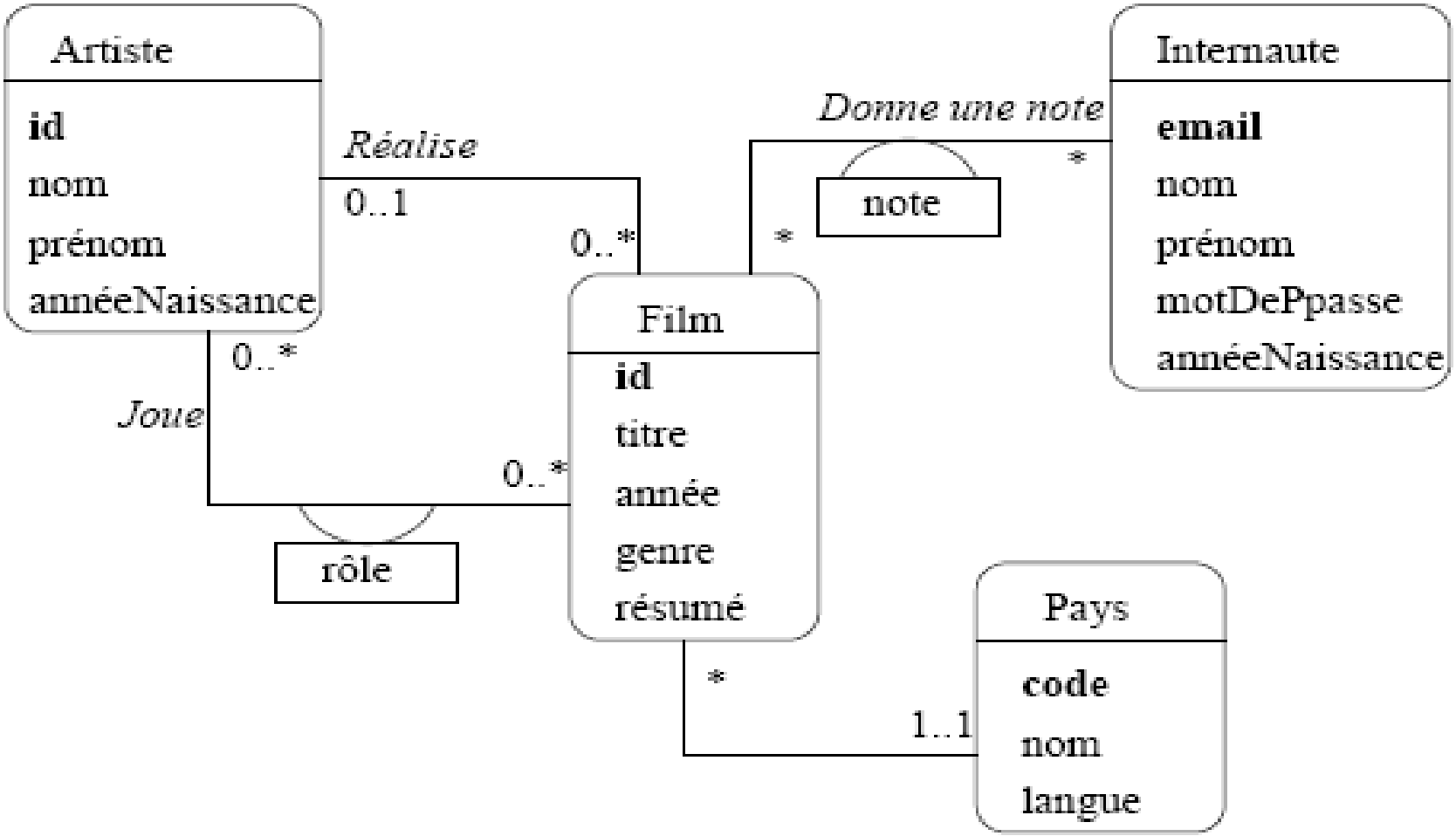
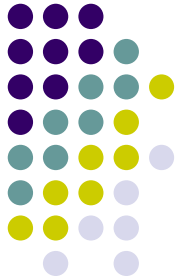


FIG. 3.1 – *Le schéma de la base de données Films*

Modèle Entité/Association



- ***Entité*** : objet concret ou abstrait, identifiable, décrit par l'information et pertinent pour l'application
  - Ex : film, réalisateurs

- Une entité est représentée par un ensemble d' ***attributs*** qui la décrive.
  - Ex : film décrit par le nom, l'année de création, …

Modèle Entité/Association

- Chaque attribut a ***un domaine*** qui correspond à l'ensemble des valeurs qu'il peut prendre
  - Ex : les années compris entre 1920-2005

- Attribut : fonction définie sur un ensemble d'entités et prenant ses valeurs dans un domain D
  - Ex : titre(1)=Alien, année(1)=1979, etc.
  - On ne peut pas avoir plus d'une valeur! : certaines méthodes admettent l'introduction de construction + complexes
    - Attributs multivalués
    - Attributs composés

Type d'entité

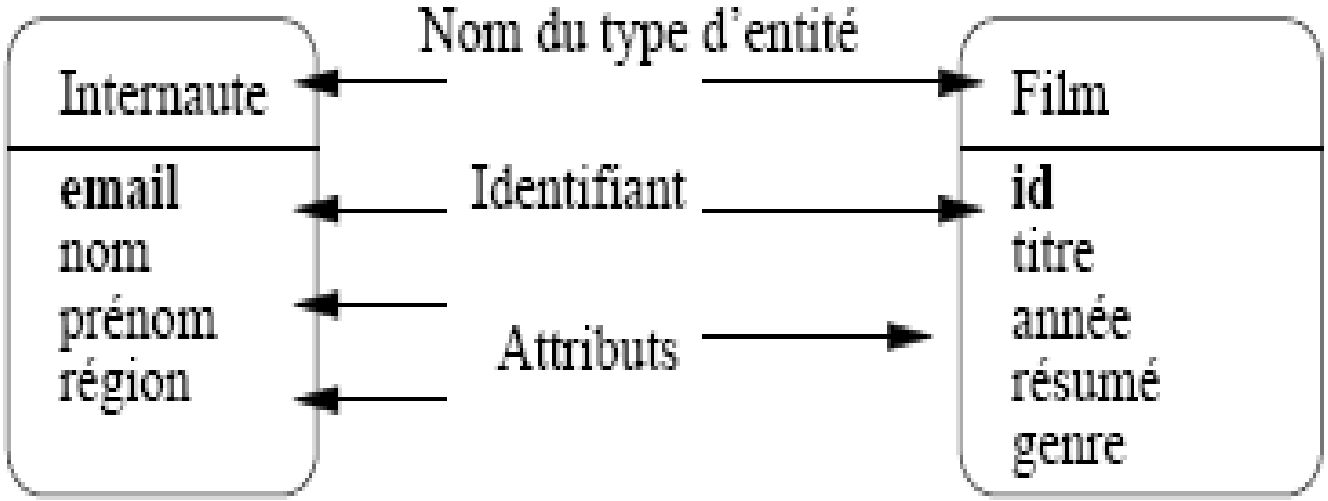


FIG. 3.2 – *Représentation des types d'entité*

Modèle Entité/Association

- ***Clé (Identificateur)*** : Un sous-ensemble d'attribut permettant d'identifier l'entité de manière unique.

- sa valeur doit être connue pour toute entité
- on ne doit jamais avoir besoin de le modifier
- sa taille de stockage doit être la plus petite possible

- Exemple:
  - internaute: e-mail
  - film: nom+pays+année?
    - Identificateur abstrait: numéro séquentiel qui sera incrémenté au fur et à mesure des insertions

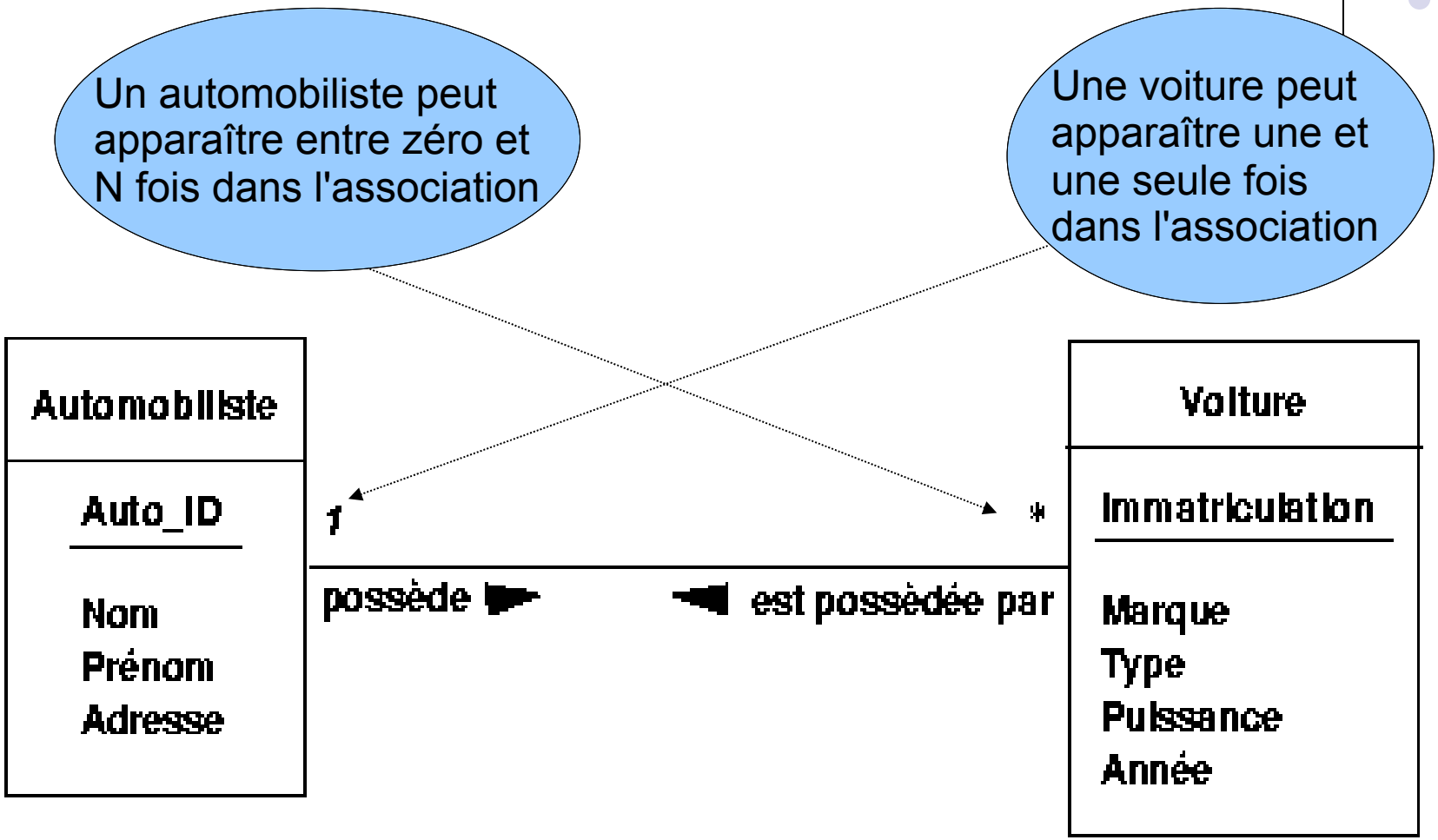
Modèle Entité/Association

- Une ***association binaire*** entre les ensembles d 'entités E_1 et E_2 est un ensemble de couple (a,b) avec $a \in E_1$ et $b \in E_2$
  - Ex: Hitchcock *a réalisé* Vertigo

- ***Cardinalité*** d'une association : nombre d'entités que l'association relie.
  - Noté (Min, Max)
  - « * » : « 0. * », « 1 »: « 1.1 »

- Une association peut avoir des attributs

Modélisation UML



Attribut d'une association

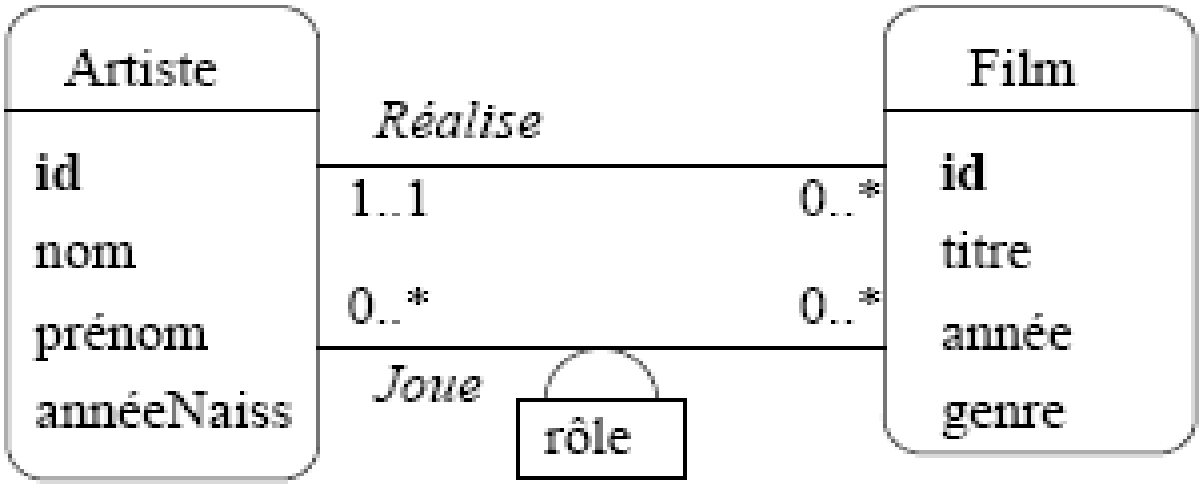
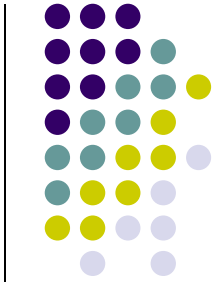


FIG. 3.6 – *Associations entre* Artiste *et* Film.

Clé d ’une association



- ***Clé d’une association*** : couple formé des clés des deux entités
  - ex: l ’association « note » entre internaute et film!
  - Comment faire si l ’internaute veut donner différentes notes pour le même film à des dates différentes? (pas de liens multiples!!)
  - Solution : ajouter un nouveau type d ’entité: date

Association n-aires

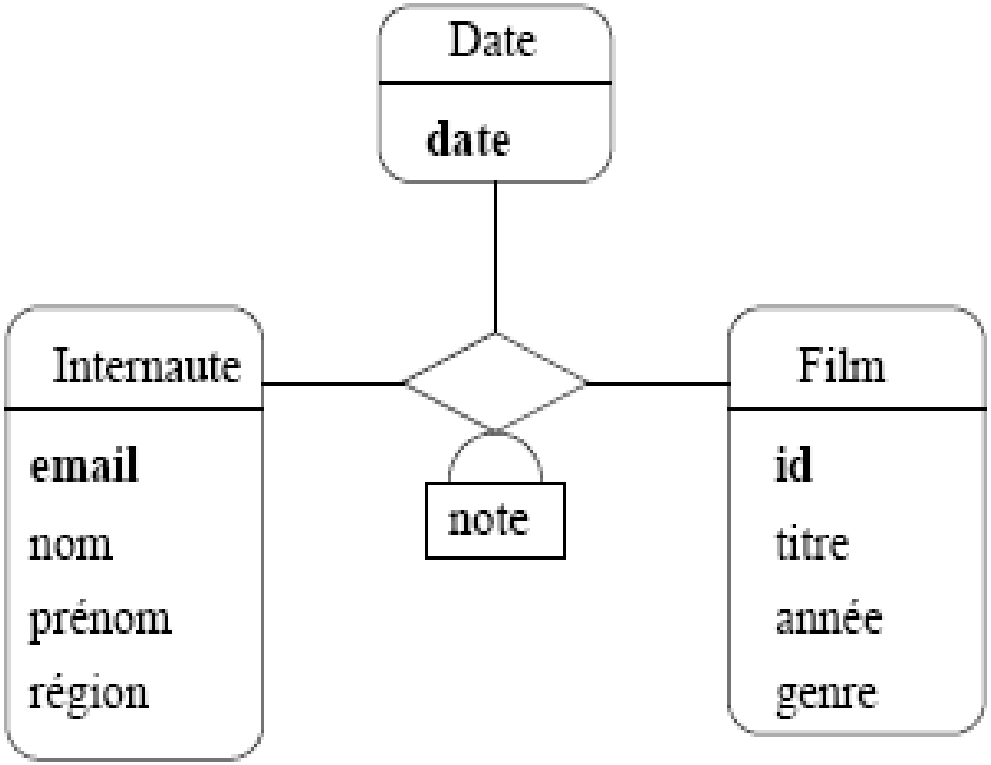


FIG. 3.7 – *Ajout d'une entité* Date *pour conserver l'historique des notations*

Association avec clé

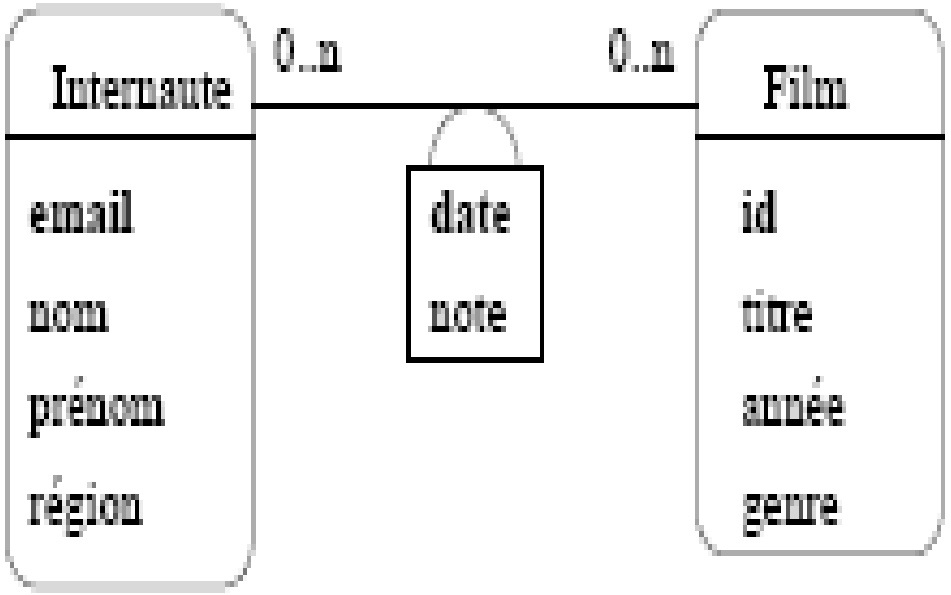


FIG. 3.9 – *Notation abrégée d'une association avec un type d'entité* Date

Association n-aire

- Une ***association n-aire*** entre n types entités E_1, E_2, …, En est un ensemble de n-uplets (e_1,e_2, …,e_3) avec $e_i \in E_i$ pour tout i.

  - difficile à comprendre
  - cardinalité ambiguë : explicitement de type « 0..* »
  - clé?
    - Ex: [nomcinéma, noSalle, idFilm, idHorraire]?

      - Contrainte du type « dans une salle pour une horaire donnée il n ’y a qu ’un seul film »… [nomcinéma, noSalle, idHorraire]?
      - « Connaissant le film et l ’horaire je connais la salle »: [idFilm, idHorraire]?
      - Donc on a des clés *candidates*

Association n-aire (cardinalité « 0.*», clé ?)

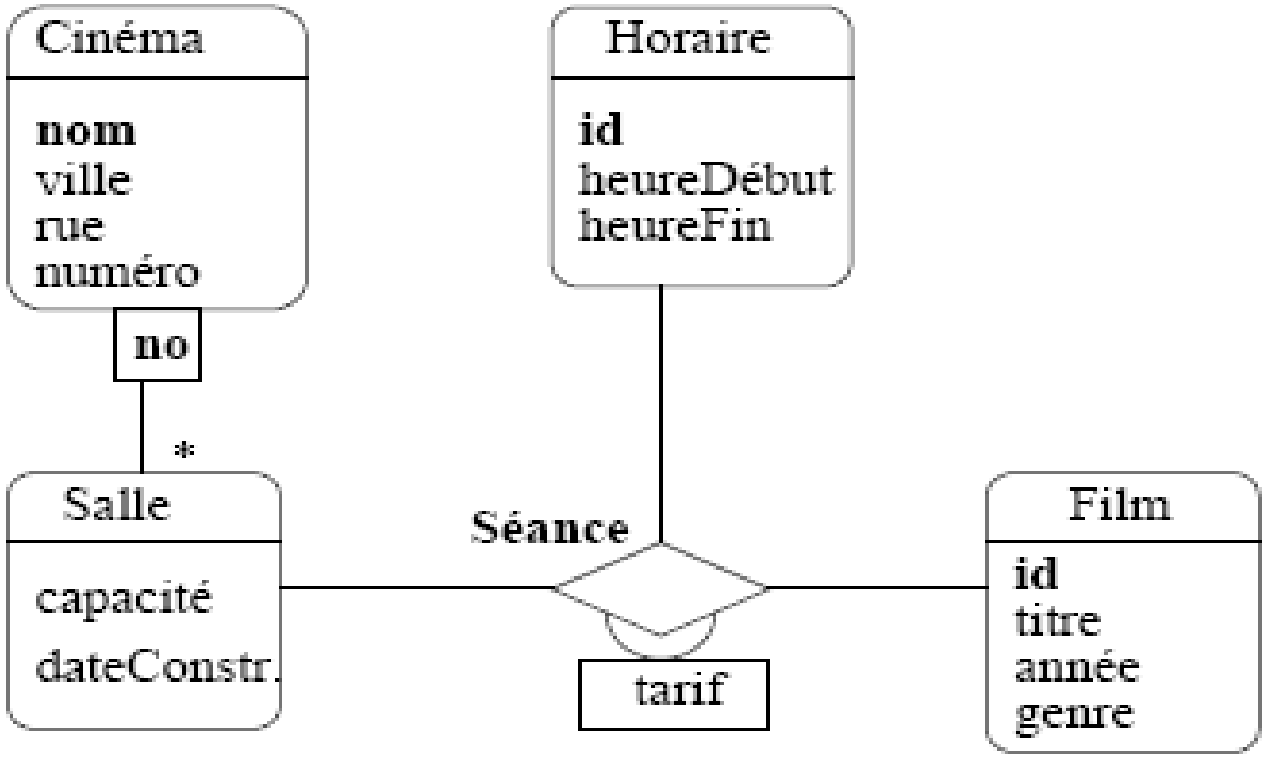


FIG. 3.11 – *Association ternaire représentant les séances*

Association n-aire (cardinalité « 0.*», clé ?)

- Remplacer l ’association par un type d ’entité!

- Règles:
  - on attribue un identifiant autonome à l’association
  - on crée une association Ai de type «1.1» « 0. *» entre l ’association et chacun des types d ’entité.

Association ternaire - - > entité

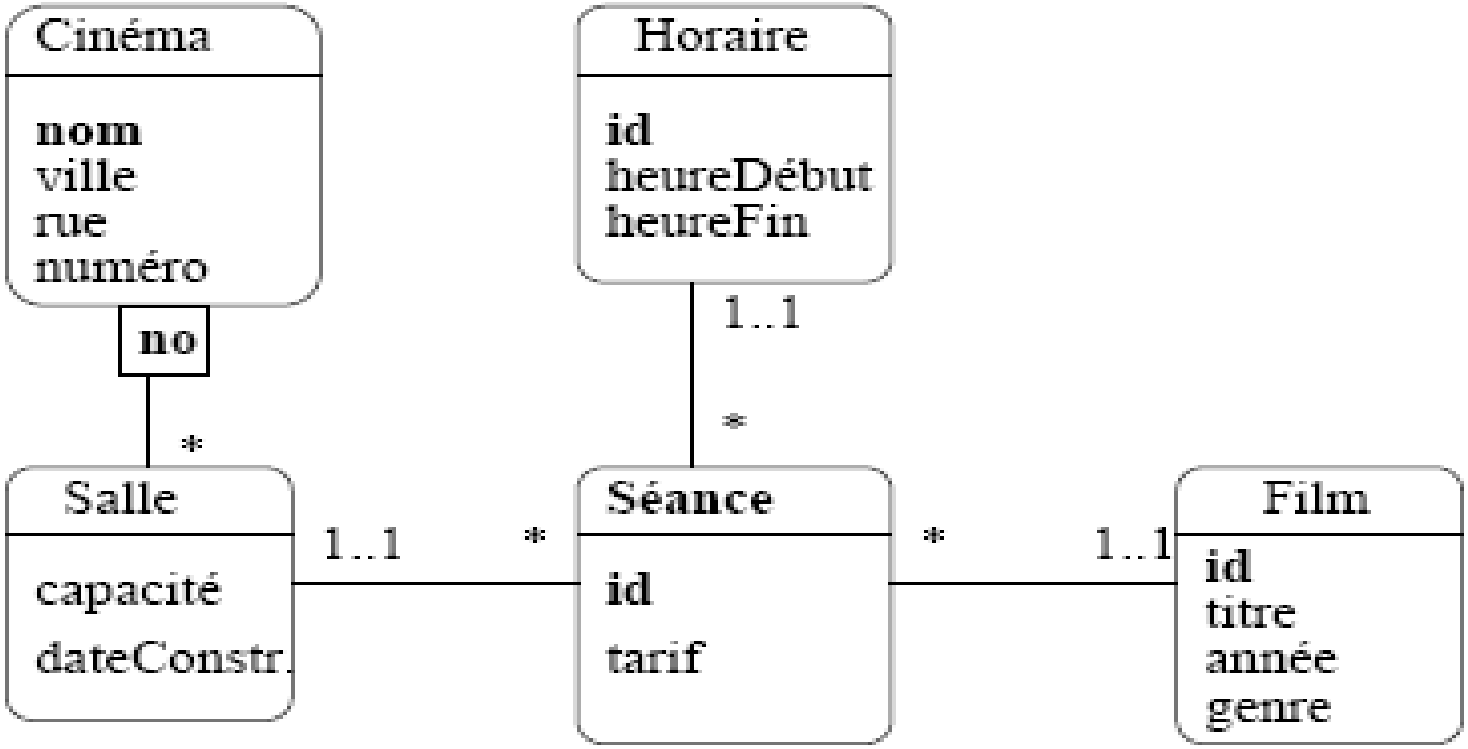


FIG. 3.13 – *L'association* Séance *transformée en entité*

Contraintes

Contraintes d'intégrité :

toutes règles implicites ou explicites que doivent suivre les données [Gar99]

- **Contraintes d'entité**: toute entité doit posséder un identificateur
- **Contraintes de domaine** : les valeurs de certains attributs doivent être prises dans un ensemble donné
- **Contraintes d'unicité** : une valeur d'attribut ne peut pas être affectée deux fois à deux entités différentes
- **Contraintes générales** : règle permettant de conserver la cohérence de la base de manière générale

Exemples de contraintes

- **Contraintes de domaine** :

  "La fonction d'un enseignant à l'Université prend sa valeur dans l'ensemble {vacataire, moniteur, ATER, MCF, Prof., PRAG, PAST}."

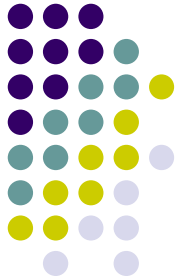
- **Contraintes d'unicité** :

  "Un département, identifié par son numéro, a un nom unique (il n'y a pas deux départements de même nom)."

- **Contraintes générales** :

  "Un même examen ne peut pas avoir lieu dans deux salles différentes à la même date et à la même heure. "

(Des)Avantages du modèle E/A

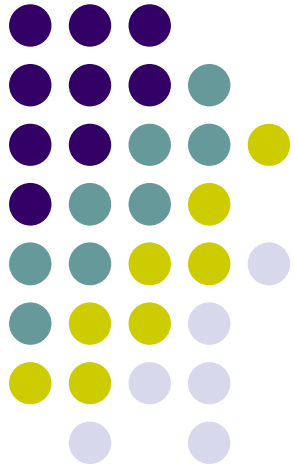


- Avantages
  - Que 3 concepts: entités, associations, attributs
  - Représentation graphique et rapide

- Désavantages
  - Pas de règle absolue pour déterminer qui est attribut, entité ou association…

Bases de Données

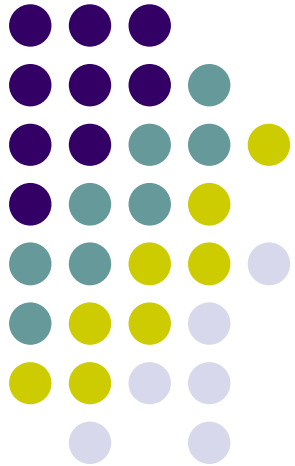
Meltem Öztürk



- Introduction
- Modèle Entité/Association
- Modèle relationnel
- SQL

Bases de Données

Meltem Öztürk

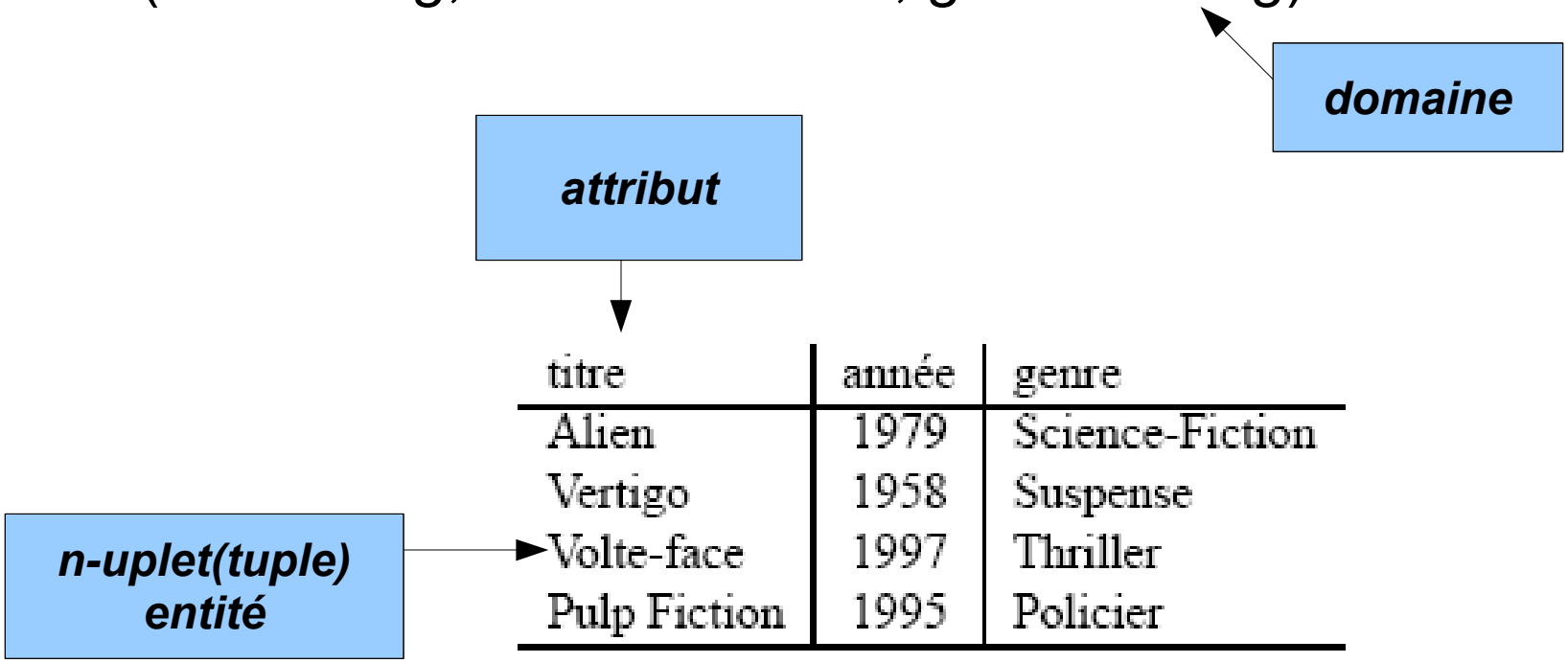


- Introduction
- Modèle Entité/Association
- Modèle relationnel
- SQL

Modèle relationnel

La relation du « nom » Film :

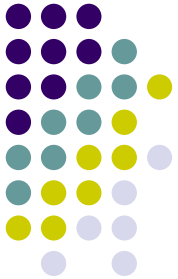
Film (titre: string, année: number, genre : string)



titre	année	genre
Alien	1979	Science-Fiction
Vertigo	1958	Suspense
Volte-face	1997	Thriller
Pulp Fiction	1995	Policier

FIG. 4.1 – *Une relation*

Modèle relationnel

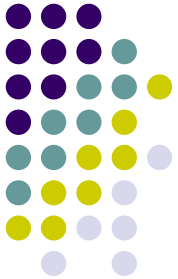


- ***Domaine*** : ensemble d'instance d'un type élémentaire
  - Ex : réels, boolean, chaine de caractères, etc.

- ***Attribut*** : colonne d'une relation, associé a un domaine

- ***Schéma de relation*** : nom suivi de la liste des attributs, chq attribut étant associé à son domain
  - $R(A_1 : D_1, A_2 : D_2, \ldots, A_n : D_n)$
  - Ex : *Film* (titre: string, année: number, genre : string)

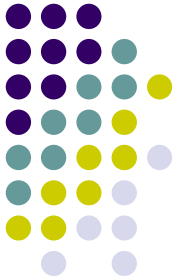
Modèle relationnel



- ***Relation (R)*** : sous-ens. fini du produit cartésien des domaines des attributs de R
  (produit cartésien $D_1 x D_2 x \ldots x D_n$ est l 'ens. de tous les tuples (v_1, … v_n) où $v_i \in D_i$)

  - représentée par une table à 2 dimensions
  - colonne = domaine du produit cartésien
  - même domaine peut apparaître +ieurs fois
  - ensemble de tuplets sans doublons (pas 2 fois même ligne)
  - ordre des lignes n'a pas d'importance
  - pas de case vide dans la table

Modèle relationnel



- ***Clé d'une relation*** : le + petit sous-ens. des attributs qui permet d'identifier chq ligne d'une manière unique
  - ex : film (**titre**, année, genre)

- ***Tuple (n-uplet)*** : une liste de n valeurs (v_1, …v_n), où chq v_i est la valeur d'un attribut A_i du domain D_i
  - ex : ('Cyrano', 1992, 'Rappeneau')

- ***Base de données*** : ensemble fini de relations.

Passage de E/A au relationnel

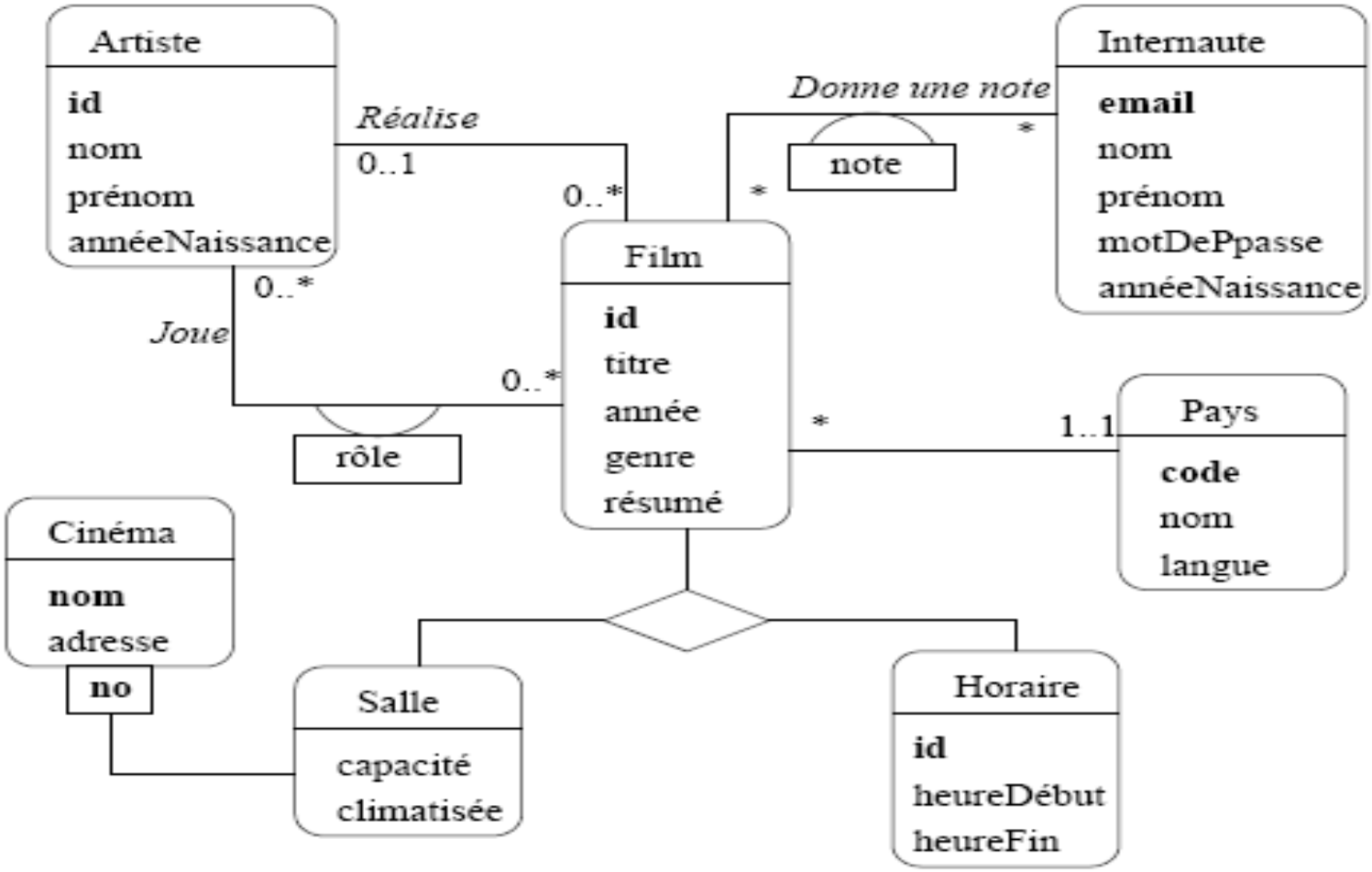


FIG. 4.2 – *Le schéma de la base de données Films*

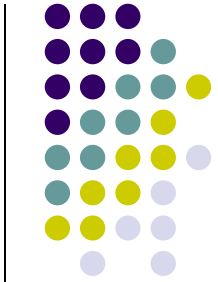
Règles de passages (1)

- Règles générales – Entité :
  1. On crée une relation de même nom que l'entité.
  2. Chaque propriété de l'entité, y compris l'identifiant, devient un attribut de la relation.
  3. Les attributs de l'identifiant constituent la clé de la relation.

  Ex:

  - *Film (**idFilm**, titre, année, genre, résumé)*
  - *Artiste (**idArtiste**, nom, prénom, annéeNaissance)*
  - *Internaute (**email**, nom, prénom, région)*
  - *Pays (**code**, nom, langue)*

Règles de passages (2)



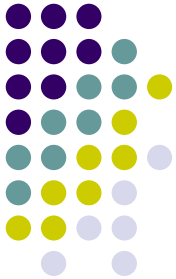
Association de un à plusieurs :

- B est en association avec plusieurs A et A est en relation avec un seul B.
- B est dit « père », A est dit « fils » (A: film, B: réalisateurs)
- On crée les relations R_a et R_b correspondant respectivement aux entités A et B.
- L'identifiant de B devient un attribut de Ra (id du père devient attribut de son fils)

ex:

- *Film (**idFilm**, titre, année, genre, résumé, idArtiste, codePays)*
- *Artiste (**idArtiste**, nom, prénom, annéeNaissance)*
- *Pays (**code**, nom, langue)*

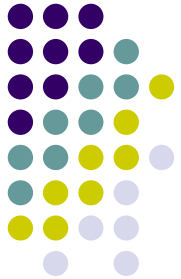
Exemple de passage



idFilm	titre	année	genre	*idMES*	*codePays*
100	Alien	1979	Science Fiction	1	US
101	Vertigo	1958	Suspense	2	US
102	Psychose	1960	Suspense	2	US
103	Kagemusha	1980	Drame	3	JP
104	Volte-face	1997	Action	4	US
105	Van Gogh	1991	Drame	8	FR
106	Titanic	1997	Drame	6	US
107	Sacrifice	1986	Drame	7	FR

La table *Film*

Règles de passage (3)

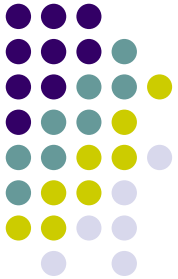


Association avec entité faible : même que le passage des associations du type de un à plusieurs, sauf que l'identifiant du père fait partie de l'identifiant du fils.

Ex:

- *Cinéma (**nomCinéma**, numéro, rue, ville)*
- *Salle (**nomCinéma, no**, capacité)*

Règles de passage (4)



Association binaire de plusieurs à plusieurs :

1. On crée les relations Ra et Rb des entités A et B.
2. On crée une relation Ra+b pour l'association
3. La clé de Ra et la clé de Rb deviennent des attributs de Ra+b
4. La clé de cette relation est la concaténation des clés des relations Ra et Rb
5. Les propriétés de l'association deviennent des attributs de Ra+b

Ex :

- *Film (**idFilm**, titre, année, genre, résumé, idMES, codePays)*
- *Artiste (**idArtiste**, nom, prénom, annéeNaissance)*
- *Rôle (**idFilm, idActeur**, nomRôle)*

Exemple de passage

Ex :

- *Film (**idFilm**, titre, année, genre, résumé, idMES, codePays)*
- *Artiste (**idArtiste**, nom, prénom, annéeNaissance)*
- *Role (**idFilm, idActeur**, nomRôle)*

idFilm	*idActeur*	rôle
20	100	William Munny
20	101	Little Bill
21	101	Bril
21	103	Robert Dean

La table *Rôle*

Règles de passage (5)

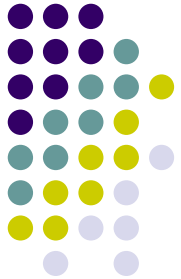
Association ternaire :

2. Même principe d'une association binaire:
   - Salle (***nomCinéma*, no**, capacité)
   - Film (**idFilm**, titre, année, genre, résumé, *idMES*, *codePays*)
   - Horaire (**idHoraire**, heureDébut, heureFin)
   - Séance (***idFilm***, ***nomCinéma***, ***noSalle***, ***idHoraire***, tarif)

idFilm	*nomCinéma*	*noSalle*	*idHoraire*	tarif
103	Le Rex	2	1	23
103	Le Rex	2	2	56
100	Kino	1	2	45
102	Le Rex	2	1	50

La table *Séance*

Exemple de passage



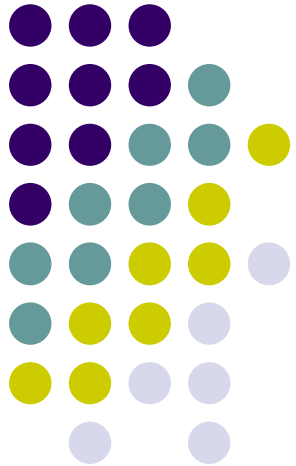
- Problème avec notre exemple : même salle présente deux films différents au même horaire

  - Salle (***nomCinéma*, no**, capacité)
  - Film (**idFilm**, titre, année, genre, résumé, *idMES*, *codePays*)
  - Horaire (**idHoraire**, heureDébut, heureFin)
  - Séance (***nomCinéma***, ***noSalle***, ***idHoraire***, tarif)

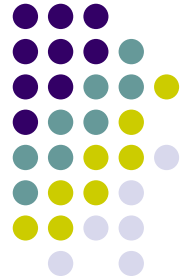
Clé de la relation est un sous-ensemble de la concaténation des clés

Bases de Données

- Introduction
- Modèle Entité/Association
- Modèle relationnel
- SQL

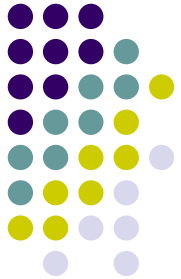


Exemple:organisme de voyage



- Station (**nomStation**, capacité, lieu, région, tarif)
- Activite (***nomStation***, **libellé**, prix)
- Client (**id**, nom, prénom, ville, région, solde)
- Séjour (***idClient***, **station**, **début**, nbPlaces)

SQL / Requêtes



```
SELECT nomStation
FROM Station
WHERE region = ’Antilles’
```

- FROM indique la (ou les) tables dans lesquelles on trouve les attributs utiles à la requête.
- SELECT indique la liste des attributs constituant le résultat.
- WHERE indique les conditions que doivent satisfaire les n-uplets de la base pour faire partie du résultat.

SQL / Requêtes

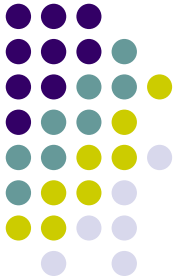
```
SELECT libelle, prix
FROM Activite
WHERE nomStation = 'Santalba'
```

```
SELECT libelle, prix / 6.56
FROM Activite
WHERE nomStation = 'Santalba'
```

```
SELECT libelle, prix / 6.56, 'Cours de l'euro = ', 6.56
FROM Activite
WHERE nomStation = 'Santalba'
```

libelle	prix / 6.56	'Cours de l'euro ='	'6.56'
Kayac	7.62	'Cours de l'euro ='	6.56

SQL / Requêtes



```
SELECT libelle, prix / 6.56 AS prixEnEuros,
                'Cours de l'euro = ', 6.56 AS cours
FROM Activite
WHERE nomStation = 'Santalba'
```

libelle	prixEnEuros	'Cours de l'euro ='	cours
Kayac	7.69	'Cours de l'euro ='	6.56

SQL / Requêtes

- Doublons :

SELECT libelle
FROM Activite

libelle
Voile
Plongee
Plongee
Ski
Piscine
Kayac

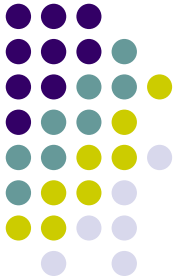
SELECT **DISTINCT** libelle
FROM Activite

SQL / Requêtes

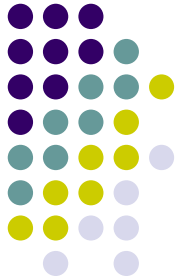
SELECT *
FROM Station

Tri du résultat
(ascendant; pour descendant ajout DESC)

SELECT *
FROM Station
ORDER BY tarif, nomStation



SQL / Requêtes

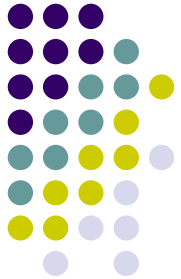


- Where :

AND, OR, NOT, <, <=, >, >=, <>, !=,BETWEEN

```
SELECT nomStation, libelle
FROM Activite
WHERE nomStation = 'Santalba'
AND (prix>50 AND prix<120)
```

SQL / Requêtes

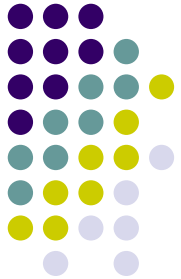


- Where :

AND, OR, NOT, <, <=, >, >=, <>, !=,BETWEEN

SELECT nomStation, libelle
FROM Activite
WHERE nomStation = ’Santalba’
AND prix **BETWEEN** 50 AND 120

SQL / Requêtes

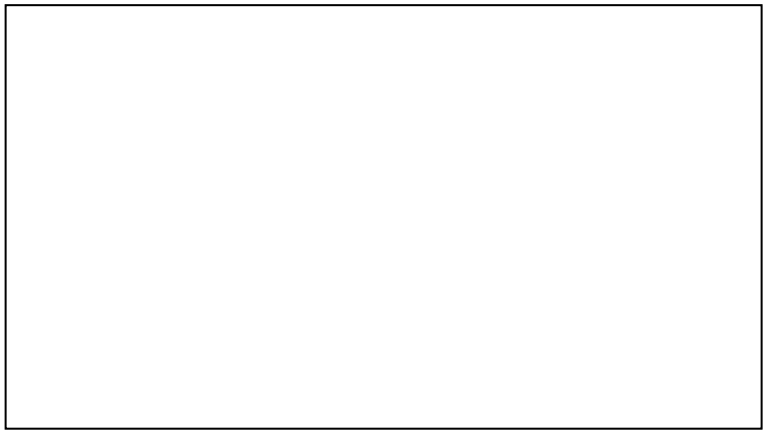


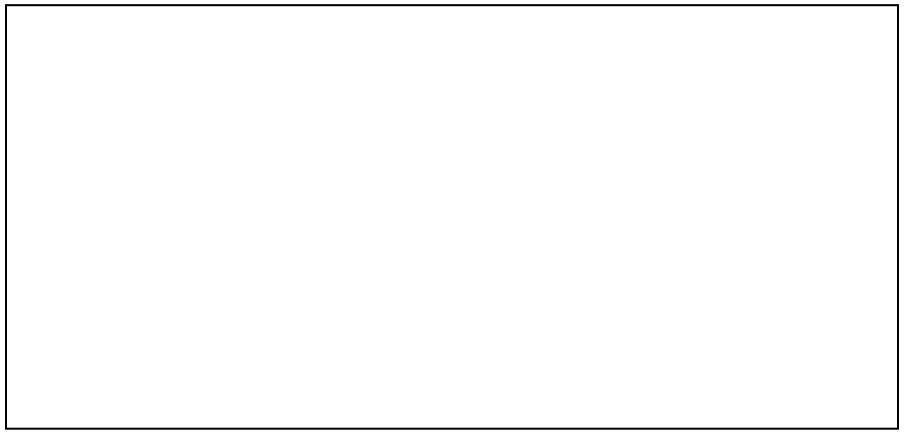
- Chaîne de caractères:
  - Attention: chaînes de longueur fixe différentes des chaînes de longueur variable.
  - Si SQL ne distingue pas majuscules et minuscules pour les mot-clés, il n'en va pas de même pour les valeurs. Donc 'SANTALBA' est différent de 'Santalba'.

SQL / Requêtes

- Chaîne de caractères:
  - LIKE : pattern matching
  - ‘_’ : n’importe quel caractère
  - ‘%’ : n’importe quelle chaîne de caractères

Ex:



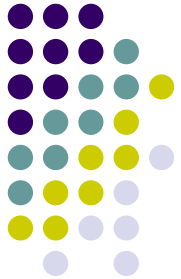


SQL/Requêtes

- Dates: aaaa-mm-jj

```
SELECT idClient
FROM Sejour
WHERE debut BETWEEN DATE ’1998-07-01’
                AND DATE ’1998-07-31’
```

SQL/Requêtes



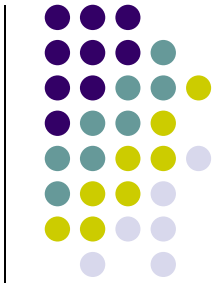
- Valeurs nulles: NULL
  - Toute opération appliquée à NULL donne pour résultat NULL.
  - Toute comparaison avec NULL donne un résultat qui n'est ni vrai, ni faux mais une troisième valeur booléenne, UNKNOWN.
  - TRUE=1, FALSE=0 et UNKNOWN=1/2.
    - x AND y=Min(x,y);
    - x OR y= Max(x,y);
    - NOT x=1-x

SQL/Requêtes

```
SELECT station
FROM Sejour
WHERE nbPlaces <= 10 OR nbPlaces >= 10
```

SEJOUR			
idClient	station	début	nbPlaces
10	Passac	1998-07-01	2
20	Santalba	1998-08-03	
30	Passac		3

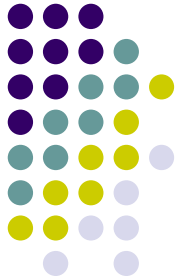
SQL/Requêtes



- nbPlaces = NULL est incorrecte.

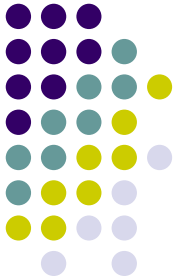
```
SELECT *
FROM Sejour
WHERE nbPlaces IS NOT NULL
```

SQL/Agrégation



- *COUNT* : compte le nombre de valeurs *non nulles*.
- *MAX* et *MIN*
- *AVG* : calcule la moyenne des valeurs de la colonne.
- *SUM* : effectue le cumul.

SQL/Agrégation



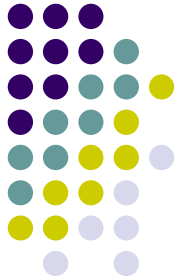
- Exemples:

  SELECT COUNT(nomStation) AVG(tarif)
  MIN(tarif) MAX(tarif)
  FROM Station

  Requête incorrecte:

  SELECT nomStation, AVG(tarif)
  MIN(tarif) MAX(tarif)
  FROM Station

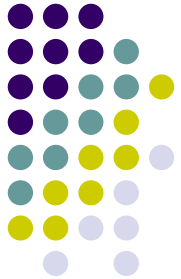
SQL/Agrégation



- Combien de place a reservé Mr Kerouac pour l ’ensemble des séjours?

```
SELECT SUM (nbPlaces)
FROM Client, Sejour
WHERE nom=`Kerouac`
AND id=idClient
```

SQL/Requêtes =ieurs tables



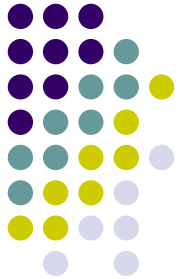
```
SELECT nom, station
FROM Client, Sejour
WHERE id = idClient
```

SQL/Requêtes =ieurs tables

```
SELECT nomStation, tarif, libelle, prix
FROM Station, Activite
WHERE Station.nomStation = Activite.nomStation

SELECT S.nomStation, tarif, libelle, prix
FROM Station S, Activite A
WHERE S.nomStation = A.nomStation
```

SQL/Requêtes =ieurs tables



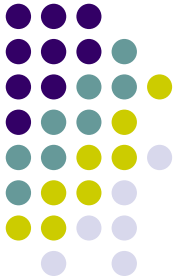
```
SELECT nom, station, debut, tarif
FROM Client, Sejour, Station
WHERE ville = ’Paris’
            AND id = idClient
            AND station = nomStation
```

SQL/Requêtes =ieurs tables

- *Donner les couples de stations situées dans la même région*:

```
SELECT s1.nomStation, s2.nomStation
FROM Station s1, Station s2
WHERE s1.region = s2.region
```

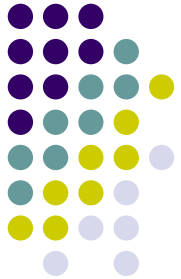
GROUP BY



- Afficher les régions avec le nombre de stations:

```
SELECT region, COUNT(nomStation)
FROM Station
GROUP BY region
```

GROUP BY



- Afficher le nombre de places reservées par client:

- SELECT nom, SUM(nbPlaces)
- FROM Client, Sejour
- WHERE id=idClient
- GROUP BY id, nom

HAVING

- Afficher le nombre de places réservées, par client, pour les clients ayant réservé plus de 10 places:

- SELECT nom, SUM(nbPlaces)
- FROM Client, Séjour
- WHERE id=idClient
- GRUOP BY nom
- HAVING SUM(nbPlaces)>=10

SQL/Requêtes imbriquées

- Noms de stations où ont séjourné des clients parisiens:

```
SELECT station
FROM Sejour, Client
WHERE id=idclient
AND ville =‘ Paris ’
```

```
SELECT station
FROM Sejour
WHERE idclient IN (SELECT id
      FROM Client
      WHERE ville =‘ Paris ’)
```

SQL/Requêtes imbriquées

- !!! Si on sait que la sous requête ramène un et un seul tuple:

```
SELECT station
FROM Sejour
WHERE idclient = (SELECT id
      FROM Client
      WHERE ville =‘ Paris ’)
```

SQL/Requêtes imbriquées

- *EXISTS R*. Renvoie *TRUE* si *R* n 'est pas vide, *FALSE* sinon.
- *t IN R*, est un tuple dont le type est celui de *R*. *TRUE* si *t* appartient à *R*, *FALSE* sinon.
- *v cmp ANY R*, où *cmp* est un comparateur SQL (<,>,=, etc.). Renvoie *TRUE* si la comparaison avec *au moins un* des tuples de la relation unaire *R* renvoie *TRUE.*
- *v cmp ALL R*, où *cmp* est un comparateur SQL (<,>,=, etc.). Renvoie *TRUE* si la comparaison avec *tous les tuples* de la relation unaire *R* renvoie *TRUE*

SQL/Requêtes imbriquées

- *Où (station, lieu) ne peut-on pas faire du ski ?*

```
SELECT nomStation, lieu
FROM Station
WHERE nomStation NOT IN
            (SELECT nomStation FROM Activité
             WHERE libelle = `ski`)
```

SQL/Requêtes imbriquées

- *Quelle station pratique le tarif le plus élevé?*

```
SELECT nomStation
FROM Station
WHERE tarif >= ALL (SELECT tarif FROM Station)
```

SQL/Requêtes imbriquées

- Dans quelle station pratique-t-on une activité aux même prix qu ’à Santalba?

```
SELECT NomStation, libelle
FROM Activite
WHERE prix IN (SELECT prix FROM Activite
                    WHERE NomStation=`Santalba`)
```

SQL/Requêtes =ieurs tables

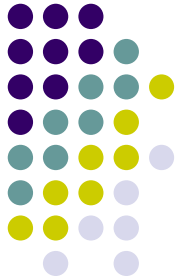
- Donnez tous les noms de région dans la base:

```
SELECT region FROM Station
UNION
SELECT region FROM Client
```

- Donnez les régions où l'on trouve à la fois des clients et des stations:

```
SELECT region FROM Station
INTERSECT
SELECT region FROM Client
```

SQL/Requêtes =ieurs tables



- Quelles sont les régions où l'on trouve des stations mais pas des clients ?

```
SELECT region FROM Station
EXCEPT
SELECT region FROM Client
```