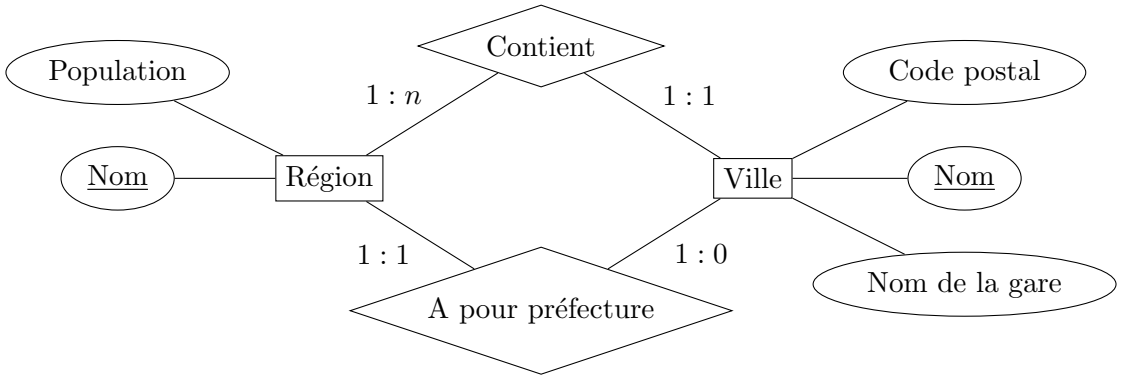
Modélisation, ANY, ALL, Vues

1 TD

1.1 Modélisation

Voici un exemple d'un schéma Entité-Association. Un schéma de base de données associé



peut être :

```
Region(Nom, population)
Ville(Nom,Nom_Gare,Code_Postale)
```

Les contraintes peuvent être de plusieurs types. Cette liste n'est pas exhaustive :

- *Contraintes de propriété-attribut* : forme, liste de valeurs, fourchette de valeurs possibles, stabilité des propriétés. Par exemple, le Code_Postal doit être un nombre à 5 chiffres.
- *Contrainte d'intégrité Fonctionnelle* : une des entités est totalement déterminée par la connaissance d'une ou plusieurs autres entités. Par exemple, une région détermine une ville, sa préfecture.
- *Contraintes inter-association* : exclusivité (même instance ne peut participer à deux associations), simultanéité (toute instance qui participe à une association participe à l'autre), totalité (toutes les instances d'une association doivent participer à au moins d'une des associations d'une collection d'association), inclusion... Par exemple, chaque ville se trouve dans une et une seule région.

Les règles de bonne conduite :

1. Chaque entité-type doit être décrite par des propriétés qui lui sont propres.
2. Distinguer entre l'élément type et l'élément individuel représenté par l'occurrence.
3. Déterminer la ou les propriétés jouant le rôle d'identifiant. Cette propriété doit permettre de distinguer entre les occurrences de l'entité type.
4. Définir les cardinalités pour les entités types impliquées dans une association
5. Mettre en évidence les associations non porteuses d'informations
6. Une association type avec une cardinalité $1,1$ ne peut pas être porteuse de propriété type.

Exercices :

1. Décrire le modèle entité-association correspondant au schéma `world` (entités, attributs, identifiants, associations, cardinalités).
2. On veut maintenant donner une version historique de ce schéma. On veut notamment mémoriser la succession des formes de gouvernement, et des chefs d'état. Proposer des modifications du schéma précédent (nouvelles identités, associations, avec attributs et identifiants, ne pas oublier les cardinalités).

3. Proposer des modifications du schéma relationnel pour rendre compte de ces données chronologiques.

1.2 ANY et ALL

`ANY` et `ALL` sont deux opérateurs qu'on utilise avec des requêtes imbriquées dans la clause `WHERE`. Par exemple, `SELECT * FROM table WHERE col < ALL(requete)` sélectionnera les lignes de `table` telles que la valeur de `col` est plus petite que toutes les valeurs retournées par la requête `requete`. Par exemple,

```
SELECT * FROM world.country
WHERE population_country >= ALL(SELECT population_country FROM world.country)
```

retournera la liste des pays les plus peuplés.

```
SELECT * FROM table WHERE col < ANY(requete)
```

sélectionnera les lignes de `table` telles que la valeur de `col` est plus petite qu'au moins une des valeurs retournées par la requête `requete`.

1. Comment simuler `IN` avec un `ANY` ?
2. Comment simuler `colonne>=ALL(requete)` ou `colonne>=ANY(requete)` avec les opérateurs d'agrégations `MIN`, `MAX` ?

2 TP

2.1 Vues

Les vues permettent de donner un nom à une requête afin de pouvoir l'appeler plus tard sans la réécrire à chaque fois. Une vue s'enregistre dans un schéma. Par exemple, dans la table `Sakila`, on pourrait créer une vue `FilmComedy` qui contient toutes les comédies de la table. On crée une vue avec `CREATE VIEW nom AS requete`. Étant donné que vous ne pouvez écrire que dans votre schéma personnel, il faudra nommer vos vues `entid.nom` où `entid` est votre identifiant ENT. Ainsi

```
CREATE VIEW entid.FilmComedy AS
WITH Comedy AS (SELECT film_id FROM film_category NATURAL JOIN category
                WHERE name='Comedy')
SELECT * FROM sakila.film
WHERE film_id IN (SELECT * FROM Comedy);
```

crée une vue dans votre schéma personnel qui renvoie les comédies de la table `sakila.film`. Désormais, si on veut sélectionner les comédies qui durent plus de deux heures, on pourra simplement écrire : SELECT * FROM entid.FilmComedy WHERE length>=120. Remarquez la différence entre `WITH` et une vue. `WITH` nomme une requête temporairement, seulement à l'échelle de la requête courante tandis qu'une vue est enregistrée de façon permanente. Cependant, chaque fois que vous appelez votre vue, elle est réévaluée par le système de base de données.

1. Proposer une vue `NeverRented` qui liste les films n'ayant jamais été loués.
2. Proposer une vue `StatRented` qui liste les identifiants des films et le nombre de fois où ils ont été loués.

2.2 Manipuler les dates et les intervalles

Deux types de données courants en SQL sont les types `date` et `interval`. Plusieurs fonctions permettent de les manipuler :

— `EXTRACT(champ FROM date)` extrait une information de la date ou de l'intervalle indiqué. Par exemple, pour extraire l'année de `rental_date`, on écrira `EXTRACT(year FROM rental_date`.
— `NOW()` renvoie la date courante.

— On peut les comparer avec les opérateurs `=`, `<`, `>`, `max`, `min` etc.
— On peut créer une date avec la fonction `make_date(year, month, day)`. On peut aussi utiliser `CAST('yyyy-mm-dd' AS DATE)` pour transformer une chaîne de caractère en date.
— On peut additionner/soustraire les dates (cela donne des intervalles) ou des intervalles.

1. Combien de DVD ont-il été loués en Juin 2005 ?
2. Combien de jours se sont-ils écoulés depuis la dernière location enregistrée dans `sakila` ?

2.3 Exercices !

1. Existe-t-il deux acteurs qui ont exactement le même prénom ? Indiquez prénom par prénom le nombre d'acteurs qui le portent, trié par fréquence décroissante.
2. Déterminer les noms des acteurs qui jouent dans au moins trente films.
3. Afficher le nom et le prénom des acteurs dont le nom commence par une lettre supérieure ou égale à `N`. Trier le résultat par nom.
4. Existe-t-il des acteurs qui jouent dans les mêmes films exactement ?
5. Existe-t-il des films sans acteurs ?
6. Calculer le nombre de DVD loués par mois.
7. Un client n'est autorisé à détenir que 3 DVDs simultanément. Vérifier qu'aucun client ne dépassait cette limite le 15 Juin 2005.
8. Quel film a-t-il été loué le plus longtemps en tout (sans compter les DVD non-rendus) ?

3 Devoir maison

Écrire dans votre schéma des fonctions pour répondre aux questions suivantes sur le schéma `sakila`. Indication : Si une location est fait le 15 avril, elle est considérée comme après le 15 avril.
Si un retours est fait le 22 avril, il est considéré comme après le 22 avril.

1. Écrire une fonction qui prend en argument une chaine de caractères `jour` (date au format 'yyyy-mm-dd' et un entier `n` et renvoie la liste des DVD loués depuis plus de `n` jours à la date spécifiée par `jour`.

```
CREATE OR REPLACE FUNCTION entid.en_retard(jour CHARACTER(10), n INTEGER)
RETURNS TABLE(inventory_id integer)
LANGUAGE sql AS $$
-- TODO
$$ ;
```

   Indication : Si une location est fait le 22 avril, elle est considérée comme après le 22 avril. Si un retours est fait le 28 avril, il est considéré comme après le 28 avril. Donc le 28 avril on considère que le DVD est loué depuis plus de 5 jours mais moins de 6.

2. Écrire une fonction qui renvoie le nombre de DVDs (instances d'`inventory`) déjà rendu par le client dont le nom et le prénom sont donnés en argument à la date définie par `jour`, date au format 'yyyy-mm-dd' (tant qu'un DVD n'est pas rendu, le champ return_date est NULL).

```
CREATE OR REPLACE FUNCTION entid.a_deja_rendu(prenom character varying(45),
                    nom character varying(45),
                    jour character(10))
RETURNS BIGINT LANGUAGE sql AS $$
 -- TODO
$$ ;
```