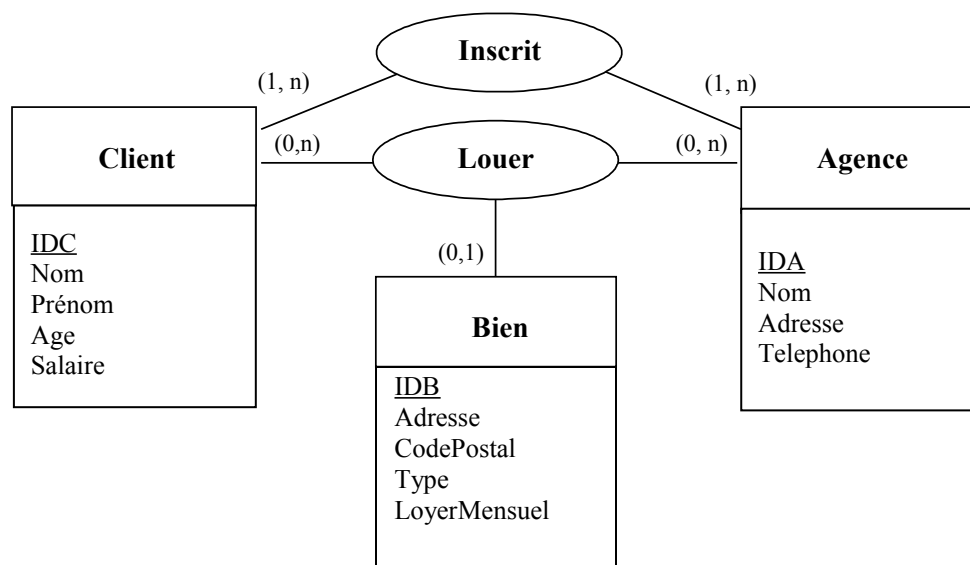


EFREI - L3/L'3/L3A A. Lahlou	Année : 2009/2010 TD-4 Bases de Données lahloukarim@free.fr
--	---

TD-4

Gestion immobilière

On se propose d'implanter le schéma Entité-Association suivant pour gérer la relation client/agence immobilière.



Le schéma ci-dessus montre que les clients s'adressent à une agence immobilière pour louer un bien. La condition pour pouvoir louer un bien est d'être préalablement inscrit sur une liste lui permettant d'avoir accès aux offres de l'agence. Un client peut s'inscrire dans plusieurs agences. Un bien est géré par une agence et est libre ou bien loué à au plus un client de l'agence mais un client peut louer plusieurs biens dans plusieurs agences.

I – MCD-MLD :

Vérifier que le schéma relationnel correspondant au schéma conceptuel ci-dessus est le suivant :

- Client (IDC, Nom, Prénom, Age, Salaire)
- Agence (IDA, Nom, Adresse, Téléphone)
- Bien (IDB, Adresse, CodePostal, Type¹, IDC, IDA, LoyerMensuel)
- Inscrit (IDC, IDA)

II - Définition de schéma et insertion de données :

1. Implanter le schéma ci-dessus en utilisant le langage SQL. Vous proposerez les types les mieux adaptés pour chaque attribut sachant que tous les identifiants seront constitués de 4 caractères alphanumériques.
2. Implanter les contraintes de clés primaires et de clés étrangères.
3. Implanter la contrainte suivante dans la table Bien : Loyer mensuel compris entre 200 et 2000 euros.
4. Implanter la contrainte indiquant que le type de bien est soit T1, T2, T3, T4 ou T5.
5. créer un jeu d'essai (remplissage des tables) qui satisfait toutes les contraintes (surtout celles de cardinalité).

¹ Le type est la catégorie du Bien qui est T1, T2, T3, T4 ou T5.

III - Interrogation :

Répondre aux requêtes suivantes :

1. Donner la liste de tous les biens situés à Paris
2. Donner l'adresse et le type de tous les biens non encore loués
3. Donner le nom et le prénom des clients n'ayant toujours rien loué. Vous pouvez utiliser l'opérateur ensembliste "minus" ou l'opérateur "in"
4. Donner le nombre de biens non encore loués
5. Donner le nombre des biens loués par chaque agence (donner le code et nom de l'agence, et nombre de biens loués).
6. Donner le salaire moyen des inscrits de l'agence d'identifiant "AA12"
7. Donner le montant total des loyers perçus chaque mois par chaque agence. On considérera que les agences perçoivent tous les loyers des biens qu'elles gèrent et qui sont loués. On donnera ce total et le code de l'agence.
8. Donner le nom de ou des agences qui perçoivent le moins de loyers.
9. Donner le ou les agences ayant des biens de tous types et libres
10. Donner le nom, le prénom et l'âge des clients qui louent un T1. On utilisera l'opérateur "exists"
11. Définition de vues

Commande : `CREATE VIEW <nomvue> [(col_alias[col_alias] ...)] as SQL_QUERY;`

- a) Définir la vue V1 représentant les biens de type T2.
- b) Donner le nombre de biens de type T2 de moins de 1000 euros
- c) Parmi les biens de type T2 de moins de 1000 euros, combien sont situés dans le département 94 ?
- d) Donner la liste des biens de type T2 les moins chers (le loyer mensuel est inférieur au loyer moyen de tous les biens de même type)
- e) Quels sont les biens de type T1 qui sont plus cher qu'au moins un T2 ?
- f) Quel sont les biens de type T3 qui sont plus cher que tous les biens de type T2 ?
- g) Que se passe-t-il si vous essayez de mettre à jour la vue V1 : insertion, suppression, mise à jour?

IV - Concurrence d'accès :

Approfondissements sur la Concurrence d'accès

Préparation

On va maintenant étudier le comportement concret d'Oracle en cas d'accès concurrents à la même ressource. Pour cela on va simuler l'exécution concurrente de programmes à l'aide du petit ensemble de lectures/écritures.

- Rajouter un attribut numérique à votre table client (champ nommé <Solde>)
- Créer 1 client nommé Joe (attention aux majuscules/minuscules)
- Soient 3 fichiers, nommés `Solde.sql`, `Depot.sql` et `Retrait.sql` :

`Solde.sql`

```
PROMPT 'Solde de Joe';  
SELECT Nom, Solde FROM client where Nom = 'Joe';
```

`Depot.sql`

```
PROMPT 'Un dépôt de 6000 euros est fait sur le compte de Joe' ;  
UPDATE client SET solde = solde + 6000 WHERE Nom = 'Joe';  
PROMPT 'Nouveau Solde de Joe';  
SELECT Nom, Solde FROM client where Nom = 'Joe';
```

`Retrait.sql`

```
PROMPT 'Joe retire 4000 euros';  
UPDATE client SET solde = solde - 4000 WHERE client = 'Joe';  
PROMPT 'Nouveau Solde de Joe';  
SELECT Nom, Solde FROM client where Nom = 'Joe';
```

Ensuite, soient 2 sessions utilisateur, nommées utilisateur1 et utilisateur2, en situation de concurrence. Dans tout ce qui suit, on note $INSTR_i$ l'exécution de l'instruction $INSTR$ par l'utilisateur i . Par exemple $Depot_1$ correspond à l'exécution du fichier `Depot` dans la première fenêtre par la commande `@Depot`. On note de même ROL_i et COM_i l'exécution des commandes `rollback;` et `commit;` dans la fenêtre i .

ORACLE pratique un verrouillage à deux phases assez libéral (degré d'isolation 1 – READ COMMITTED), qui ne garantit pas la sérialisabilité. De plus, un protocole multi-versions est utilisé. Du coup, aucun verrou n'est placé sur une ligne lors d'une lecture. Expérimentez les exécutions suivantes et essayez de comprendre le fonctionnement du verrouillage d'Oracle dans ce mode.

Simuler l'exécution des séquences d'instruction décrites ci-dessous.

1. L'utilisateur 1 (Joe) effectue un retrait. L'utilisateur 2 ne fait que consulter les soldes...

$COM_1, COM_2, Solde_1, Solde_2, Retrait_1, Solde_2, ROL_1, Solde_1, Solde_2$.

2. idem, mais avec un *commit* :

$COM_1, COM_2, Solde_1, Solde_2, Retrait_1, Solde_2, COM_1, Solde_1, Solde_2$,

3. L'utilisateur 1 (Joe) fait un retrait, alors que son salaire est crédité par sa banque (utilisateur 2). Un blocage est apparu. Pourquoi ?

$COM_1, COM_2, Solde_1, Solde_2, Retrait_1, Solde_2, Depot_2, Solde_1, Solde_2, COM_1, COM_2$