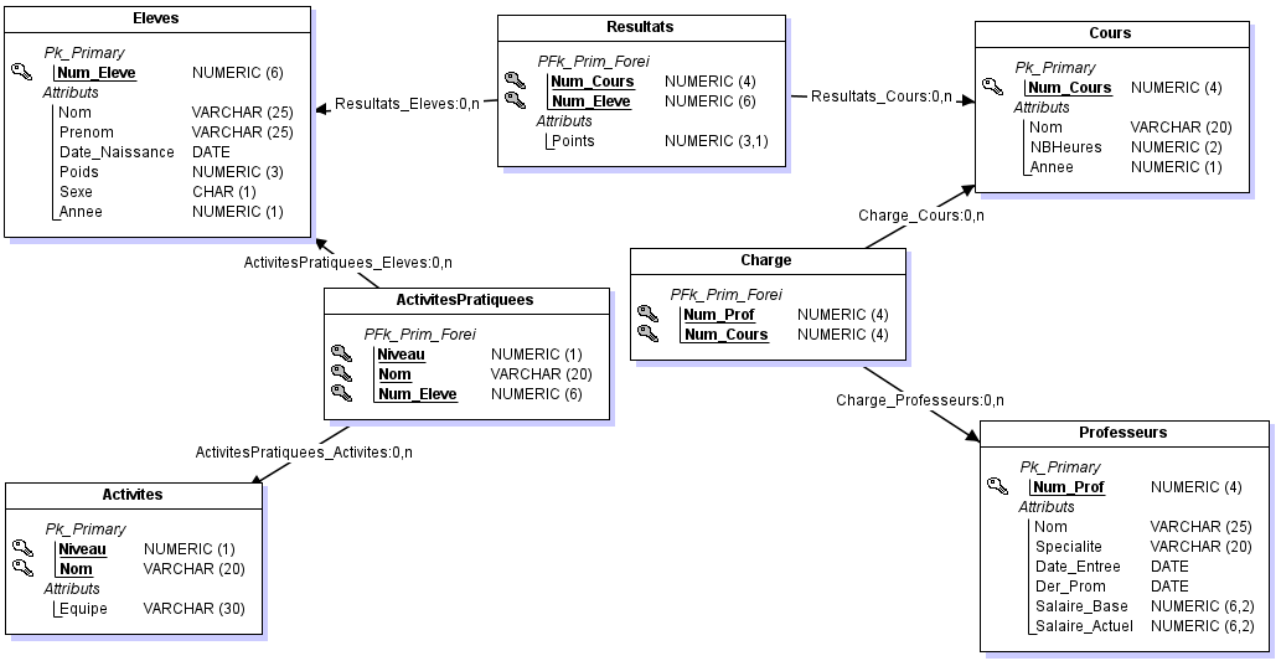
TP 3

Vues, droits, transactions et concurrence d'accès

Cas École

Description de la base de données :

Soit la base de données école dont le schéma relationnel est le suivant (le script vous est fourni):



Les vues et droits d'accès :

1. Créer la vue Eleves_ValDeMarne renfermant tous les champs des élèves du Val de Marne (Département 94)
2. Est-il possible de mettre à jour l'adresse de l'élève Buck DANNY qui déménage de 94800 Villejuif à 75013 PARIS depuis la vue Eleves_ValDeMarne ? Pourquoi ? Vérifier ?

3. Est-il possible de supprimer depuis la vue Eleves_ValDeMarne l'élève Gil JOURDAN habitant à 93800 EPINAY SUR SEINE qui a quitté l'école ? Pourquoi ? Vérifier ?
4. Peut-on ajouter un nouvel élève arrivant à l'école et venant de 94000 CRETEIL puis un autre élève venant de 91000 EVRY depuis la vue Eleves_ValDeMarne ? Pourquoi ? Vérifier ?
5. En utilisant la vue Eleves_ValDeMarne, donner les noms et prénoms des élèves du Val de Marne qui pratiquent du Tennis et qui ont une moyenne générale supérieure à 10.
6. Créer l'utilisateur User1 de mot passe 123 et lui donner un droit de consultation sur la table Eleves.
7. Tester s'il arrive bien à consulter le contenu de la table Eleves ?
8. Que faut-il faire pour que user1 puisse consulter toutes les tables de la base de données école ?
9. Le propriétaire de la base de données ecole est root. Il décide d'insérer un nouvel élève DUPONT Fantome né 12 avril 2001, de poids 60 kgs, de sexe masculin, qui est en 1ere année et qui vient de Villejuif 94800. Insérer ce nouvel élève et faire le nécessaire pour que User1 puisse le voir.
10. On souhaite donner tous les droits nécessaires à User1 pour gérer tous nos élèves du Val de Marne (Tous les élèves dont le code postal commence par 94). Donner les commandes nécessaires pour qu'il ait ces privilèges. Tester ces privilèges en choisissant des exemples adéquats.

Transactions et reprise sur panne :

On s'intéresse ici à prévoir le contenu de la base à chaque instant au cours du déroulement d'une séquence d'ordres SQL[1].

On considère une table `T(A INTEGER, B INTEGER)` contenant à l'origine le(s) n-uplet(s) indiqué(s) au début de chaque tableau. Tout ordre dessus est autorisé à tout utilisateur. On considère un client p_1 qui lance les séquences d'ordres suivantes, immédiatement après sa connexion, avec le mode de confirmation automatique non activé. Un autre client quelconque est représenté par p_2. Compléter les tableaux suivants. Indiquez le début et la fin de chaque transaction. Vérifiez ensuite sur machine avec les comptes root et user1.

ordres	T ds trans p_1	T vue par p_2
	(1,2)	(1,2)
INSERT (3,4)		
INSERT (1,2)		
ROLLBACK		
COMMIT		
INSERT (1,2)		
EXIT		

Ordres	T ds trans p_1	T vue par p_2

[1]Un exercice apparemment un peu scolaire, mais… incontournable pour bien utiliser ensuite les outils dans ses propres programmes.

	(1,2)	(1,2)
ROLLBACK		
INSERT (1,2)		
ALTER TABLE T ADD COLUMN (C INTEGER)		
UPDATE T SET B = 3 WHERE A = 1		
Panne : simuler une panne en stoppant le service mysql		

ordres	T ds trans p_1	T vue par p_2
	(1,2)	(1,2)
INSERT (3,4)		
INSERT (5,6)		
SET AUTOCOMMIT ON		
INSERT (7,8)		
ROLLBACK		
INSERT (1,2,3,4)		
DELETE FROM T WHERE A = 1		
Panne : simuler une panne en stoppant le service mysql		

Introduction au contrôle de concurrence :

MySQL gère des verrous pour contrôler les processus concurrents, au niveau des tables ou des n-uplets. Certains sont posés implicitement (sur les n-uplets lors des mises à jour d'instance, d'autres explicitement (sur les tables). Il n'est pas possible que deux processus possèdent des verrous globaux sur la même table ou locaux sur le même n-uplet. Les verrous sont relâchés en fin de transaction.

On s'initie ci-dessous à faire de l'intuition sur déroulement des ordres posant des verrous, puis à en poser nous-même là où il faut.

On considère une table `T(A integer, B integer)` contenant au début les n-uplets `(1,1)` et `(2,2)`. Tout ordre dessus est autorisé à tout utilisateur. On considère deux processus clients p1 et p2 qui effectuent les séquences d'ordres suivantes, immédiatement après leur connexion. Un autre client quelconque est représenté par p3.

Remplir les tableaux dans l'ordre dans lequel les ordres sont effectivement exécutés. Puis vérifiez sur machine.

1. `1 (p1) : DELETE T WHERE B=1;`

   `2 (p1) : COMMIT;`

Client	Ordre	T ds trans p_1	T ds trans p_2	T vue par p_3
		(1,1) (2,2)	(1,1) (2,2)	(1,1) (2,2)

2.

1 (p_1) : DELETE T WHERE B=1;

2 (p_2) :UPDATE T SET B=B-1;

3 (p_1) : COMMIT;

4 (p_2) : COMMIT;

Client	Ordre	T ds trans p_1	T ds trans p_2	T vue par p_3
		(1,1) (2,2)	(1,1) (2,2)	(1,1) (2,2)

3.

1 (p_2) :UPDATE T SET B=B-1;

2 (p_1) : DELETE T WHERE B=1;

3 (p_2) : COMMIT;

4 (p_1) : COMMIT;

Client	Ordre	T ds trans p_1	T ds trans p_2	T vue par p_3
		(1,1) (2,2)	(1,1) (2,2)	(1,1) (2,2)

4.

1 (p_1) :UPDATE T SET A=1 WHERE B=2;

2 (p_2) : DELETE T WHERE A=1;

3 (p_1) : COMMIT;

4 (p_2) : COMMIT;

Client	ordre	T ds trans p_1	T ds trans p_2	T vue par p_3
		(1,1) (2,2)	(1,1) (2,2)	(1,1) (2,2)