EXERCICE 1:

1. On observe qu'il y a une deadlock.
Cette erreur survient même avec un niveau d'isolation repeatable read.

2. Dans le niveau repeatable read, on est bloqué dans la seconde transaction jusqu'au moment où l'on fait un commit dans la première transaction.

4. On s'attendait à ce qu'au moins 1 objet soit supprimé de la table.
Or aucun élément a été supprimé de la table.

EXERCICE 2

1. Une partie est fermée alors qu'elle n'a pas assez de joueurs inscrits.

BEGIN ISOLATION LEVEL READ COMMITTED;	BEGIN ISOLATION LEVEL READ COMMITTED;
DELETE FROM demande WHERE pid = 8;	
	SELECT * FROM demande;
	UPDATE partie SET etat = 'fermé' WHERE pid = 8;
COMMIT;	COMMIT;

READ COMMITTED = X
REPEATABLE READ = X
SERIALIZABLE = X

L'erreur persiste à chaque niveau d'isolation

2. Un joueur a une demande en attente pour une partie annulée.

BEGIN ISOLATION LEVEL READ COMMITTED;	BEGIN ISOLATION LEVEL READ COMMITTED;
UPDATE partie SET etat = 'annulée' WHERE pid = 8;	
UPDATE demande SET statut = 'refusé' WHERE pid = 8;	
	SELECT * FROM partie WHERE etat = 'annulée';
	INSERT INTO demande (jid, pid) VALUES (3, 8);
COMMIT;	COMMIT;

READ COMMITTED = X
REPEATABLE READ = X
SERIALIZABLE = OK

3. Une partie annulée a des demandes validées.

BEGIN ISOLATION LEVEL READ COMMITTED;	BEGIN ISOLATION LEVEL READ COMMITTED;
UPDATE partie SET etat = ‘annulée’ WHERE pid = 8;	
UPDATE demande SET statut = ‘refusé’ WHERE pid = 8;	
	UPDATE demande SET statut = ‘validé’ WHERE pid = 8;
COMMIT;	COMMIT;

READ COMMITTED = X
REPEATABLE READ = OK
SERIALIZABLE = OK

4. Une demande est refusée alors qu’elle a une date plus ancienne qu’une demande acceptée pour la même partie. (non résolu)

il faut sûrement 3 transactions (2 joueurs et 1 organisateur) pour pouvoir créer l’incohérence.

BEGIN ISOLATION LEVEL READ COMMITTED;	BEGIN ISOLATION LEVEL READ COMMITTED;
SELECT * FROM joueur NATURAL JOIN demande NATURAL JOIN partie WHERE pid = 8;	
INSERT INTO demande(jid, pid) VALUES (4, 8);	
SELECT * FROM joueur NATURAL JOIN demande NATURAL JOIN partie WHERE pid = 8;	
UPDATE demande SET statut = ‘refusé’ WHERE jid = 4 and pid = 8;	
UPDATE demande SET statut = ‘accepté’ WHERE statut = ‘en attente’ and pid = 8;	SELECT * FROM joueur NATURAL JOIN demande NATURAL JOIN partie WHERE pid = 8;
	INSERT INTO demande(jid, pid) VALUES (5, 2);
COMMIT;	COMMIT;

READ COMMITTED = ?
REPEATABLE READ = ?
SERIALIZABLE = ?

5. Une partie contient plus de demandes validées que le maximum souhaité.

BEGIN ISOLATION LEVEL READ COMMITTED;	BEGIN ISOLATION LEVEL READ COMMITTED;
INSERT INTO demande(jid, pid) VALUES (4, 8);	
	SELECT * FROM demande NATURAL JOIN partie NATURAL JOIN joueur WHERE pid = 8;
COMMIT;	
	UPDATE partie SET etat = ‘fermé’ WHERE pid = 8;
	UPDATE demande SET statut = ‘validé’ WHERE pid = 8;
	COMMIT;

READ COMMITTED = X
REPEATABLE READ = OK (l’élément inséré est ‘en attente’)
SERIALIZABLE = OK

EXERCICE 3