# Travail pratique 2 :

Situation : **Compétition**

Modalité : seul ou en équipe de 2 maximum

Travail complet vaut : 20% de la session. L’addition des parties donne 100.

### Base de données

Nous utiliserons la même Base de données que le travail pratique 1. Vous devez mettre à jour le script de la BD (du travail pratique 1, celui-ci ajoutait beaucoup de données) et y ajouter toutes vos parties du travail pratique 2. Celui-ci doit s’exécuter sans erreur. Avec EF (.edmx), il est important que votre version de BD soient bien à jour pour leur compatibilité. (5 points)

## Partie 1 : Les vues modifiables (25% du travail)

Faites un formulaire qui me permet de faire l’entrée des résultats pour une compétition.

1. Comme nous n’avons pas beaucoup de compétition, nous pourrons nous servir d’un combobox pour rechercher la compétition.
2. Vous utiliserez un DataGridView pour afficher le no, prénom, nom de votre coureur (participant) et son résultat. Vous devez vous faire une vue. Celle-ci devra permettre d’enregistrer les résultats à la compétition choisie.
3. Vous devez faire l’enregistrement au fur et à mesure dès qu’on sort de la cellule du résultat, si celle-ci a bien été modifiée. Il est possible d’enregistrer à la sortie de la forme s’il y a lieu. Vous devez gérer les erreurs s’il y a lieu.
4. Seul le résultat sera modifiable, les autres champs seront non sélectionnables.
5. Vous devez gérer le cas où l’on ne respecte pas votre contrainte check. Votre message devra être personnalisé. Assurez-vous de revenir automatiquement à la valeur précédente dans ce cas.
6. Vous devez avoir ajouté des inscriptions à une compétition pour en avoir au moins 5. Faites l’ajout de ces données dans votre script.

## Partie 2 : Les transactions ( 20% du travail)

On vous demande de faire une procédure stockée qui permettra de gérer le cas où un coureur décide de se désinscrire de toutes ces compétitions.

1. Vous devrez utiliser les transactions pour vous assurer du bon fonctionnement complet de ces actions.
2. Vous devez retourner un message d’erreur en cas de non-complétion.
3. Pour aider à la correction de votre procédure stockée, vous ferez du code transact sql pour créer un coureur, utilisez son scope\_identity et lui créer au moins 2 inscriptions. Dans ce même code, vous afficherez le résultat obtenu pour ce coureur en affichant ses ajouts sur chacune des tables. Servez-vous de declare, scope\_identy pour faire ce travail en une exécution.
4. Appeler votre procédure stockée et afficher la preuve pour le coureur spécifié en affichant le contenu des tables.
5. Vous placerez en commentaire dans votre procédure stockée un ajout erroné sur votre table et la gestion de celle-ci. Nous décommenterons votre commentaire, puis nous ferons un alter de votre procédure et la ferons réexécuter afin de voir l’effet des transactions lors d’une erreur.
6. Faites un formulaire C#, qui recherchera le coureur et qui permettra de faire la destruction de celui-ci. Vérifier par une question pour être sûr de la volonté de l’usager avant de détruire.
7. Gérer le message de votre raiserror dans c#. Utiliser un catch DataException pour afficher notre message de notre transaction. Bien entendu celui-ci arrive quand votre transaction a volontairement votre instruction erronée.

### remise partiel (partie 1-2) L le jeudi 29 avril avant minuit

Remettre :

* Vos formulaires C#
* Votre script fonctionnel et à jour pour créer votre BD. Celui-ci contiendra les données nécessaires pour tester.

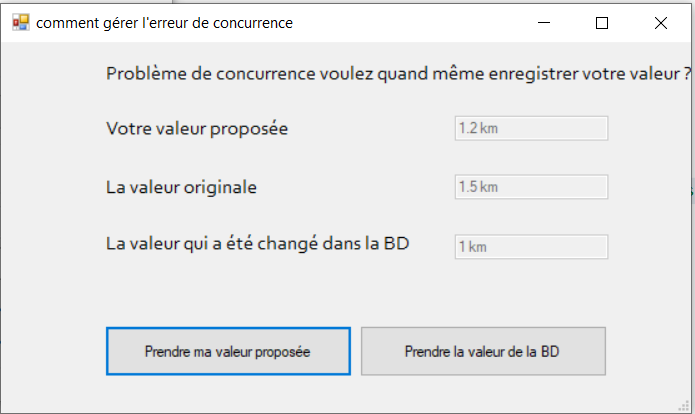
## Partie 3 : Les déclencheurs (trigger) (25% du travail)

1. Nous désirons imposer un maximum de coureur dans nos compétitions afin de pouvoir en faire la gestion. Dans le cadre du cours et pour faciliter les vérifications, nous baisserons ce nombre à maximum 5 coureurs par compétition. Faites le déclencheur qui permet d’imposer ce maximum. Votre déclencheur nous enverra un message (throw).
2. Pour tester votre déclencheur, nous aurons besoin d’au moins 4 compétitions. Ajouter les instructions pour ajouter les données manquantes dans votre script.
3. Donnez-moi les instructions dans le cas où le déclencheur **laissera** ajouter en batch (au moins 4 ajouts. Il faut tester au moins 2 fois le même coureur, des inscriptions dans des compétitions différentes.
4. Donnez-moi les instructions dans le cas où le déclencheur **empêchera** notre ajout en batch. (Au moins 4 ajouts. Il faut tester au moins 2 fois le même coureur, des inscriptions dans des compétitions différentes.
5. Faites un formulaire qui pourra recevoir l’erreur de votre trigger lorsqu’elle s’applique et qui l’affichera à l’usager (DataException). Faites une recherche sur notre coureur (beaucoup de données) et une recherche sur nos compétitions en ordre alphabétique (peu de données) et bouton enregistrer l’inscription.

## Partie 4 : La concurrence (25% du travail)

Nous allons gérer la concurrence sur une petite table de notre système. Vous ferez la gestion de la concurrence sur la table de compétition. Nous ferons comme si plusieurs utilisateurs de notre système font des mises à jour en même temps sur cette table.

1. Faites une modification des champs de votre compétition. Faites une recherche préalable (nous avons peu de données).
2. Permettez la modification de la ou des valeurs en les affichant dans un textbox.
3. Vous devrez ajouter dans votre script l’instruction qui vous permet d’ajouter un champ timestamp dans votre table de compétition. Faites la mise à jour dans votre edmx, assurez-vous de mettre la propriété (mode d’accès concurrentiel) à fixed.
4. Gérer la concurrence (System.Data.Entity.Infrastructure.DbUpdateConcurrencyException). Pour prendre une décision éclairée, affichez les 3 valeurs (courante, originale et valeur de la BD). Servez-vous d’un formulaire pour faire un bel affichage. Deux choix s’offrent à l’usager, gardez sa valeur ou gardez celle de la BD.



1. Tester votre application en démarrant un autre formulaire grâce au .exe de votre projet.

### **Remise partie 3-4**: Le lundi 10 mai avant minuit

Remettre :

* Vos formulaires C#
* Votre script fonctionnel et à jour. Celui-ci contiendra les données nécessaires pour tester.