Comment rédiger un compte rendu de travaux pratique en informatique ?

christophe.cerin@univ-paris13.fr

8 octobre 2017

1 Introduction

Rédiger un compte rendu de TP consiste à retracer les différentes étapes de la démarche adoptée pour résoudre un problème scientifique et technique avec l'aide d'un ordinateur. Au moyen d'observations réalisées par des expériences mettant en jeu l'ordinateur, il s'agit aussi d'analyser ce qui s'est passé dans l'expérience.

Selon le problème à traiter, votre compte rendu doit se composer des parties que nous listons plus loin. . .qui, d'ailleurs, peuvent être explicitées plus ou moins dans l'énoncé que l'on vous propose. En effet, dans le contexte scolaire, vous êtes guidés dans ce que vous avez à faire. Plus tard, dans la vie active, il faudra sans aucun doute poser le problème et le résoudre. Autrement dit, à l'université, le plan expérimental est plus ou moins explicite et l'examinateur vous attend sur votre analyse !

Selon l'agade que c'est en faisant qu'on apprend, rappelons que le TP est le moyen d'apprendre la programmation en profondeur. Comme nul ne peut programmer à votre place, vous avez donc un effort conséquent à produire. Vous avez aussi un effort conséquent à faire sur la rédaction du compte rendu. N'oubliez donc pas le bon vieux sujet-verbe-complément.

2 Contenu d'un TP

Votre compte rendu de TP comprendra les points suivants :

1. Décrire le problème posé : il s'agit d'expliquer (en reformulant l'énoncé à votre sauce) le problème et de définir les objectifs à atteindre ;
2. Présenter les hypothèses : proposer en 2 ou 3 lignes les hypothèses que vous allez suivre lors de l'expérience ;
3. Décrire le protocole expérimental (s'il n'est pas complètement explicité dans l'énoncé de TP) a) Décrire la stratégie mise en œuvre pour tester une hypothèse. C'est le point le plus important. Vous pouvez vous demander quel est le résultat attendu. . .et discuter plus tard si ce que vous observez est conforme à ce qui est attendu. b) Décrire ce qui est réalisé pendant la manipulation :
   — le matériel utilisé (si le matériel est imposé par le protocole, faites référence à celui-ci et ne le rappelez pas) ; Présenter la fréquence du

processeur, en particulier s'il s'agit de mesurer des temps d'exécution des programmes, la quantité de RAM ou de disque s'il s'agit de discuter de l'occupation en mémoire des données du programme, du système d'exploitation ou des librairies (avec les numéros de version) s'il s'agit de discuter de précision arithmétique par exemple...
   - les précautions éventuelles à prendre ; Il s'agit souvent pour l'informaticien de préciser les pré-conditions c'est-à-dire sous quelles conditions expérimentales on peut exécuter le programme ; Cette partie doit expliciter les conditions expérimentales ;
   - les grandeurs mesurées et leurs unités ;
   - les paramètres que vous ferez varier.
4. Présenter les résultats expérimentaux. Présenter les observations faites sous forme de schéma, d'illustrations légendées, de commentaires...Je vous rappelle qu'une bonne figure ou schéma bien légendé vaut tous les commentaires du monde ! Je vous rappelle également que toute figure ou schéma doit être commenté / légendé. Ce n'est pas la peine de faire une figure si elle n'est pas commentée ! Présenter les résultats expérimentaux (les mesures et leurs unités, écrites en respectant le nombre de chiffres significatifs) sous une forme appropriée : tableau de valeurs, représentation graphique...
5. Traiter et analyser les résultats expérimentaux. Il ne s'agit pas de dire ce que l'on obtient (j'ai copié collé un code, et j'ai réussi à l'exécuter) mais plutôt d'analyser pourquoi ce que l'on obtient est conforme ou pas à ce qui était attendu (et formulé par une hypothèse). Tenir compte des incertitudes liées aux mesures. Par exemple si le temps d'exécution du programme est la micro-seconde et que la librairie utilisée pour la mesure du temps est connue pour avoir une précision de l'ordre de la micro-seconde, alors on peut avoir un doute sur ce que vous racontez. Mettre en évidence des facteurs d'influence, une relation entre les grandeurs : si x croit en même temps que y alors cela a peut être un sens particulier.
6. Conclure et faire la critique de l'expérience. Indiquer si les hypothèses de départ sont validées ou non. Si c'est pas le cas, vous avez peut être du modifier tel et tel paramètre avant que les hypothèses soient vérifiées. Proposer une réponse au problème posé en une phrase de synthèse. Dire si l'objectif est atteint. Dans un cursus de licence ou master, vous pouvez aussi comparer les résultats que vous avez obtenus à ceux de la littérature (articles de référence, Web, encyclopédie...) Les questions posées dans l'énoncé d'un TP sont là pour vous guider dans votre rédaction, et donc leurs réponses doivent apparaitre dans votre compte rendu.

3 Cas d'usage

Considérons le TD/TP numéro 04 - ACCÈS CONCURRENTS SOUS POSTGRESQL dont les objectifs sont d'étudier les modes d'isolation et les transactions sérialibles (READ COMMITED / SERIALIZABLE). Vous pouvez noter que la notion de transaction est aux bases de données ce que les exceptions sont dans un langage de programmation (comme Java par exemple). Il s'agit de contrôler ce qui se passe en présence d'une erreur.

Les éléments de réponse aux premières questions du TP sont les suivants.

1.1) Au moyen des définitions du cours, il y a deux transactions, la première est délimitée par BEGIN TRANSACTION;....; COMMIT; et la seconde est SELECT *.

1.2) et 1.3) Lorsque nous exécutons le code qui suit, que nous avons commenté, il y a une erreur. Pour comprendre l'erreur il faut remonter aux schémas des relations.

```
BEGIN TRANSACTION; -- ouverture d'une transaction
INSERT INTO Commande VALUES ('F2','P4',25); -- insertion
INSERT INTO Commande
VALUES ('F2','P8',25); -- viole l'int\'egrit\'e de la clef \'etrang\'ere, la
COMMIT;                -- transaction  est avort\'ee + rollback
SELECT * FROM commande
WHERE co_piece='P4'; -- comme la transaction a \'et\'e avort\'ee
                     -- on ne retrouve pas le tuple ('F2','P8',25)
                     -- dans le r\'esultat de la requ\^ete
```

L'insertion de la pièce P8 ne peut pas avoir lieu car cette pièce doit exister dans la table Pièce, (voir en particulier la contrainte sur le deuxième attribut dans la table COMMANDE).

En modifiant le code précédent en :

```
-------------------------------------------------
BEGIN TRANSACTION;
         INSERT INTO Commande
                VALUES ('F2','P4',25);
         INSERT INTO Commande
                VALUES ('F2','P8',25);
         SELECT * FROM commande WHERE co_piece='P4';
END TRANSACTION;
-------------------------------------------------
```

Le SELECT ne s'exécute pas car la transation a été arrétée à l'erreur.

1.4) Par contre en modifiant le comme comme suit :

```
-------------------------------------------------
BEGIN TRANSACTION;
         INSERT INTO Commande
                VALUES ('F2','P4',25);
         SELECT * FROM commande WHERE co_piece='P4';
         INSERT INTO Commande
                VALUES ('F2','P8',25);
END TRANSACTION;
SELECT * FROM commande WHERE co_piece='P4';
-------------------------------------------------
```

le premier SELECT va afficher le tuple inséré, par contre le deuxième non car comme la transaction est avortée. Tout ce qui est fait pendant cette transaction n'affecte pas la base de données. C'est un peu comme si nous avions travaillé sur une copie...qui est effacée en cas d'erreur.

1.5) Pour déterminer le mode d'isolation, nous pouvons simuler une execution concurrente en ouvrant deux terminaux et en tapant alternativement le code suivant :

```
Terminal 1 BEGIN TRANSACTION ; SELECT * FROM fournisseur ;
Terminal 2 UPDATE fournisseur
                    SET fo_categorie=fo_categorie+2 ;
Terminal 1 SELECT * FROM fournisseur ;
    COMMIT;
```

Si les résultats des deux selects sont identiques alors le mode est SERIALIZABLE ici les deux sont identiques on peut donc être certains que le mode est SERIALIZABLE. Si les résultats sont différents alors cela veut dire que l'instruction UPDATE est vue par le dernier SELECT et dans ce cas nous aurions été en mode READ COMMITTED.

2.1) Il y a 6 transactions : 4 SELECT du terminal 1 plus 2 transactions explicites du terminal 2.

2.2) Il s'agit d'expliquer les résultats des étapes de la transaction O1 proposée.

Etapes 1-4 : la transaction 1 et la transaction 2 lisent les mêmes données car la requête 3 fait partie d'une transaction non terminée : transaction 2 ne voient pas les mises à jour de la requête 3.

Etapes 4-6 : la transaction 6 voit les mises à jour de la requête 3 qui fait partie d'une transaction terminée à l'étape 5

Etapes 6-10 : la transaction 6 et la transaction à l'étape 10 lisent les mêmes données car la requête 8 fait partie d'une transaction annulée.

2.3) Le terminal 1 est bloqué puisque le terminal 1 et 2 cherchent a modifier le même tuple au même moment.

2.4) Le terminal 1 est en mode READ COMMITED car les deux select sont différents et que l'exécution s'est terminée.

2.5) Par contre ici, avec l'ajout explicite de l'instruction qui fixe le mode d'isolation puis l'erreur sur l'étape 7 :

```
SELECT fo_nom,fo_categorie FROM fournisseur
WHERE fo_nom='Dupont' ;
NOTICE: current transaction is aborted, queries ignored
until end of transaction block
*ABORT STATE*
```

nous déduisons que le mode d'isolation est SERIALIZABLE ce qui a eu pour effet d'empécher les deux mises à jour concurrentes : celle à l'étape 4 et celle à l'étape 5.