Services d'un SGBD

1 Gestion des accès concurrents

Un des intérêts des bases de données est le fait qu'elles puissent être consultées et modifiées par plusieurs utilisateurs simultanément.

Comment faire en sorte que ces accès concurrents ne posent pas de problème à l'état de la base ?

Regardons cela sur un exemple simple :

Imaginons deux utilisateurs Alice et Bob réalisant une suite d'actions sur une base de données "très simple" contenant une seule table `Employes(ne,nom,sal)`

NE	NOM	SAL
0	Charlie	2000
1	Diana	2100
2	Eric	1600

On notera $R_A(id)$ une requête de lecture sur le n-uplet repéré par id, par l'utilisateur A pour Alice et $R_B(id)$ une requête de lecture sur le n-uplet repéré par id, par l'utilisateur B pour Bob

Par exemple cette requête peut être :

Select salaire from Employes where prenom = 'Charlie'

On notera $W_A(id)$ une requête d'écriture sur le n-uplet repéré par id, par l'utilisateur A pour Alice

Par exemple cette requête peut être :

Update Employes Set sal = 2500 where nom = 'Diana'

Chaque utilisateur ne soupçonne pas forcément l'existence de l'autre et exécute par exemple

1. Pour Alice : $R_A(1)$ puis $W_A(1)$
2. Pour Bob : $R_B(1)$ puis $R_B(2)$ puis puis $W_B(1)$

Le problème est la place de ces actions sur l'axe du temps, il existe plusieurs possibilités d'entrelacements de ces actions

Par exemple la séquence conjointe des actions appelé **ordonnancement** ,d'Alice et Bob peut-être :

$R_A(1)$;$W_A(1)$;$R_B(1)$;$R_B(2)$;$W_B(1)$

On appelle **transaction** d'un utilisateur une suite d'actions terminée soit par un COMMIT ;soit par un ROLLBACK ;

Pour qu'une transaction soit **validée** on utilise la commande **COMMIT** et pour qu'elle soit **annulée** on utilise la commande **ROLLBACK**

Dans ce cas on observe que toutes les actions d'Alice (on parle de la **transaction**

d'Alice T_A) forment une unité dans le temps isolée de la transaction de Bob T_B on dit alors que **la suite des transactions est sérielle**

Mais on aurait pu avoir par exemple :

$R_B(2)$;$R_A(1)$;$W_A(1)$;$R_B(1)$;$W_B(1)$

et on a perdu la sérialité de la suite d'actions **mais on constate que l'on peut placer $\mathbf{R}_B(2)$ après $\mathbf{W}_A(1)$** et ensuite rendre sériel l'ordonnancement

Par quel processus va-t-on placer $R_B(2)$ après $W_A(1)$?

Il existe des processus de mise en attente d'une transaction pour laisser à d'autres transactions la possibilité de faire un tout

Est ce toujours possible ?

Ces mécanismes de mise en attente peuvent conduire à un **interblocage ou deadlock** , la transaction 1 attend la transaction 2 qui attend la transaction 1 . Il faut donc éliminer les interblocages

Voici un exemple d'interblocage dans l'ordonnancement suivant :

Pour la suite on adopte la convention suivante, une opération d'écriture par la transaction n sur l'objet A est notée Wn(A).

Une opération de lecture est notée Rn(A).

Par exemple l'écriture de l'objet B par la transaction 2 est notée W2(B).

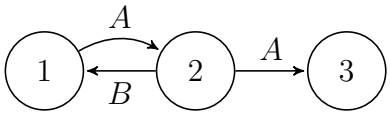
Soit l'ordonnancement :

R1(A) ;W2(A) ;R2(B) ;R3(A) ;W1(B) ;

L'utilisateur 1 commence en premier dans le temps à lire sur A puis l'utilisateur 2 va écrire sur A, **puisque ces deux actions ne peuvent pas commuter dans le temps** donc l'utilisateur 1 doit précéder l'utilisateur 2

Ensuite l'utilisateur 2 lit sur B et l'utilisateur 1 écrit sur B donc 2 doit précéder 1

On voit mieux cela en dessinant un **graphe orienté de précédence**



On observe un **cycle** dans le graphe de précédence donc il y a interblocage

Il existe des techniques pour soit prévenir soit traiter les interblocages, cependant ces techniques ralentissent le fonctionnement de la base de données

Que retenir des propriétés des transactions ?

On retiendra l'acronyme **ACID**

1. Le SGBD fait en sorte qu'une transaction est soit acceptée dans son ensemble soit refusée dans son ensemble **(propriété d'insécabilité ou Atomicité)**
2. Les transactions permettent de passer d'un état **Cohérent** de la base à un autre
3. Le SGBD fait en sorte que chaque utilisateur "ait l'impression qu'il est seul avec la base de données" **(propriété d'Isolation)**
4. Une fois une transaction validée son effet ne peut pas être perdu suite à une panne quelconque **(propriété de Durabilité)**

2 Persistance des données

A cause de pannes possibles et du volume des données ces dernières ne peuvent pas se trouver dans la mémoire RAM et sont donc stockées sur disques

Comment assurer la durabilité des transactions validées ?

A côté de la base on tient un journal des transactions (de mise à jour) , c'est un fichier séquentiel dans lequel on enregistre toutes les mises à jour.

En cas de panne on s'aide de ce journal pour respecter la durabilité des transactions

3 Efficacité de traitement des requêtes

Index

aire et intégrale, 47, 60
aire et primitive, 37, 74
algorithme, 5, 19, 24, 25, 37, 48, 54, 60, 71, 72, 76
arbre pondéré, 41, 56
arithmétique, 14, 18, 30, 43, 59, 64

complexes, 4, 24, 32, 40, 54, 57, 62, 79
congruence, 14

dérivée, 23, 29, 37, 45, 74
division euclidienne, 30, 50

équation complexe, 4, 32, 62
équation de plan, 15, 35, 38, 58, 63, 64, 68
équation diophantienne, 64
équation paramétrique, 22, 38, 47, 58, 64
équation trigonométrique, 79

fonction exponentielle, 11, 17, 22, 33, 37, 45, 53, 61, 74, 80
fonction logarithme, 5, 18, 33, 57, 67

trigonométrie, 24, 54

vecteur normal, 13, 15, 35, 58, 63
vrai-faux, 47

Comment trouver "rapidement" une information dans une base de données ?

On sait depuis longtemps utiliser des index pour pouvoir trouver dans un document une information précise par exemple dans la capture d'écran ci-dessus on voit un index pour un document de mathématiques ainsi si je cherche des exercices concernant la notion de fonction dérivée je vais regarder directement à la page 23, 29, 37 , 45, ou 74

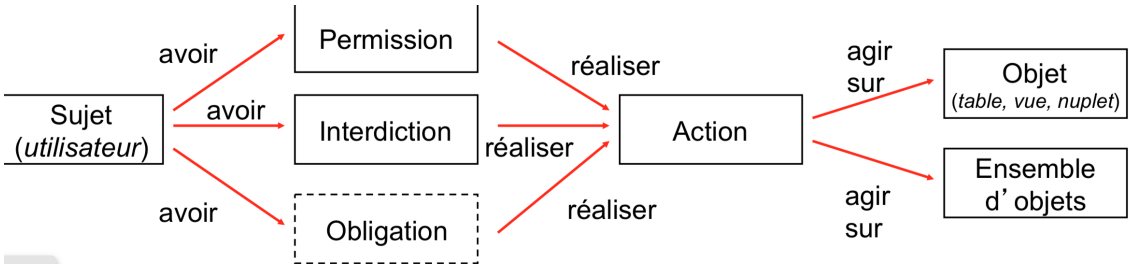
D'un point de vue informatique ce genre de technique a été mis en place avec un peu plus de raffinement : On découpe les disques de la base en blocs (et non en pages) à chaque bloc est associé une clé puis est créé un index avec ces clés

SQL est un **langage déclaratif** et non impératif comme Python : On écrit une requête mais on délègue au SGBD la tâche de trouver la façon la plus performante d'un point de vue algorithmique d'exécuter cette requête

4 Sécurisation des accès

Les bases de données contiennent des informations de valeur d'où la sécurisation des accès à plusieurs niveaux

Par Exemple sur PRONOTE en fonction de son statut (Elève, Enseignant, Administration, Parent) on peut voir et agir suivant certaines règles



C'est par une **authentification de l'utilisateur par identifiant et mot de passe** que l'utilisateur a accès à son environnement

Pour augmenter la sécurité des échanges d'informations entre l'utilisateur et la base de données les communications sont **chiffrées**

5 Exercices : Services d'un SGBD

Ex 1

Que signifie l'acronyme ACID ?

Ex 2

NE	NOM	SAL
0	Charlie	2000
1	Diana	2100
2	Eric	1600

Quel est l'état du compte de Charlie après la transaction suivante ?

UPDATE Employes SET Sal = Sal + 50 WHERE Nom = 'Charlie' ; UPDATE Employes SET Sal = Sal + 100 WHERE Nom ='Eric' ; SELECT Sal FROM Empl WHERE Nom ='Diana' ; COMMIT ;

Ex 3

Que signifie l'instruction ROLLBACK ; ?

Ex 4

Alice effectue la transaction suivante :

Select avg(Sal) from Employes

Bob effectue la transaction suivante :

INSERT into Employes(ne,nom,sal) VALUES,
(4,'Paul',3000),
(5,'Patrick',4000) ;

Peut-on permuter ces deux transactions dans le temps ?

Ex 5

Une opération d'écriture par la transaction n sur l'objet A est notée Wn(A).

Une opération de lecture est notée Rn(A).

Par exemple l'écriture de l'objet B par la transaction 2 est notée W2(B).

Parmi les ordonnancements suivants donner ceux qui sont sérialisables

1. R1(A) ;W1(A) ;R2(A) ;W2(A) ;
2. R1(A) ;W2(A) ;R2(A) ;W1(B) ;
3. R1(A) ;W2(A) ;R2(A) ;W1(A) ;
4. R1(A) ;W2(B) ;R2(B) ;W1(A) ;
5. R1(A) ;W2(A) ;W1(A) ;R2(A) ;
6. R2(A) ;R1(A) ;W2(A) ;W1(B) ;

Ex 6

Construire le graphe de précédence des ordonnancements suivants

1. R1(A) ;W1(A) ;R2(A) ;W2(A) ;
2. R1(A) ;W2(A) ;R2(A) ;W1(B) ;

3. R1(A) ;W2(A) ;R2(A) ;W1(A) ;
4. R1(A) ;W2(B) ;R2(B) ;W1(A) ;
5. R1(A) ;W2(A) ;W1(A) ;R2(A) ;
6. R2(A) ;R1(A) ;W2(A) ;W1(B) ;

Ex 7

Parmi les ordonnancements suivants, donnez ceux qui peuvent être sérialisés en R2(A) ;R1(A) ;W1(A) ;

1. R1(A) ;W1(A) ;R2(A)
2. R2(A) ;W1(A) ;R1(A)
3. R1(A) ;R2(A) ;W1(A)