Assertions (contraintes inter-relations mais indisponibles sous Oracle) et des déclencheurs (outriggers) (= élément actifs qui vont jouer leur rôle lors de la venue de certains événements : insertion, modification ou délétion au sein d'une relation). Les assertions et les triggers vont être des éléments actifs et font partie du schéma de la base au même titre que les relations ou les vues.

Le trigger s'active lorsqu'un évènement de type insertion, modification ou d'élétion survient.

1. d´eclencheur LMD : opérations de mise à jour des tuples d'une table
2. d´eclencheur LDD : opérations survenant au niveau d'un schéma utilisateur de la base
(interdit toute suppression de table ou autre objet de la base le vendredi.)
3. déclencheur d'instance : opérations survenant au niveau de la base

1. déclencheur LMD :
Création d'un trigger global

```
create table log (
txt varchar2 (20), date_maj date, user_maj varchar2 (15) ) ;
create or replace trigger Global_update
before update on EMP
begin
insert into log values ('update trigger', sysdate, user) ;
end ;
/
update EMP set nom = nom || 't'
where substr(nom,1,1) = 'M' ;
-- un seul enregistrement dans log quel que soit le nombre de tuples modifi´es.
select * from log ;
drop table log ;
```

Création d'un d´eclencheur d'historisation

```
create table historique (dateOperation date, typeOperation
varchar(15), nomUser varchar(15), AnciennumEmploye number,
NouveauNumEmploye number) ;
create or replace trigger monitor_historique
after insert or delete or update on emp for each row
declare
typeOp varchar(15);
BEGIN
if inserting then
typeOp := 'Insertion';
elsif updating then
typeOp := 'Modification';
elsif deleting then
typeOp := 'Suppression';
end if ;
insert into historique values (sysdate, typeOp, user, :old.num, :new.num);
end ;
/
```

1.2.3 Déclencheur LDD

```
set serveroutput on
create or replace trigger majRefusee
before drop on isa.schema
begin
if trim(to_char(sysdate,'DAY')) = 'FRIDAY' then
raise_application_error (-20001,'forbidden on friday') ;
else
dbms_output.put_line(to_char(sysdate,'DAY'));
```

```
end if ;
end ;
/
```

Un corps de programme PL/SQL peut comprendre des :
— commandes SQL
— opérateurs de comparaison
— instructions de contrôle conditionnel
— instructions de contrôle itératif
— gestion des curseurs
— gestion des erreurs

2.4 Les curseurs
Les curseurs (CURSOR) permettent de traiter les résultats d'une requête en vue d'analyses ou d'interprétations diverses. Ils peuvent être comparés à des tables temporaires.

Curseur implicite
Un ordre SQL peut ne retourner qu'un n-uplet, il n'est pas nécessaire de définir un curseur de
manière explicite, on parle alors de curseur implicite.

```
declare
name emp.nom%type;
numero emp.num%type;
begin
select num, nom into numero, name from emp where num=79 ;
dbms_output.put_line('j''affiche le numero et nom '||numero||' '||name) ;
end ;
/
```

Curseur explicite
Dès que l'ordre SELECT SQL retourne une réponse comprenant plus de deux n-uplets,

```
declare
cursor mon_curseur is select num, nom, fonction from EMP ;
begin
For c in mon_curseur
loop
dbms_output.put_line(c.num||' '||c.nom||' '||c.fonction);
end loop ;
end ;
/
```

2.8 Les procédures et les fonctions

***Paquetages**
— sécurité et contrôle au sein des bases de données
— performance, efficacité des applicatifs mis en place
— modularité, clarté, flexibilité et évolutivité des applicatifs

Propriétés d'une transaction
Atomicité : lors d'une exécution d'une transaction, toutes ses actions sont exécutées ou bien aucune ne l'est.
Cohérence : les modifications apportées à la bd lors d'une transaction doivent être valides cad respecter les contraintes d'intégrité.
Isolation : chaque transaction est isolée, pour s'affranchir des incohérences lors d'exécutions concurrentes
Durabilité ou Permanence : les effets d'une transaction, qui s'est exécutée correctement, doivent survivre à une panne

Transaction sérialisable :
Une exécution d'un ensemble de transactions est sérialisable si elle est équivalente à une exécution séquentielle (ou en série) de trans-actions. Quand les transactions sont arbitraires, la sériabilité est la seule `a pouvoir assurer un entrelacement correct.
Isoler au moyen de verrous
Isoler un élément dans une transaction en le verrouillant (lock). Les verrous sont définis par deux opérations : Verrouillage à deux phases (2PL)

Verrouiller(A) (lock(A)): cette opération oblige toute transaction `a attendre le déverrouillage de l'´element A si elle a besoin de cet élément
Déverrouiller(A) (unlock(A)): la transaction effectuant cette opération libère le verrou qu'elle avait obtenu sur A et permet `a une autre transaction candidate, en attente, de poser, `a son tour, un verrou.

La pose de verrous dégrade les performances du système et impose des temps d'attente

Transactions sérialisables et verrous

La sériabilité impose aux transactions que tous les verrouillages précèdent tous les déverrouillages. Les transactions sont dites à deux phases : une phase d'acquisition des verrous puis une phase de libération (`a la validation ou `a l'annulation). Aucun granule ne reste verrouillé après la fin d'une transaction.

Read-uncommitted : transaction (avec actions en écriture) sans besoin de validation pour être visible par les autres
Transactions read-committed : transaction (avec actions en écriture) devant être validée pour être visible par les autres transactions
Repeatable-read : transaction (incluant des actions en lecture et écriture) devant être validée pour être visible par les autres transactions
Serializable : transaction isolée : insensible aux changements intermédiaires des autres

Architecture Oracle
Principale structure : System Global Area (SGA)
> 2% de la taille totale de la base données (fichiers).
Répartition :
50% Cache de données (database buffer cache)
40% Shared Pool
10% Redo log Buffers
Autres structures facultatives : large pool, java pool
La vue V$SGASTAT : taille des structures SGA par exemple
espace libre dans la SGA :
select * from v$sgastat where name='free memory'

La Shared Pool : réduire le temps et le coût de l'exécution
Vues du méta-schéma d'intérêt
v$librarycache, v$sqlarea, vsql, vsessstat, v$sysstat, v$sqltext, v$db_object_cache

Buffer Cache
Blocs : donnée, index, rollback (ou undo), temporaire
Le Redo Log Buffer (tampon journalisation)
Informations sur les transactions réalisées (bloc : ancienne et nouvelle valeurs)

Les Processus « Background »
PMON, SMON, DBWR, LGWR
Si l'un de ces quatre processus échoue, l'instance sera détruite et devra être redémarrée.

PMON : Processus Monitor

Nettoie les connexions terminées de façon anormale
Défait les transactions non validées
Libère les verrous qui avaient été posés par un processus qui s'est terminé en erreur
Libère les ressources SGA allouées par le processus en erreur

SMON : Processus System Monitor
Réalise la restauration automatique d'instance
Récupère l'espace occupé par des segments temporaires qui ne sont plus utilisés
Fusionne les zones contiguës d'espace libre dans les fichiers de données

DBWR : Processus Database Writer
Le DBWR gère les tampons (buffers) de la base de données pour que les processus serveurs puissent toujours trouver des buffers libres en SGA.
Le processus DBWR écrit les buffers modifiés vers les fichiers de données.
Utilise un algorithme (LRU) qui garde les blocs les plus récemment utilisés en mémoire,
? Diffère les écritures en vue d'optimiser les E/S

LGWR : Processus Log Writer
Le LGWR écrit les entrées du buffer redo log dans les fichiers redo log lorsque :
- Un commit se produit,
- le buffer est rempli au tiers,
- le DBWR achève l'écriture des buffers de données lors du checkpoint,
- Un time-out se produit.

Tablespace
Une base de données est divisée en unités logiques de stockage appelées tablespaces, servant à réunir des structures connexes.Il est courant de réunir dans un tablespace tous les objets d'une application afin de simplifier certaines opérations administratives.Il existe deux types de tablespace : le tablespace SYSTEM et les tablespace utilisateurs destinés à contenir les données.

LES INDEX

Structure de données sous forme d'arbre qui maintient dynamiquement un ensemble d'elements de manière a ce que l'arbre soit équilibre :
L'équilibre est important pour toutes les opérations usuelles sur une table :
recherche, insertion, suppression
- pour chaque noeud branche de l'arbre : n cles et n+1 pointeurs,
- un noeud (sauf la racine) est de a moitie plein a plein
- pour les noeuds feuilles : elles sont toutes au même niveau et elles contiennent les cles et les pointeurs sur les données

Plusieurs variants d'arbres – B+-tree et B*tree
- Dans un B-arbre, les noeuds intermediaires peuvent contenir des pointeurs sur des donnees
- Dans un B+arbre, seules les feuilles contiennent des pointeurs sur les donnees et les noeuds sont doublement chaines
- Dans un B*arbre les noeuds sont au moins a 2/3 plein

Opérations d'accès aux tables L'opération d'accès aux tables (TABLE ACCESS) possède differentes options :

1. TABLE ACCESS FULL : pour parcourir toutes les lignes d'une table. L'optimiseur lit toutes les données d'une table et ne retient que celles qui satisfont le ou les critères de sélection. Cette option est toujours choisie par l'optimiseur lorsqu'il n'y a pas d'index.

2. TABLE ACCESS HASH : pour parcourir les lignes d'une table `a partir des cl´es de l'index hash

3. TABLE ACCESS BY INDEX ROWID : pour parcourir les lignes d'une table `a partir des

valeurs d'index. L'objet ROWID spécifie l'identifiant de fichier de donn´ees, le num´ero de bloc et le numéro de la ligne dans le bloc pour atteindre la donnée recherchée. L'optimiseur choisit ce chemin d'exécution lorsqu'il y a un index et que la colonne sur lequel porte cet index est exploitée dans la requête. Lorsqu'il s'agit d'une opération de sélection sur la colonne indexée dans la clause WHERE, l'optimiseur utilise les opérations d'index range scan ou index unique scan et lorsqu'il s'agit d'une opération de projection, l'optimiseur utilise l'opération d'index full scan. Quand la projection s'effectue sur la seule colonne porteuse d'un index, l'opération HMIN328 M2 Info 2018-2019 9
index full scan permet d'éviter d'aller interroger dans les blocs de données car les blocs d'index suffisent). Il est toujours possible d'influencer, au travers de ce qui est appelé un **HINT** (en français directive), la tâche de l'optimiseur. Par exemple, l'optimiseur est invité ci dessous à utiliser l'option TABLE ACCESS FULL (nom f est porteur d'une contrainte de clé primaire)

Introduction aux Bases de Données Réparties (BDR)

Avantages
• extensibilité
• partage de données hétérogènes et réparties
• performances
• disponibilité des données

Inconvénients
• administration complexe
• distribution du contrôle

Constituants du schéma global
• schéma conceptuel global
• donne la description globale et unifiée de toutes les données de la BDR (e.g., des relations globales)
• indépendance à la répartition

• schéma de placement
• règles de correspondance avec les données locales
• indépendance à la localisation, la fragmentation et la duplication

Objectifs de la décomposition
• fragmentation
• trois types : horizontale, verticale, mixte
• performances en favorisant les accès locaux
• équilibrer la charge de travail entre les sites (parallélisme)
• duplication (ou réplication)
• favoriser les accès locaux
• augmenter la disponibilité des données

Allocation des fragments aux sites
Non-dupliquée
• partitionnée : chaque fragment réside sur un seul site
Dupliquée
• chaque fragment sur un ou plusieurs sites
• maintien de la cohérence des copies multiples
Règle intuitive:
• si le ratio est [lectures/màj] > 1, la duplication est avantageuse

Lien à une table dans une BD distante specifié par :
• nom de lien

• nom de l'utilisateur et password
• chaîne de connexion SQL*Net (protocole réseau, nom de site, options, etc…)

Exemple

• CREATE DATABASE LINK empLien
• CONNECT TO user1
• IDENTIFIED BY mdp1
• USING ‘master’

Difficultés des bases réparties

?Choix et maintien des fragments
•En fonction des besoins des applications
•Heuristiques basées sur l’affinité d’attributs et le regroupement
?**Disponibilité des données**
•Dépend de la robustesse du protocole 2PC; implique une grande fiabilité du réseau et des participants
?Echelle
• Le nombre de sessions simultanées est limité par l’architecture 2-tiers; grande échelle nécessite un moniteur transactionnel

Bases de données répliquées

Définitions

?Réplica ou copie de données
• Fragment horizontal ou vertical d’une table stockée dans une base de données qui est copiée et transféré vers une autre base de données
• L’original est appelé la copie primaire et les copies sont appelées copies secondaires
?Transparence
• Les applications clientes croient à l’existence d’une seule copie des données qu’ils manipulent :
? soit « logique » dans le cas d’une vue
? soit physique dans le cas de vues matérialisées

Les avantages de la réplication

?Amélioration des performances
• lecture de la copie la plus proche
• évitement du goulot d'étranglement du serveur unique
?Amélioration de la disponibilité
• lors d'une panne d'un serveur, on peut se replier sur l'autre
• Disponibilité = 1 - probabilité_panneN
? probabilité de panne = 5% et 2 copies => disponibilité = 99.75%
?Meilleure tolérance aux pannes
• possibilité de détecter des pannes diffuses

Les problèmes de la réplication

?Convergence
• les copies doivent être maintenues à jour
• à un instant donné, elles peuvent être différentes
• mais elles doivent converger vers un même état cohérent où toutes les mises à jour sont exécutées partout dans le même ordre
?Transparence : le SGBD doit assurer
• la diffusion et la réconciliation des mises à jour
• la résistance aux défaillances

Diffusion synchrone

?Une transaction met à jour toutes les copies de toutes les données qu ’elle modifie.
+ mise à jour en temps réel des données
- trop coûteux pour la plupart des applications
- pas de contrôle de l ’instant de mise-à-jour

Diffusion asynchrone

?Chaque transaction met à jour une seule copie et la
mise-à-jour des autres copies est différée (dans d’autres transactions)
?Réplication asymétrique : toutes les transactions mettent à jour la même copie
?Réplication symétrique : les transactions peuvent mettre à jour des copies
différentes
+ mise-à-jour en temps choisi des données
+ accès aux versions anciennes puis nouvelles
- l'accès à la dernière version n'est pas garanti

```
SELECT table_schema, table_name, grantor, grantee, privilege FROM all_tab_privs
WHERE grantor != 'TBONDETTI' AND privilege = 'SELECT' AND grantee IN ('PUBLIC');
```