Chapitre 5

Bases de données distribuées

Master II STIC

Année universitaire 2025/2026

1. Introduction

Une base de données distribuée (BDD) est un ensemble de données :

- logiquement liées,
- mais physiquement réparties sur plusieurs sites interconnectés,
- et contrôlées par un SGBD distribué

Objectifs :

- Disponibilité
- Répartition de charge
- Performance locale
- Tolérance aux pannes
- Autonomie locale

BDD Oracle

Oracle supporte les BDD distribuées via :

- **Database Links**
- **Materialized Views**
- **Advanced Replication**
- **Two-Phase Commit (2PC)**

2. Architecture des BDD distribuées

2.1. Types d'architectures

- Système homogène : tous les sites → Oracle.
- Système hétérogène : Oracle + MySQL + SQL Server…

Oracle permet :

- Heterogeneous Services (HS)
- Oracle Gateway (vers SQL Server, DB2...)

3. Fragmentation et Allocation

La fragmentation consiste à diviser une table en morceaux distribués sur plusieurs sites.

3.1. Fragmentation horizontale

Séparation par lignes.

Exemple :On veut stocker les employés par région :

Site 1 : employés d'Algérie

Site 2 : employés de Tunisie

Oracle : Table partitionnée (similaire à fragmentation)

```
CREATE TABLE employe (
    id NUMBER PRIMARY KEY,
    nom VARCHAR2(50),
    pays VARCHAR2(20)
)
PARTITION BY LIST (pays) (
    PARTITION p_algerie VALUES ('Algerie'),
    PARTITION p_tunisie VALUES ('Tunisie')
);
```

3. Fragmentation et Allocation

3.2. Fragmentation verticale

Séparation par colonnes.

Exemple :

Site A : infos personnelles

Site B : infos professionnelles

3. Fragmentation et Allocation

```
-- Fragment site A
CREATE TABLE emp_info_pers (
    id NUMBER PRIMARY KEY,
    nom VARCHAR2(30),
    adresse VARCHAR2(50)
);

-- Fragment site B
CREATE TABLE emp_info_prof (
    id NUMBER PRIMARY KEY,
    salaire NUMBER,
    poste VARCHAR2(30)
);
```

3. Fragmentation et Allocation

3.3. Fragmentation mixte

Découpage vertical + horizontal.

3. Fragmentation et Allocation

3.4. Allouer les fragments

Avec Oracle, on utilise :

- Oracle RAC
- Database Links
- Materialized Views

3.4 Stratégies de Réplication

La réplication consiste à dupliquer tout ou partie des données sur plusieurs sites du système distribué afin

- d'améliorer les performances,
- la disponibilité et la tolérance aux pannes.

Oracle propose plusieurs mécanismes pour gérer ces copies en assurant la cohérence et la synchronisation entre elles.

3.4 Stratégies de Réplication

3.4.1 Objectifs de la réplication

- **Disponibilité accrue**

Si un site tombe en panne, les autres peuvent continuer à fonctionner grâce aux copies locales.

- **Performances améliorées**

Les utilisateurs accèdent à une copie proche géographiquement → réduction de la latence.

- **Répartition de charge**

Plusieurs copies peuvent servir les requêtes simultanément.

- **Tolérance aux pannes**

Les copies permettent la récupération rapide des données.

3.4.2 Types de réplication

A. Réplication complète (Full Replication)

Toutes les données sont copiées sur tous les sites.

Avantages :

- Disponibilité maximale
- Lecture très rapide

Inconvénients :

- Synchronisation coûteuse
- Risque élevé de conflits

Exemple Oracle : réplication complète via Multi-Master Replication.

3.4.2 Types de réplication

B. Réplication partielle (Fragment Replication)

Seules certaines tables ou certains fragments sont répliqués.

Avantages :

- Moins de trafic réseau
- Meilleure adaptation au besoin local

Inconvénients :

- Complexité de gestion des fragments

Exemple Oracle : matérialiser seulement des tables fréquemment consultées.

3.4.2 Types de réplication

C. Réplication synchrone

Les mises à jour sont propagées instantanément vers les autres copies.

Avantages : cohérence forte (tous les sites voient les mêmes données)

Inconvénients : latence réseau, blocages éventuels

Exemple Oracle :

- RAC (Real Application Cluster) assure une mise à jour synchrone entre nœuds.
- Oracle Streams AQ peut forcer des validations distribuées (2PC).

3.4.2 Types de réplication

D. Réplication asynchrone

Les mises à jour sont d'abord locales, puis propagées en différé.

Avantages :

Très bonnes performances

Moins de blocages

Inconvénients :

Risque de conflits (mises à jour concurrentes)

Exemple Oracle :

- Materialized Views (MVs) FAST REFRESH
- Oracle GoldenGate

3.4.3 Techniques de synchronisation dans Oracle

A. Materialized Views (MV)

Oracle utilise les MVs pour répliquer les données de manière asynchrone.

Modes de rafraîchissement :

- FAST : seules les modifications dans les logs MV sont appliquées.
- COMPLETE : reconstruction totale de la vue matérialisée.
- FORCE : Oracle choisit FAST si possible sinon COMPLETE.

3.4.3 Techniques de synchronisation dans Oracle

Materialized Views (MV)

Exemple Oracle :

```
CREATE MATERIALIZED VIEW emp_mv
REFRESH FAST START WITH SYSDATE NEXT SYSDATE + 1/24
AS SELECT * FROM employees@site_remote;
```

3.4.4 Gestion des conflits dans Oracle

Quand deux sites modifient la même donnée avant synchronisation, Oracle doit arbitrer.

Types de conflits :

- Update conflicts — deux mises à jour concurrentes
- Delete conflicts — une mise à jour arrive après une suppression
- Unique key conflicts — violation de contrainte d'unicité

Méthodes de résolution :

- Last update wins
- Timestamp-based
- Priorité des sites
- Règles personnalisées (PL/SQL)

3.4.4 Gestion des conflits dans Oracle

Exemple Oracle :

```
BEGIN
  DBMS_REPCAT.ADD_UPDATE_RESOLUTION(
    sname => 'HR',
    oname => 'EMPLOYEES',
    conflict_type => 'UPDATE',
    method_name => 'LATEST_TIMESTAMP',
    priority_group => NULL
  );
END;
```

3.4.5 Exemple complet : réplication Oracle avec Materialized Views

Sur le site maître :

```
GRANT CREATE MATERIALIZED VIEW TO site_replica;

CREATE MATERIALIZED VIEW LOG ON employees
WITH PRIMARY KEY, ROWID (first_name, last_name, salary)
INCLUDING NEW VALUES;
```

3.4.5 Exemple complet : réplication Oracle avec Materialized Views

Sur le site répliqué :

```
CREATE MATERIALIZED VIEW emp_mv
REFRESH FAST
START WITH SYSDATE
NEXT SYSDATE + (1/96)  -- rafraîchissement toutes les 15 minutes
AS SELECT * FROM employees@link_master;
```

3.4.6 Bonnes pratiques Oracle

- Utiliser FAST REFRESH quand possible.
- Limiter les colonnes enregistrées dans les logs.
- Synchroniser en dehors des heures de pointe.
- Prévoir une stratégie de résolution automatique des conflits.
- Surveiller les MVs avec :

```
SELECT mview_name, staleness FROM user_mviews;
```

4. Communication inter-sites dans Oracle

Oracle permet à une base de communiquer avec d'autres bases distantes grâce à des Database Links.

Ces liens permettent d'exécuter des requêtes sur un autre serveur comme si les tables étaient locales.

4. Communication inter-sites dans Oracle

4.1. Database Links

Un database link est un objet Oracle stocké dans le dictionnaire.

Il contient :

- l'adresse réseau du serveur distant,
- l'identifiant utilisateur,
- le mot de passe,
- le service Oracle à contacter.

4. Communication inter-sites dans Oracle

4.2. Syntaxe

Créer un database link privé

```
CREATE DATABASE LINK site_remote
CONNECT TO remote_user IDENTIFIED BY remote_pwd
USING 'remotesid';
```

Créer un database link public

```
CREATE PUBLIC DATABASE LINK site_remote
CONNECT TO remote_user IDENTIFIED BY remote_pwd
USING 'remotesid';
```

Vérifier le lien

```
SELECT * FROM dual@site_remote;
```

4. Communication inter-sites dans Oracle

4.3. Requêtes distribuées

Sélection distribuée

```
SELECT nom, salaire
FROM employes@site_remote;
```

Jointure entre sites

```
SELECT e.nom, d.nom_dept
FROM employes@site_local e
JOIN departements@site_remote d
    ON e.dept_id = d.id;
```

Oracle choisit automatiquement le plan d'exécution optimal (semi-join, shipping, etc.).

5. Réplication et synchronisation

La réplication consiste à **dupliquer des données** entre plusieurs sites pour améliorer :

- Disponibilité
- Performance
- Tolérance aux pannes

5. Réplication et synchronisation

5.1. Types de réplication

Réplication unidirectionnelle

Site maître → Site(s) esclave(s)

Réplication bidirectionnelle (multi-maître)

Sites pouvant tous modifier les données.

– Utilisée pour applications critiques.
– Risques de conflits (doivent être résolus par Oracle Replication Manager).

Réplication matérielle via Materialized Views

C’est la plus utilisée.

5. Réplication et synchronisation

5.2. Materialized Views

Synchronisation automatique d'une table distante.

Création d'un materialized view

```
CREATE MATERIALIZED VIEW emp_mv
REFRESH FAST
START WITH SYSDATE
NEXT SYSDATE + 1/24
AS SELECT * FROM employes@remote_link;
```

5. Réplication et synchronisation

5.2. Materialized Views

Modes de rafraîchissement

- **FAST :** uniquement les modifications
- **COMPLETE :** recrée entièrement la vue matérialisée
- **ON COMMIT :** rafraîchissement à chaque transaction
- **ON DEMAND :** refresh manuel
- **Rafraîchir manuellement**

```
EXECUTE DBMS_MVIEW.REFRESH('EMP_MV');
```

6. Mise à jour distribuée et contrôle de cohérence

6.2. Le protocole 2PC (Two-Phase Commit)

Oracle utilise automatiquement **2PC** pour coordonner les transactions distribuées.

- **Phase 1 – PREPARE**

Chaque site vote COMMIT ou ROLLBACK.

- **Phase 2 – COMMIT**

Si tous les sites votent COMMIT → transaction validée globalement.
Sinon → ROLLBACK global.

6. Mise à jour distribuée et contrôle de cohérence

6.3. Gestion des blocages

Oracle maintient :

- **DBA_2PC_PENDING**
- **DBA_2PC_NEIGHBORS**

pour analyser les transactions bloquées.

- **Forcer la résolution**

COMMIT FORCE 'transaction_id';

ROLLBACK FORCE 'transaction_id';

6. Mise à jour distribuée et contrôle de cohérence

6.3. Gestion des blocages

Oracle maintient :

- **DBA_2PC_PENDING**
- **DBA_2PC_NEIGHBORS**

pour analyser les transactions bloquées.

- **Forcer la résolution**

COMMIT FORCE 'transaction_id';

ROLLBACK FORCE 'transaction_id';

7. Transparence dans les BDD distribuées

Objectif : rendre la distribution **invisible** pour l'utilisateur.

7.1. Transparence de localisation

L'utilisateur ne sait pas où les données sont stockées.

Requête Oracle :

SELECT * FROM produit@site2;

La syntaxe reste la même pour n'importe quel site.

7. Transparence dans les BDD distribuées

7.2. Transparence de fragmentation

Utilisateur ne sait pas si la table est fragmentée horizontalement, verticalement, ou hybride.

Les vues Oracle cachent les fragments :

```
CREATE VIEW clients_global AS
SELECT * FROM clients_dz
UNION ALL
SELECT * FROM clients_fr;
```

7. Transparence dans les BDD distribuées

7.3. Transparence de réplication

Si un site est indisponible, Oracle peut utiliser une copie répliquée.

7.4. Transparence de traitement (requêtes distribuées)

Oracle choisit automatiquement le plan d’exécution optimal.

8. Mini-projet complet Oracle distribué

Mini-projet : **Système de gestion de commandes distribué**
3 sites :

- **SITE1** : Clients
- **SITE2** : Produits
- **SITE3** : Commandes (jointure clients + produits)

8. Mini-projet complet Oracle distribué

8.1. Étape 1 – Création des database links

- Sur SITE3 :

```
CREATE DATABASE LINK link_site1
CONNECT TO user1 IDENTIFIED BY pwd1
USING 'site1';

CREATE DATABASE LINK link_site2
CONNECT TO user2 IDENTIFIED BY pwd2
USING 'site2';
```

8. Mini-projet complet Oracle distribué

8.2. Étape 2 – Tables distribuées

- **SITE1 – Clients**

```
CREATE TABLE clients (
   id NUMBER PRIMARY KEY,
   nom VARCHAR2(50),
   ville VARCHAR2(50));
```

- **SITE2 – Produits**

```
CREATE TABLE produits (
   id NUMBER PRIMARY KEY,
   nom VARCHAR2(50),
   prix NUMBER);
```

SITE3 – Commandes

```
CREATE TABLE commandes (
   id NUMBER PRIMARY KEY,
   id_client NUMBER,
   id_produit NUMBER,
   qte NUMBER);
```

8. Mini-projet complet Oracle distribué

8.3. Étape 3 – Jointures inter-sites

```
SELECT c.nom AS client,
       p.nom AS produit,
       co.qte
FROM commandes co
JOIN clients@link_site1 c ON co.id_client = c.id
JOIN produits@link_site2 p ON co.id_produit = p.id;
```

8. Mini-projet complet Oracle distribué

8.4. Étape 4 – Réplication des produits sur SITE3

```
CREATE MATERIALIZED VIEW produits_mv
REFRESH FAST
AS SELECT * FROM produits@link_site2;
```

8. Mini-projet complet Oracle distribué

8.5. Étape 5 – Mise à jour distribuée (2PC)

Insertion d’une commande (SITE3) :

INSERT INTO commandes VALUES (1, 10, 20, 3);

- COMMIT; -- Oracle déclenche automatiquement 2PC

SITE3 écrit localement

SITE1 vérifie existence du client

SITE2 vérifie existence du produit

Les 3 sites votent COMMIT → validation globale

8. Mini-projet complet Oracle distribué

8.6. Étape 6 – Résolution de panne (simulateur)

- Si un site tombe en panne :

```
SELECT * FROM DBA_2PC_PENDING;

ROLLBACK FORCE 'transaction-ID';
```