

Bases de Données Réparties Concepts et Techniques

Matthieu Exbrayat

ULP Strasbourg - Décembre 2007

[Définition]

- Une base de données répartie (distribuée) est une base de données logique dont les données sont distribuées sur plusieurs SGBD et visibles comme un tout.
- Les données sont échangées par messages
- Si les données sont dupliquées, on parle plutôt de BD répliquée

[Principe fondamental]

- « To the user, a distributed system should look exactly like a nondistributed system. »
(C. Date, Introduction to Database Systems)
- ▣ Autonomie locale
- ▣ Egalité entre sites (pas de site « central »)
- ▣ Fonctionnement continu (pas d'interruption de service)
- ▣ Localisation transparente
- ▣ Fragmentation transparente
- ▣ Indépendance à la réplication
- ▣ Exécution de requêtes distribuées
- ▣ Gestion de transactions réparties
- ▣ Indépendance vis-à-vis du matériel
- ▣ Indépendance vis-à-vis du Système d'Exploitation
- ▣ Indépendance vis-à-vis du réseau
- ▣ Indépendance vis-à-vis du SGBD



■ Autonomie locale

- La BD locale est complète et autonome (intégrité, sécurité, gestion), elle peut évoluer indépendamment des autres (upgrades...)

■ Égalité entre sites

- Un site en panne ne doit pas empêcher le fonctionnement des autres sites (mais perturbations possibles)

■ Fonctionnement continu

- Distribution permet résistance aux fautes et aux pannes (en théorie)



■ Localisation transparente

- Accès uniforme aux données quel que soit leur site de stockage

■ Fragmentation transparente

- Des données (d'une même table) éparpillées doivent être vues comme un tout

■ Indépendance à la réplication

- Les données répliquées doivent être maintenues en cohérence (délai possible)



■ Requêtes distribuées


- L'exécution d'une requête peut être répartie (automatiquement) entre plusieurs sites (si les données sont réparties)

■ Transactions réparties

- Le mécanisme de transactions peut être réparti entre plusieurs sites (si ...)

■ Indépendance vis-à-vis du matériel

- Le SGBD fonctionne sur les différentes plateformes utilisées

- 
- Indépendance vis-à-vis du SE
 - Le SGBD fonctionne sur les différents SE...
 - Indépendance vis-à-vis du réseau
 - Le SGBD est accessible à travers les différents types de réseau utilisés
 - Indépendance vis-à-vis du SGBD
 - La base peut être distribuée sur des SGBD hétérogènes
 - En théorie...

Quelques termes équivoques...

- BD distribuée
 - Un schéma global
 - Les données sont réparties sur plusieurs sites, accessibles à partir du site central ou de tous les sites
- BD fédérée
 - Chaque site a son schéma local, pas forcément inclus entièrement dans le schéma global (il y a un site central)
- Système multi-bases
 - Pas de schéma global, pas de site central. Accès à (une partie des) données distantes.
- Mais il existe d'autres définitions !
 - Ex : fédérée = approche ascendante partielle ...

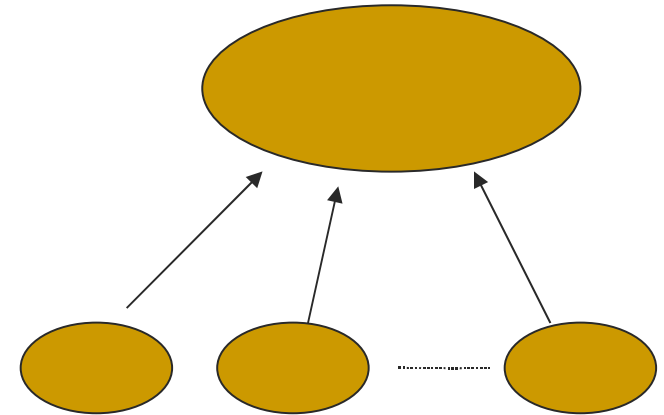
[Conception de BD répartie]

- On ne met en place une BD répartie qu'en cas de réel besoin
 - Démarche de conception délicate
 - Gestion complexe
 - L'évolution du SI peut invalider la solution retenue...
- Des raisons valables :
 - Volumes de données, sites distants, etc.
 - Fusions de SI

Deux approches de conception

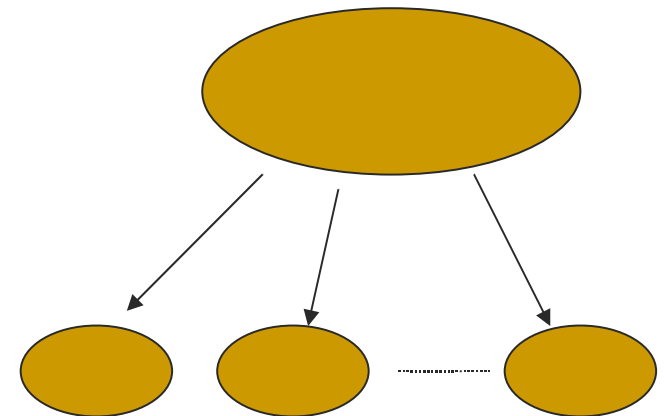
■ Conception ascendante

- Part de l'existant
- Intègre bases locales dans schéma global



■ Conception descendante

- On part du schéma global
- On le scinde en schémas locaux



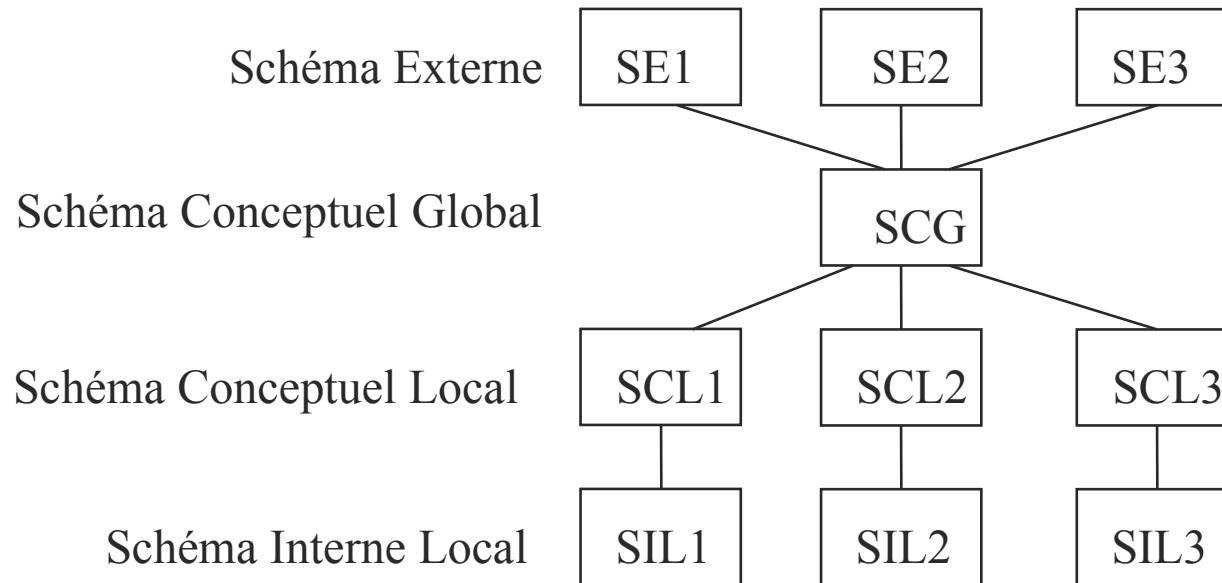
■ Conception descendante

- On part de zéro (nouvelle base)
- Recherche de performance (pas forcément de répartition géographique)
- Assez peu fréquent

■ Conception ascendante

- Distribution pré-existante
- Nécessite consolidation, uniformisation (« réconciliation sémantique »)
 - Identifier les données semblables
 - Accorder leurs types, gérer leur cohérence...
 - Interfacer ou adapter les SGBD...
- Ex : fusion, mise en place DW

Schémas d'une BD répartie



[Fragmentation des données]

- Fragmentation horizontale

- Les tuples sont répartis

Yellow	Yellow	Yellow	Yellow	Yellow
Brown	Brown	Brown	Brown	Brown
Green	Green	Green	Green	Green
Brown	Brown	Brown	Brown	Brown

- Fragmentation verticale

- Les tuples sont découpés et fragmentés
- Nécessite colonne commune (clé ou unique) dupliquée

Brown	Brown	Yellow	Green	Green
Brown	Brown	Yellow	Green	Green
Brown	Brown	Yellow	Green	Green
Brown	Brown	Yellow	Green	Green

[Fragmentation horizontale]

- En conception descendante
 - Adéquation géographique
 - Recherche de performance (I/O, traitements)
- En conception ascendante
 - Des données comparables dans différentes bases
 - Pb : consolidation correcte (unicité des clés, types des attributs...)

[Fragmentation verticale]

- On projette la table sur des attributs différents suivant site.
- Comme frag. horizontale, peut correspondre à consolidation ou recherche de perf.
- La reconstruction des tuples doit être possible (et validée)
- Mêmes problèmes que f.h.

[Frag. horizontale dérivée]

- Placer deux tables en relation de manière à localiser les jointures
- Une des deux tables doit être fragmentée en fonction de l'autre (semi jointure)
- Plutôt en approche descendante
- Peut introduire des redondances

Mise en pratique de la fragmentation

- Historique : des SGBDs distribués
 - R*, etc.
 - Plutôt expérimental
- Dans les SGBD commerciaux actuels
 - Pas de fragmentation explicite au niveau du schéma
 - Assemblage = création de vue (ou de snapshot)
 - Distribution des données ?
 - Une solution = triggers

Mise en œuvre sous SQL (Assemblage)

- Frag. Horizontale
 - CREATE VIEW V1
AS SELECT Table1.cle, Table1.attr1
FROM Table1@site1
UNION
SELECT Table2.cle, Table2.attr1
FROM Table2@site2

Mise en œuvre sous SQL (Assemblage)

■ Frag. Verticale

- CREATE VIEW V1
AS SELECT Table1.cle, Table1.attr1,
Table2.attr2
FROM Table1@site1, Table2@site2
WHERE Table1.cle=Table2.cle

■ Remarque :

- l'attribut de fragmentation n'est pas forcément la clé primaire...
- En frag. verticale, il faut au néanmois que ce soit une clé

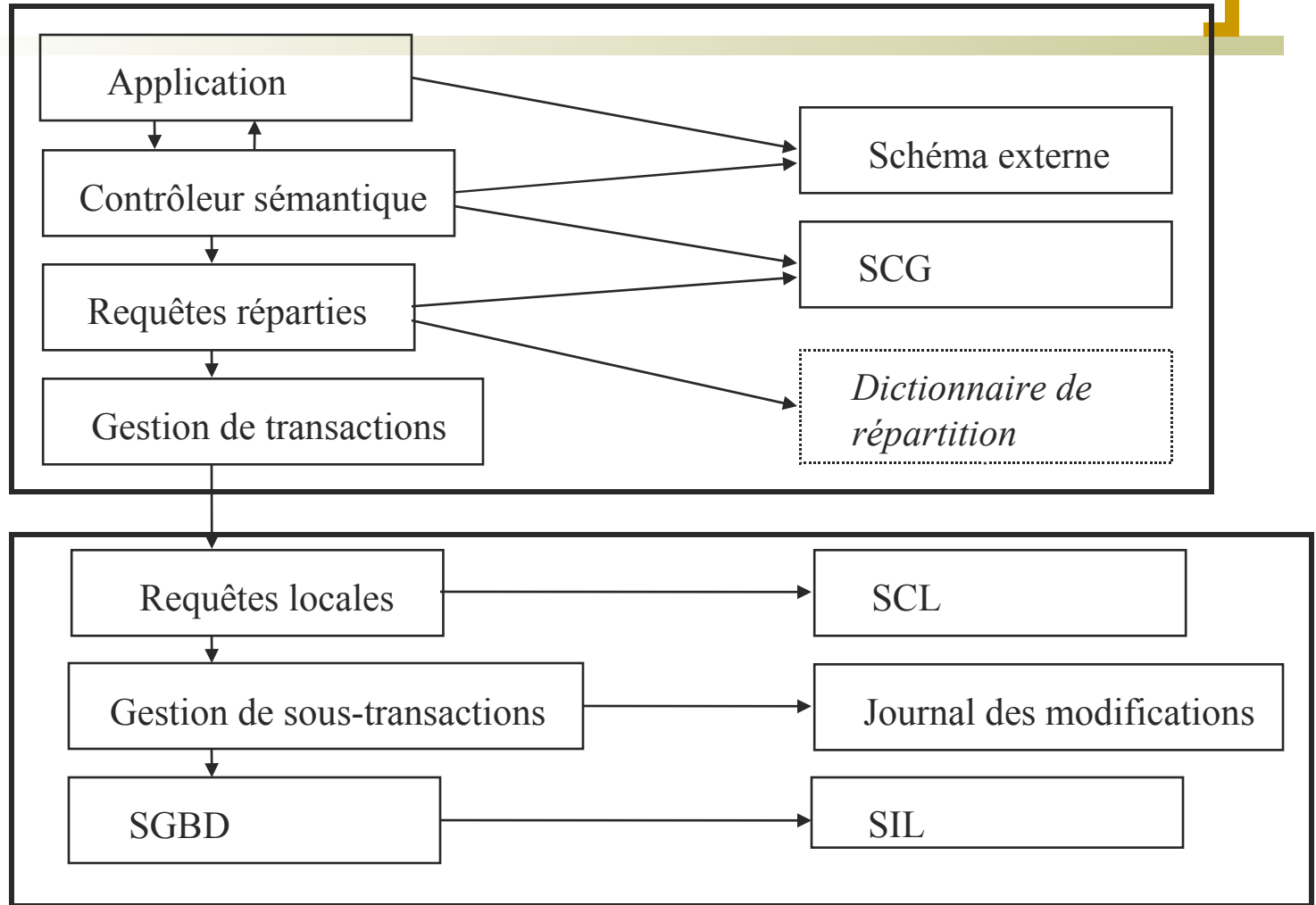
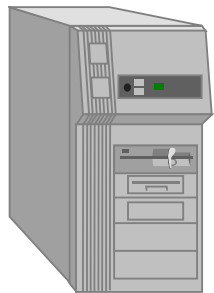
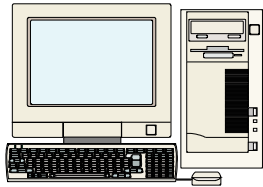
[Requête répartie]

- Les opérations précédentes (par exemple) accèdent à des données situées sur différents sites.
- Le SGBD gère les accès distants en répartissant la requête
 - Exécution distante de la sous requête
 - Récupération de données résultat
 - Assemblage local (requête en lecture)

Gestion de l'hétérogénéité

- Hétérogénéité « sans problème »
 - SE et réseau : géré par SGBD (si « bon » SGBD)
 - Version de SGBD : niveau de SGBD le plus ancien
- Hétérogénéité plus délicate
 - SGBD : pb des dialectes de SQL
 - ▮ passerelles entre SGBD
 - ▮ Ex : ODBC (au départ sous Windows mais porté sous d'autres OS)
 - ▮ Ex : passerelles propriétaires SGBD à SGBD

BDR comment ça marche



Communication Inter-sites

- Chaque SGBD dispose d'un démon permettant les connexions distantes, sur un mode client - serveur
 - *Listener (médiateur)*
- Chaque SGBD dispose d'une table des BDs accessibles
 - *Nom >> doit être unique !!!*
 - *Adresse*
 - *Protocole*
- Cette approche permet aussi un équilibrage de charge transparent...

[Exemple : Oracle]

- Permet la distribution et la réplication
- Assure une bonne transparence à différents niveaux
- Système de nommage simple
 - sales.france.europe.computers
- Accès BD distante : LINK
 - CREATE DATABASE LINK sales...
- Accès table distante : schéma.table@base
 - svc_maint.emp@sales.france.europe.computers
- Lien public, lien privé

[Transparence]

- Localisation : synonymes
 - CREATE PUBLIC SYNONYM employes FOR
svc_maint.emp@sales.france.europe.computers
- Requêtes et transactions
- Opérations internes
- Réplication

[Caractéristiques]

- Autonomie des sites
 - Gestion indépendante, upgrades localisés...
- Sécurité
 - Les utilisateurs et leurs rôles doivent être connus sur chaque site accédé
 - Possibilité d'utiliser un Security Server
 - Encryptage
- Administration globale : entreprise manager

Mise en oeuvre en SQL

(Insertion avec les triggers Oracle)

- CREATE TRIGGER Tr1
INSTEAD OF INSERT on Table
BEGIN
IF :New.cle < 1000 THEN
 INSERT INTO Table1@site1(cle,attr1)
 VALUES(:New.cle,:New.attr);
ELSE
 INSERT INTO Table2@site2(cle,attr2)
VALUES (:New.cle,:New.attr);
END IF;
END;

[Complément sur triggers]

- Utilisables également pour update et delete (:New et :Old) pour
 - Suppression (:Old)
 - Modification hors attribut de distribution (:New)
 - Modification attribut de distribution (:New et :Old pour vérifier si bon site)
- Doit être écrit « à la main » par le dba...

[Gestion des contraintes distribuées]

- Types de contraintes
 - Domaine
 - Unicité
 - Intégrité référentielle
- Dans un SGBD centralisé, c'est bien géré
 - Dans la définition des tables (principalement)
- Le côté manuel de la distribution limite les possibilités de gestion des contraintes
 - Utilisation de vues >> pas de contraintes

[Exemples]

- Check / NOT NUL
 - Se gère par fragment
- Unicité
 - Gérable par fragment
 - Globalement ? Réplication OK, Fragmentation KO
 - Utilisation de trigger
 - Site maître pour numérotation >> perte autonomie
- Intégrité Référentielle
 - Pas géré en inter-bases
 - Réplication OK, Fragmentation KO
 - Utilisation trigger
- Ca devient vraiment lourd à gérer...

[Liens BD : notions avancées]

- 3 types de liens :
 - Connected user
 - Mode de base : même utilisateur en local et distant
 - Ce n'est pas nécessairement le créateur du lien
 - Current user
 - Si lien utilisé à partir de procédure, identification en temps que propriétaire de la procédure
 - Intérêt ?
 - Fixed user
 - Utilisateur fixe.
 - Intérêt ?

[Exemples]

- `CREATE DATABASE LINK ventes.ulp.fr
USING 'ventes';`
- `CREATE DATABASE LINK ventes.ulp.fr
CONNECT TO scott IDENTIFIED BY tiger
USING 'ventes';`
- `CREATE DATABASE LINK ventes.ulp.fr
CONNECT TO CURRENT_USER
USING 'ventes';`
- Remarque : USING -> nom du service!

[Liens BD : notions avancées]

- Visibilité :
 - Public
 - Pratique si utilisé par un grand nombre d'utilisateurs
 - Private (défaut)
 - + sécurisé, accessible uniquement au propriétaire et à ses sous-programmes
 - Global
 - Accessible de n'importe quelle base, mais suppose une gestion centrale (directory server)

[exemples]

- `CREATE PUBLIC DATABASE LINK ventes.ulp.fr USING 'ventes';`
- `CREATE PUBLIC DATABASE LINK ventes.ulp.fr CONNECT TO scott IDENTIFIED BY tiger USING 'ventes';`
- `CREATE PUBLIC DATABASE LINK ventes.ulp.fr CONNECT TO CURRENT_USER USING 'ventes';`
- Différence avec privé ?

[Opérations sur liens]

- Fermeture

- Utile si forte charge
- ALTER SESSION CLOSE DATABASE LINK lien;

- Suppression

- DROP DATABASE LINK lien

[Infos utiles...]

■ Tables

- DBA_DB_LINKS : tous les liens
- ALL_DB_LINKS : tous ceux accessibles par l'utilisateur
- USER_DB_LINKS : tous ceux qui lui appartiennent
- V\$DBLINK : tous les liens ouverts par la transaction

■ Paramétrage

- OPEN_LINKS : nombre de liens simultanés dans une même session (défaut 4, 0 = pas de limite)
- (INIT.ORA)

[Liens partagés]

- Un lien peut être partagé par plusieurs utilisateurs
- Mot clé « shared »
- **CREATE SHARED DATABASE LINK nomlien**
[CONNECT TO util IDENTIFIED BY passwd]|
[CONNECT TO CURRENT_USER]
AUTHENTICATED BY schéma IDENTIFIED BY passwd
[USING 'service']
- Authentification pour le lien, connexion au schéma distant suivant mode de connexion choisi
- Intérêt : si beaucoup de connexions, sinon surcoût (pooling)
- Peut être connecté vers serveur à processus dédiés ou partagés

[Notion de nommage global]

- Oracle peut utiliser le nommage global
 - Chaque base est identifiée par un nom et un domaine. Le nom complet doit être unique
 - Utile avec les liens, indispensable pour la réplication
- Paramétrage
 - GLOBAL_NAMES (INIT.ORA)

[Manipuler les noms globaux]

- `SELECT NAME,VALUE FROM V$PARAMETER WHERE NAME=« db_domain »;`
 - Modifier : dans INIT.ORA
- `SELECT * FROM GLOBAL_NAME;`
- `ALTER DATABASE RENAME GLOBAL_NAME TO nom.domain;`

[Bases de données répliquées]

- Ensemble de bases de données identiques, dont une appelée *copie maître* permet de créer les autres appelées *copies esclaves*.
- Techniques de mise en cohérence
 - Synchrones: la mise à jour des copies est faite dans la même transaction
 - Asynchrones: la mise à jour des copies est faite le plus tôt possible

Bases de données répliquées

■ Alimentation d'entrepôt de données



■ Dissémination de données



■ Consolidation de données



Bases de données répliquées

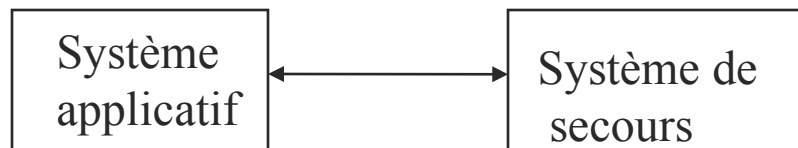
Découpage d'un processus par activité



Accès délocalisé



Systèmes 24h/24



Bases de données répliquées

Réplication sans conflits

En évitant les mises à jour multiples (réplication asymétrique)

- **Système maître unique**
 - Alimentation des entrepôts de données
 - Dissémination d'information
 - Consolidation d'information
- **Système maître désigné en dynamique**
 - Découpage d'un processus par activité

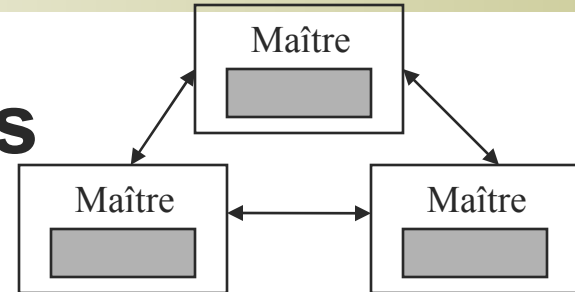
Réplication avec résolution des conflits

Une règle de priorité permet de résoudre les conflits (r. symétrique)

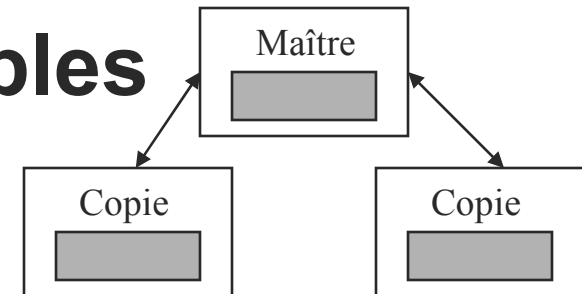
- **Systèmes maîtres multiples**
 - Accès délocalisé
 - Système 24h/24

Bases de données répliquées

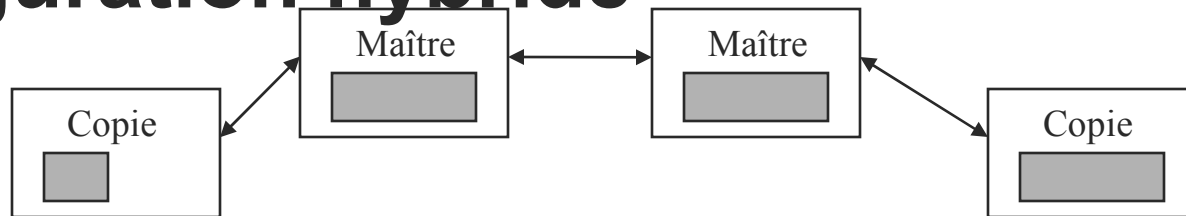
- **Copies maîtres multiples**



- **Copies esclaves modifiables**



- **Configuration hybride**



[Réplication]

- Réplication simple (maître-esclave, read only)
 - Snapshots
 - MAJ : complète ou incrémentale
 - Snapshots complexes : plusieurs tables, pas de MAJ
- Réplication avancée (multi maîtres, rw)
 - multimaître, hybride
 - réplication objet ou groupe
 - site et catalogue de réplication

Conflits d'accès

- Unicité, modif, suppression
 - Des méthodes dans les deux premiers cas
 - pour suppr, suggère l'utilisation d'un flag « suppr »
- Possession statique : une donnée \in à un seul serveur
- Possession dynamique : les serveurs demandent l'exclusivité
- Possession partagée : les conflits éventuels sont gérés de manière asynchrone
- Gestion synchrone : synchro immédiate

[Création d'une vue matérialisée]

- CREATE MATERIALIZED VIEW nommv
[TABLESPACE ... STORAGE ...]
[REFRESH FAST|COMPLETE|FORCE
START WITH sysdate
NEXT sysdate+1 // *en jours...*
WITH PRIMARY KEY // *si possible ...*
USING ROLLBACK SEGMENT ...]
AS
SELECT...

[Vue simple]

- Rafraîchissement rapide possible
- Repose sur clé
 - WITH PRIMARY KEY
- Repose sur ROW ID
 - WITH ROW ID
- Repose sur une requête adaptée
 - sous requête de type exists
 - union

[Des vues « simples » ...]

- ```
CREATE MATERIALIZED VIEW
info_produit REFRESH FAST AS
SELECT * FROM info_produit ip
WHERE id =10 AND EXISTS
 (SELECT * FROM descr_produit dp
 WHERE...)
UNION
...
...
```
- Cela reste rafraîchissable en fast !

# [ Des vues plus complexes... ]

- Pas fast refresh si contient :
  - DISTINCT, UNIQUE
  - INTERSECT, MINUS, UNION ALL
  - CONNECT BY
  - Jointure
  - Agrégation
  - UNION si pas les mêmes types (ou requête complexe)
- Pour tester rafraîchissement rapide
  - DBMS\_VIEW.EXPLAIN\_MVIEW('vue');

# [ Compléments sur les V.M. ]


- pour que le REFRESH fonctionne il faut paramétrer Oracle en conséquence...
  - Noms globaux obligatoires
  - Utilisation d'un lien en fixed user
- Pour autoriser le fast refresh il faut créer un snapshot log sur la table source
  - CREATE SNAPSHOT LOG ON Table  
WITH PRIMARY KEY  
TABLESPACE ... STORAGE (...)

Refresh gérable par le biais de fonctions PL/SQL

- DBMS\_REFRESH.REFRESH('vue');

# [ Groupe de rafraîchissement ]

- DBMS\_REFRESH.MAKE(  
name => 'nomgrp',  
list => "",  
next\_date => SYSDATE,  
interval => 'SYSDATE+... ',  
implicit\_destroy => FALSE,  
rollback\_seg => "",  
push\_deferred => TRUE,  
refresh\_after\_errors => FALSE);

- 
- DBMS\_REFRESH.ADD(  
name=>'nomgrp',  
list=>'nomvue',  
lax=TRUE);

# [ Vues matérialisées modifiables ]

- Intérêt ?
- Ajouter « FOR UPDATE » avant le « AS SELECT... »
- Rattacher à un groupe de vues matérialisées

# [ Groupe de vues matérialisées ]

- Simplification de l'administration
- D'autres avantages...
- Création du groupe :
  - DBMS\_REPCAT.CREATE\_MVIEW\_REPGROUP(  
gname=>'nomgrp',  
master=>'base d'origine',  
propagation\_mode=>'ASYNCHRONOUS');
- Ajout dans le groupe
  - DBMS\_REPCAT.CREATE\_MVIEW\_REPOBJECT(  
gname=>'nomgrp',  
sname=>'schema',  
oname=>'nommv',  
type=>'SNAPSHOT',  
min\_communication=>'TRUE');

# [ Privilèges ]

- ALTER ANY SNAPSHOT
- CREATE ANY SNAPSHOT
- DROP ANY SNAPSHOT
  - Plutôt pour le DBA...
- CREATE DATABASE LINK
  - Utile car permet d'utiliser les liens privés
- CREATE SNAPSHOT
  - permet ajout, modif et suppr
- CREATE VIEW
  - Permet gestion de vue sous-jacente au snapshot



# [ Vues multi tiers ]

---

- On peut avoir des vues matérialisées qui reposent elles mêmes sur des vues matérialisées
- Contraintes : essentiellement utilisation des clés primaires.

# [ Systèmes multi maîtres ]

- Intérêt :
  - Équilibrage de charge
  - Résistance aux pannes
  - Interopérabilité entre applications
- Cohérence ?
  - Synchrone : MAJ immédiate
  - Asynchrone : files d'attente (et d'erreur...)

# [ Mise en oeuvre ]

- Rôles

- Administrateur

- Ajout de sites, mise en sommeil, etc.

- Propagateur

- Envoi des requêtes

- Récepteur

- Mise en œuvre des requêtes reçues

# [ Structures ]

- Liens planifiés
  - Liens avec utilisateur fixe + planification des envois (transactions)
- Purge
- Groupe maître
  - Contient les objets répliqués
  - Il peut y en avoir plusieurs...
  - Notion de site de définition
- Site maîtres
  - Ajoutés sur le site de définition

# [ Résolution de conflits ]

- Méthodes proposées par Oracle:
  - Plus récente modif (timestamp)
  - Réécriture
  - Max et min
  - Moyenne et somme
  - Estampillage
  - Priorité de groupe
  - Priorité de site

# [ Résolution de conflit ]

- Plus récente modif
  - La plus récente est la bonne
  - (pas de lien entre valeurs)
  - Suppose datage uniforme
- Réécriture
  - Essentiellement pour mono maître

# [ Résolution de conflit ]

- Additive
  - On ajoute la différence
  - OK pour transactions financières par ex.
- Moyenne
  - OK pour donnée évoluant vers moyenne
- Abandon
  - OK avec mono maître

# [ Résolution de conflit ]

- Plus ancienne date
  - Utilisation limitée... (mono maître)
- Maximum, Minimum
  - Pour des données s'y prêtant
- Groupes prioritaires
  - Priorité suivant la valeur d'une colonne donnée
- Site prioritaire



# [Autres types de conflits]

- Insertion
  - Séquences
- Suppression
  - Colonne de suppression

# [ Éviter les conflits ]

---

- Groupes de colonnes
  - On peut modifier sur deux groupes indépendants sans générer de conflit
- Site propriétaire
  - Granularité réglable
- Possession dynamique

# [ Compléments sur la réplication ]

- Template de réplication
  - Pour simplifier réplication massive
  - Ex : copies sur machines nomades
  - Description des tables distribuées
- Réplication d'autres informations
  - Triggers
  - Types d'Objets
  - ...