Base de données répartie (BDR)

Une BDR est un ensemble de bases de données gérées par des sites différents et apparaissant à l'utilisateur comme une base unique.

SGBD réparti (SGBDR)

Un SGBDR est un système qui gère une collection de BDs logiquement reliées, distribuées sur un réseau, en fournissant un mécanisme d'accès qui rend la répartition transparente aux utilisateurs.

Avantages : - Prendre en compte la répartition géographique des données. - Prendre en compte la distribution fonctionnelle des données. - Une meilleure disponibilité des données en présence de panne ou de dysfonctionnement des applications. - Une plus grande flexibilité afin d'assurer le partage des données hétérogènes et réparties. - Meilleures performances (données réparties sur plusieurs bases gérées par des serveurs différents mieux adaptés).

Inconvénients : - Complexité des SGBDRs. - Problèmes de cohérence dus à la distribution du contrôle de données entre plusieurs sites. - Difficulté de changement (ex. intégration d'un nouveau type de SGBD).

Objectifs des SGBDR

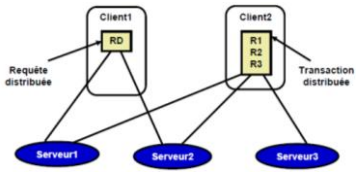
1) Multiclients multiserveurs

- Fournir un mécanisme de contrôle des accès concurrents adapté. - Garantir que l'effet de l'exécution simultanée des transactions est le même que celui d'une exécution séquentielle (sérialisabilité des transactions).

- Permettre l'exécution des requêtes distribuées.

-> Une requête distribuée est une requête émise par un client dont l'exécution nécessite l'exécution de N sous-requêtes sur N serveurs avec N>1 .

-> Une transaction distribuée est une transaction qui met en oeuvre plusieurs serveurs.



2) Transparence à la localisation des données

- Propriété d'un SGBDR permettant d'écrire des requêtes avec des noms d'objets ne contenant pas la localisation des données.

Avantages : - Simplifier la vue utilisateur et l'écriture de ses requêtes. - Introduire la possibilité de déplacer les objets sans modifier les requêtes. Inconvénient : - Contraindre le SGBDR à rechercher les sites capables de générer des éléments de réponse à une requête pour l'exécuter (fonction pas évidente). Solution : Utilisation d'un nom hiérarchique pour les objets : <objet>@<base>. -Utilisation d'un alias pour retrouver l'indépendance à la localisation.

3) Meilleure disponibilité : - Disponibilité des données : une des justifications essentielles des SGBDR. - La répartition permet de ne plus dépendre d'un site central. - Gestion des copies : se replier sur une copie lorsqu'une autre est indisponible (site en panne) ==> Réplication. - Assurer une meilleure disponibilité, c'est aussi garantir l'Atomicité des transactions.

4) Autonomie locale : - Garder une administration locale séparée et indépendante pour chaque serveur participant à la base de données répartie (pas d'administration globale) ==> les reprises après panne et les mises à niveau des logiciels doivent être accomplies localement et ne pas impacter les autres sites.

5) Hétérogénéité : - Capacité d'unifier des modèles et langages afin de générer des bases fédérées. - Intégration sémantique des bases.

Architecture des SGBDR

Fonctions d'un SGBDR => Traitement des requêtes référençant des objets d'une BDR.

- Assurer la décomposition de la requête distribuée en sous-requêtes locales envoyées à chaque site. - Prendre en compte les règles de localisation lors de la décomposition. - Evaluer globalement la requête distribuée en minimisant le transfert de données et en maximisant le parallélisme. - Traduire les requêtes exprimées dans un langage Pivot (ex. SQL) en requêtes compréhensibles par le SGBD local (dans un contexte de BD hétérogènes). - Router les mises à jour vers les sites concernés en assurant la gestion des transactions réparties (vérification des règles d'intégrité, contrôle des accès concurrents, gestion de l'atomicité des transactions distribuées).

Organisation des schémas : Une BDR est décrite par différents niveaux de schémas

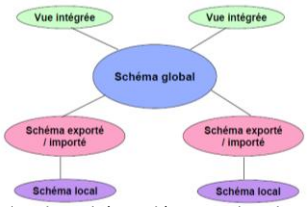
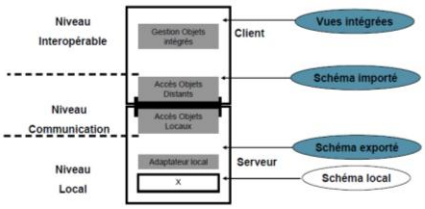


Schéma local : Schéma décrivant les données d'une BD locale gérée par le SGBD local. // Schéma exporté : Schéma décrivant les données exportées par un site vers les sites clients. // Schéma importé : Schéma exporté reçu par un site client. // Schéma global : L'ensemble des schémas exportés par tous ou certains sites clients exprimé dans le modèle de référence du SGBDR et décrivant globalement la base répartie (le schéma global est non matérialisée dans les sites clients). // Vue intégrée : Schéma décrivant dans le modèle du SGBDR les données de la BDR accédées par une application.

Architecture de référence

Architecture s'articulant autour de 3 niveaux de fonctionnalités :

- Niveau local : présent sur chaque serveur permet d'exporter les données locales selon le modèle pivot du SGBDR. - Niveau communication : permet de transmettre les sous-requêtes en provenance d'un site client au serveur dans le langage pivot d'échange. Ces sous-requêtes référencent le schéma exporté vers ce site client. - Niveau interopérabilité : permet de formuler des requêtes mettant en jeu des vues intégrées de la base. Il assure la décomposition des requêtes en sous-requêtes et le passage des vues intégrées aux différents schémas importés.



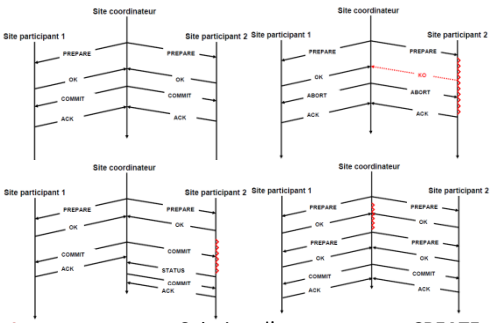
Conception des BDR

Conception descendante : - Utilisée lors de la constitution de nouvelles bases de données. - Un schéma conceptuel est tout d'abord élaboré puis les diverses entités de ce schéma sont distribuées sur les sites ==> définition des schémas locaux.

Conception ascendante : - Intégration des BD locales existantes dans une base fédérée ==> intégration des schémas locaux existants afin de constituer un schéma global.

Les 12 règles d'un SGBD : Autonomie locale / Pas de site fédérateur / Exploitation en continue / Indépendance à la localisation / Règles de fragmentation / Duplication multiple / Requêtes distribuées / Transactions distribuées / Indépendance du matériel, des systèmes d'exploitation et du réseau / Indépendance du SGBD.

Transactions distribuées : Propriétés des transactions : - Atomicité : Une transaction doit effectuer toutes ses mises à jour ou ne rien faire du tout. // - Cohérence : Une transaction doit faire passer la BD d'un état cohérent à un autre. // - Isolation : Les résultats d'une transaction ne doivent être visibles aux autres transactions qu'une fois la transaction validée. // - Durabilité : Dès qu'une transaction valide ses modifications, le système doit garantir que ces modifications seront conservées en cas de panne. ------ Dans le cadre d'un système réparti, chaque site peut décider de valider ou d'annuler une transaction ==> il faut donc coordonner les validations.



Les synonymes : - Création d'un synonyme : **CREATE SYNONYM Client FOR Client@lien_base2; //** - Suppression d'un synonyme : **DROP SYNONYM Client; //** - Synonyme d'une séquence distante : **CREATE SYNONYM Sequence_Client FOR Sequence_Client@lien_base2;**

Distribution : - Bases de données distribuées ou réparties. -Sans redondance.

Duplication : - Duplication locale d'un fragment éloigné maître. - Fragment locale en lecture seule. - Utilisation de la notion de cliché. - Duplication synchrone ou asynchrone. - **Duplication synchrone :** propager immédiatement les modifications apportées aux données sources vers les copies -> Utilisation des Trigger ou Trigger instead of. - **Duplication asynchrone :** propager les modifications apportées aux données sources vers les copies à des intervalles prédéfinis. -> Utilisation d'un programmateur, Oracle utilise la notion de SNAPSHOT ou vues matérialisées.

Réplication : - Pas de fragment maître. - Réplication synchrone : utilisation de jetons. - Réplication asynchrone : problèmes de cohérence. Avantages : - Améliorer les performances (Utilisation de la copie la plus proche du client => évite des transferts inutiles). - Augmenter la disponibilité des données (Lors d'une panne d'un serveur, on peut se replier sur un autre disposant d'une copie des données). Avec N copies sur des serveurs différents => Disponibilité = 1 – probabilité_panne N. Inconvénients : - Assurer la convergence des copies. - Offrir une transparence de gestion aux clients : les clients doivent croire à l'existence d'une seule copie. -> Le SGBD doit assurer la diffusion des mises à jour aux copies et le choix de la meilleure copie lors des accès.

Client (#No, Nom, Prénom, Adresse, Ville)

Agence (#No, Nom, Adresse, Ville)

Compte (#No, Type_Compte_No, DateOuverture, Decouvert_autorise, Solde, Client_No, Agence_No)

Type_Compte (#No, Nom, Description)

Operation (#No, Type_Operation_No, Compte_No)

Type_Operation (#No, Description)

Création du lien inter – base (database link)

CREATE DATABASE LINK dbl_ensias1 CONNECT TO Inwis IDENTIFIED BY azerty USING 'ensias1';

CREATE DATABASE LINK dbl_ensias2 CONNECT TO Inwis IDENTIFIED BY azerty USING 'ensias2';

Test des liens :

CONNECT Inwis/azerty@ensias2; SELECT * FROM Client_1@dbl_ensias1;

CONNECT Inwis/azerty@ensias1; SELECT * FROM Client_2@dbl_ensias2;

Création de la base répartie : Créer les objets virtuels répartis

CONNECT Inwis/azerty@ensias1; CREATE VIEW Client (No, Nom, Prenom, Adresse, Ville) AS SELECT No, Nom, Prenom, Adresse, 'Casablanca' FROM Client_1 UNION SELECT No,Nom,Prenom,Adresse,'Rabat' FROM Client_2@dbl_ensias2;

CONNECT Inwis/azerty@ensias2; CREATE VIEW Client (No, Nom, Prenom, Adresse, Ville) AS SELECT No, Nom, Prenom, Adresse, 'Casablanca' FROM Client_1@dbl_ensias1 UNION SELECT No,Nom,Prenom,Adresse,'Rabat' FROM Client_2;

Fragmentation horizontale pour la table Client

Serveur1 : Client_1 des clients de Casablanca sans 'Ville'.

CREATE TABLE Client_1 (No, Nom, Prénom, Adresse CONSTRAINT pk1 PRIMARY KEY(No)) AS SELECT No,Nom, Prénom, Adresse FROM Client WHERE ville = 'Casablanca';

Serveur2 : Client_2 des clients de Rabat sans 'Ville'.

COPY FROM Inwis/azerty@ensias1 TO Inwis/azerty@ensias2 REPLACE Client_2 (No,Nom, Prénom, Adresse) USING SELECT No,Nom, Prénom, Adresse FROM Client WHERE ville = 'Rabat';

Fragmentation horizontale pour la table Compte :

Serveur1 : Compte_1 des comptes des clients de Casablanca.

CREATE TABLE Compte_1 (No, Type_Compte_No, DateOuverture, Decouvert_autorise, Solde, Client_No, Agence_No, CONSTRAINT pk2 PRIMARY KEY(No)) AS SELECT No, Type_Compte_No, DateOuverture, Decouvert_autorise, Solde, Client_No, Agence_No FROM Compte WHERE Client_No IN (SELECT No FROM Client_1);

Serveur2 : Compte_2 des comptes des clients de Rabat.

COPY FROM Inwis/azerty@ensias1 TO Inwis/azerty@ensias2 REPLACE TABLE Compte_2 (No, Type_Compte_No, DateOuverture, Decouvert_autorise, Solde, Client_No, Agence_No) USING SELECT No, Type_Compte_No, DateOuverture, Decouvert_autorise, Solde, Client_No, Agence_No FROM Compte WHERE Client_No IN (SELECT No FROM Client WHERE ville = 'Rabat');

Fragmentation horizontale pour la table Operation :

Serveur1 : Operation_1 des opérations des comptes de Compte_1. CREATE TABLE Operation_1 (No, Type_Operation_No, Compte_No, CONSTRAINT pk3 PRIMARY KEY(No)) AS SELECT No, Type_Operation_No, Compte_No FROM Operation WHERE Compte_No IN (SELECT Compte_No FROM Compte_1);

Serveur2 : Operation_2 des opérations des comptes de Compte_2. COPY FROM Inwis/azerty@ensias1 TO Inwis/azerty@ensias2 REPLACE Operation_2 (No, Type_Operation_No, Compte_No) USING SELECT No, Type_Operation_No, Compte_No FROM Operation Where Compte_No IN (SELECT Compte_No FROM Compte WHERE Client_No IN (SELECT No FROM Client WHERE ville = 'Rabat')); (ou bien par jointures).

Déplacement complet de la table Type_Compte sur Serveur2 : COPY FROM Inwis/azerty@ensias1 TO Inwis/azerty@ensias2 REPLACE Type_Compte_2(no, libelle_compte, description) USING SELECT no, libelle_compte, description FROM Type_Compte; DROP TABLE Type_Compte;

Déplacement complet de la table Type_Operation sur Serveur2 : COPY FROM Inwis/azerty@ensias1 TO Inwis/azerty@ensias2 REPLACE Type_Operation_2(no, libelle_operation, decription) USING SELECT no, libelle_operation, description from Type_Operation; DROP TABLE Type_Operation;

Ajout des contraintes de base

Les contraintes de clé primaire :

ALTER TABLE Client_2 ADD PRIMARY KEY (No);

ALTER TABLE Agence_2 ADD PRIMARY KEY (No);

ALTER TABLE Compte_2 ADD PRIMARY KEY (No);

ALTER TABLE Type_Compte_2 ADD PRIMARY KEY (No);

ALTER TABLE Operation_2 ADD PRIMARY KEY (No);

ALTER TABLE Type_Operation_2 ADD PRIMARY KEY (No);

Les contraintes de références si la table mère est sur le même site : (+ Compte_1 (2 FK) + Operation_1 (1 FK))

ALTER TABLE Compte_2 ADD CONSTRAINT fk1 FOREIGN KEY(Client_No) REFERENCES Client_2(No);

ALTER TABLE Operation_2 ADD CONSTRAINT fk2 FOREIGN KEY(Compte_No) REFERENCES Compte_2(No);

Les contraintes de références par trigger si la table 'mère' est sur un site distant => Deux triggers :

- un trigger sur la fille remplaçant la FOREIGN KEY,

- un trigger sur la mère interdisant de supprimer une ligne référencée.

CONNECT Inwis/azerty@ensias1;

CREATE OR REPLACE TRIGGER trig_agence_mother

BEFORE DELETE OR UPDATE OF No ON Agence

FOR EACH ROW

DECLARE x number := 0;

BEGIN

SELECT count(*) INTO x from Compte_2@dbl_ensias2

WHERE Agence_No = :OLD.No;

IF x>0 THEN

RAISE_APPLICATION_ERROR(-20175,'Cette agence est utilisée.');

END IF; END; /

CONNECT Inwis/azerty@ensias2;

CREATE OR REPLACE TRIGGER trig_agence_daughter

BEFORE INSERT OR UPDATE OF Agence_No ON Compte_2

FOR EACH ROW

DECLARE x number := 0;

BEGIN

SELECT count(*) INTO x FROM Agence@dbl_ensias1

WHERE No = :NEW.Agence_No;

IF x=0 THEN

RAISE_APPLICATION_ERROR(-20175,'Cette agence est inexistante.');

END IF: END: /

Mise en œuvre de la réplication synchrone

CONNECT Inwis/azerty@ensias1; CREATE TABLE Appareil(no_appareil number(7) PRIMARY KEY, designation varchar(30), prix number(7,2), caracteristiques_techniques varchar(50));

Copier la table centrale APPAREIL dans le Serveur 2.

CONNECT Inwis/azerty@ensias2; CREATE TABLE Appareil_Copy AS SELECT * FROM Appareil@dbl_ensias1;

Ecrire un trigger sur la base du siège (Serveur1) qui permet d'assurer que toute modification au niveau de la table centrale APPAREIL soit répercutée immédiatement vers l'image de cette table à Rabat.

CREATE TRIGGER app

AFTER INSERT OR DELETE OR UPDATE ON Appareil

FOR EACH ROW

BEGIN

IF INSERTING THEN

INSERT INTO Appareil_Copy@dbl_ensias2 VALUES(:NEW.no_appareil, :NEW.designation, :NEW.prix, :NEW.caracteristiques_techniques);

ELSE IF DELETING THEN

DELETE FROM Appareil_Copy@dbl_ensias2 WHERE no_appareil = :OLD.no_appareil;

ELSE IF UPDATING THEN

UPDATE Appareil_Copy @dbl_ensias2

SET no_appareil = :NEW.no_appareil, designation = :NEW.designation, prix = :NEW.prix, caracteristiques_techniques = :NEW.caracteristiques_techniques WHERE no_appareil = :OLD.no_appareil;

END IF;

END;

/

Mise en œuvre de la réplication asynchrone

Créer une image (cliché) de la table centrale APPAREIL dans chacun des autres sites. Le rafraîchissement doit être rapide et sa mise à jour doit être effectuée toutes les 30 jours.

CREATE SNAPSHOT LOG ON Appareil;

CREATE SNAPSHOT image_appareil REFRESH FAST

START WITH SYSDATE NEXT SYSDATE + 30

AS SELECT * FROM Appareil@dbl_ensias1;

Syntaxe générale :

CREATE SNAPSHOT [nom_schéma.]Nom_image [Spécification de stockage] [REFRESH [FAST|COMPLETE|FORCE][START WITH date1] [NEXT date2] AS Requête;