Coté Labo - Réplication avec MySQL 5.6 – Partie 1

Description du thème

Le thème comporte deux parties :

- La première partie vise à assurer la réplication d'une base de données MySQL entre deux serveurs, maître et esclave.
- La deuxième partie consiste à préparer une évolution de l'architecture vers plusieurs serveurs esclaves et "scripter" le paramétrage de ces serveurs.

Propriétés	Description
Intitulé long	Mise en place d'une architecture répliquée avec MySQL Server
Formation concernée	BTS Services Informatiques aux Organisations
Matière	Intégration et adaptation d'un service
Présentation	Cette première partie propose d'installer et de configurer un serveur MySQL, de créer une base de données sur ce serveur, puis d'installer et de configurer une réplication de cette base sur un serveur esclave.
Notions	**Domaines de compétences** D1.1 - Analyse de la demande • A1.1.2 Étude de l'impact de l'intégration d'un service sur le système informatique D1.3 – Mise en production d'un service • A1.3.1 Test d'intégration et d'acceptation d'un service • A1.3.4 Déploiement d'un service **Savoir-faire** • Sauvegarder et restaurer une base de données • Automatiser l'installation d'un service • Valider et documenter la mise en exploitation d'un service **Savoirs associés** • Stratégies et techniques de sauvegarde et de restauration de données • Stratégies et techniques de répartition et de réplication
Prérequis	Langage SQL, bases de Linux, bases des SGDR (et si possible de MySQL), les savoirs associés ont déjà été abordés pour tous les étudiants.
Transversalité	SLAM3 - Conception et adaptation d'une base de données
Outils	Deux STA (éventuellement virtualisées). Dans l'exemple décrit par la suite, une machine Ubuntu Server 12.04 et une machine serveur Windows Server 2012 R2.
Mots-clés	Bases de données, sauvegarde, restauration, réplication, MySQL
Durée	2h (TP SI7)
Version	v 1.0
Date de publication	Mars 2016

Contexte

La **DICCC** (Direction Inter-provinciale du Commerce de la Concurrence et de la Compétitivité) est une collectivité territoriale chargée de réaliser des enquêtes sur les échanges commerciaux à l'intérieur de la Nouvelle-Calédonie.

L'organisation possède un parc de véhicules, destinés à ses agents, pour un usage professionnel.
L'attribution de ces véhicules est décidée par la cellule logistique, qui est basée à Nouméa.
Afin d'améliorer la gestion de son parc automobile, la cellule logistique de la DICCC a mis en place une base de données comportant 4 tables et contenant toutes les informations qui concernent l'attribution des véhicules aux agents. Cette attribution est effectuée par le responsable de la cellule logistique ; c'est lui qui est également responsable de la mise à jour des données de la base.
Le modèle relationnel est fourni en **annexe 1** et la description des tables est détaillée en **annexe 2**. Le responsable de la cellule logistique a besoin de cette base de données pour savoir à quel agent est attribué chaque véhicule, quelles réparations ont été effectuées sur les véhicules, etc. Il garde un historique des attributions des véhicules et un objectif est, à terme, de pouvoir réaliser des statistiques sur ceux-ci (coût des réparations, surveillance du kilométrage des véhicules, etc.).

La DICCC est implantée sur tout le territoire de Nouvelle-Calédonie ; elle dispose d'un siège à Nouméa, des agences provinciales de Koné et de Wé et des sites annexes du Mont-Dore, de La Foa, de Bourail, de Poindimié et de Tadine.
L'accès aux données concernant son parc de véhicules est possible depuis n'importe quelles agences principales ou annexes de l'organisation.

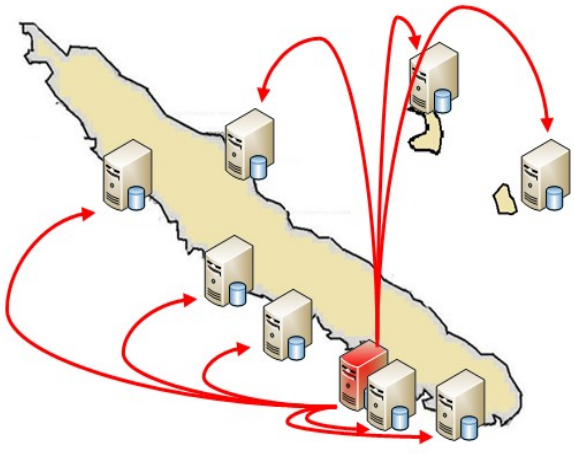
Au siège, à Nouméa, une application client/serveur développée en python permet d'accéder à cette base de données implémentée sur un serveur Ubuntu 12.04 en local hébergeant MySQL 5.6. Cette application constitue une interface permettant de visualiser et gérer le parc de véhicules attribué aux agents.

Dans chaque agence principale ou annexe, une autre application client/serveur beaucoup plus limitée que celle du siège permet uniquement de visualiser le parc et d"éditer un certain nombre de statistiques. Le système de gestion de base de données Mysql 5.6 est implémenté en local sur un serveur Windows 2012 R2. Un système de sauvegarde de la base du siège et de restauration quotidienne permet d'alimenter chaque nuit la base de données des agences.

Une équipe de développeur travaille actuellement sur l'enrichissement de cette base et des applications associées. Une exploitation optimum de celle destinée aux agences nécessitera de bénéficier pour celles-ci de données fiables en temps réel.

Pour notamment des raisons de sécurité, il a été décidé de ne pas permettre à l'application déployée en agence d'accéder à la base de données du siège. Aussi, la DICCC souhaiterait tester la mise en place d'une architecture répliquée de cette base de données vers chaque agence du territoire calédonien hébergera la base sur une machine Ubuntu Server 12.04 et l'ensemble des données sera répliqué sur un serveur .

L'architecture prévue peut être représentée par le schéma ci-dessous :



Dans un premier temps, il a été décidée de mettre en place une maquette de cette architecture répliquée pour vérifier son bon fonctionnement uniquement entre le serveur du siège et une agence (partie 1 du Côté Labo), un script SQL de création de la base de données de test est fourni. Puis, une fois cette architecture validée, il s'agira de préparer une évolution permettant la réplication vers les différents serveurs des agences (partie 2 du Côté Labo).

Nous mettrons de côté toutes les problématiques liées aux accès distants, votre maquette ne comportera que des postes situés dans le même réseau logique.

Préambule

Expliquer dans une note succincte en quoi consiste la réplication de bases de données. Cette note doit notamment préciser les avantages attendus pour la DICCC.

Phase 1 – Installation et configuration du serveur maître

1.1 - Installer le serveur MySQL sur l'ordinateur destiné à héberger le serveur maître.

1.2 - Configurer le serveur MySQL afin qu'il soit accessible à l'administrateur « root » de MySQL depuis une machine distante.
Vous trouverez ici une documentation détaillée qui vous sera utile : http://doc.ubuntu-fr.org/mysql

1.3 - Tester l'accès au serveur MySQL depuis une machine distante.

1.4 - Paramétrer le serveur afin qu'il soit serveur « maître »
Vous trouverez ici des documentations détaillées qui vous seront utiles :
http://dev.mysql.com/doc/refman/5.6/en/replication.html
http://dev.mysql.com/doc/refman/5.6/en/replication-howto-masterbaseconfig.html
http://dev.mysql.com/doc/refman/5.6/en/replication-howto-repuser.html

Phase 2 – Restauration des données sur le serveur maître

2.1 - À l'aide du script SQL qui accompagne cet énoncé, restaurer la base de données *BDD_vehicules* sur le serveur maître MySQL.

2.2 - Tester la présence des données restaurées.

2.3 – Modifier la configuration du serveur afin que seule cette base de données puisse être répliquée.
Vous trouverez ici une documentation détaillée qui vous sera utile :
http://dev.mysql.com/doc/refman/5.6/en/replication-rules.html

Phase 3 - Installation et configuration du serveur esclave

3.1- Tester la connexion TCP/IP entre les deux machines.

3.2 - Installer le serveur MySQL sur l'ordinateur destiné à être le serveur esclave.

3.3 - À l'aide du script SQL qui accompagne cet énoncé, restaurer la base de données *BDD_vehicules* sur ce nouveau serveur MySQL. Ensuite, paramétrer ce serveur en tant que serveur esclave.

3.4 Paramétrer le serveur afin qu'il soit serveur esclave.

Phase 4 – Test de la réplication

4.1 - Tester le bon fonctionnement de la réplication.

Annexe 1 – Modèle relationnel

t_vehicules (NumV, MarqueV, ModeleV, TypeV, PrixV, AnneeV, EtatV)

Clé primaire : *NumV*

t_agents (MatA, NomA, PrenomA, PB, PC)

Clé primaire : *MatA*

t_reparations (NumR, DateR, NumV, DescR, KmR, PrixR)

Clé primaire : *NumR*

Clé étrangère : *NumV* en référence à *NumV* de la relation *t_vehicules*

t_attribuer (NumAtt, MatA, NumV, DateAtt)

Clé primaire : *NumAtt*

Clés étrangères : *MatA* en référence à *MatA* de la relation *t_agents*

NumV en référence à *NumV* de la relation *t_vehicules*

Annexe 2 – Description des tables

t_vehicules	
NumV	*Chaine de caractères - Numéro minéralogique du véhicule. Clé primaire de la table.*
MarqueV	*Chaine de caractères - Marque du véhicule.*
ModeleV	*Chaine de caractères - Nom du modèle du véhicule.*
TypeV	*Chaine de caractères - Type de véhicule. Ce champ contient obligatoirement un des 4 mots suivants : « Utilitaire », « 4x4 », « Berline », « Camion ».*
PrixV	*Nombre décimal - Prix d'achat du véhicule.*
AnneeV	*Entier - Année d'acquisition du véhicule. Ne peut pas être inférieure à 1990.*
EtatV	*Chaine de caractères - État du véhicule. Ce champ contient généralement les mots « A réviser », « OK », « H.S. » ou « Réforme ».*
t_agents	
MatA	*Chaine de caractères - Matricule de l'agent. Clé primaire de la table.*
NomA	*Chaine de caractères - Nom de l'agent*
PrenomA	*Chaine de caractères - Prénom de l'agent*
PB	*Booléen - Champ pouvant prendre la valeur 1 ou 0 et indiquant si l'agent possède le permis B.*
PC	*Booléen - Champ pouvant prendre la valeur 1 ou 0 et indiquant si l'agent possède le permis C (poids lourds).*
T_reparations	
NumR	*Entier - Clé primaire de la table.*
DateR	*Date - Date de la réparation.*
NumV	*Chaine de caractères - Numéro du véhicule sur lequel a eu lieu la réparation.*
DescR	*Texte - Description de la réparation.*
KmR	*Entier - Kilométrage relevé sur le véhicule lors de la réparation.*
PrixR	*Nombre décimal - Prix de la réparation.*
T_attribuer	
NumAtt	*Entier - Clé primaire de la table.*
MatA	*Chaine de caractères - Matricule de l'agent à qui est attribué le véhicule.*
NumV	*Chaine de caractères - Numéro du véhicule attribué.*

DateAtt	*Date - Date d'attribution du véhicule à l'agent concerné.*