Alexis TOURRENC--LECERF

Louis-Alexandre LAGUET

Cédric DOLLET

ING 2 - GIA

Rapport de projet

GLPI (Gestion du parc informatique de CY Tech)

Enseignant : Redouane Bouhamoum

Matière : Traitement et Administration des Données

Sommaire

Présentation du groupe et du projet

Notre équipe, composée d'Alexis Tourrenc-Lecerf, Louis-Alexandre Laguet et Cédric DOLLET, s'est engagée dans un mini-projet d'optimisation du système de gestion du parc informatique de CY Tech, en se concentrant sur l'utilisation de GLPI. Nous avons entrepris une révision en profondeur de la base de données GLPI pour simplifier et améliorer sa performance, tout en prenant en compte les spécificités multi-sites de l'entreprise, avec des sites à Cergy et à Pau.
Chacun de nous a joué un rôle spécifique et complémentaire dans ce processus.

Alexis et Louis ont dirigé la phase de modélisation de la base de données, en travaillant méticuleusement pour concevoir une structure cohérente qui répond aux besoins spécifiques de CY Tech, tout en tenant compte des aspects multi-sites de l'entreprise.

Alexis et Cédric ont pris en charge le développement des triggers et des séquences, en veillant à ce que chaque élément contribue à l'optimisation des performances du système.

Louis et Alexis se sont concentrés sur le développement des index et des clusters pour améliorer les performances de la base de données.

Cédric et Alexis ont collaboré sur le développement des procédures stockées pour garantir une exécution efficace des opérations.

Cédric s'est également chargé de générer des jeux de données conséquents et de tester notre solution.

Enfin, Louis s'est chargé de la création du rapport final, mettant en lumière les différentes optimisations apportées au système.

Modélisation

Fragmentation

Dans le cadre de notre projet GLPI visant à optimiser la gestion du parc informatique de CY TECH, nous avons choisi une approche de fragmentation horizontale de notre base de données. Cette fragmentation consiste à diviser logiquement la base de données en deux ensembles distincts, chaque ensemble étant dédié à un site spécifique de l'école, à savoir Pau et Cergy.

Cette fragmentation horizontale a été réalisée en se basant sur le champ de localisation présent dans nos tables. En effet, le champ de localisation permet d'identifier les sous-sites des sites de Pau et Cergy. Cette fragmentation nous a permis de séparer les données de manière logique et physique, garantissant ainsi une gestion efficace et sécurisée des données propres à chaque site.

Les techniciens travaillant sur un site spécifique n'ont pas besoin d'accéder aux données de l'autre site, ce qui rend la fragmentation horizontale appropriée pour notre système. En fragmentant la base de données en fonction de la localisation, nous avons limité les accès inutiles à des données externes, ce qui contribue à améliorer les performances et la sécurité du système.

De plus, cette fragmentation horizontale offre une meilleure évolutivité et une gestion simplifiée des ressources informatiques sur plusieurs sites. Elle permet également une meilleure isolation des données entre les différents sites, renforçant ainsi la confidentialité et la sécurité des informations.

En adoptant cette approche de fragmentation horizontale, nous avons créé une structure de base de données solide, adaptée aux besoins spécifiques de CY Tech et offrant une gestion efficace des ressources informatiques sur plusieurs sites.

Tables

Comme énoncé plus tôt, notre modélisation repose sur la création de deux bases de données distinctes, chacune dédiée à un site spécifique de l'école, à savoir Pau et Cergy. Chaque base de données est gérée localement par un administrateur désigné (*glpiAdmin_pau* et *glpiAdmin_cergy*), assurant ainsi une gestion efficace et sécurisée des données propres à chaque site.

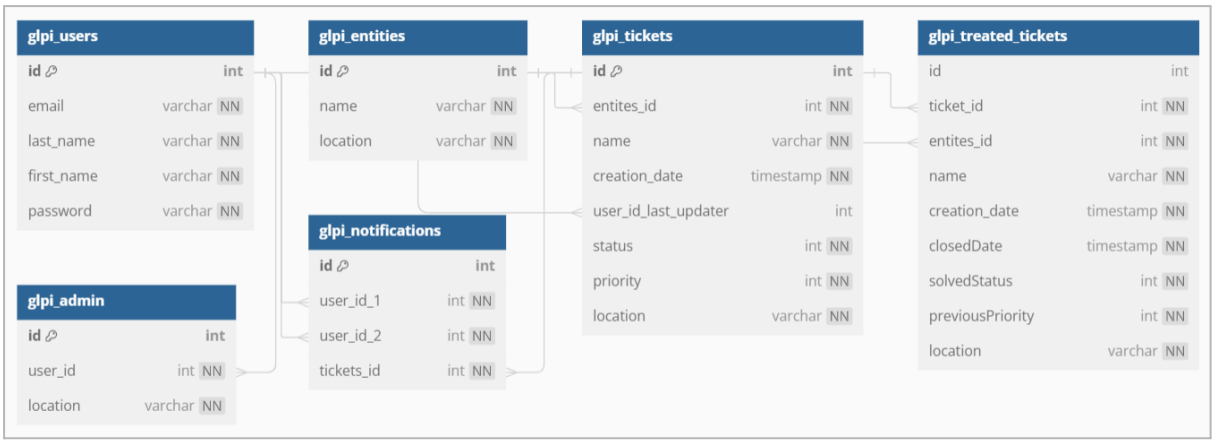

Diagramme de notre solution

La modélisation de la base de données s'appuie sur plusieurs entités clés, notamment les utilisateurs, les entités, les tickets, les tickets traités, les administrateurs et les notifications.

- La table **glpi_users** est conçue pour représenter les utilisateurs du système, avec des champs essentiels tels que l'identifiant, l'adresse e-mail, le nom, le prénom et le mot de passe. Cette table constitue le fondement de l'identification et de l'authentification des utilisateurs au sein du système.

- La table **glpi_entities** enregistre les diverses entités défaillantes liées aux tickets, telles que des équipements informatiques (par exemple, PC, imprimantes, serveurs) ou des logiciels spécifiques (par exemple, système d'exploitation, suite bureautique), offrant ainsi des informations essentielles sur leur localisation. De plus, le champ location permet d'identifier les sous-sites des sites Pau et Cergy.
  Pour Cergy, les sous-sites incluent Condorcet, Cauchy, Turing, Fermat, Saint Martin et Saint Germain.
  Cependant, pour Pau, le champ de localisation peut sembler moins utile car, à notre connaissance, il n'existe qu'un seul site. Néanmoins, maintenir la même modélisation qu'à Cergy permet d'avoir des tables suivant un même schéma. De plus, cela offre la flexibilité à Pau de déclarer plusieurs sous-sites si l'administrateur local le souhaite, prévoyant ainsi une éventuelle expansion ou une meilleure gestion des ressources sur le site.

- Les tables **glpi_tickets** et **glpi_treated_tickets** sont dédiées à la gestion des tickets, couvrant à la fois les tickets en cours et ceux qui ont été traités. Ces tables contiennent des données telles que l'identifiant du ticket, l'entité associée, le nom du ticket, la date de création, l'état, la priorité, l'emplacement et d'autres informations

pertinentes. La conception de ces tables vise à rationaliser le processus de gestion des incidents et des demandes de service au sein de l'organisation.

- La table **glpi_admin** est destinée à stocker les informations sur les administrateurs du système, avec une référence à la table **glpi_users** pour garantir l'intégrité référentielle. Chaque administrateur est associé à un emplacement spécifique, reflétant sa responsabilité dans la gestion du parc informatique local.
- Enfin, la table **glpi_notifications** permet de gérer les notifications liées aux tickets, en enregistrant les utilisateurs impliqués et les tickets associés.

Rôles et utilisateurs

Dans le cadre de notre stratégie de gestion des autorisations et des privilèges des utilisateurs, nous avons mis en place différents rôles pour catégoriser et attribuer des autorisations spécifiques.

Ainsi, les rôles *technician_role* et *simple_user_role* ont été créés pour répondre aux besoins des différentes catégories d'utilisateurs. Le *technician_role* octroie des privilèges étendus, permettant aux techniciens d'accéder à des fonctionnalités avancées telles que la création d'utilisateurs, ce qui n'est pas permis pour les simples utilisateurs qui sont associés au rôle *simple_user_role*.

En parallèle, nous avons établi un administrateur "super admin", *glpiAdmin*, qui détient des droits spéciaux sur l'ensemble du système, facilitant ainsi une gestion globale et centralisée des opérations administratives. Ce super administrateur bénéficie d'autorisations étendues sur les deux bases de données, ce qui lui permet de superviser efficacement l'ensemble des activités.

Cette approche garantit la sécurité et la confidentialité des données tout en offrant une structure de base de données solide, adaptée aux exigences opérationnelles de CY Tech et à la gestion efficace des ressources informatiques sur plusieurs sites.

Utilisation

Vues

Le choix que nous avons fait pour notre modélisation est de passer par des vues pour la sélection, l'insertion, la mise à jour et la suppression des données plutôt que d'utiliser directement les tables. Cela permet de vérifier au préalable la validité des opérations sur les tables au travers de triggers.

De plus, le fait de pouvoir effectuer des sélection sur ces vues permet dans le cadre de Cergy de ne requêter que sur des localisations précises et donc de compartimenter les données et de pouvoir interagir avec la base de donnée de manière précise.
Par exemple, insérer sur la vue Fermat, permet de directement insérer dans la table globale la ligne avec la localisation renseignée à Fermat grâce au trigger *insert_ticket_fermat* du fichier "trigger_cergy.sql".

Trois vues sont accessibles depuis l'administrateur global *glpiAdmin* dans le fichier "view.sql". Elles permettent à ce dernier de récupérer tous les tickets de CY Tech en unissant ceux de Pau et Cergy. La différence entre les deux premières vues se trouve dans le statut des tickets retournés. La première retourne les tickets ouverts, la seconde les tickets fermés. La troisième permet de récupérer tous les utilisateurs de chaque site.

Déclencheurs (Triggers)

Les déclencheurs que nous avons implémentés sont conçus pour générer des identifiants uniques avant l'insertion de nouvelles données dans certaines tables, telles que **glpi_entities**, **glpi_tickets**, **glpi_treated_tickets**, **glpi_users**, **glpi_admin**, et **glpi_notifications**. Cette approche assure que chaque nouvelle entrée dans ces tables est dotée d'un identifiant unique, essentiel pour maintenir l'intégrité des données et permettre les opérations de lecture, d'écriture et de mise à jour de manière efficace.

Par exemple, le déclencheur *glpi_entities_trigger* est associé à la table **glpi_entities**. Avant chaque insertion de données dans cette table, ce déclencheur récupère la prochaine valeur de la séquence *glpi_entities_seq* et l'assigne à l'ID de la nouvelle ligne. Des déclencheurs similaires sont définis pour les autres tables mentionnées.

Outre la génération d'identifiants uniques, nous avons également mis en place un déclencheur spécifique, *trg_user_id_last_updater_notification*, qui s'active après la mise à

jour du champ *user_id_last_updater* dans la table **glpi_tickets**. Ce déclencheur est conçu pour gérer la création de notifications associées à des mises à jour spécifiques des tickets.
Lorsqu'une mise à jour est effectuée sur le champ *user_id_last_updater* d'un ticket, ce déclencheur récupère les identifiants des utilisateurs concernés, puis insère une nouvelle notification dans la table **glpi_notifications**, enregistrant ainsi les détails de la mise à jour et les utilisateurs impliqués.

Les déclencheurs spécifiques à Cergy sont chacun associés à un bâtiment spécifique, tel que Fermat, Cauchy, Turing, Condorcet, Saint Martin et Saint Germain, et sont conçus pour intercepter les opérations d'insertion, de mise à jour et de suppression sur les tickets de ces emplacements.

Par exemple, le déclencheur *insert_ticket_fermat* (cité plus tôt) est spécifique au bâtiment Fermat. Il est activé lorsqu'une opération d'insertion, de mise à jour ou de suppression est effectuée sur les tickets de ce bâtiment. Ce déclencheur assure que les données des tickets liés à Fermat sont correctement gérées dans la table **glpi_tickets**. De manière similaire, d'autres déclencheurs sont définis pour chaque bâtiment de Cergy, garantissant ainsi une gestion précise et efficace des tickets au niveau local.

Ces déclencheurs sont étroitement intégrés à notre approche basée sur des vues pour la manipulation des données. Par exemple, la vue *selectTicketFromFermat* est connectée au déclencheur *insert_ticket_fermat*. Lorsqu'une opération d'insertion est effectuée sur cette vue pour créer ou mettre à jour un ticket dans le bâtiment Fermat, le déclencheur correspondant est déclenché, assurant ainsi que les données sont correctement synchronisées avec la table **glpi_tickets**, et ce, de manière transparente pour l'utilisateur.
Il existe aussi pour Pau un déclencheur similaire à ceux de Cergy pour gérer les tickets de Pau.

De cette manière, les déclencheurs contribuent à maintenir la cohérence et l'intégrité des données entre les différentes vues et tables de la base de données, garantissant ainsi une expérience utilisateur fluide et cohérente lors de la gestion des tickets sur les différents sites de CY TECH. En adoptant une approche basée sur des déclencheurs, nous renforçons la robustesse de notre système de gestion du parc informatique, offrant ainsi une expérience utilisateur fluide et sécurisée pour les administrateurs et les utilisateurs finaux.

Procédures

Les procédures stockées jouent un rôle essentiel dans le cadre de notre système de gestion du parc informatique de CY TECH. Elles automatisent divers processus, facilitant ainsi la gestion efficace des utilisateurs, des tickets et d'autres entités du système. Voici un aperçu des principales procédures que nous avons mises en œuvre :

La procédure *close_ticket*, est conçue pour fermer un ticket existant en fonction de sa priorité. Elle prend en entrée l'ID du ticket à fermer et vérifie d'abord si l'utilisateur actuel possède des rôles de technicien pour Cergy ou Pau. Ensuite, elle récupère la priorité du ticket, insère les données du ticket dans la table **glpi_treated_tickets**, supprime le ticket de la table **glpi_tickets**, et envoie une notification aux administrateurs si la priorité du ticket est élevée.

La procédure *reopen_ticket* est destinée à re-ouvrir un ticket précédemment traité. Elle prend en entrée l'ID du ticket à rouvrir, récupère la priorité précédente du ticket à partir de la table **glpi_treated_tickets**, puis insère les données du ticket dans la table **glpi_tickets**, supprimant simultanément le ticket traité de la table **glpi_treated_tickets**. Elle envoie également une notification aux administrateurs si la priorité du ticket est élevée.

La procédure *create_users_procedure* parcourt la table **glpi_users** pour créer des utilisateurs dans la base de données Oracle correspondants aux enregistrements de la table. Elle vérifie d'abord si un utilisateur correspondant existe déjà dans la base de données Oracle. Si ce n'est pas le cas, elle crée un nouvel utilisateur avec les privilèges appropriés.

La procédure *add_admin_procedure* permet d'attribuer le rôle de technicien à un utilisateur spécifié. Elle prend en entrée l'ID de l'utilisateur et utilise ces informations pour attribuer le rôle de technicien correspondant à l'utilisateur dans la base de données Oracle.

La procédure *scan_and_check_users* parcourt tous les utilisateurs Oracle de la base de données et vérifie s'il existe des utilisateurs GLPI correspondants. Si aucun utilisateur GLPI correspondant n'est trouvé, la procédure supprime l'utilisateur Oracle correspondant de la base de données.

Ces procédures contribuent à automatiser et à rationaliser divers aspects de la gestion du parc informatique, garantissant ainsi un fonctionnement efficace et une maintenance facile du système. En intégrant ces procédures dans notre système, nous améliorons la productivité de l'administration du parc informatique et offrons une expérience utilisateur plus fluide.

Optimisation des performances

Index

Dans le cadre de l'optimisation des performances de notre système de gestion du parc informatique de CY TECH, nous avons mis en place plusieurs index sur les tables pertinentes. Les index sont des structures de données supplémentaires qui permettent d'accélérer les opérations de recherche et de filtrage dans la base de données. Les détails de la création de ces index sont disponibles dans le fichier “index.sql”.

Ces index ont été choisis pour améliorer les performances des requêtes les plus fréquemment exécutées dans notre système, en accélérant les opérations de recherche et de filtrage basées sur des colonnes telles que la localisation, la priorité, l'identifiant de l'entité, l'email des utilisateurs, et les identifiants d'utilisateurs dans les notifications.

Cluster

En plus des index, nous avons mis en place un cluster sur la colonne location de la table **glpi_tickets**. Les clusters regroupent physiquement les lignes de données en fonction des valeurs des colonnes de regroupement spécifiées, ce qui peut améliorer les performances des requêtes en réduisant les accès disque et en regroupant les données similaires ensemble. Les détails de la création de ce cluster dans le fichier “cluster.sql”.

Ce cluster sur la colonne location permet de regrouper physiquement les tickets en fonction de leur localisation, ce qui peut accélérer les opérations de recherche et de traitement des tickets basées sur cette colonne. Bien que pour Pau, où il n'existe actuellement qu'un seul site, l'utilité apparente de ce champ de localisation puisse sembler limitée dans l'immédiat. Toutefois, nous avons maintenu la même modélisation que pour Cergy, avec des sous-sites définis, pour plusieurs raisons.

D'abord, cette uniformité de modélisation entre les sites de Cergy et Pau simplifie la gestion et la maintenance du système dans son ensemble. En maintenant un schéma cohérent, nous facilitons la scalabilité future et l'adaptation à de nouveaux besoins éventuels.

De plus, même si actuellement Pau ne comporte qu'un seul site, notre approche offre une flexibilité précieuse pour l'avenir. En permettant la déclaration de plusieurs sous-sites, nous prévoyons une éventuelle expansion ou une meilleure gestion des ressources sur le site de Pau, tout en maintenant une structure de base de données robuste et adaptable.