slapd-sql

Backend sql

Backend SQL pour slapd. Il permet de présenter les informations stockées dans un RDBMS sous forme d'entrée LDAP.

options - Configuration de source de données

dbname <datasource name> Le nom de la source ODBC à utiliser

dbhost <hostname>

dbpasswd <password>

dbuser <username> Informations de connections à la source ODBC

options - Configuration de scope

subtree_cond <SQL expression> template where-clause pour former une condition de recherche de subtree **(dn="(.+,)?<dn>$")**. (ex : "<upper_func>(ldap_entries.dn) LIKE CONCAT('%', ?)")

children_cond <SQL expression> template where-clause pour former une condition de recherche enfant **(dn=".+,<dn>$")**. (ex : "<upper_func>(ldap_entries.dn) LIKE CONCAT('%,', ?)")

use_subtree_shortcut YES Ne pas utiliser de condition de subtree quand la base de recherche est le suffix de la base.

options - Configuration de déclarations

oc_query <SQL expression> requête utilisée pour collecter le mapping des objectClass depuis la table ldap_oc_mappings. (défaut : "SELECT id, name, keytbl, keycol, create_proc, delete_proc, expect_return FROM ldap_oc_mappings")

at_query <SQL expression> requête utilisée pour collecter le mapping des attributeType depuis la table ldap_attr_mappings

id_query <SQL expression> requête utilisée pour mapper un DN à une entrée dans la table ldap_entries (défaut : "SELECT name, sel_expr, from_tbls, join_where, add_proc, delete_proc, param_order, expect_return FROM ldap_attr_mappings WHERE oc_map_id=?"))

insentry_stmt <SQL expression> requête utilisée pour insérer une nouvelle entrée dans la table ldap_entries (défaut : "INSERT INTO ldap_entries (dn, oc_map_id, parent, keyval) VALUES (?, ?, ?, ?)")

delentry_stmt <SQL expression> requête utilisée pour supprimer une entrée existante depuis la table ldap_entries (défaut : "DELETE FROM ldap_entries WHERE id=?")

delobjclasses_stmt <SQL expression> requête utilisée pour supprimer l'ID d'une entrée dans la table ldap_objclasses. (défaut : "DELETE FROM ldap_entry_objclasses WHERE entry_id=?"

options - Configuration helper

upper_func <SQL function name> Spécifie le nom d'un fonction qui convertit une valeur en majuscule (ex : UCASE, UPPER)

upper_needs_cast NO Définis si upper_func doit explicitement être utilisé sur des chaînes littérales.

strcast_func <SQL function name> Spécifie le nom d'une fonction qui convertis une valeur donnée en chaîne pour l'ordonnancement. Utilisé dans SELECT DISTINCT.

concat_pattern <pattern> Définis le pattern utilisé pour concaténer les chaînes. le pattern doit contenir 2 ' ?' qui seront remplacés par 2 chaînes à concaténer. (défaut : CONCAT(?, ?), PostfreSQL : ?|| ?)

aliasing_keyword <string> Définis le mot clé d'alias. défaut : AS

aliasing_quote <string> Définis le caractère de quoting pour les mots clé d'alias. défaut : aucun.

has_ldapinfo_dn_ru NO Informe explicitement le backend si la colonne dn_ru (DN sous forme majuscule inversé) est présent dans la table ldap_entries.

fail_if_no_mapping NO Force les opérations d'écriture d'attribut à échouer si aucun mapping approprié n'est disponible. uniquement pour les attributs. N'a pas d'impacte sur les classec d'objet.

allow_orphans NO Permet d'ajouter des entrées orphelines.

baseObject [<filename>] Instruit la base de créer et gérer une entrée baseObject en mémoire au lieu d'en chercher un dans le RDBMS. Si filename est fournis, l'entrée est lue depuis ce ldif. Utile quand les informations ldap_entries sont stockés dans une vue et union n'est pas supporté pour les vues

create_needs_select NO Instruit la base si la création d'entrée dans la table ldap_entries a besoin d'un select pour collecter l'ID assigné automatiquement, au lieu d'être retourné par une procédure stockée.

fetch_attrs <attrlist>

fetch_all_attrs NO Le premier état permet de fournir un liste d'attributs qui doivent toujours être remplis en plus de ceux requis par un opération spécifique. Pour l'instant, tous les attributs utilisés dans les ACL devraient être listés ici. Le second état est un raccourci pour tous les atributs

check_schema YES Instruit la base de vérifier l'adhérence des entrées au schéma après modification.

sqllayer <name> [...] Charge la couche name dans la pile d'helpers qui sont utilisés pour mapper les DN de LDAP vers la représentation SQL et inversement. Les arguments suivants sont passés à la routine de configuration.

autocommit NO Active l'autocommit. défaut : off

Exemples

Supposons que l'on stocke des informations sur des personnes travaillant dans notre organisation dans 2 tables :
PERSONS PHONES
————-
id integer id integer
first_name varchar pers_id integer references persons(id)
last_name varchar phone
middle_name varchar
...
Une personne peut avoir plusieurs téléphones, phone peut contenir plusieurs enregistrements avec le pers_id correspondant, ou aucun.
Une classe d'objet LDAP pourrait présenter de tels informations comme ceci :
person
MUST cn
MAY telephoneNumber $ firstName $ lastName
...
Pour renseigner toutes les valeurs cn, on construit la requête :
SELECT CONCAT(persons.first_name,' ',persons.last_name) AS cn FROM persons WHERE persons.id=?
Pour telephoneNumber on peut utiliser :
SELECT phones.phone AS telephoneNumber FROM persons,phones WHERE persons.id=phones.pers_id AND persons.id=?
Si on veut des requêtes LDAP avec des filtres tels que (telephoneNumber=123*), on peut construire :
SELECT ... FROM persons,phones WHERE persons.id=phones.pers_id AND persons.id=? AND phones.phone like '%1%2%3%'
On a donc les informations sur quelles tables contiennent les valeurs de chaque attribut, comment joindre ces tables et arranger les valeurs, on pourrait essayer de générer automatiquement de tels déclaration, et traduire les filtres de recherche en clause WHERE.
Pour stocker de tels informations, on ajoute 3 tables de plus à notre schéma :
ldap_oc_mappings (some columns are not listed for clarity)