# ANSI-INCITS_359-2004

Le modèle de contrôle d'accès basé sur des rôles

# Scope

Ce standard consiste de 2 parties - le modèle de référence RBAC, et la spécification fonctionnelle système et administrative de RBAC.

Le modèle de référence RBAC définis un jeu d'éléments de base RBAC (par exemple, des utilisateurs, des rôles, des permissions, opérations et objets) et les relations comme les types et fonctions qui sont incluses dans ce standard. Le modèle de référence RBAC sert 2 buts. D'abord, il référence le périmètre des fonctionnalités RBAC qui sont incluses dans le standard. Il identifie le jeu minimum de fonctionnalités inclus dans tous les système RBAC, les aspects de la hiérarchie de rôle, de relations de contraintes statiques, et de relations de contraintes dynamiques. Ensuite, le modèle de référence fournis un langage précis et consistant, en termes de jeux d'éléments et de fonctions à utiliser pour la définition de spécification fonctionnelles.

Le système RBAC et la spécification fonctionnelle système et administrative RBAC spécifient les fonctionnalités qui sont requises d'un système RBAC. Ces fonctionnalités remplissent 3 catégories, les opérations administratives, les revues administratives, et la fonctionnalité de niveau système. Les opérations administratives définissent les fonctions en terme d'interface administrative et un jeu associé de sémantiques qui fournissent la capacité de créer, supprime et maintenir des éléments RBAC et des relations (ex, pour créer et supprimer des assignements de rôle utilisateur). Les fonctionnalités de revue administratives définissent des fonction en terme d'interface administratives et un jeu associé de sémantiques qui fournissent la capacité d'effectuer des opérations de requêtes sur des éléments et des relations RBAC. La fonctionnalité niveau système définie les fonctionnalité pour la création de sessions utilisateurs pour inclure l'activation/désactivation de rôles, les contraintes forcées dans l'activation de rôle, et pour le calcul d'une décision d'accès.

# Conformité

Toutes les fonctionnalités RBAC ne sont pas appropriées pour toutes les applications. Ce standard fournis une méthode de packaging de fonctionnalité via la sélection de composants fonctionnels et optionnels dans un composant, commençant avec un jeu cœur des fonctionnalité RBAC qui doivent être inclus dans tous les packages.

# Conformité

Pour ce conformer à ce standard, un système RBAC doit être conforme avec tout le jeu core des spécification fonctionnelles RBAC dans la clause 6.1. La conformité d'un système RBAC à d'autres spécification fonctionnelles pour un composant particulier et options, trouvés dans les clauses 6.2 à 6.4, est optionnel et dépendent des exigences fonctionnelles d'une application particulière.

# Termes et définitions

**Composant** réfère à un des blocks majeurs des fonctionnalités RBAC.

**Objets** Peut être une ressource système sujet à contrôle d'accès, tel qu'un fichier, une imprimante, etc.

**Opérations** image exécutable d'un programme, qui à l'invocation exécute une fonction pour l'utilisateur.

**Permissions** Approbation pour effectuer une opération dans un ou plusieurs objets Protégés par RBAC.

**role** job d'un humain. bien que le concept d'utilisateur peut être étendus pour inclure des machines, réseaux, etc., la définition est limitée à une personne dans ce document.

# Le modèle de référence RBAC

Il est définis en terme de 4 composants : le cœur RBAC, La hiérarchie RBAC, séparation statique des privilèges, et séparation dynamique des privilèges. Le cœur RBAC définis une collection minimum d'éléments RBAC, de jeux d'éléments, et des relations pour achever le système RBAC. Cela inclus l'assignement de rôle utilisateur et les relations d'assignement permission/rôle, considérés comme fondammental dans un système RBAC. De plus, le cœur RBAC introduit le concept d'activation de rôle comme faisant partie d'une session utilisateur. Le cœur RBAC est requis dans tout système RBAC, mais les autres composants sont indépendants des autres et peuvent être implémentés séparémments.

La hiérarchie RBAC ajoute les relations pour supporter les hiérarchies de rôle. Une hiérarchie est mathématiquement un ordre partiel définissant une relation d'apparentée entre les rôles, où les rôles séniors acquièrent les permissions de leurs juniors et les rôles juniors acquièrent les utilisateurs de leurs sénior. De plus, la hiérarchie RBAC va au-delà du simple assignement utilisateur et permission en introduisant le concept de jeu de rôles d'utilisateurs autorisé et de permissions autorisées. La séparation statique des relations de privilège ajoute des relations exclusives avec les rôles en respect des assignements utilisateur. À cause de potentiels inconsistances d'un hiérarchie de rôle, le modèle de relation SSD définis les relations sur la présence et l'absence des hiérarchies de rôle. La séparation dynamique des relations de privilèges définis les relations exclusives en respect des rôles qui sont activés, faisant partie de la session utilisateur.

Chaque composant est définis par les sous-composants suivants :

- Un jeu de jeux d'éléments de base
- Un jeu de relations RBAC impliquant ces éléments (contenant des sous-jeu de produits cartésiens dénotant une assignement valide)
- Un jeu de fonction de mappage, qui maintiennent des instances de membre d'un éléments, définis pour une instance donnée depuis un autre jeu d'élément.

Il est important de notser que le modèle de référence RBAC définis une taxonomie des fonctionnalités RBAC qui peuvent être composés en un nombre de package de foncionnalités. Au lieu de tenter de définir un jeu complet de fonctionnalité RBAC, ce modèle de concentre à produire un jeu standard de termes pour définir les foncitonnalités les plus saillantes comme représenté dans les modèles existants et implémenté dans les produits commerciaux.

# cœur RBAC

Le cœur RBAC inclus les jeux de 5 éléments de données de base appelés des utilisateurs (USERS), des rôles (ROLES), des objets (OBS), des opérations (OPS), et des permissions (PRMS). Le modèle RBAC dans son ensemble est fondamentalement définis en terme d'utilisateurs individuels étant assignés aux rôles et de permissions étant assignés aux rôles. Ainsi, un rôle est un moyen de nommer des relations many-to-many entre des individus et des permissions. De plus, le modèle du cœur RBAC inclus un jeu de sessions (SESSIONS) où chaque session est un mappage entre un utilisateur et un sous-jeu activé de rôle qui sont assignés à l'utilisateur.

Un utilisateur est définis comme une personne. Bien que le concept d'un utilisateur peut être étendus pour inclure des machines, réseaux, ou des agents autonomes intelligents, la définition est limitée à une personne dans ce document pour des raisons de simplicité. Un rôle est une fonction (job) dans le contexte de l'organisation avec des sémantiques associées avec l'autorité et la responsabilité conférée à l'utilisateur assigné au rôle. Les permissions sont une approbation pour effectuer une opération sur un ou plusieurs objets protégés par RBAC. Une opération est une image exécutable d'une programme, qui à l'invocation exécute certaines fonctions pour l'utilisateur. Les types d'opérations et objets que RBAC contrôle sont dépendant du type de systèmes dans lequel il est implémenté. Par exemple, dans un système de fichier, les opérations peuvent inclure lire, écrire, et exécuter ; dans une base de données, les opérations peuvent inclure insert, delete, append, et update.

Le but de tout mécanisme de contrôle d'accès est de protéger les ressources système. Consistant avec les anciens modèles de contrôle d'accès un objet est une entité qui contient ou reçois des informations. Pour un système qui implémente RBAC, les objets peuvent

représenter des conteneurs d'informations (par exemple des fichiers, répertoires, un système d'exploitation, colonnes/lignes/tables/views dans une base de données), ou des objets qui représentent des ressources systèmes exhaustives, tels qu'une imprimante, espace disque, et cycles CPU. Le jeu d'objets couvert par RBAC inclus tous les objets listés dans les permissions qui sont assignés aux rôles.

Le centre de RBAC est le concept de relation de rôle, autour duquel un rôle est une construction sémantique pour formuler des stratégies. La figure ci-dessous illustre les relations d"assignement utilisateur (UA) et d'assignement de permission (PA). Les flèches indiquent une relation many-to-many (ex : un utilisateur peut être assigné à un ou plusieurs rôles, et un rôle peut être assigné à un ou plusieurs utilisateurs. Cet arrangement fournis une grande flexibilité et granularité d'assignement des permissions aux rôles et des utilisateurs aux rôles. Sans cela il y a un risque qu'un utilisateur puisse obtenir plus d'accès aux ressources que nécessaire à cause du contrôle limité sur le type d'accès qui peut être associé avec les utilisateurs et les ressources. Les utilisateurs peuvent souhaiter lister les répertoires et modifier les fichiers existants, par exemple, sans créer de nouveaux fichiers, ou ils peuvent souhaiter ajouter des enregistrements à un fichier sant modifier d'enregistrements existants. Tout augmentation dans la flexibilité du contrôle d'accès aux ressources renforce également l'application du principe de moins de privilège.

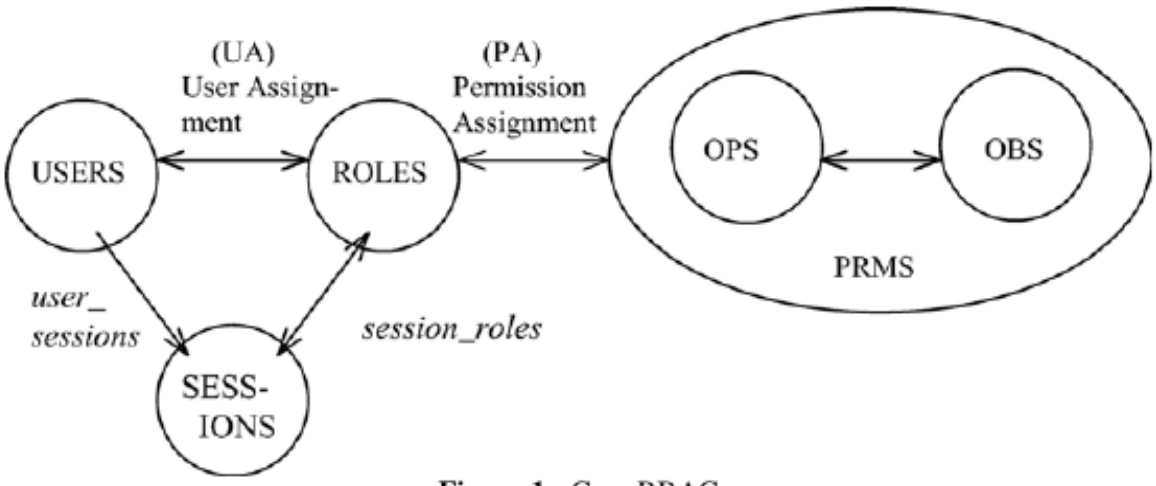


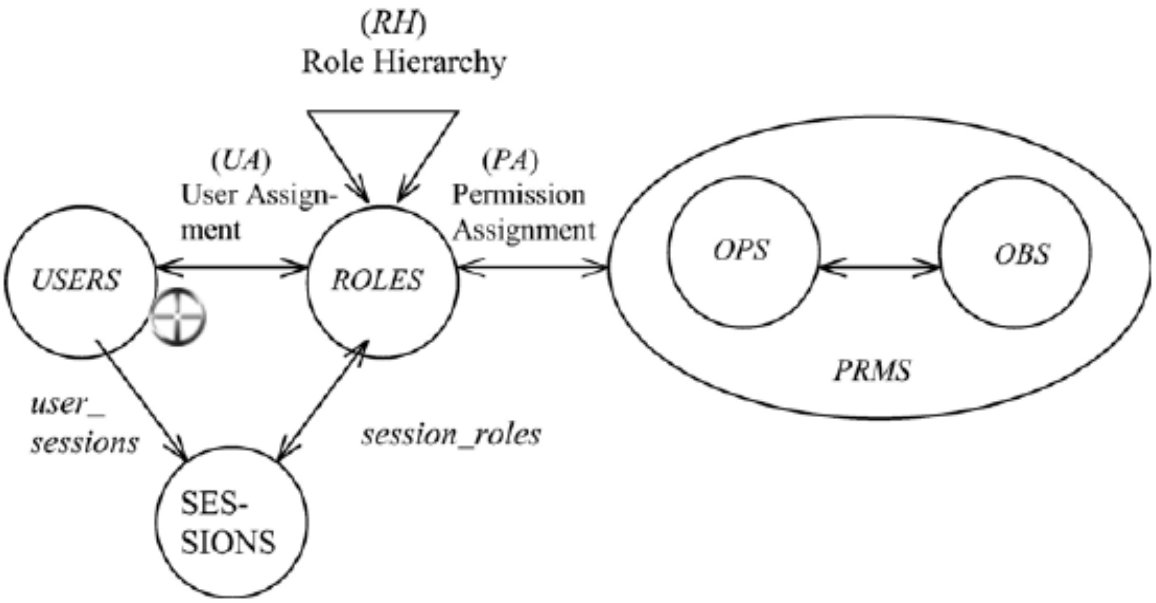

**Figure 1:** Core RBAC

Chaque session est un mappage d'un utilisateur à possiblement plusieurs rôles. Un utilisateur établis une session durant laquelle l'utilisateur active des sous-jeux de rôles auquel il est assigné. La fonction session_roles nous donne les rôle activés par la session et la fonction session_users nous donne l'utilisateur qui est associé avec une session. Les permissions disponibles à l'utilisateur sont les permissions assignées aux rôle qui sont actuellement actifs au travers de toutes les sessions utilisateur.

# Hiérarchie RBAC

Ce modèle introduit des hiérarchies de rôle (RH), communément inclus comme un aspect clé des modèles RBAC. Les hiérarchies sont un moyen naturel de structurer les rôles pour refléter les lignes d'autorité et de responsabilité d'une organisation.

Les hiérarchies de rôle définissent une relation d'héritage avec les rôles. L'héritage a été décrit en terme de permission, par exemple, r1 hérite du rôle r2 si tous les privilèges de r2 sont également privilèges de r1. Pour certaines distribution de RBAC les permissions de rôle ne sont pas gérés centralement, bien que ces hiérarchies de rôle le soient. Pour ces systèmes, les hiérarchies de rôle sont gérés en terme de relations de contention utilisateur. Les rôle r1 contient le rôle r2 si tous les utilisateurs autorisés pour r1 sont également autorisés pour r2. Noter, cependant, que la contention d'utilisateur implique qu'un utilisateur de r1 ais (au moins) tous les privilèges de r2, alors que l'héritage de permissions pour r1 et r2 n'impliquent rient sur l'assignement de l'utilisateur.




Ce standard reconnaît 2 types de hiérarchies de rôle - les hiérarchies de rôle générales et les hiérarchies de rôle limitées. Les hiérarchies de rôle générales fournissent un support pour un ordre arbitraire partiel pour servir de hiérarchie de rôle, pour inclure le concept d'héritage multiple des permissions et de membership via les rôles. Les hiérarchies de rôle limitées imposent des restrictions résultant en une structure plus simple (par exemple, un rôle peut avoir un ou plusieurs ascendants immédiats, mais est restreints à un seul descendant immédiat).

# Hiérarchies de rôle générales

Les hiérarchies de rôle générales supportent le concept d'héritage multiple, qui fournis la capacité d'hériter les permissions de 2 ou plusieurs rôles source et d'hériter des utilisateurs membre de 2 ou plusieurs rôles sources. l'héritage multiple fournis 2 propriétés de hiérarchie importantes. La première est la capacité de composer un rôle depuis plusieurs rôles subordonnés (avec moins de permissions) en définissant des rôles et relations qui sont caractéristiques de l'organisation et de structures métier, que ces rôle représentent. Ensuite, l'héritage multiple fournis un traitement uniforme des relations d'assignement utilisateur/rôle et des relations d'héritage rôle/rôle.

Les rôle dans une hiérarchie de rôle limitée sont restreints à un simple descendant immédiat. Bien que les hiérarchies de rôle limités ne supportent pas l'héritage multiple, elles ne fournissent pas d'avantage administratif claire sur Core RBAC.

Une hiérarchie de rôle générale peut être représenté en diagramme de Hasse. Les nœuds dans le graphique représent les rôles de la hiérarchie et il y a un ligne direct entre r1 et r2 si r1 est un descendant immédiat de r2.

# Contraintes RBAC

Les contraintes RBAC ajoute les relations de séparation des privilèges au modèle RBAC. Ces relation sont utilisées pour forcer un conflit d'interêt des stratégies que les organisations peuvent employer pour empêcher les utilisateurs d'excéder un niveau raisonnable d'autorité pour leur positions.

Le conflit d'interêt dans un système basé sur les rôles peut se produire comme résultat d'un gain d'autorisation pour des permissions associés avec les rôles en conflit. Pour empêcher cette forme de conflit, la séparation de privilèges statique force les contraintes dans l'assignement des utilisateurs aux rôles. Les contraintes statiques peuvent prendre un variété de formes différentes.

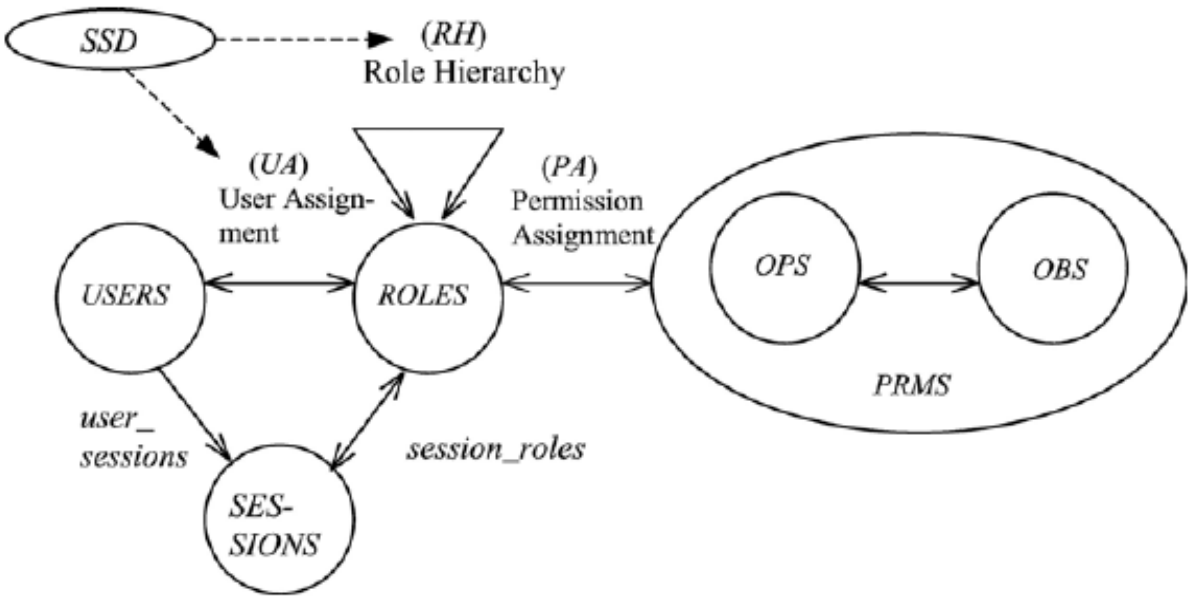
Les contraintes statiques définies dans ce modèle sont limitées à ces relations qui placent des restrictions dans les jeux de rôle et en particulier dans leur capacité à former les UA. Cela signifie que si un utilisateur est assigné à un rôle, l'utilisateur se voit refuser l'appartenance d'un second rôle. Une stratégie SSD peut être centralement spécifiée et imposée uniformément dans des rôles spécifiques.

Les modèles RBAC ont définis des relations SSD en respect aux contraintes dans les assignements user-role sur des paires de rôles. Bien que dans le monde réel des stratégies SSD existent, cette définition est trop restrictive sous 2 aspects. D'abord, la taille du jeu de rôles dans le SSD et ensuit la combinaison des rôles dans le jeu pour lequel l'assignement utilisateur est restreint. Dans ce modèle, SSD est définis avec 2 arguments. Un jeu de rôle qui inclus 2 ou plusieurs rôles et une cardinalité supérieur à un indiquent une combinaison de rôle qui constiturait une violation de la stratégie SSD. Par exemple, une organisation peut exiger qu'aucun utilisateur ne puisse être assigné à 3 des 4 rôles qui représentent la fonction d'achat.

Comme illustr ci-dessous, les relations SSD peuvent exister dans la hiérarchie RBAC. En appliquant des relations SSD dans la présence d'une hiérarchie de rôle, une attention particulière doit être appliquée pour s'assurer que l'héritage utilisateur n'amoindri pas les stratégies SSD. Ainsi, les hiérarchies de rôle ont été définis pour inclure l'héritage des contraintes SSD. Pour adresser cette inconsistance potentielle, SSD est définis dans les utilisateurs autorisés des rôles qui ont une relation SSD.

Les relations SSD réduisent le nombre de permissions potentielles qui peuvent être faites à un utilisateur en plaçant des contraintes dans les utilisateurs qui peuvent être assignés à un jeu de rôles. Les relations de séparation de privilège dynamique (DSD), comme les relations SSD, sont prévues pour limiter les permissions qui sont disponibles à un utilisateur. Cependant, les relations DSD diffèrent des relations SSD par le contexte dans lequel ces limitations sont imposées. Les relations SSD définissent et placent les contraintes dans l'espace de permission total de l'utilisateur. Ce modèle définis les propriétés DSD qui limitent la capacité des permissions sur une espace de permission utilisateur en plaçant des contraintes dans les rôles qui peuvent être activés avec ou au travers d'une session utilisateur. Les propriétés DSD fournissent un support étendu pour le principe du moins de privilège dans cela que chaque utilisateur a différents niveaux de permissions à des moments différents, en fonction du rôle effectué. Ces propriétés s'assurent que les permissions ne persistent pas au-dela du temps requis pour la performance du privilège. Cet aspect du moins de privilège est sous référré à une révocation en temps utile de privilège. La révocation dynamique des permissions peut être un problème plus complexe sans les facilités de la séparation dynamique de privilèges, et a été largement ignoré dans le passé.

Ce modèle fournis la capacité de forcer une stratégie spécifique à une organisation de séparation dynamique des privilèges (DSD). Les relations SSD fournissent la capacité d'adresser de potentiels conflits d'interêt au moment où un utilisateur est assigné à un rôle. DSD permet à un utilisateur d'être autorisé pour 2 ou plusieurs rôles qui ne créent pas de conflit lorsqu'ils agissent indépendamment, mais produit un problème de stratégie quand activé simultanément. Par exemple, un utilisateur peut être autorisé pour les rôles de caissier et de contrôleur de caisse, RBAC exige que l'utilisateur abandonne le rôle de caissier et fermet son tirroir caisse avant de passer contrôleur. Tant que ce même utilisateur n'est pas autorisé à assumer ces 2 rôles simultanément, un conflit d'interêt ne peut pas se produire.




# Système RBAC et spécification fonctionnelle administrative

La spécification fonctionnelle RBAC spécifie les opérations administratives pour la création et la maintenance de jeux d'éléments et de relations RBAC ; les fonctions administratives pour effectuer les requêtes administratives ; et les fonctions système pour créer et gérer les

attributs RBAC dans les sessions utilisateur et prendre les décisions de contrôle d'accès. Les fonctions sont définies avec une précision suffisante pour répondre aux besoins de conformité de test et d'assurance, tout en fournissant aux développeurs la capacité d'incorporer des fonctionnalités additionnelles pour répondre aux besoins des utilisateurs.

# Commandes administratives pour Core RBAC

**AddUser** Cette commande créé un nouvel utilisateur RBAC. La commande est valide seulement si le nouvel utilisateur n'est pas déjà membre du jeu de données USERS. Ce jeu USERS est mis à jours. Le nouvel utilisateur ne possède pas de session au moment de sa création.

**DeleteUser** Cette commande supprime un utilisateur existant de la base RBAC. La commande est valide si et seulement si l'utilisateur à supprimer est un membre du jeu de données USERS. Les jeux de données USER et UA et la fonction assigned_users sont mis à jours. C'est une décision d'implémentation de savoir comment procéder avec les sessions possédés par l'utilisation à supprimer.

**AddRole** Cette commande crée un nouveau rôle. La commande est valide si et seulement si le nouveau rôle n'est pas déjà membre du jeu de données ROLES. Le jeu ROLES et les fonctions assigned_users et assigned_permissions sont mis à jours. Initialement, aucun utilisateur ou permission n'est assigné au nouveau rôle.

**DeleteRole** Cette commande supprime un rôle existant de la base RBAC. La commande est valide si et seulement si le rôle est membre du jeu de données ROLES. C'est une décision d'implémentation de décidé de la manière de procéder avec les sessions dans le rôle à supprimer sont actifs.

**AssignUser** Cette commande assigne un utilisateur à un rôle. La commande est valide si et seulement si l'utilisateur est un membre du jeu de données USERS et le rôle est membre du jeu de données ROLES, et que l'utilisateur n'est pas déjà assigné au rôle. Le jeu de données UA et la fonction assigned_users sont mis à jours pour refléter l'assignement.

**DeassignUser** Cette commande supprime l'assignement d'un utilisareur à un rôle. Cette commande est valide si et seulement si le rôle est un des rôles actif de l'utilisateur.

**GrantPermission** Cette commande octroie à un rôle la permission d'effectuer une opération sur un objet à un rôle. La commande peut être implémentée pour octroyer les permissions à un groupe correspondant à ce rôle, par ex., en définissant la liste de contrôle d'accès à l'objet en question.

**RevokePermission** Cette commande révoque la permission d'effectuer une opération sur un objet depuis le jeu de permissions assignés à un rôle. La commande peut être implémentée pour révoquer les permissions d'un groupe correspondant à ce rôle, par ex., en définissant la liste de contrôle d'accès de l'objet en question.

# Fonctions système pour Core RBAC

**CreateSession(user,session)** Cette fonction créé une nouvelle session avec un utilisateur donné comme propriétaire et un jeu de rôle actif. La fonction est valide si et seulement si l'utilisateur est membre du jeu de données USERS et que le jeu de rôle actif est un sous-jeu des rôles assignés à cet utilisateur.

**DeleteSession(user,session)** Cette fonction supprime une session données avec un utilisateur. La fonction est valide si et seulement si l'identifiant de session est un membre du jeu de données SESSIONS, l'utilisateur est un membre du jeu de données USERS, et la session est possédée par l'utilisateur donné.

**AddActiveRole** Cette fonction ajoute un rôle comme rôle actif d'une session dont le propriétaire est l'utilisateur donné. La fonction est valide si et seulement si l'utilisateur est membre du jeu de données USERS et le rôle est membre du jeu de données ROLES, et l'identifiant de session est membre du jeu de données SESSIONS, et le rôle est assigné à l'utilisateur, et la session est possédée par l'utilisateur.

**DropActiveRole** Cette fonction supprime un rôle du jeu de rôle actif d'une session possédé par un utilisateur donné. La fonction est valide si et seulement si l'utilisateur est membre du jeu de données USERS, l'identifiant de session est un membre du jeu de données SESSION, la session est possédées par l'utilisateur, et le rôle est un rôle actif de cette session.

**CheckAccess** Cette fonction retourne une valeur booléenne indiquant si le sujet d'une session données est autorisée ou non à effectuer l'opération données sur l'objet donné. La fonction est valide si et seulement si l'identifiant de session est un membre du jeu de données SESSIONS, l'objet est un membre du jeu de données OBJS, et l'opération est un membre du jeu de données OPS. Le sujet de la session a la permission d'effectuer l'opération sur cet objet si et seulement si cette permission est assignée à (au moins) un des rôles actifs de la session. Une implémentation peut utiliser les groupes qui correspondent aux rôle actifs du sujet et leur permission comme enregistré dans la liste de contrôle d'accés de l'objet.

# Fonctions d'examen pour Core RBAC

**AssignedUsers** Cette fonction retourne le je d'utilisateurs assignés à un rôle donné. La fonction est valide si et seulement si le rôle est un membre du jeu de données ROLES.

**AssignedRoles** Cette fonciton retourne le je de rôles assignés à l'utilisateur donné. La fonction est valide si et seulement si l'utilisateur est membre du jeu de données USERS.

# Fonctions d'examen avancé pour Core RBAC

**RolePermissions** Cette fonction retourne le jeu de permissions (op,obj) octroyés à un rôle donné. La fonction est valide si et seulement si le rôle est un membre du jeu de données ROLES.

**UserPermissions** Cette fonction retourne les permissions qu'un utilisateur obtient via ses rôles assignés. La fonction est valide si et seulement si l'utilisateur est un membre du jeu de données USERS.

**SessionRoles** Cette fonction retourne les rôles actifs associés avec une session. La fonction est valide si et seulement si l'identifiant de session est un membre du jeu de données SESSIONS.

**SessionPermissions** Cette fonction retourne les permissions de la session, par ex., les permission assignées à ses rôles actifs. La fonction est valide si et seulement si l'identifiant de session est un membre du jeu de données SESSIONS.

**RolesOperationsOnObject** Cette fonction retourne le jeu d'opérations d'un rôle donné autorisé sur un objet donné. Cette fonction est valide seulement si le rôle est un membre du jeu de données OBJS

**UserOperationsOnObject** Cette fonction retourne le jeu d'opérations qu'un utilisateur donné est autorisé à effectuer sur un objet donné, obtenu soit directement ou via ses rôles assignés. La fonction est valide si et seulement si l'utilisateur est un membre du jeu de données USERS et l'objet est un membre du jeu de données OBJS.

# Commandes administratives pour les hiérarchies de rôles générales

**AddInheritance** Cette commande établis une nouvelle relation d'héritage immédiat entre les rôle. La commande est valide si et seulement si les 2 rôles sont membres du jeu de données ROLES.

**DeleteInheritance** Cette commande supprime une relation d'héritage immédiat existante. Cette commande est valide si et seulement si les rôles sont membre du jeu de données ROLES.

**AddAscendant** Cette commande créé un nouveau rôle r_asc, et l'insert dans la hiérarchie de rôle comme ascendant immédiat du rôle existant r_desc. La commande est valide si et seulement si r_asc n'est pas membre du jeu de données ROLES, et r_desc est un membre du jeu de données ROLES.

**AddDescendant** Cette commande créé un nouveau rôle r_desc, et l'insert dans la hiérarchie de rôle comme descendant immédiat du rôle existant r_asc. La commande est valide si et seulement si r_desc n'est pas membre du jeu de données ROLES, et r_asc est un membre du jeu de données ROLES.

# Fonctions système pour les hiérarchies de rôles générales

**CreateSession(user,session)** Cette fonction créé une nouvelle session avec un utilisateur donné comme propriétaire, et un jeu donné de rôles actifs. La fonction est valide si et seulement si l'utilisateur est membre du jeu de données USERS, et le jeu de rôle actif est un sous-jeu autorisé pour cet utilisateur. Noter que si un rôle est actif pour une session, ses descendants ne sont pas nécessairement actifs pour cette session. Dans une implémentation RBAC, les rôles actifs d'une session peuvent être les groupes qui représentent ces rôles.

**AddActiveRole** Cette fonction ajoute un rôle comme rôle actif d'une session dont le propriétaire est un utilisateur donné. La fonction est valide si et seulement si l'utilisateur est membre du jeu de données USERS, le rôle est membre du jeu de données ROLES, l'identifiant de session est membre du jeu de données SESSIONS, l'utilisateur est autorisé pour ce rôle, et la session est possédée par cet utilisateur.

# Fonctions d'examen pour les hiérarchies de rôles générales

Toutes les fonctions de la section "Fonctions d'examen pour Core RBAC" sont valides. Cette section définis les fonctions suivantes :

**AuthorizedUsers** Cette fonction retourne le jeu d'utilisateurs autorisés pour un rôle donné, par ex., les utilisateurs qui sont assignés à un rôle qui hérite du rôle donné. La fonction est valide si et seulement si le rôle donné est un membre du jeu de données ROLES.

**AuthorizedRoles** Cette fonction retourne le jeu de rôles autorisés pour un utilisateur donné. La fonction est valide si et seulement si l'utilisateur est un membre de jeu de données USERS.

# Fonction d'examen avancé pour les hiérarchies de rôles générales

Cette section redéfinis les fonction RolePermissions et UserPermissions de la section "Fonctions d'examen avancé pour Core RBAC". toutes les autres fonction de cette section sont valides.

**RolePermissions** Cette fonction retourne le jeu de permission(ob,obj) octroyé à ou hérité par un rôle donné. La fonction est valide si et seulement si le rôle est un membre du jeu de données ROLES.

**UserPermissions** Cette fonction retourne le jeu de permissions qu'un utilisateur obtient via ses rôles assignés. La fonction est valide si et seulement si l'utilisateur est un membre du jeu de données USERS.

**RoleOperationsOnObject** Cette fonction retourne le jeu d'opérations qu'un rôle donné est autorisé à effectuer sur un objet donné. Le jeu contient toutes les opérations octroyées directement à ce rôle ou hérité par ce rôle depuis un autre rôle. La fonction est valide seulement si le rôle est un membre du jeu de données ROLES, et l'objet est un membre de jeu de données OBJS.

**UserOperationsOnObject** Cette fonction retourne le jeu d'opérations qu'un utilisateur est autorisé à effectuer sur un objet donné. Le jeu consiste de toutes les opérations obtenues par l'utilisateur directement, ou via ses rôle autorisés. La fonction est valide si et seulement si l'utilisateur est un membre de jeu de données USERS et l'objet est un membre du jeu de données OBJS.

# Commandes administratives pour les hiérarchies de rôles limitées

Cette section redéfinis la fonction AddInheritance de la section "Commandes administratives pour les hiérarchies de rôles générales". Toutes les autres fonctions de cette section restent valides

**AddInheritance** Cette commande établis une nouvelle relation d'héritage immédiate entre les rôles existants r_asc et r_desc. La commande est valide si et seulement si les rôles sont membre du jeu de données ROLES.

# Fonctions système pour les hiérarchies de rôle limitées

Toutes les fonctions de la section "Fonctions système pour les hiérarchies de rôles générales" sont valides

# Fonction d'examen pour les hiérarchies de rôle limitées

Toutes les fonctions de la section "Fonctions d'examen pour les hiérarchies de rôles générales" sont valides

# Fonction d'examen avancé pour les hiérarchies de rôle limitées

Toutes les fonctions de la section "Fonctions d'examen avancé pour les hiérarchies de rôles générales" sont valides

# Relations SSD

La propriété de séparation des privilèges statique, tel que définis dans ce modèle, utilise une collection SSD de paires d'un jeu de rôle et un cardinalité associée. Cette section définis le nouveau type de données SSD, qui dans un implémentation peut être le jeu de noms utilisé pour identifier les paires dans la collection.

# Commandes administratives pour les relations SSD

Cette section redéfinis la fonction AssignUser de la section "Commandes administratives pour Core RBAC", et définis un nouveau jeu de fonctions spécifiques. Les autres fonction de cette section sont valides.

**AssignUser** Cette commande assigne un utilisateur à un rôle. La commande est valide si et seulement si l'utilisateur est membre du jeu de données USERS, le rôle est membre du jeu de données ROLES, l'utilisateur n'est pas déjà assigné au rôle, et les contraintes SSD sont satisfaites après assignement.

**CreateSsdSet** Cette commande créé un jeu SSD de rôle et définis la cardinalité n de ses sous-jeu qui ne peuvent pas avoir d'utilisateurs communs. La commande est valide si et seulement si le nom du jeu SSD n'est pas déjà utilisé, tous les rôle dans le jeu SSD sont membres du jeu de données ROLES, n est un nombre naturel supérieur ou égal à 2 et inférieur ou égal à la cardinalité du jeu de rôle SSD, et la containte SSD pour le nouveau rôle est satisfait.

**AddSsdRoleMember** Cette commande ajoute un rôle à un jeu SSD de rôles. La cardinalité associée avec le jeu de rôle reste inchangée. La commande est valide si et seulement si le rôle SSD existe, le rôle à ajouté est un membre du jeu de données ROLES mais pas membre du jeu de rôle SSD, et la contrainte SSD est satisfaite après l'ajout du rôle dans le jeu de rôle SSD.

**DeleteSsdRoleMember** Cette commande supprime un jeu de rôle SSD complètement. La commande est valide si et seulement si le jeu de rôle SSD existe.

**SetSsdSetCardinality** Cette commande définis la cardinalité associée avec un jeu de rôle SSD. La commande est valide si et seulement si le jeu de rôle SSD existe, et la nouvelle cardinalité est un nombre naturel supérieur ou égal à 2 et inférieur ou égal au nombre d'éléments du jeu de rôle SSD, et la contrainte SSD est satisfaite après avoir définis la nouvelle cardinalité.

# Fonctions système pour SSD

Toutes les fonctions de la section "Fonctions système pour Core RBAC" sont valides

# Fonctions d'examen pour SSD

Toutes les fonctions de la section "Fonctions d'examen pour Core RBAC" sont valide. De plus, les fonctions suivantes sont définis :

**SsdRoleSets** Cette fonction retourne la liste des jeux de rôle SSD.

**SsdRoleSetRoles** Cette fonction retourne le jeu de rôles d'un jeu de rôle SSD. La fonction est valide si et seulement si le jeu de rôle existe.

**SsdRoleSetCardinality** Cette fonction retourne la cardinalité associée avec un jeu de rôle SSD. La foncion est valide si et seulement si le jeu de rôle existe.

# Fonctions d'examen avancé pour SSD

Toutes les fonctions de la section "Fonctions d'examen avancé pour Core RBAC" sont valide.

# Commandes administratives pour SSD avec les hiérarchies de rôle générales

Cette section redéfinis les fonctions AssignUser et AddInheritance de la section "Commandes administratives pour les hiérarchies de rôles générales", et les fonctions CreateSsdSet, AddSsdRoleMember, SetSsdSetCardinality de la section "Commandes administratives pour les relations SSD". Les autres fonctions de ces 2 section sont valides.

**AssignUser** Cette commande assigne un utilisateur à un rôle. La commande est valide si et seulement si l'utilisateur est un membre du jeu de données USERS, et le rôle est un membre du jeu de données ROLES, et l'utilisateur n'est pas déjà assigné au rôle, et les contraintes SSD sont satisfaites après l'assignement.

**AddInheritance** Cette commande établis une nouvelle relation d'héritage entre les rôles existants r_asc, et r_desc. La commande est valide si et seulement si les rôles sont membres du jeu de données ROLES, r_asc n'est pas un ascendant direct de r_desc, _desc n'hérite pas de r_asc, et les contraintes SSD sont satisfaites après avoir établis l'héritage.

**CreateSsdSet** Cette commande créé un jeu SSD de rôle et les jeux de cardinalité associés n de ses sous-jeux qui n'ont pas d'utilisateurs communs. La commande est valide si et seulement si le nom du jeu SSD n'est pas déjà utilisé, tous les rrôles dans le jeu SSD sont membre du jeu de données ROLES, n est un nombre naturel supérieur ou égal à 2 et inférieur ou égal à la cardinalité du jeu de rôle SSD, et la contrainte SSD pour le nouveau jeu de rôle est satisfaite.

**AddSsdRoleMember** Cette commande ajoute un rôle à un jeu SSD de rôles. La cardinalité associée avec le jeu de rôle reste inchangée. La commande est valide si et seulement si le jeu de rôle SSD existe, et le rôe à ajouter est un membre du jeu de données ROLES mais pas membre du jeu de rôle SSD.

**SetSsdSetCardinality** Cette commande définis la cardinalité associée avec un jeu de rôle SSD. La commande est valide si et seulement si le jeu de rôle SSD existe, la nouvelle cardinalité est un nombre naturel supérieur ou égal à 2 et inférieur ou égal au nombre d'éléments du jeu de rôles SSD, et la containte SSD est satisfaite après avoir définis la nouvelle cardinalité.

# Fonctions système pour SSD avec les hiérarchies de rôle générales

Toutes les fonctions de la section "Fonctions système pour les hiérarchies de rôles générales" sont valides

# Fonctions d'examens pour SSD avec les hiérarchies de rôle générales

Toutes les fonctions des sections "Fonctions d'examen pour les hiérarchies de rôles générales" et "Fonctions d'examen avancé pour SSD" sont valides

# Fonction d'examens avancé pour SSD avec les hiérarchies de rôle générales

Toutes les fonctions de la section "Fonction d'examen avancé pour les hiérarchies de rôles générales" sont valides

# Commandes administratives pour SSD avec les hiérarchies de rôle limitées

Cette section redéfinis la fonction AddInheritance de la section "Commandes administratives pour SSD avec les hiérarchies de rôle générales". Toutes les autres fonctions de cette section sont valides.

**AddInheritance** Cette commande établis une nouvelle relation d'héritage immédiat entre les rôle r_asc et r_desc. La commande est valide si et seulement si les rôles sont membre du jeu de données ROLES, r_asc n'a pas de descendants, et r_desc n'hérite pas de r_asc.

[SECTION] name="Fonctions système pour SSD avec les hiérarchies de rôle limitées" table="paragraphes" imbrication="0"

**Toutes les fonctions de la section "Fonctions système pour SSD avec les hiérarchies de rôle générales" sont valides.**

# Fonctions d'examens pour SSD avec les hiérarchies de rôle limitées

Toutes les fonctions des sections "Fonctions système pour les hiérarchies de rôles générales" et "Fonctions d'examen pour SSD" sont valides

# Fonction d'examens avancé pour SSD avec les hiérarchies de rôle limitées

Toutes les foncions de la section "Fonction d'examen avancé pour les hiérarchies de rôles générales" sont valides.

# Commandes administratives pour les relations DSD

Toutes les fonctions de la section "Commandes administratives pour Core RBAC" sont valide. Cette section définis les fonctions supplémentaires suivantes :

**CreateDsdSet** Cette commande créé un jeu DSD de rôle et définis une cardinalité n associée. La contrainte DSD stipule que le jeu de rôle DSD ne peut pas contenir n rôle ou plus, actifs simultanément dans la même session. La commande est valide si et seulement si le nom du jeu DSD n'est pas déjà utilisé, tous les rôles dans le jeu DSD sont membre du jeu de données ROLES, n est un nombre naturel supérieur ou égal à 2 et inférieur ou égal à la cardinalité du jeu de rôle DSD, et la contrainte DSD pour le nouveau rôle est satisfaite.

**AddDsdRoleMember** Cette commande ajeute un rôle à un jeu de rôles DSD. La cardinalité associée avec le jeu de rôle reste inchangé. La commande est valide si et seulement si le jeu de rôle DSD existe, le rôle à ajouter est membre du jeu de données ROLES, mais non membre du jeu de rôle DSD, et la contrainte DSD est satisfaite après l'ajout du rôle dans le jeu de rôle DSD.

**DeleteDsdRoleMember** Cette commande supprime un rôle depuis un jeu de rôles DSD. La cardinalité associée avec le jeu de rôle reste inchangé. La commande est valide si et seulement si le jeu de rôle DSD existe, le rôle à supprimer est un membre du jeu de rôle DSD, et la cardinalité associée avec le jeu de rôle DSD est inférieur au nombre d'éléments du jeu de rôle DSD.

**DeleteDsdSet** Cette commande supprime un jeu de rôle DSD complètement. Cette commande est valide si et seulement si le jeu de rôle DSD existe.

**SetDsdSetCardinality** Cette commande définis la cardinalité associée avec un jeu de rôle DSD donné. La commande est valide si et seulement si le jeu de rôle DSD existe, la nouvelle cardinalité est un nombre naturel supérieur ou égal à 2 et inférieur ou égal au nombre d'éléments du jeu de rôle DSD, et la contrainte DSD est satisfaite après avoir définis la nouvelle cardinalité.

[SECTION] name="Fonctions système pour les relations DSD" table="paragraphes" imbrication="0"

**Cette section redéfinis les fonctions CreateSession et AddActiveRole de la section "Fonctions système pour Core RBAC". Toutes les au**

**CreateSession** Cette fonction créé une nouvelle session dont le propriétaire est l'utilisateur donné et un jeu de rôles actif. La fonction est valide si et seulement si l'utilisateur est un membre du jeu de données USERS et le jeu de role actif de la session est un sous-jeu des rôle assigné pour le propriétaire de la session, le jeu de rôle actif de la session est un sous-jeu des rôles assignés au propriétaire de la session, et le jeu de rôle actif de la session satisfait les contraintes DSD.

**AddActiveRole** Cette fonction ajoute un rôle comme rôle actif d'une session dont le propriétaire est un utilisateur donné. La fonction est valide si et seulement si l'utilisateur est un membre du jeu de données USERS, le rôle est un membre du jeu de données ROLES, l'identifiant de session est membre du jeu de données SESSIONS, le rôle est assigné à cet utilisateur, et l'ancien jeu de rôle actif complété avec le rôle à activer satisfont les contraintes DSD, et la session est possédée par cet utilisateur.

# Fonctions d'examen pour les relations DSD

Toutes les fonction de la section "Fonctions d'examen pour Core RBAC" sont valide. De plus, cette section définis de nouvelles fonctions :

**DsdRoleSets** Cette fonction retourne la liste de tous les jeux de rôle DSD

**DsdRoleSetRoles** Cette fonction retourne le jeu de rôles d'un jeu de rôle DSD. La fonction est valide si et seulement si le jeu de rôle existe.

**DsdRoleSetCardinality** Cette fonction retourne la cardinalité associée avec un jeu de rôle DSD. La fonction est valide si et seulement si le jeu de rôle existe.

# Fonctions d'examen avancé pour les relations DSD

Toutes les fonctions de la section "Fonctions d'examen avancé pour Core RBAC" sont valides.

# Commandes administratives pour les relations DSD avec les hiérarchies de rôle générales

Toutes les fonctions des sections "Commandes administratives pour les hiérarchies de rôles générales" et "Commandes administratives pour les relations DSD" sont valides

# Fonctions système pour les relations DSD avec les hiérarchies de rôle générales

Cette section redéfinis les fonctions CreateSession et AddActiveRole de la section "Fonctions système pour Core RBAC". Toutes les autres fonctions de cette section sont valides

**CreateSession** Cette fonction créé une nouvelle session dont le propriétaire est l'utilisateur donné et un jeu de rôles actif. La fonction est valide si et seulement si l'utilisateur est un membre du jeu de données USERS et le jeu de role actif de la session est un sous-jeu des rôle autorisés pour le propriétaire de la session, et le jeu de rôle actif de la session satisfait les contraintes DSD.

**AddActiveRole** Cette fonction ajoute un rôle comme rôle actif d'une session dont le propriétaire est un utilisateur donné. La fonction est valide si et seulement si l'utilisateur est un membre du jeu de données USERS, le rôle est un membre du jeu de données ROLES, l'identifiant de session est membre du jeu de données SESSIONS, le rôle est autorisé pour cet utilisateur, et l'ancien jeu de rôle actif complété avec le rôle à activer satisfont les contraintes DSD, et la session est possédée par cet utilisateur.

# Fonctions d’examen pour les relations DSD avec les hiérarchies de rôle générales

Toutes les fonctions des sections "Fonctions d’examen pour les relations DSD" et "Fonctions d’examen pour les hiérarchies de rôles générales" sont valides

# Fonctions d’examen avancé pour les relations DSD avec les hiérarchies de rôle générales

Toutes les fonctions de la section "Fonction d’examen avancé pour les hiérarchies de rôles générales" sont valides

# Commandes administratives pour les relations DSD avec les hiérarchies de rôle limitées

Toutes les fonctions des sections "Commandes administratives pour les hiérarchies de rôles limitées" et "Commandes administratives pour les relations DSD" sont valides

# Fonctions système pour les relations DSD avec les hiérarchies de rôle limitées

Toutes les fonctions de la section "Fonctions système pour les relations DSD avec les hiérarchies de rôle générales" sont valides

# Fonctions d’examen pour les relations DSD avec les hiérarchies de rôle limitées

Toutes les fonctions de la section "Fonctions d’examen pour les relations DSD avec les hiérarchies de rôle générales" sont valides

# Fonctions d’examen avancé pour les relations DSD avec les hiérarchies de rôle limitées

Toutes les fonctions de la section "Fonction d’examen avancé pour les hiérarchies de rôles générales" sont valides