Sécurité des bases de données

Jacques Le Maitre
Université du Sud Toulon-Var

Références bibliographiques

- Silvana Castano, Mariagrazia Fugini, Giancarlo Martella, Pierangela Samarati, *Database Security,* Addison-Wesley, 1995.
- Ron Ben Nathan, *Implementing Database Security and Auditing*, Elsevier Digital Press, 2005.
- Pierre Delmal, *SQL2–SQL3, chapitre 6,* De Boeck Université, 2001.

Introduction

Sécurisation d'une BD : objectifs

- Assurer la sécurité d'une BD c'est maintenir :
  - la **confidentialité**,
  - l'**intégrité**,
  - et la **disponibilité**

  des données.

Maintien de la confidentialité

- Il s'agit de détecter ou d'empêcher des accès non autorisés.
- C'est crucial :
  - dans des environnements critiques ou stratégiques : militaires ou commerciaux, par exemple.
  - pour respecter le droit des individus à décider comment et dans quel but les informations les concernant peuvent être extraites, mémorisées ou transmises à d'autres individus.

Maintien de l'intégrité

- Il s'agit de détecter ou d'empêcher des modifications illicites des données qu'elles soient dues à :
  - des pannes de système,
  - des manipulations erronées,
  - des sabotages.

Maintien de la disponibilité

- Il s'agit de détecter ou d'empêcher des **dénis de service**.
- Il y a déni de service lorsqu'un utilisateur ne parvient pas à accéder dans un **délai raisonnable**, à une information ou à une ressource pour laquelle il a une autorisation d'accès.
- Par exemple, une attaque consistant à saturer un serveur de fausses requêtes empêchant les requêtes valides d'être exécutées.

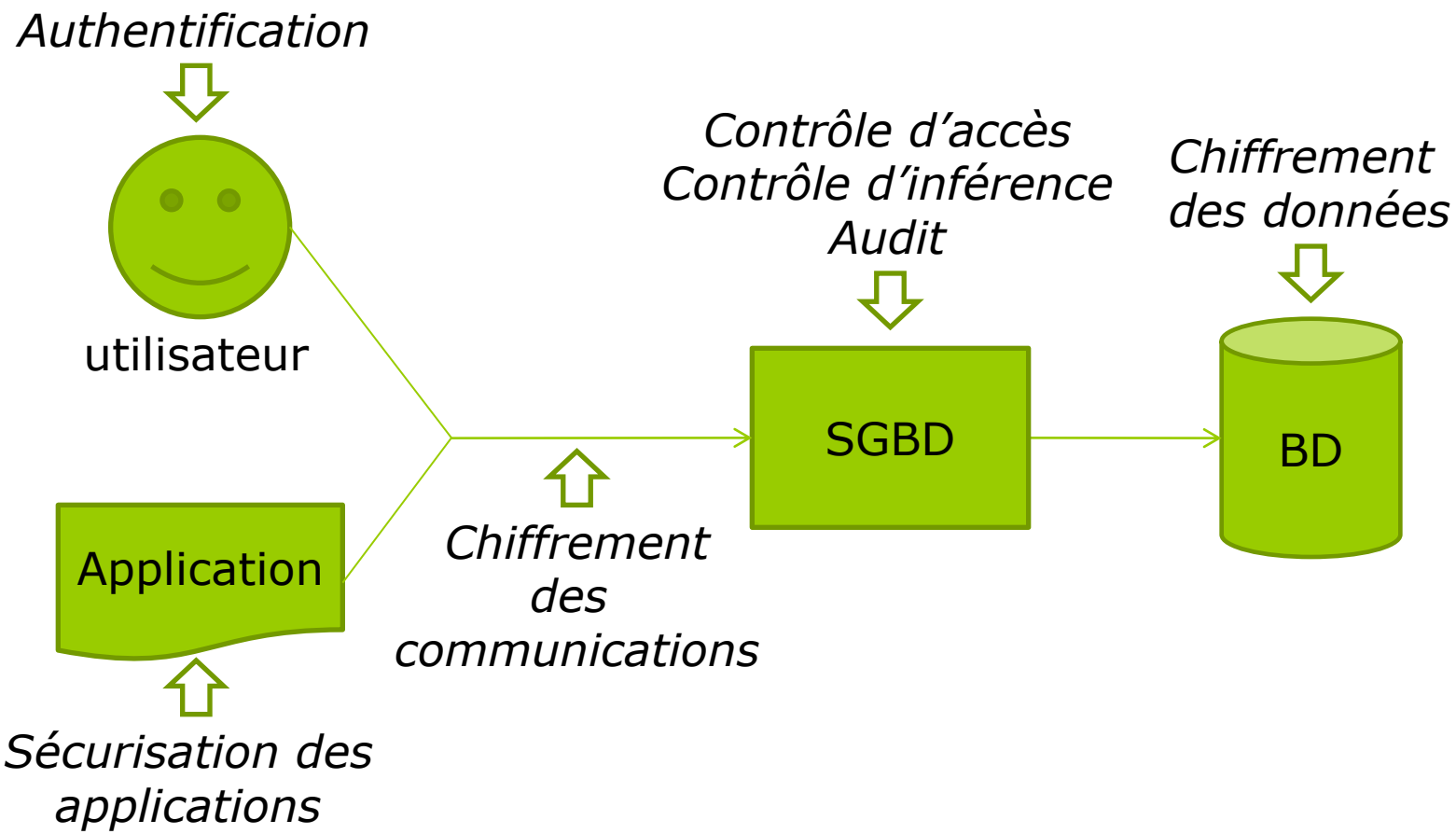
Attaques

- Les violations de la sécurité d'une BD consistent en des lectures ou des mises à jour illicites.
- Les événements qui portent ces violations sont appelées des **attaques**.

Types d'attaques

- On distingue :
  - les attaques **non frauduleuses** :
    - catastrophes naturelles,
    - pannes de logiciel ou de matériel,
    - erreurs humaines…
  - les attaques **frauduleuses** :
    - utilisation abusive de leurs droits par les utilisateurs,
    - agents hostiles exécutant des actions de destruction du logiciel ou du matériel, ou lisant ou mettant à jour des données protégées,
    - ces agents peuvent être cachés dans des actions légales : **chevaux de Troie**.

Protections contre les attaques



Authentification

- L'**authentification** a pour objectif d'assurer que l'utilisateur qui se connecte à la BD :
  - est autorisé à se connecter,
  - est bien celui qui s'annonce.
- L'authentification repose sur :
  - la sécurité des mots de passe,
  - des techniques d'identification biométriques.

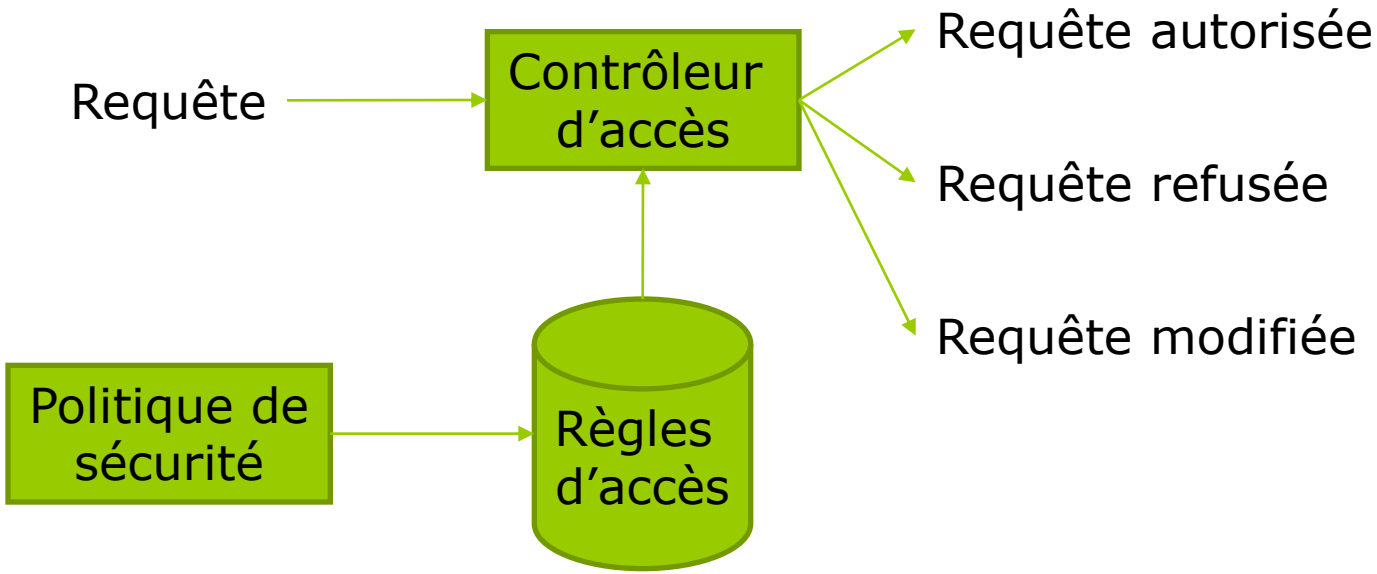
Contrôle d'accès

- Un système de contrôle d'accès comprend :
  - des **sujets** : utilisateurs, processus… qui peuvent être classés par groupes,
  - des **objets** : données, programmes…
  - des **opérations** : lecture, ajout, modification, suppression…, déclenchées par les sujets sur les objets
  - un **règlement de sécurité** constitué d'un ensemble de **règles d'accès** traduisant la **politique de sécurité** du système d'information.
  - un **processeur de sécurité** qui vérifie que les requêtes adressées au système ne violent pas les règles d'accès et selon le cas autorise, modifie ou interdit la requête.

Politique de sécurité

- Les ITSEC définissent la politique de sécurité comme l'ensemble des *lois règles ou pratiques qui régissent la façon dont l'information sensible et les autres ressources sont gérées, protégées et distribuées à l'intérieur d'un système d'information*.

Processeur de sécurité



Principe du moindre privilège

- Ce principe stipule qu'un sujet ne doit disposer que des droits d'accès minimum pour assurer l'exécution des tâches qui lui sont assignées, pas un de plus.
- Ex : ne pas donner les droits de l'administrateur à tout utilisateur d'un système (système d'exploitation, SGBD).

Autorisation, interdiction et obligation

- Les règlements les plus simples ne contiennent que des **autorisations** :
  - ce qui n'est pas autorisé est interdit.
- Certains règlements incluent des **interdictions** afin de spécifier des exceptions à des permissions générales.
  - Ex : les patients ont droit de consulter leur dossier médical sauf Jean Dupont.
- D'autres enfin, plus sophistiqués, incluent des **obligations** :
  - difficiles à implanter dans les systèmes informatiques.

Modèles de contrôle d'accès

- On distingue deux grandes catégories de modèles de contrôle d'accès :
  - les modèles **discrétionnaires** :
    - Le créateur d'un objet en est son propriétaire et le transfert des privilèges sur ces objets est à sa discrétion.
  - les modèles **obligatoires** :
    - Il s'agit de protéger le secret et l'intégrité.
    - Les sujets (ou utilisateurs) et objets sont classés par niveaux (ex: Top secret, Secret, Confidentiel, Public).

Contrôle d'inférence

- L'objectif du **contrôle d'inférence** est protéger une BD des attaques consistant à déduire des données non autorisées à partir de données autorisées.
- Ex : interdire l'accès à des données individuelles dans une BD statistiques à partir de requêtes agrégatives (comptage, somme, moyenne...).

Sécurisation des applications

- Les attaques à une BD peuvent exploiter les failles des applications opérant sur cette BD :
  - stockage des mots de passe dans les fichiers de configuration de l'application,
  - scripts de connexion à la BD accessibles dans le code source de l'application,
  - attaques par injection SQL,
  - attaques exploitant les débordements de tampons

  ...

Injection SQL (1)

- On suppose qu'une application récupère le nom *username* et le mot de passe *password* saisi, dans un formulaire, par un utilisateur et se connecte à la BD en exécutant la requête affectée à la variable `$requete` composée par concaténation :
  - `$requete =`

    `"SELECT *`

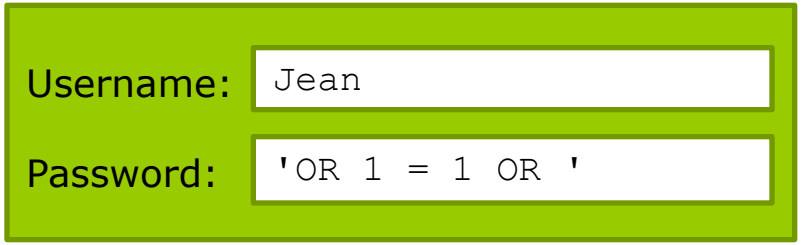
    `FROM users`

    `WHERE username = '" & username & "'`

    `AND password = '" & password & "' ;"`

Injection SQL (2)

- Un attaquant désirant se connecter comme administrateur pourra saisir dans le formulaire :



- La requête générée sera :

```
SELECT *
FROM users
WHERE username = 'Jean'
AND password = '' OR true OR '';
```

- L'attaquant se retrouvera connecté comme 1er utilisateur de la BD qui a toutes les chances d'être l'administrateur.

Audit

- Un outil indispensable pour assurer la sécurité d'une BD est l'**audit** basé sur un journal des différents types d'accès à la BD :
  - audit des entrées dans la base,
  - audit des utilisations de la BD en dehors des heures ouvrables,
  - audit de la manipulation du schéma,
  - audit des erreurs,
  - audit des modifications des sources des procédures stockées et des triggers,
  - audit des modifications des attributs de sécurité (login, privilèges...)

  ...

Modèles de contrôle d'accès

Principaux modèles

- Modèles discrétionnaires à base de matrice d'accès
- Modèle à base de rôles
- Modèles obligatoires
  - Modèle de Bell et LaPadula
    - protection du secret
  - Modèle de Biba
    - protection de l'intégrité
- Les modèles à base de rôle et les modèles obligatoires s'appuient aussi sur une matrice d'accès.

Modèles discrétionnaires

- Chaque objet a un propriétaire (en général, son créateur) qui décide quels sont les autres sujets qui ont accès à cet objet :
  - d'où le nom de « **discrétionnaire** »
- Il y a donc décentralisation du contrôle d'accès.
- Ce contrôle peut malgré tout être centralisé en confiant tous les droits à l'administrateur du système.

Modèle à base de matrice d'accès

- Initialement proposé par B. W. Lampson en 1971 pour les systèmes d'exploitation.
- Etendu G. S. Graham et D. E. Denning en 1972
- Formalisé par M. A. Harrison, W. L. Ruzzo, J. D. Ullman en 1976 :
  - modèle HRU

Modèle HRU

- Le modèle HRU est basé sur l'utilisation d'une matrice d'accès indiquant les modes d'accès à un objet autorisés pour un sujet.
- Le système est défini par le quadruplet (S, O, R, M) :
  - S est l'ensemble des **sujets**,
  - O est l'ensemble des **objets**,
  - R est l'ensemble des **mode d'accès**,
  - M est la **matrice d'accès**.

Sujets

- Un sujet est une entité active qui peut exécuter des actions et de qui le système doit être protégé.
- Un sujet peut être un utilisateur, un ensemble d'utilisateurs, un programme...

Objets

- Un objet est une entité qui doit être protégée.
- Un objet est soit :
  - une entité passive (fichiers, programmes, base de données...),
  - un sujet.

Modes d'accès

- read (**r**)
  - lecture d'une information contenue dans un objet
- write (**w**)
  - écriture d'une information dans un objet en voyant son contenu
- append (**a**)
  - ajout d'une information à un objet sans voir son contenu
- execute (**x**)
  - exécution d'un programme
- **own**
  - possession d'un objet

Matrice d'accès

- $M[s, o]$ = ensemble des opérations autorisées pour le sujet *s* sur l'objet *o* (ensemble des **privilèges** de *s* sur *o*).
- Si **own** $\in M[s, o]$ alors *s* est le propriétaire de *o* et l'administrateur des autorisations sur *o* :
  - il peut les transmettre à d'autre utilisateurs.

Exemple de matrice d'accès

objets / sujets	fichier F_1	fichier F_2
utilisateur U_1	**own** U_1 a tous les droits sur F_1 dont celui de transmettre tout ou partie de ces droits à un autre utilisateur	**x*** U_1 a le droit d'exécuter F_2 et de transmettre ce droit à un autre utilisateur
utilisateur U_2	_	**own**
programme P	**r, w** P a le droit de lire ou de modifier F_1	_

État du système

- L'état du système est le triplet (*S*, *O*, *M*).
- L'application d'une opération à un état produit un nouvel état qui diffère de l'ancien par au moins l'une des 3 composantes.

Administration des autorisations

- Le créateur d'un objet reçoit automatiquement **own** sur cet objet :
  - il en est le **propriétaire**.
- Le propriétaire d'un objet peut octroyer/révoquer à un autre sujet n'importe quel autre privilège sur cet objet excepté le privilège **own**.
- Un sujet ayant un privilège sur un objet sans en être le propriétaire peut transmettre ce privilège.
- Si *m* est un privilège **r**, **w**, **a** ou **x** et $m^* \in M[s, o]$ est autorisé à octroyer à *s* le privilège *m* sur l'objet *o*.

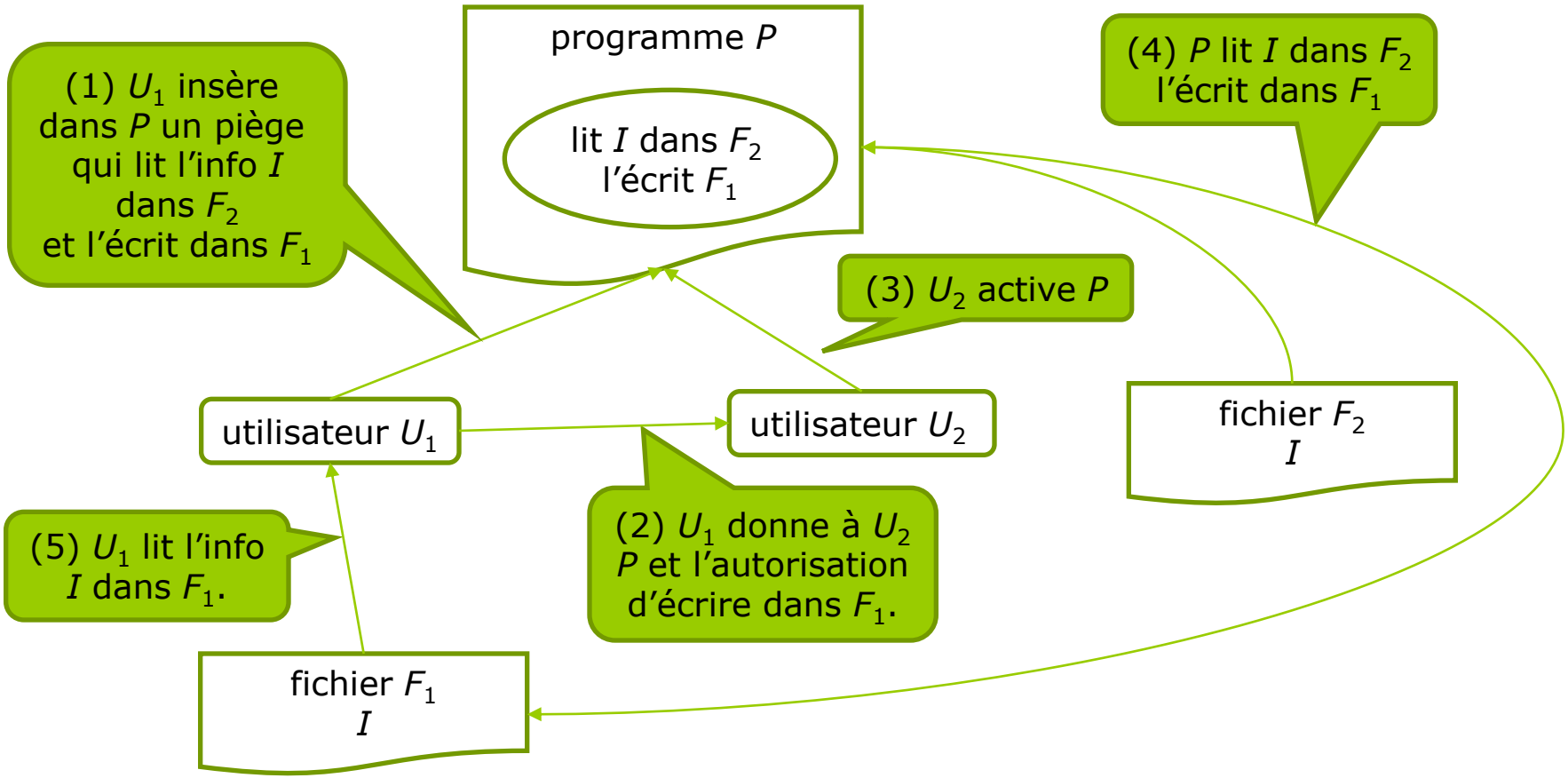
Modèles discrétionnaires : avantages et inconvénients

- Ils sont simples à mettre en œuvre.
- Ce sont les plus implantés : UNIX, SQL…
- Le contrôle de la propagation des droits et celui de la révocation des droits propagés posent des problèmes difficiles.
- Ils sont vulnérables aux **chevaux de Troie**.

Exemple de cheval de Troie

- Soit :
  - 1 programme P
  - 2 fichiers F_1 et F_2
  - 1 utilisateur U_1 :
    - propriétaire de P et de F_1
    - sans privilèges sur F_2
  - 1 utilisateur U_2 :
    - propriétaire de F_2
    - sans privilèges sur F_1
- Comment U_1 peut-il obtenir une information I contenue dans F_2 qu'il n'a pas le droit de lire ?

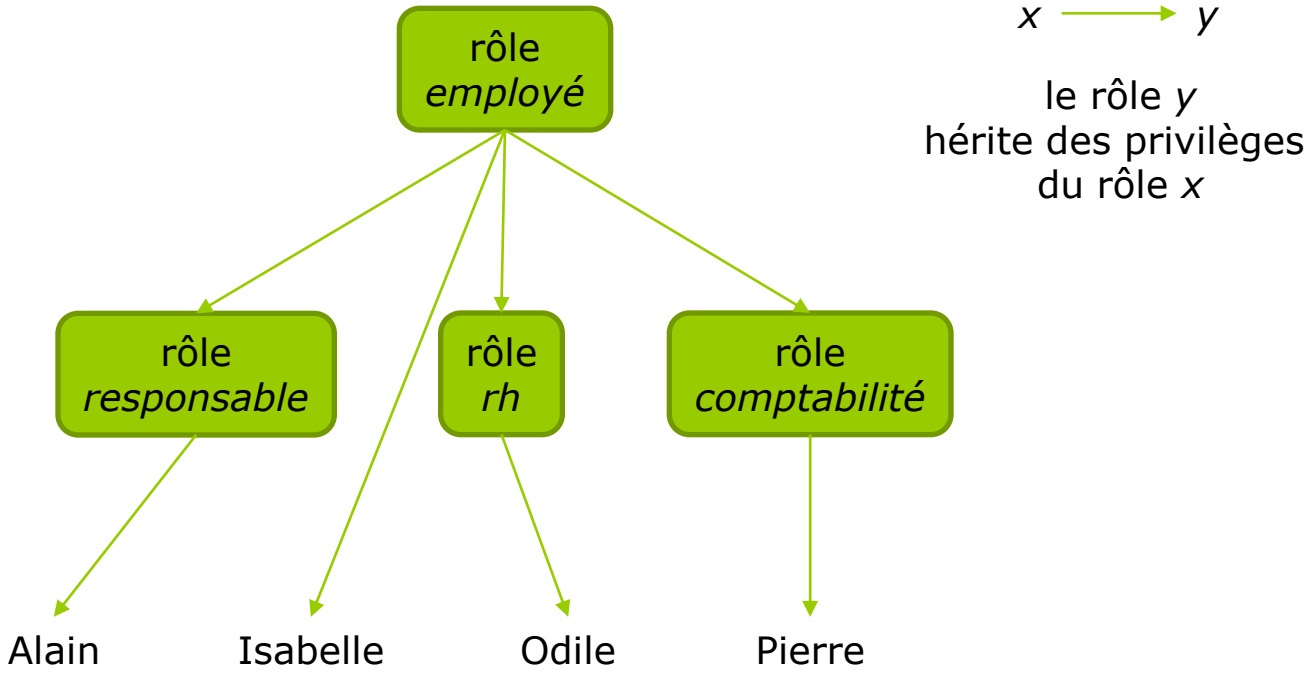
Exemple de cheval de Troie



Modèles à base de rôles

- On peut améliorer les modèles discrétionnaires en créant des **rôles** qui sont des ensembles d'autorisation.
- Les autorisations sont octroyés aux rôles et les rôles aux utilisateurs.
- Pour pouvoir réaliser une action sur un objet un utilisateur doit ouvrir une session et activer celui de ses rôles qui contient l'autorisation de réaliser cette action.
- Un rôle peut hériter des privilèges d'un autre rôle.
- Les modèles à base de rôles :
  - sont bien adaptés aux applications de gestion : un rôle pour chaque fonction dans l'entreprise.
  - permettent une administration souple des privilèges.

Héritage de rôles : exemple



x ⟶ *y*

le rôle *y*
hérite des privilèges
du rôle *x*

Modèles obligatoires

- Ils ont été développés pour des systèmes d'information pour lesquels la préservation du secret est primordiale :
  - militaires, par exemple.
- Ce sont des modèles **multi-niveaux**.

Modèle de Bell et LaPadula (BLP)

- Le modèle de Bell et LaPadula a pour objectif la protection du secret des informations.
- Il est basé sur la classification des **objets** et des **sujets** par **niveaux de secret**.
- L'ensemble des niveaux de secret est muni d'un ordre partiel (≥). Par exemple :
  - Top secret (TS) > Secret (S) > Confidentiel (C) > Non classifié (NC).

Définition du système

- Le système est défini par le 7-uplet (S, O, R, b, M, f, H) :
  - S, O, R, M comme dans le modèle HRU,
  - b est l'ensemble des **accès courants**, il est composé de triplets (s, o, m) spécifiant que le sujet s a un accès courant à l'objet o en mode m,
  - f est la fonction de niveau qui associe à chaque sujet ou à chaque objet un niveau de sécurité :
    - $f_o(o)$ est le niveau de sécurité de l'objet o,
    - $f_s(s)$ est le niveau de sécurité affecté à un sujet lorsqu'il est créé dans le système,
    - $f_c(s)$ est le niveau de sécurité courant du sujet s, celui avec lequel il entre dans une session ($f_c(s) \leq f_s(s)$).
  - H une classification hiérarchique des objets.

Objets

- Les objets sont des éléments passifs du système qui contiennent des informations.
- Chaque objet reçoit un niveau de secret.
- Ce niveau reflète la sensibilité de l'information stockée dans cet objet et donc le problème que poserait la transmission de cette information à un utilisateur non autorisé à la consulter.
- Le niveau de secret d'un objet doit dominer celui de ses parents dans la hiérarchie de classification *H*.

Utilisateurs et sujets

- Chaque utilisateur reçoit un niveau de secret.
- Ce niveau traduit la **confiance** faite à un utilisateur de ne pas transmettre une information à un utilisateur ayant un niveau de secret inférieur au niveau de secret de cette information.
- Un utilisateur peut entrer dans le système à tout niveau ≤ à son niveau de secret.
- Les sujets sont des processus activés par les utiisateurs.
- Les sujets activés par un utilisateur reçoivent le même niveau de secret que celui-ci.

Modes d'accès aux objets

- read (**r**)
  - lecture d'information dans un objet
- append (**a**)
  - ajout d'information dans un objet sans voir son contenu
- write (**w**)
  - écriture dans un objet avec la permission de voir son contenu
- execute (**x**)
  - exécution d'un objet (programme)

Politique de sécurité

- Un état du système est sécuritaire s'il vérifie les trois propriétés suivantes :
  - propriété de sécurité discrétionnaire (**sd**)
  - propriété de simple sécurité (**ss**)
  - propriété étoile (*)

Propriété de sécurité discrétionnaire **sd**

- **Règles**. Chaque accès courant doit être présent dans la matrice d'accès : un sujet ne peut effectuer que les accès pour lesquels il a les autorisations nécessaires.
- Un état (b, M, f, H) satisfait la propriété **sd** si, et seulement si, pour chaque (s, o, m) $\in$ b on a $m \in M[s, o]$.

Propriété de simple sécurité (**ss**)

- **Règle**. Un sujet *s* peut lire ou écrire une information dans un objet *o* seulement si le niveau de secret de *s* est ≥ au niveau de secret de *o*.
- Un état (*b*, *M*, *f*, *H*) satisfait la propriété **ss** si, et seulement si, pour chaque $(s, o, m) \in b$ tel que $m = \mathbf{r}$ ou $m = \mathbf{w}$ on a $f_S(s) \geq f_O(o)$.
- La satisfaction de cette propriété assure qu'un sujet n'accédera pas à une information classée à un niveau plus haut que lui.

Sujet de confiance

- On appelle **sujet de confiance** un sujet dont on est sûr qu'il ne transmettra pas une information à un utilisateur ayant un niveau de secret inférieur au niveau de secret de cette information.

Propriété *

- **Règles**. Un sujet *s* qui n'est pas de confiance peut :
  - ajouter une information dans un objet *o* seulement si le niveau de sécurité courant de *s* est ≤ au niveau de secret de *o*,
  - écrire une information dans un objet *o* seulement si le niveau de secret courant de *s* est = au niveau de secret de *o*,
  - lire une information dans un objet *o* seulement si le niveau de secret courant de *s* est ≥ au niveau de secret de o.
- Un état (*b*, *M*, *f*, *H*) satisfait la propriété * si, et seulement si, pour chaque (*s*, *o*, *m*) $\in$ *b* tel que *s* n'est pas un sujet de confiance on a :
  - $m = \mathbf{a} \Rightarrow f_c(s) \leq f_o(o)$
  - $m = \mathbf{w} \Rightarrow f_c(s) = f_o(o)$
  - $m = \mathbf{r} \Rightarrow f_c(s) \geq f_o(o)$

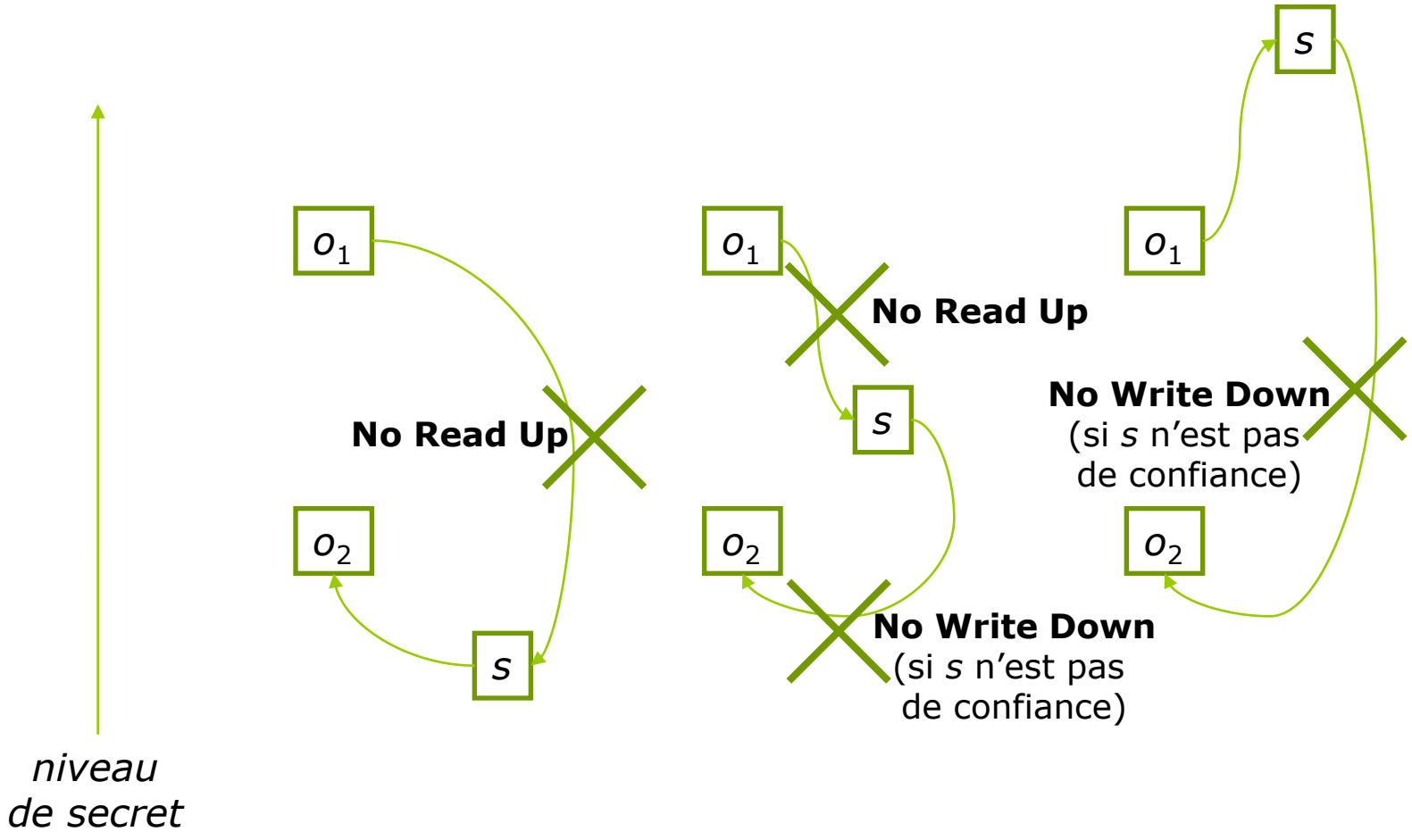
Liens entre **ss** et *

- La propriété * implique la propriété **ss**.
  (à démontrer !)
- Cependant ces deux propriétés sont nécessaires car :
  - la propriété **ss** concerne tous les sujets,
  - la propriété * ne concerne que les sujets qui ne sont pas de confiance.

En résumé

- La politique de sécurité du modèle de Bell et LaPadula se résume dans les deux principes :
  - **No Read Up Secrecy**
    - préserve de la lecture d'une information dans un objet par un sujet de niveau de secret inférieur.
  - **No Write Down Secrecy**
    - préserve d'un transfert d'information d'un objet vers un autre objet de niveau de secret inférieur, par un utilisateur qui n'est pas de confiance.

No Read Up / No Write Down



Exemple d'état sécuritaire

niveaux de secret = 1 < 2 < 3 < 4 < 5 < 6 < 7 < 8
sujets : Alice et Odile (de confiance),
Pierre (non de confiance)

M	F_1	P	F_2	F_3
Alice	**r**, **w**			**r**
Pierre	**r**		**r**	**w**
Odile		**x**	**a**	**r**

	F_1	P	F_2	F_3
f_O	4	8	3	1

	f_S	f_C
Alice	7	4
Pierre	5	1
Odile	6	2

b = {(Alice, F_1, **r**), (Pierre, F_3, **w**), (Odile, P, **x**)}

Exemple d'état non sécuritaire

niveaux de secret = 1 < 2 < 3 < 4 < 5 < 6 < 7 < 8
sujets : Alice et Odile (de confiance),
Pierre (non de confiance)

M	F_1	P	F_2	F_3
Alice	**r, w**	**r**		**r**
Pierre	**r**		**r**	**w**
Odile		**x**		**r**

	F_1	P	F_2	F_3
f_O	4	8	3	1

	f_S	f_C
Alice	7	4
Pierre	5	3
Odile	6	2

b = {(Alice, *P*, **r**), (Pierre, *F3*, **w**), (Odile, *F2*, **a**)}

- l'accès d'Alice à P contrevient à **ss**
- l'accès de Pierre à F_3 contrevient à ***
- l'accès d'Odile à F_2 contrevient à **sd**

Leurres

- Un **leurre** est une information fausse introduite dans une BD multi-niveaux (en général) pour protéger l'existence d'une information sensible.

Exemple de leurre

- Supposons que les 2 informations suivantes soient classées *Secret* :
  - « Le salaire mensuel de Jean est de 600 € »,
  - « Le salaire de Jean est classé secret ».
- A la question:
  - « Quel est le salaire de Jean ? »

  on ne peut donc pas répondre :
  - « Le salaire de Jean est secret ou inconnu. »
- On affecte donc à Jean un faux salaire (200 €, par exemple) mais public.
  - Ce faux salaire est un leurre.

Modèle de Biba

- Le modèle de Biba (K. J. Biba, 1977) a pour objectif la protection de l'intégrité des informations.
- Il applique à la protection de l'intégrité une stratégie similaire à celle de la protection du secret par le modèle BLP : les sujets et les objets sont classés par **niveaux d'intégrité**.
- L'ensemble des niveaux d'intégrité est muni d'un ordre partiel (≥). Par exemple :
  - Crucial (TS) > Très important (TI) > Important (I) > Non classifié (NC)

Classification des objets

- Chaque objet reçoit un niveau d'intégrité.
- Ce niveau reflète la confiance qui peut être accordée à l'information stockée dans cet objet et donc le problème que pourrait poser une modification non autorisée de cet objet.

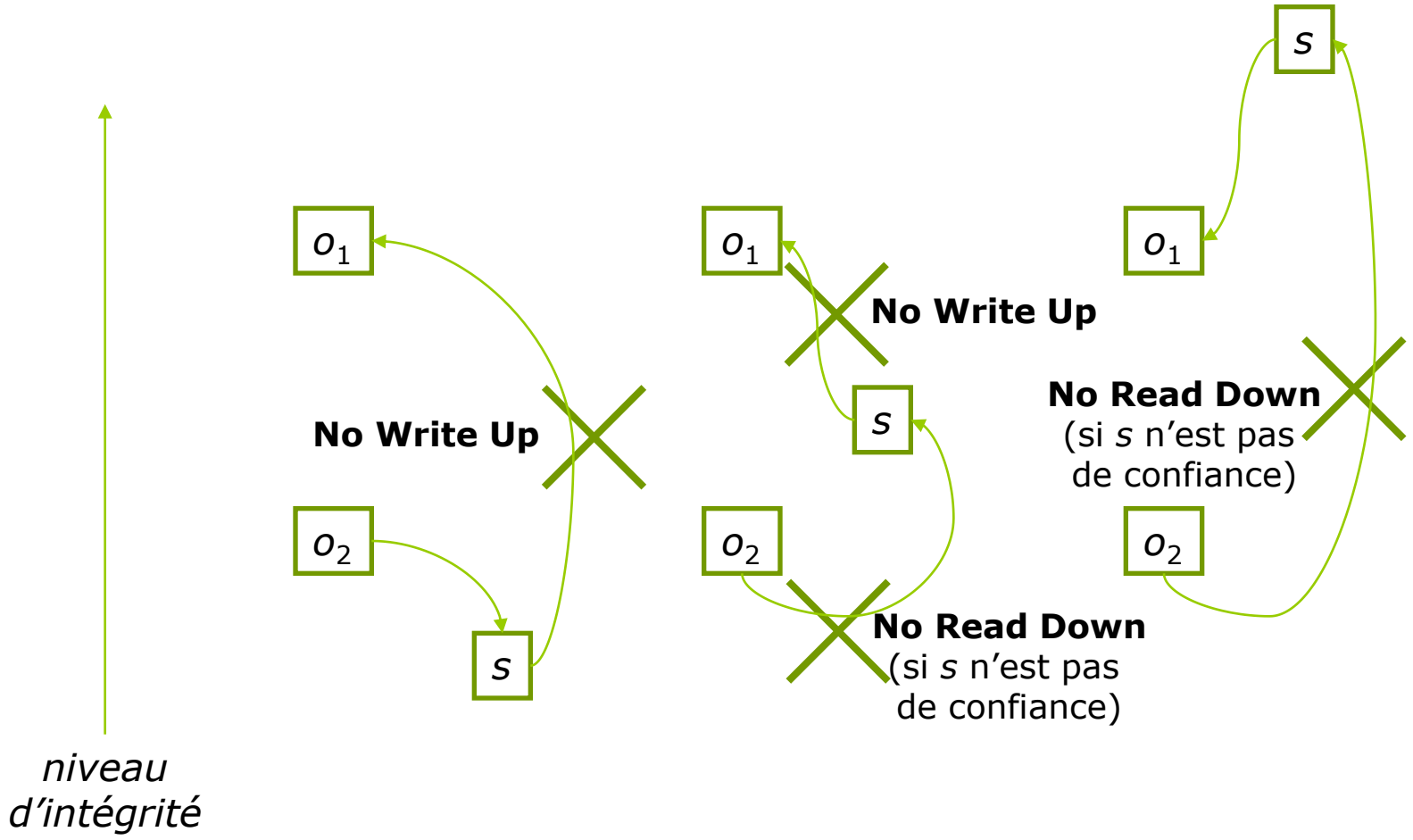
Classification des utilisateurs

- Chaque utilisateur reçoit un niveau d'intégrité.
- Ce niveau traduit la **confiance** faite à un utilisateur pour insérer, modifier ou supprimer de l'information.
- Les sujets activés par un utilisateur reçoivent le même niveau de secret que celui-ci.

Politique de sécurité

- La politique de sécurité du modèle de Biba se résume dans les deux principes :
  - **No Write Up Integrity**
    - préserve de l'écriture d'une information dans un objet par un sujet de niveau d'intégrité inférieur.
  - **No Read Down Integrity**
    - préserve d'un transfert d'information d'un objet vers un autre objet de niveau d'intégrité supérieure par un utilisateur qui n'est pas de confiance.

No Read Down / No Write Up



Modèles obligatoires : avantages et inconvénients

- Il sont bien adaptés aux applications où la protection du secret et de l'intégrité est primordiale.
- Mais ils sont trop rigides et centralisés:
  - La confidentialité est assurée au détriment de la disponibilité.

Contrôle d’accès en SQL

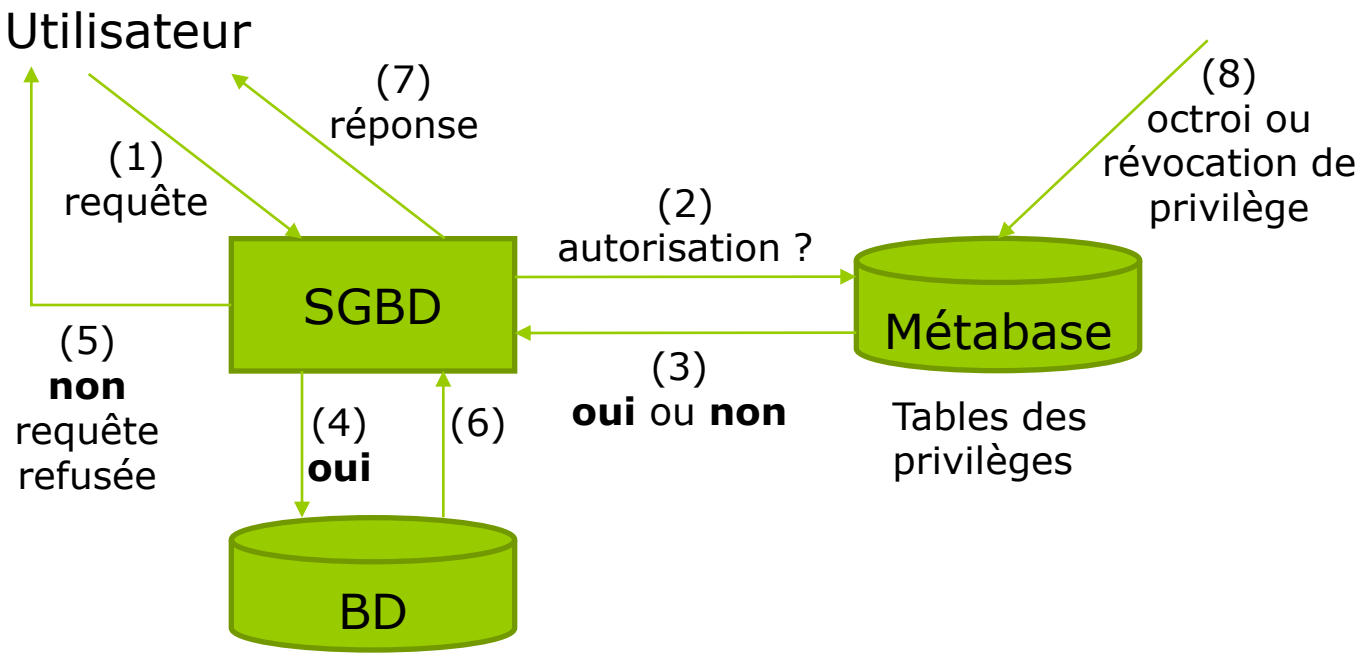
Sujets, objets, privilèges et rôles

- Sujets
  - utilisateurs, groupe d'utilisateurs, tous les utilisateurs
- Objets
  - BD, tables, vues, index, procédures…
- Privilèges (ou autorisations)
  - sur les tables,
  - sur le schéma,
  - sur la base de données…
- Rôles
  - groupes de privilèges

Modèle de contrôle d'accès

- Le modèle de contrôle d'accès SQL est un modèle discrétionnaire :
  - Le créateur d'un objet possède tous les privilèges sur cet objet.
  - Pour pouvoir accéder à un objet qu'il n'a pas créé, un utilisateur doit avoir reçus les privilèges nécessaires.

Processus de contrôle



Création des utilisateurs

- La création des utilisateurs est du ressort de l'administrateur de la base de données qui peut créer :
  - des utilisateurs,
  - des groupes d'utilisateurs.
- SQL2 ne spécifie pas de commande pour la création des utilisateurs :
  - cette commande est donc spécifique à chaque SGBD.

Privilèges

- On distingue :
  - les **privilèges objets** qui concernent des opérations précises sur des tables, des vues, des procédures stockées…, dont le nom est spécifié ;
  - les **privilèges systèmes** qui concernent des opérations sur tous les objets d'un certaine catégorie.
- Ces privilèges varient sensiblement d'un SGBD à l'autre.

Privilèges objets (1)

- SELECT SELECT(c_1, …, c_n)
  - pour pouvoir lire le contenu de toutes ou de certaines colonnes d'une table.
- INSERT INSERT(c_1, …, c_n)
  - pour pouvoir insérer une valeur dans toutes ou certaines colonnes d'une table.
- UPDATE UPDATE(c_1, …, c_n)
  - pour pouvoir modifier le contenu de toutes ou de certaines colonnes d'une table.
- DELETE
  - pour pouvoir supprimer des lignes d'une table.

Privilèges objets (2)

- REFERENCES REFERENCES(C_1, …, C_n)
  - pour pouvoir faire référence à une table ou à certaines colonnes d'une table dans une contrainte d'intégrité.
- TRIGGER
  - pour pouvoir placer un trigger sur une table.
- USAGE
  - pour pouvoir manipuler les domaines.
- EXECUTE
  - pour pouvoir exécuter une procédure stockée.

Privilèges système

- ORACLE, par exemple, propose un jeu très complet de privilèges systèmes (plus d'une centaine) qui permettent de réaliser des opérations sur tous les objets d'une certaine catégorie :
- Par exemple :
  - `INSERT ANY TABLE`
  - `DELETE ANY INDEX`

Octroi ou révocation de privilèges

- SQL offre deux commandes pour octroyer ou révoquer des privilèges :
  - `GRANT`
  - `REVOKE`

GRANT

- Syntaxe

  `GRANT` (*liste de privilèges* | `ALL`)
  `ON` *liste d'objets*
  `TO` *liste d'utilisateurs* | `PUBLIC`
  [`WITH GRANT OPTION`]

  avec :

  - `ALL`
    - tous les privilèges que le donneur peut accorder
  - `PUBLIC`
    - tous les utilisateurs connus du système
  - `WITH GRANT OPTION`
    - indique que le receveur pourra transmettre les privilèges qui lui sont octroyés

REVOKE

- Syntaxe

  REVOKE [GRANT OPTION FOR] (*privilèges* | ALL)
  ON *liste d'objets*
  FROM *liste d'utilisateurs*
  [{RESTRICT | CASCADE}]

  avec :

  - CASCADE
    - la révocation concerne les utilisateurs cités dans la clause FROM ainsi que ceux à qui ces privilèges ont été récursivement transmis.
  - RESTRICT
    - la révocation ne concerne que les utilisateurs cités dans la clause FROM.
  - GRANT OPTION FOR
    - ce n'est pas les privilèges qui sont révoqués, mais le droit de le transmettre.

Règle d'octroi des privilèges

- Un utilisateur ne peut octroyer que les privilèges qu'il possède.

Règles de révocation des privilèges

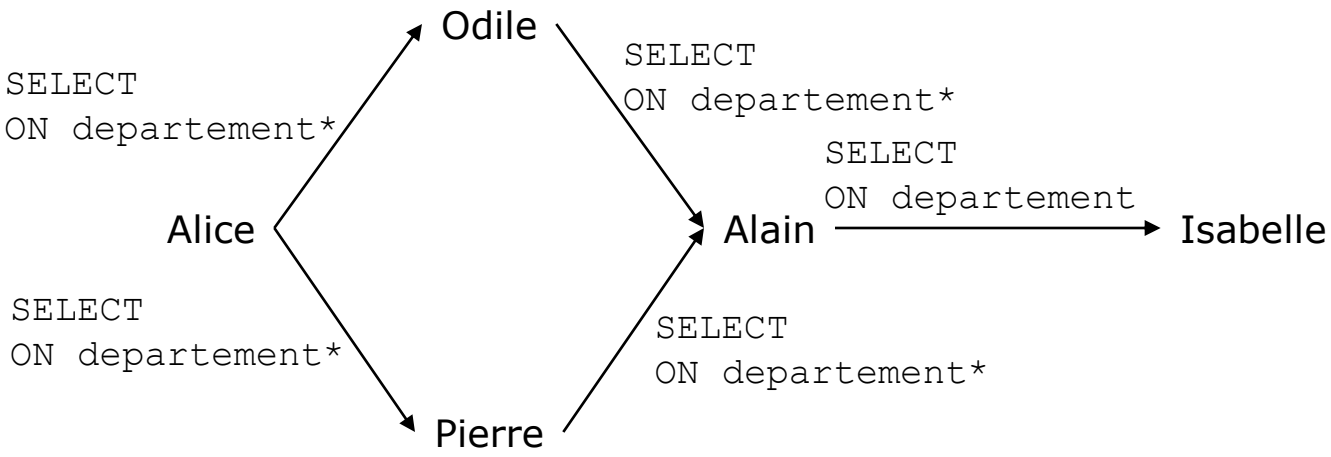
- Un utilisateur ne peut révoquer que les privilèges qui lui ont été transmis.
- Si l'option `CASCADE` est spécifiée, la révocation d'un privilège est récursive.
- Si l'option `RESTRICT` est spécifiée, la révocation d'un privilège à un utilisateur n'est possible que si celui-ci n'a pas transmis ce privilège à un autre utilisateur.
- Si un utilisateur a reçu un privilège de plusieurs utilisateurs, il ne perd ce privilège que si tous ces utilisateurs le lui retirent.

Graphe d'octroi des privilèges

- C'est un outil pratique pour gérer la propagation des privilèges et leur révocation.
  - chaque nœud représente un utilisateur, il est étiqueté par le nom de cet utilisateur,
  - chaque arc représente un octroi de privilège, il est étiqueté par ce privilège.

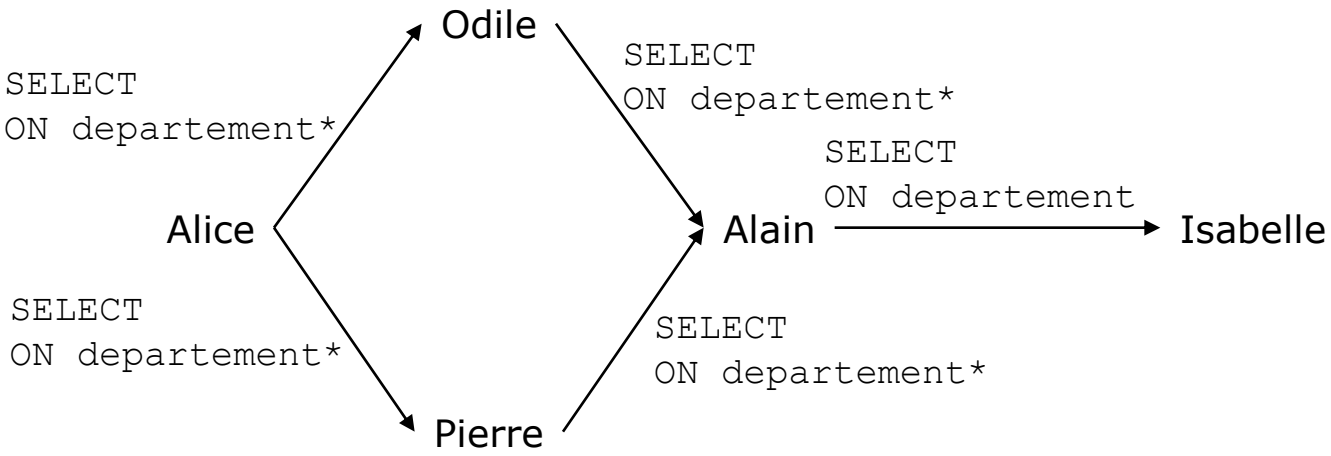
Octroi de privilèges : exemple

Alice : `GRANT SELECT ON departement TO odile WITH GRANT OPTION;`
Alice : `GRANT SELECT ON departement TO pierre WITH GRANT OPTION;`
Odile : `GRANT SELECT ON departement TO alain WITH GRANT OPTION;`
Pierre : `GRANT SELECT ON departement TO alain WITH GRANT OPTION;`
Alain : `GRANT SELECT ON departement TO isabelle;`

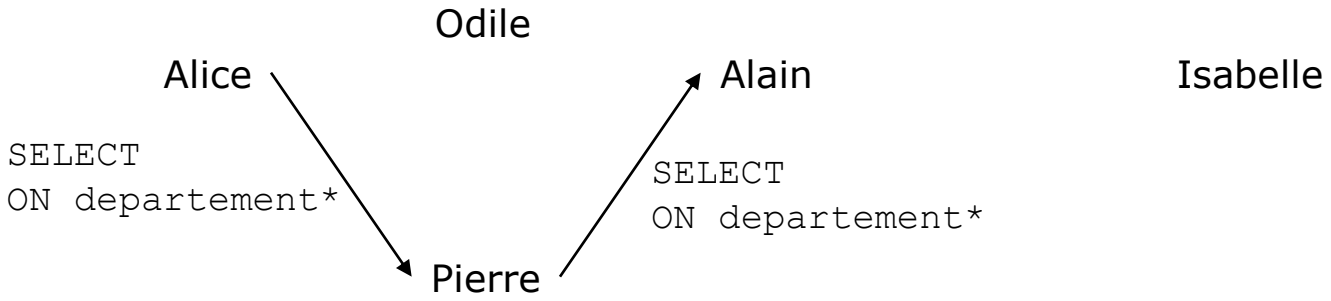


* indique que le privilège a été accordé `WITH GRANT OPTION`.

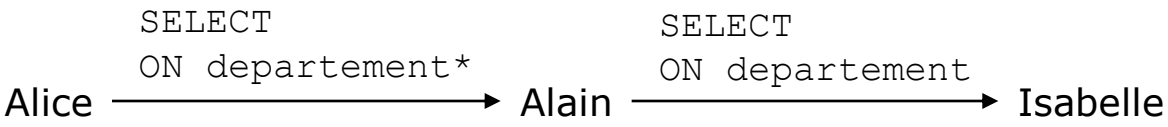
Révocation en cascade : exemple



Alice : `REVOKE SELECT ON département FROM odile CASCADE`



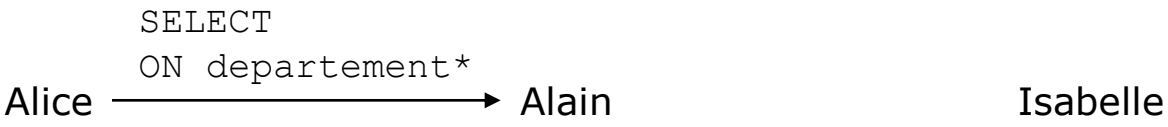
Révocation avec restriction : exemple



Alice : `REVOKE SELECT ON département FROM alain RESTRICT`

Interdit !

Alain : `REVOKE SELECT ON departement FROM isabelle RESTRICT`



Importance des vues

- Les vues jouent un rôle important car elles permettent de définir de façon précise les portions d'une BD sur lesquelles des privilèges sont accordés.
- Par exemple, si la BD est formée de la table :
  - `employe(nom, departement, salaire)`

  on pourra restreindre l'accès à l'affectation ou au salaire d'un employé en définissant les deux vues suivantes :
  - `create view affectation_employe(nom, departement) as select nom, departement from employe`
  - `create view salaire_employe(nom, salaire) as select nom, salaire from employe`

  et en accordant des privilèges sur chacune de ces deux vues, par exemple :
  - `grant update on salaire_employe to pierre`

Exemple de mise en place des contrôles d'accès sur une base de données

- On considère une BD décrivant une petite entreprise :
  - La BD décrit :
    - les employés : nom, salaire, nom du département,
    - les départements : nom, nom du responsable.
  - Les utilisateurs de la BD sont ses employés :
    - Alice est l'administratrice de la BD,
    - Odile est la directrice du personnel,
    - Pierre est l'agent comptable,
    - Alain est responsable du département Informatique,
    - Isabelle est une employé du département Informatique.

Exemple : création des utilisateurs

- Alice, l'administratrice, crée la BD :

```
createdb ma_petite_entreprise
```

- Alice se connecte à la BD :

```
CONNECT TO ma_petite_entreprise USER DBA;
```

- Alice crée les utilisateurs de cette BD:

```
CREATE USER alain;
CREATE USER isabelle;
CREATE USER odile;
CREATE USER pierre;
```

Exemple : création des tables

- Alice crée deux tables décrivant les employés et les départements:

```
CREATE TABLE employe (
 nom VARCHAR(20) PRIMARY KEY,
 nom_dept VARCHAR(20),
 salaire FLOAT);
CREATE TABLE departement (
 nom VARCHAR(20) PRIMARY KEY,
 responsable VARCHAR(20)
  REFERENCES employe(nom));
ALTER TABLE employe ADD FOREIGN KEY (nom_dept)
REFERENCES departement(nom);
```

Exemple : création des vues

- Alice crée une vue qui cache les salaires des employés :

```
CREATE VIEW affectation(nom, nom_dept) AS
 SELECT e.nom, e.nom_dept FROM employe e;
```

- Alice crée une vue qui donne le nom et le salaire de chaque employé du département dont l'utilisateur de cette vue est le responsable :

```
CREATE VIEW mon_employe(nom, salaire) AS
 SELECT e.nom, e.salaire
 FROM employe as e
 WHERE e.nom_dept IN
       (SELECT nom
        FROM departement
        WHERE responsable = USER);
```

Exemple : octroi des privilèges

- Alice transmet à tous les utilisateurs le privilège de consulter les affectations :

```
GRANT SELECT ON affectation TO PUBLIC;
```

- Alice transmet à Alain, responsable du département Informatique, le privilège de consulter le salaire des employés de ce département.

```
GRANT SELECT ON mon_employe TO alain;
```

Exemple : octroi des privilèges

- Alice transmet à Odile, la directrice du personnel, les privilèges de consulter et de modifier le département dans lequel travaille un employé et le responsable d'un département.

```
GRANT SELECT, UPDATE(nom_dept)
ON employe TO odile;
GRANT SELECT, UPDATE(responsable)
ON departement TO odile;
```

- Alice transmet à Pierre, l'agent comptable, les privilèges de consulter et de modifier le salaire des employés.

```
GRANT SELECT, UPDATE(salaire)
ON employe TO pierre;
```

Exemple : matrice d'accès

	Employé	*Département*	*Affectation*	*Mon_employé*
Alain			`SELECT`	`SELECT`
Alice	*tous les privilèges*	*tous les privilèges*	*tous les privilèges*	*tous les privilèges*
Isabelle			`SELECT`	
Odile	`UPDATE( nom_dept)`	`UPDATE( responsable)`	`SELECT`	
Pierre	`UPDATE(salaire)`		`SELECT`	

Exemple : effets sur les requêtes

	`SELECT *`
	`FROM affectation`
		`UPDATE employe`
		`SET salaire = 2000`
		`WHERE nom =`
		`'isabelle'`
			`UPDATE departement`
			`SET responsable =`
			`'isabelle'`
			`WHERE nom =`
			`'informatique'`

alain	acceptée	refusée	refusée
alice	acceptée	acceptée	acceptée
isabelle	acceptée	refusée	refusée
odile	acceptée	acceptée	acceptée
pierre	acceptée	acceptée	refusée

Exemple : révocation de privilèges

- Alice révoque le privilège de Pierre de modifier le salaire d'un employé :

```
REVOKE UPDATE(salaire)
ON employé
FROM pierre;
```

Pierre possède encore le privilège de consulter les affectations.

Rôles

- Un **rôle** est un ensemble de privilèges.
- Les rôles sont assignés aux utilisateurs qui peuvent avoir plusieurs rôles.
- Les rôles sont organisés hiérarchiquement.
- Il ne faut pas confondre rôle et utilisateur :
  - un utilisateur peut être propriétaire d'un objet, un rôle ne le peut pas.

Gestion des rôles en SQL Oracle

- Création d'un rôle

  `CREATE ROLE` *nom de rôle*

- Octroi d'un privilège à un rôle

  `GRANT` *liste de privilèges*

  `ON` *liste d'objets*

  `TO` *liste de rôles ou d'utilisateurs*

- Octroi d'un rôle à un rôle ou à un utilisateur

  `GRANT` *liste de rôles*

  `TO` *liste de rôles ou d'utilisateurs*

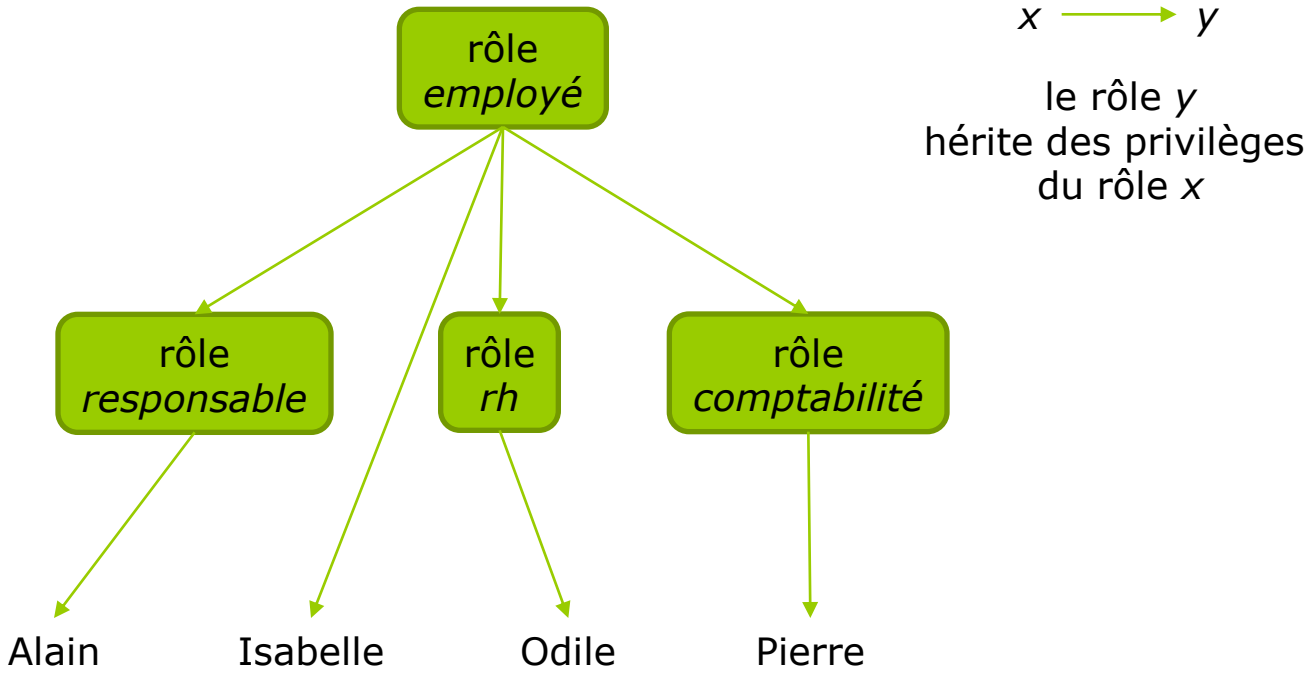
- Activation d'un rôle

  `SET ROLE` *liste de rôles*

Exemple : les rôles

- Les 4 rôles de la BD *ma_petite_entreprise*:
  - *employé* :
    - privilège de consulter les affectations
  - *responsable* (de département) :
    - privilège *employé*
    - privilège de consulter les salaires de ses employés
  - *srh (service des ressources humaines)* :
    - privilège *employé*,
    - privilège de modifier le département dans lequel travaille un employé et le responsable d'un département.
  - *agent_comptable* :
    - privilège *employé*
    - privilège de mettre à jour le salaire des employés

Exemple : graphe d'octroi des rôles



Exemple : création du rôle *employé*

- `CREATE ROLE employe;`
- `GRANT SELECT`
  `ON affectation`
  `TO employe;`
- `GRANT employe`
  `TO isabelle;`

Exemple : création du rôle *responsable*

- `CREATE ROLE responsable;`
- `GRANT employe TO responsable;`
- `GRANT SELECT`
  `ON mon_employe`
  `TO responsable;`
- `GRANT responsable TO alain;`

Exemple : création du rôle *srh*

- `CREATE ROLE srh;`
- `GRANT employe TO srh;`
- `GRANT UPDATE(nom_dept)`
  `ON employe`
  `TO srh;`
- `GRANT UPDATE(responsable)`
  `ON departement`
  `TO srh;`
- `GRANT srh TO odile;`

Exemple : création du rôle *comptabilité*

- `CREATE ROLE comptabilite;`
- `GRANT employe TO comptabilite;`
- `GRANT UPDATE(salaire)`
  `ON employe`
  `TO comptabilite;`
- `GRANT comptabilite TO pierre;`

Privilèges en PostgreSQL

- Les privilèges sur une table ne peuvent pas être restreints à certaines des colonnes de cette table.
- Pour placer un privilège `INSERT` ou `UPDATE` sur certaines colonnes d'une table *T*, il faut :
  1. construire une vue de cette table restreinte :
     - aux colonnes à mettre à jour,
     - aux colonnes permettant de sélectionner les lignes à mettre à jour ;
  2. créer une règle `ON INSERT` ou `ON UPDATE`, déclenchant les mises à jour sur la table *T*.

Utilisateurs et rôles en PostgreSQL version 8

- Le concept d'utilisateur est subsumé par celui de rôle.
- Un rôle est une entité qui peut posséder des objets ou bien des privilèges sur des objets.
- On distingue :
  - les **rôles de connexion**, qui possèdent un login et peuvent se connecter à la BD,
  - les **rôles de groupe**, qui ne possèdent pas de login.

Exercice : gestion des notes du M2 (1)

- La base de données contient les informations concernant :
  - les étudiants et les enseignants :
    - nom, prénom, e-mail, adresse, téléphone, nom de la personne à prévenir en cas d'accident ;
  - les UE :
    - code, intitulé, responsable, enseignants, nombre d'ECTS.
  - les notes obtenues par chaque étudiant dans chaque UE :
    - note réelle,
    - note finale ;
  - les notes obtenues par chaque étudiant au M2 :
    - note réelle,
    - points de jury,
    - note finale,
    - mention.

Exercice : gestion des notes du M2 (2)

- Les utilisateurs de la base de données sont :
  - le secrétaire,
  - les étudiants,
  - les enseignants parmi lesquels :
    - le responsable du M2,
    - les responsables des UE.

Exercice : gestion du M2 (3)

- L'administrateur de la base de données a l'autorisation de créer les tables et de transmettre les privilèges sur ces tables.
- Le secrétaire a l'autorisation de créer et de modifier les informations concernant :
  - les étudiants et les enseignants : nom, e-mail, adresse, téléphone, personne à prévenir en cas d'accident…
  - les UE : code, responsable, intervenants, intitulé, ECTS…
- Le responsable d'une UE a l'autorisation d'ajouter ou de modifier la note réelle d'un étudiant dans cette UE.
- Le responsable du M2 a l'autorisation de créer ou modifier la note finale d'un étudiant à une UE et au M2. Il dispose de points de jury accordés par les membres du jury (les enseignants) qu'il répartit entre les différentes UE.

Exercice : gestion du M2 (3)

- Tous les utilisateurs ont l'autorisation de consulter :
  - les informations qui les concernent,
  - les noms, prénoms et e-mails des étudiants et des enseignants,
  - les informations concernant les UE : code, intitulé, enseignants, ECTS, …
  - les notes finales.
- Le secrétaire a l'autorisation de consulter les informations personnelles des étudiants et des enseignants (adresse, téléphone, personne à prévenir en cas d'accident…).
- Les étudiants ont l'autorisation de consulter leurs propres notes réelles et leurs points de jury.
- Les enseignants ont l'autorisation de consulter les notes réelles et les points de jury.
- **Tout ce qui n'est pas autorisé est interdit !**