Contrôle d'accès pour BD relationnelles

- Introduction
- Modèle discrétionnaire : Discretionary Access Control (DAC)
- Gestion des vues
- Modèle basé sur les rôles : Role-Based Access Control (RBAC)

Objectifs des SGBD

Système de gestion de bases de données

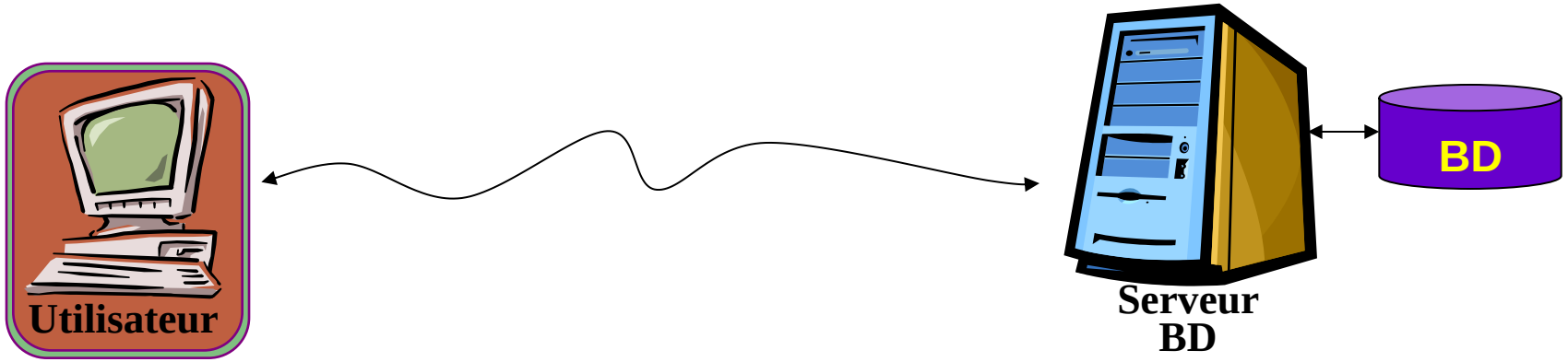
- I- Indépendance Physique
- II- Indépendance Logique
- III – Langage de manipulation
- IV - Gestion des vues
- V - Optimisation des questions
- VI - Gestion de la cohérence
- VII - Gestion des pannes
- VIII - Concurrence d'accès
- IX - Gestion de la confidentialité
- X - Standards

BD

Confidentialité

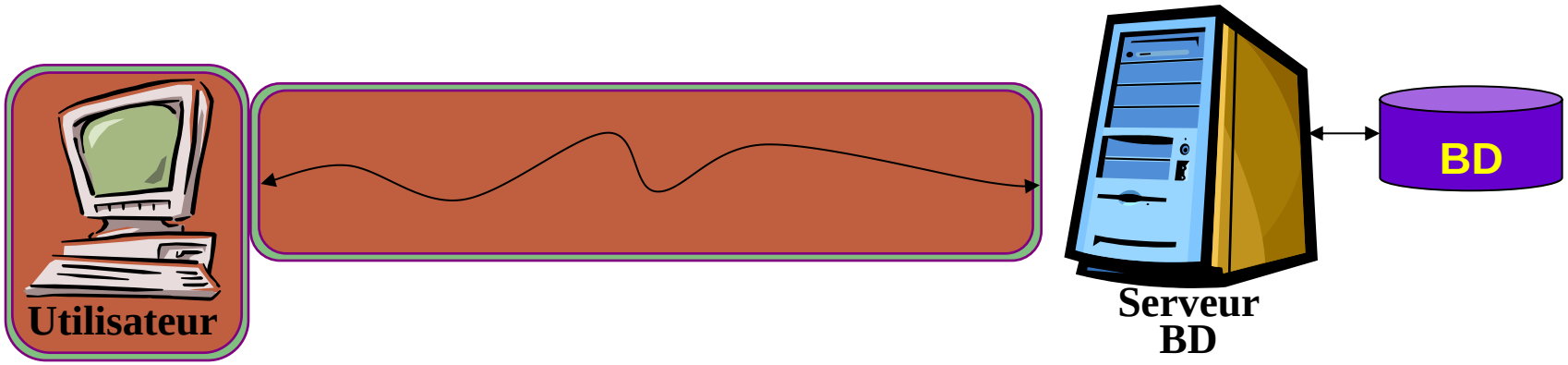
- **Objectif : Protéger les données de la BD contre des accès non autorisés**
- Deux niveaux :
  - Connexion restreinte aux **usagers répertoriés** (mot de passe)
  - **Privilèges** d'accès aux objets de la base
- Usagers : Usager ou groupe d'usagers
- Objets : Relation, **Vue**, autres objets (procédures, etc.)

T1 : Identification/authentification



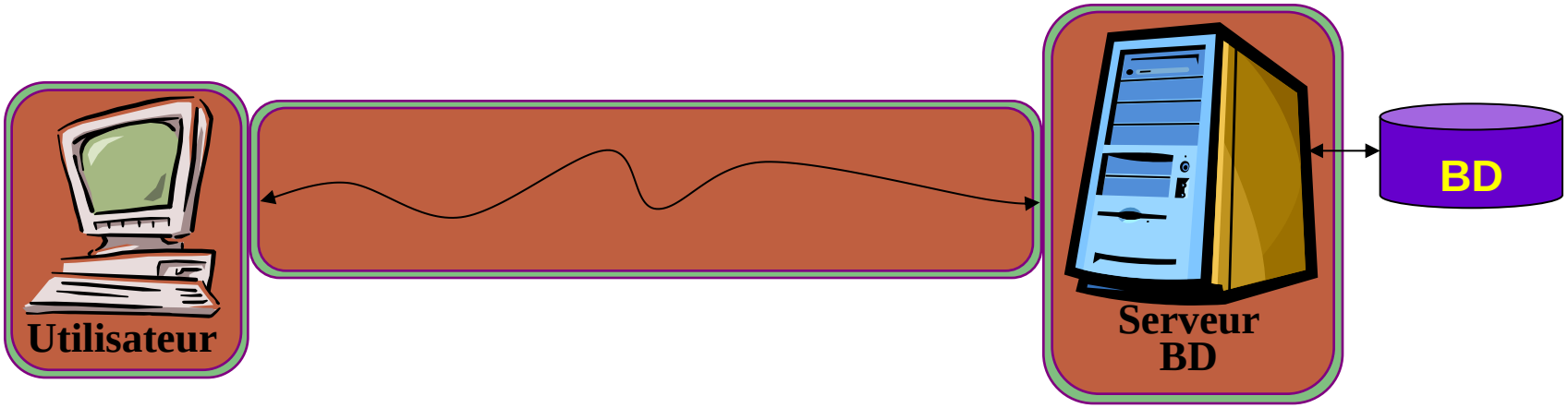
- Au minimum : login (identification) + password (authentification)
- Assuré par le SGBD et/ou l'OS et/ou l'application
- Authentification forte :
  - Conjonction de 2 éléments d'authentification distincts parmi :
    - Ce que l'entité connaît : password, pin code, etc
    - Ce que l'entité détient : carte à puce, token, badge RFID, etc
    - Ce que l'entité est : empreinte biométrique

T2 : Chiffrement des communications



- Technologie éprouvée (ex: SSL)
- Assure la confidentialité des messages
- Techniques cryptographiques complémentaires
  - Hachage (ex: SHA) : intégrité des messages
  - Signature (ex: via PKI) : authentification et non répudiation du message

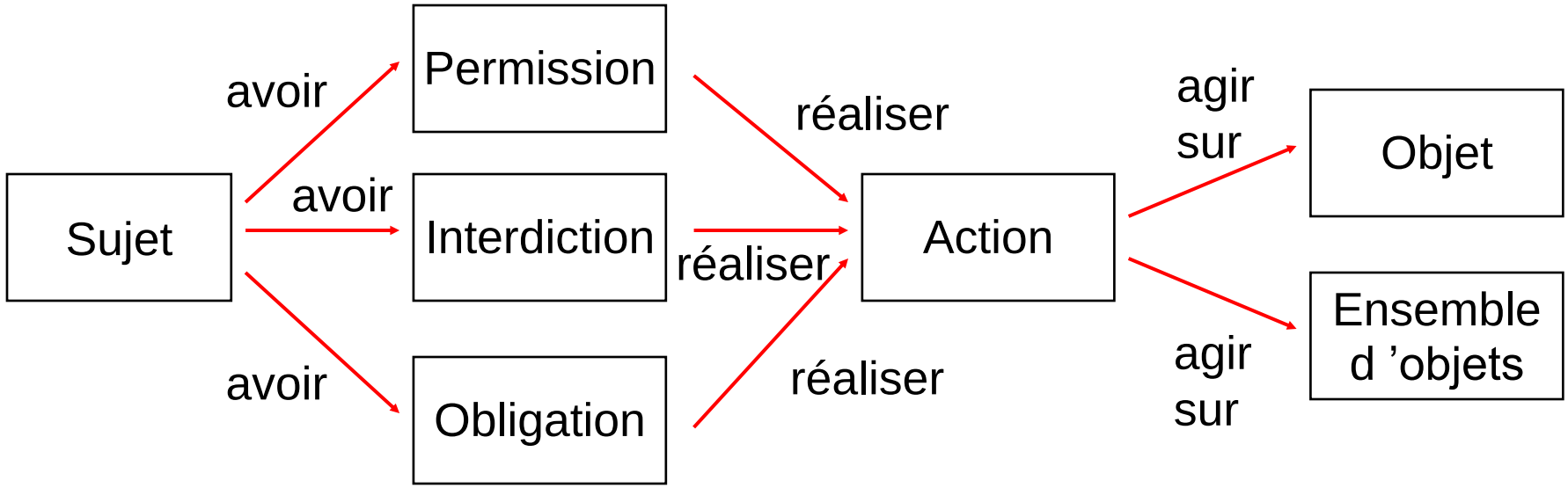
T3 : Mécanismes de contrôle d'accès



- Contrôle d'accès sophistiqué dans les SGBD
  - Autorisations affectées à des utilisateurs ou rôles
  - Peut porter sur des objets d'une granularité variée : tables, vues, procédures stockées …

Politique de contrôle d'accès = ensemble de règles

- Précise qui est autorisé à faire quoi sur quelles données et dans quelles conditions
- Format des règles :



Modèle discrétionnaire (DAC)

- DAC = Discretionary Access Control
  - Contrôle d'accès discrétionnaire
- Principes de DAC
  - Le créateur d'un objet fixe la politique de contrôle d'accès sur cet objet
  - Les sujets reçoivent des permissions pour réaliser des actions sur des objets
  - Les sujets ont l'autorisation de transférer certaines permissions à d 'autres sujets
  - Modèle par essence décentralisé
    - ⇗ très souple
    - ⇘ difficile à administrer

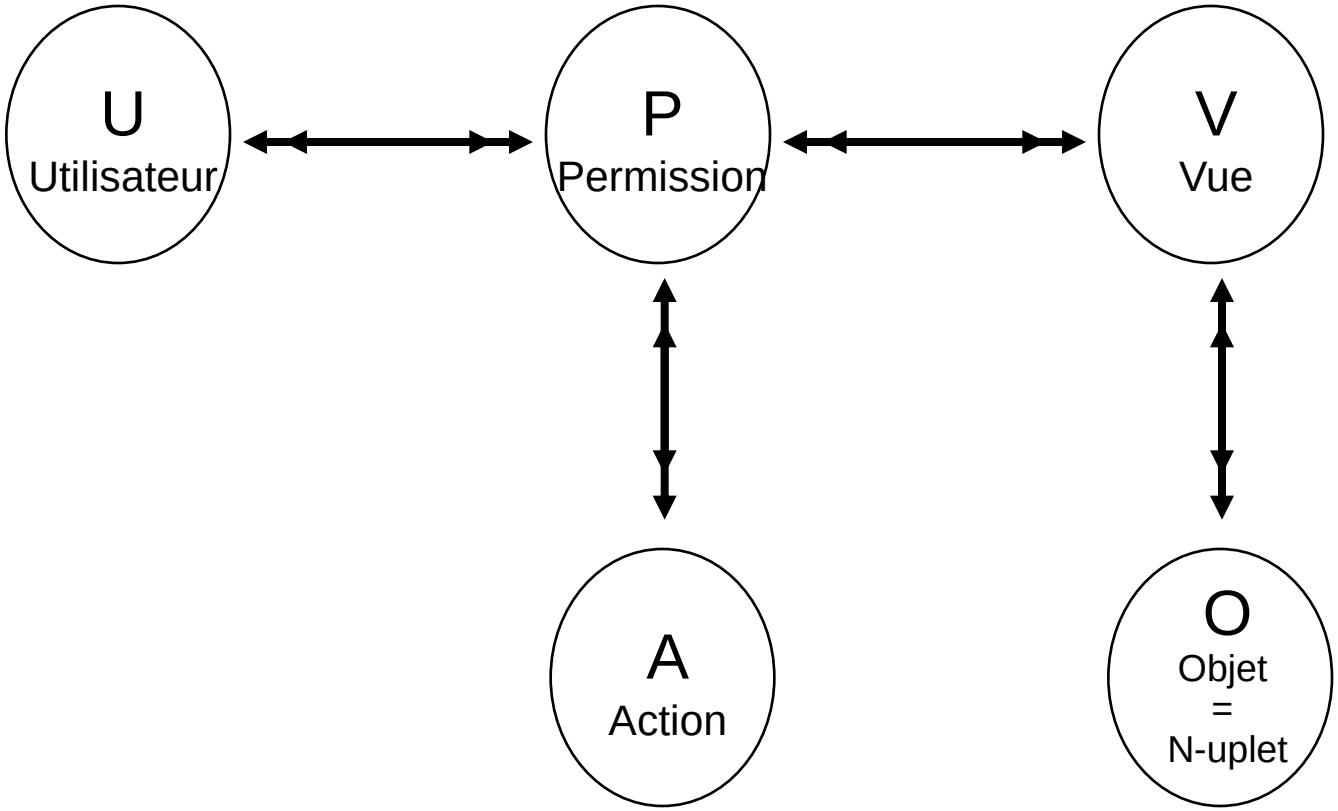
Modèle discrétionnaire (DAC)

- Exemple de matrice d'accès
  - La matrice peut être immense

	Nom	Salaire	. . .
Dupont	read write	read	
Robert			
Durand	read		

- Deux approches pour renseigner la matrice accès
  - Capacité (Capability List)
    - la matrice est gérée par ligne
    - une liste d'autorisations, appelée capability list, est affectée à chaque utilisateur
  - ACL (Access Control List)
    - la matrice est gérée par colonne
    - une liste d'autorisations est affectée à chaque objet

Principe du modèle DAC proposé par SQL



Remarque : modèle fermé, basé exclusivement sur des autorisations

Commandes SQL Grant

GRANT <liste privileges>

ON <table ou vue>

TO <liste utilisateurs>

[WITH GRANT OPTION] ;

– WITH GRANT OPTION

- est optionnel
- signifie que l ’utilisateur qui obtient le privilège peut ensuite accorder ce privilège à un autre utilisateur

Privilèges SQL

- Principaux privilèges
  - SELECT : permet la consultation de la table
  - INSERT : permet l ’insertion de nouvelles données dans la table
  - UPDATE : permet la mise à jour de n ’importe quelle colonne de la table
  - UPDATE(nom_colonne) : permet la mise à jour d ’une colonne spécifique de la table
  - DELETE : permet de supprimer n’importe quelle donnée de la table
- S’applique aussi aux fonctions d’administration
  - CREATE/ALTER/DROP TABLE : Modifier la définition d’un objet
  - EXECUTE : Compiler et exécuter une procédure utilisée dans un programme
  - REFERENCE : référencer une table dans une contrainte
  - INDEX : Créer un index sur une table
  - …

Commande SQL Revoke

REVOKE [GRANT OPTION FOR] <liste privilèges>
ON <table ou vue>
FROM <liste utilisateurs>
[option] ;

- [GRANT OPTION FOR]
  - signifie que seul le droit de transfert est révoqué
- [option] = RESTRICT ou CASCADE
  - Supposons que A accorde le privilège p à B et B accorde ensuite p à C
  - CASCADE : si A révoque p à B alors C perd aussi le privilège
  - RESTRICT : si A révoque p à B alors la révocation s'arrête à B et C garde le privilège reçu de B

Et si un utilisateur U a reçu le privilège p de A et de B (sans relation entre A et B) ?

Exemples GRANT et REVOKE

- GRANT SELECT INSERT
  ON Patients
  TO Alice

- GRANT SELECT DELETE
  ON Prescription
  TO Alice
  WITH GRANT OPTION

- REVOKE UPDATE
  ON Medecins
  FROM Bob CASCADE

- REVOKE GRANT OPTION FOR DELETE
  ON Prescription
  FROM Alice

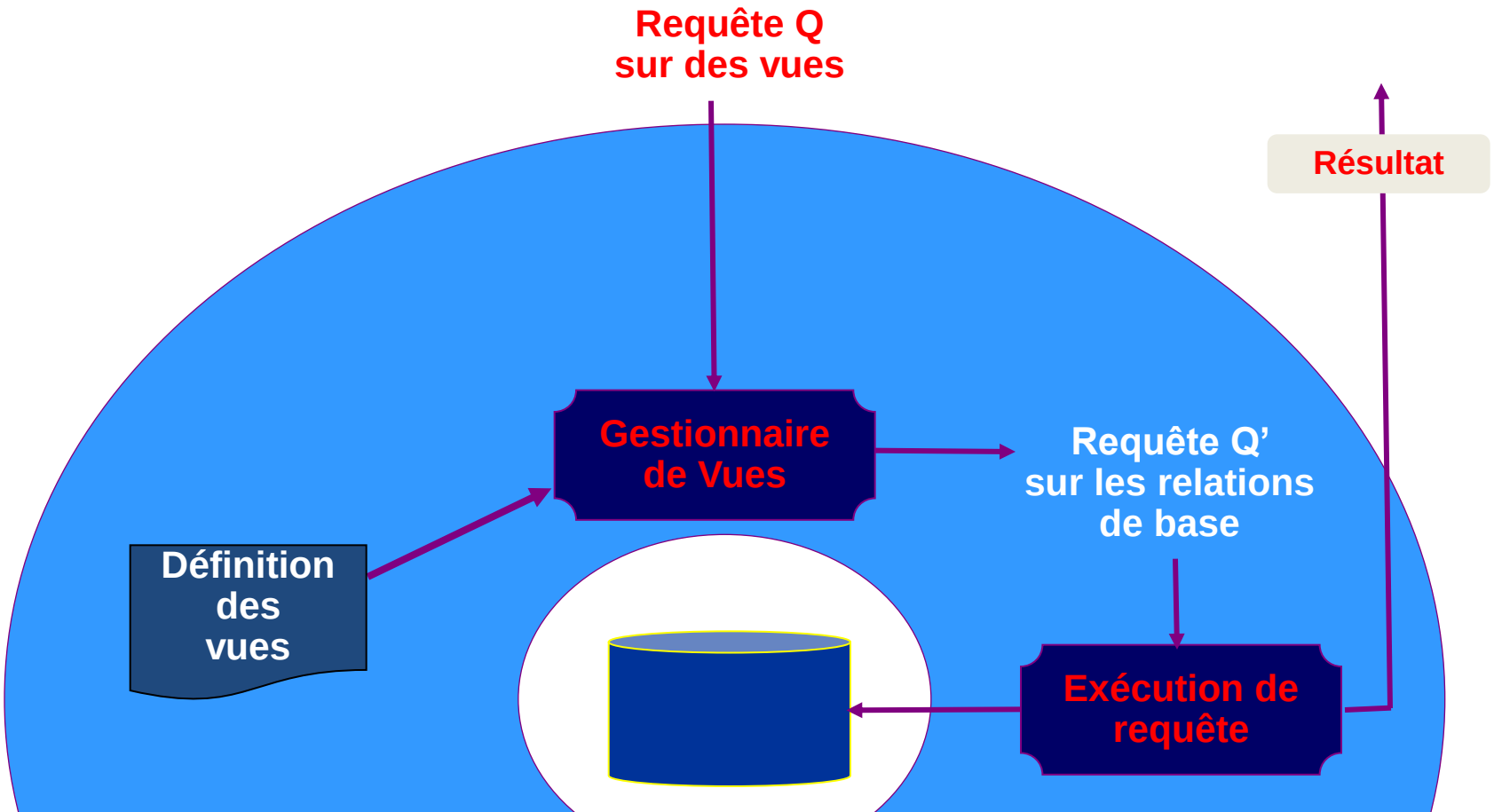
Gestion des vues

- Les vues permettent d'implémenter l'indépendance logique en créant des objets virtuels
- Vue = expression d'un requête SQL
- Le SGBD stocke la définition et non le résultat
- Exemple : la vue du dossier patient

```
CREATE VIEW dossier_patient AS
SELECT *
FROM   dossier_admin DA, dossier_medical DM, dossier_soins_infirmiers DSI
WHERE          DA.id_patient = DM.id_patient AND
               DA.id_patient = DSI.id_patient
```

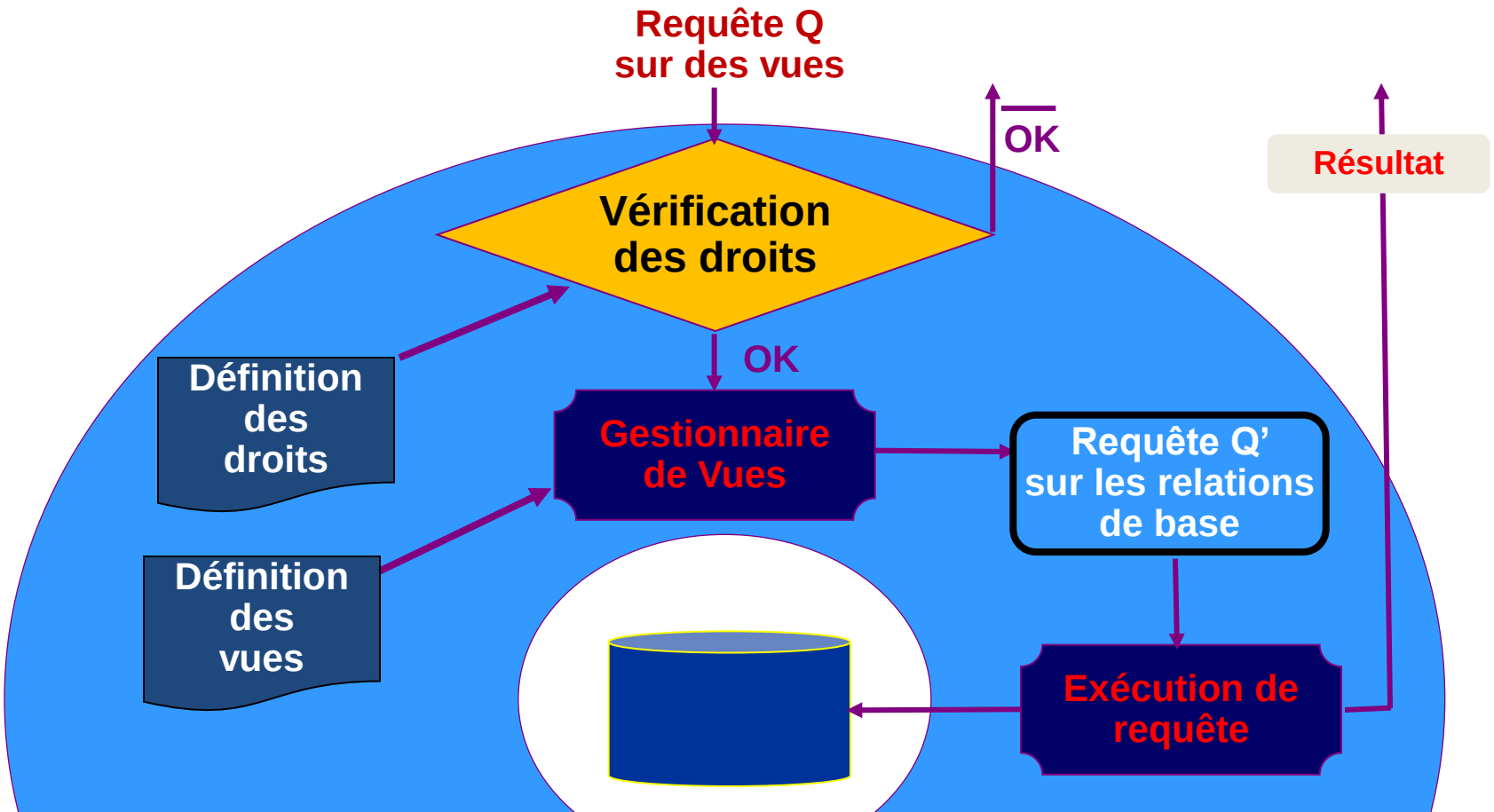
Gestion des vues

Le SGBD transforme la question sur les vues en question sur les relations de base

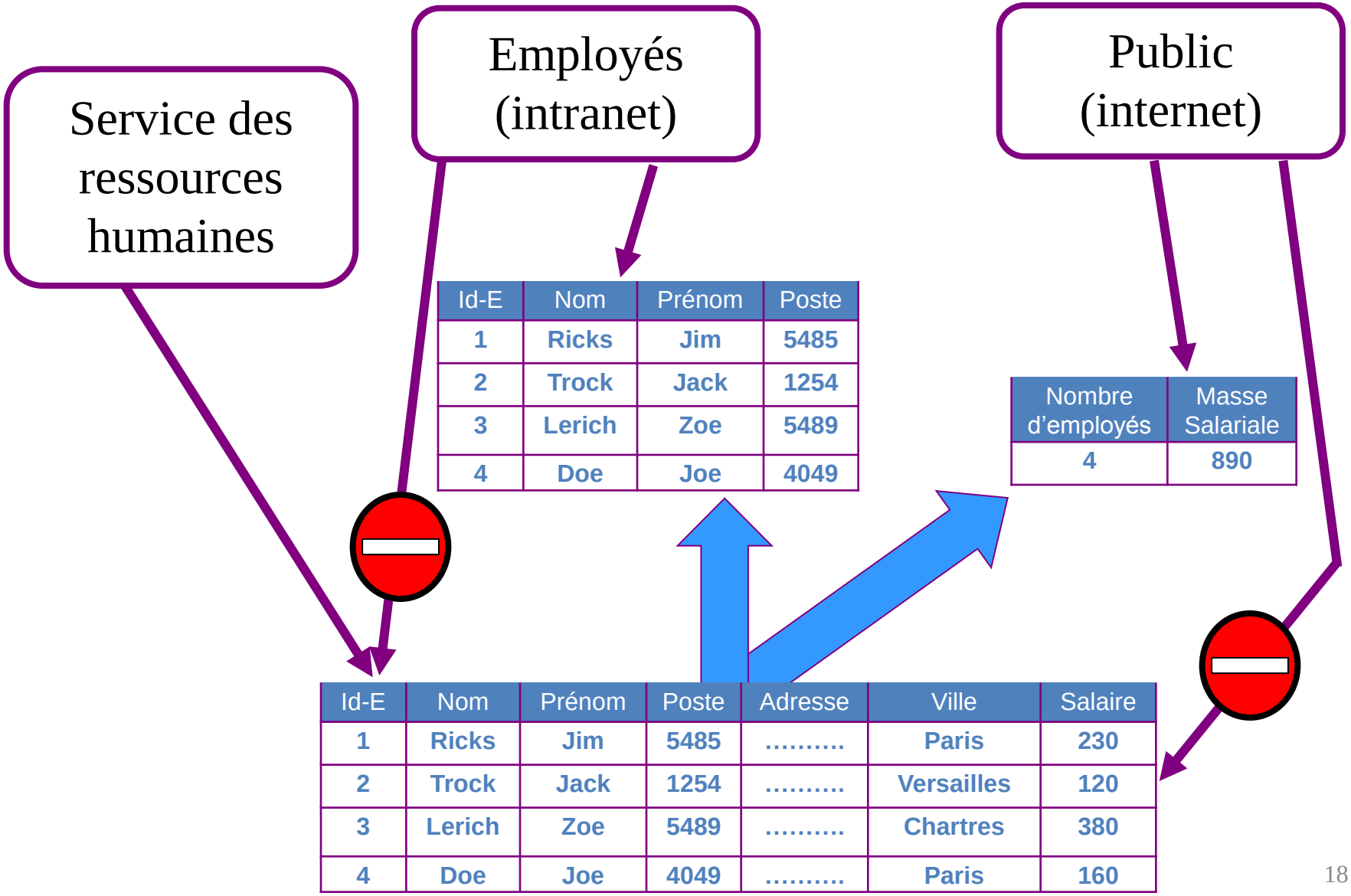


Confidentialité via les vues

Principe : Restreindre l'accès à la BD en distribuant les droits via des vues :

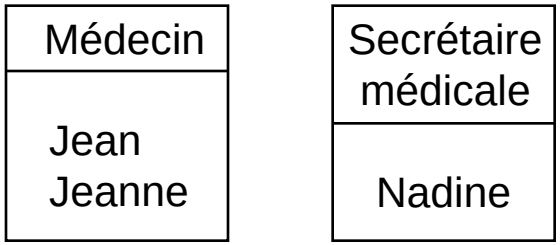


Confidentialité via les vues

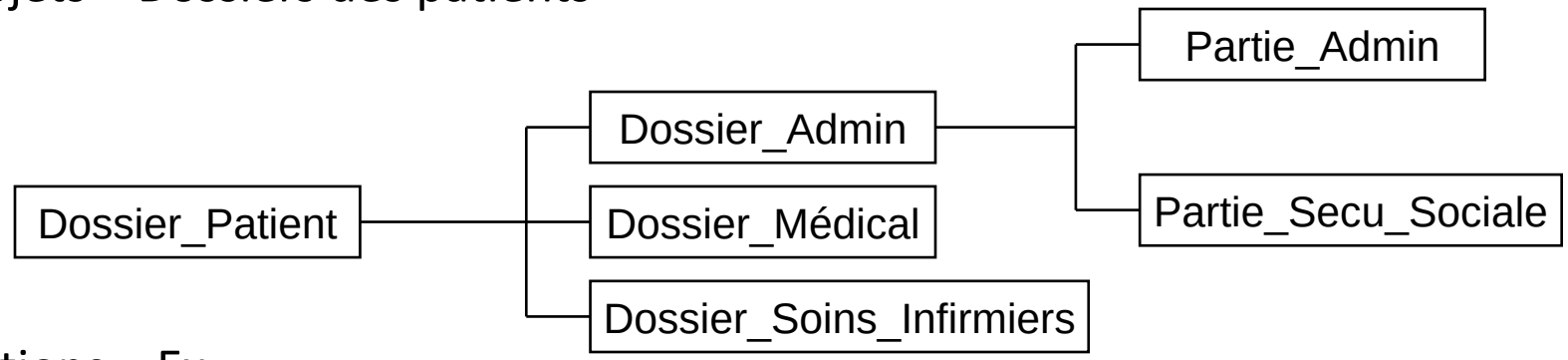


Ex. Système d'information médical

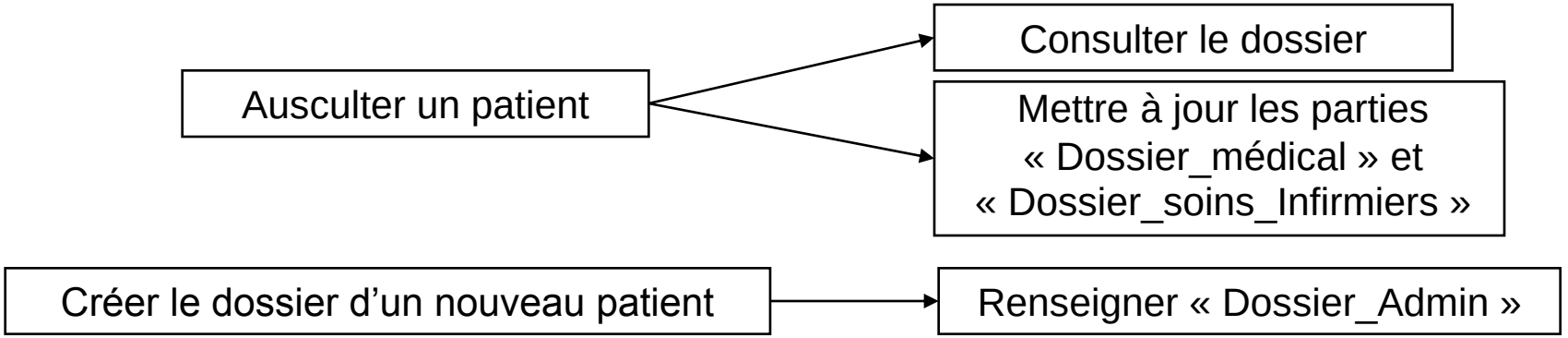
- Sujets = Personnels du groupe médical



- Objets = Dossiers des patients



- Actions = Ex.



Expression des règles (exemples)

- R1 : La secrétaire médicale a la permission de gérer le « Dossier_Admin » des patients du groupe médical (règle simple ne dépendant pas du contenu)

```
GRANT ALL PRIVILEGES
    ON dossier_admin
    TO Nadine ;
```

- R2 : Le médecin a la permission de consulter l'intégralité du dossier de ses propres patients (règle dépendant du contenu)

```
CREATE VIEW dossier_patient_du_medecin AS
        SELECT *
        FROM dossier_patient
        WHERE dossier_patient.medecin_traitant = CURRENT_USER ;
```

(CURRENT_USER : opérateur prédéfini SQL)

```
GRANT SELECT
    ON dossier_patient_du_medecin
    TO Jean, Jeanne ;
```

Expression des règles (exemples)

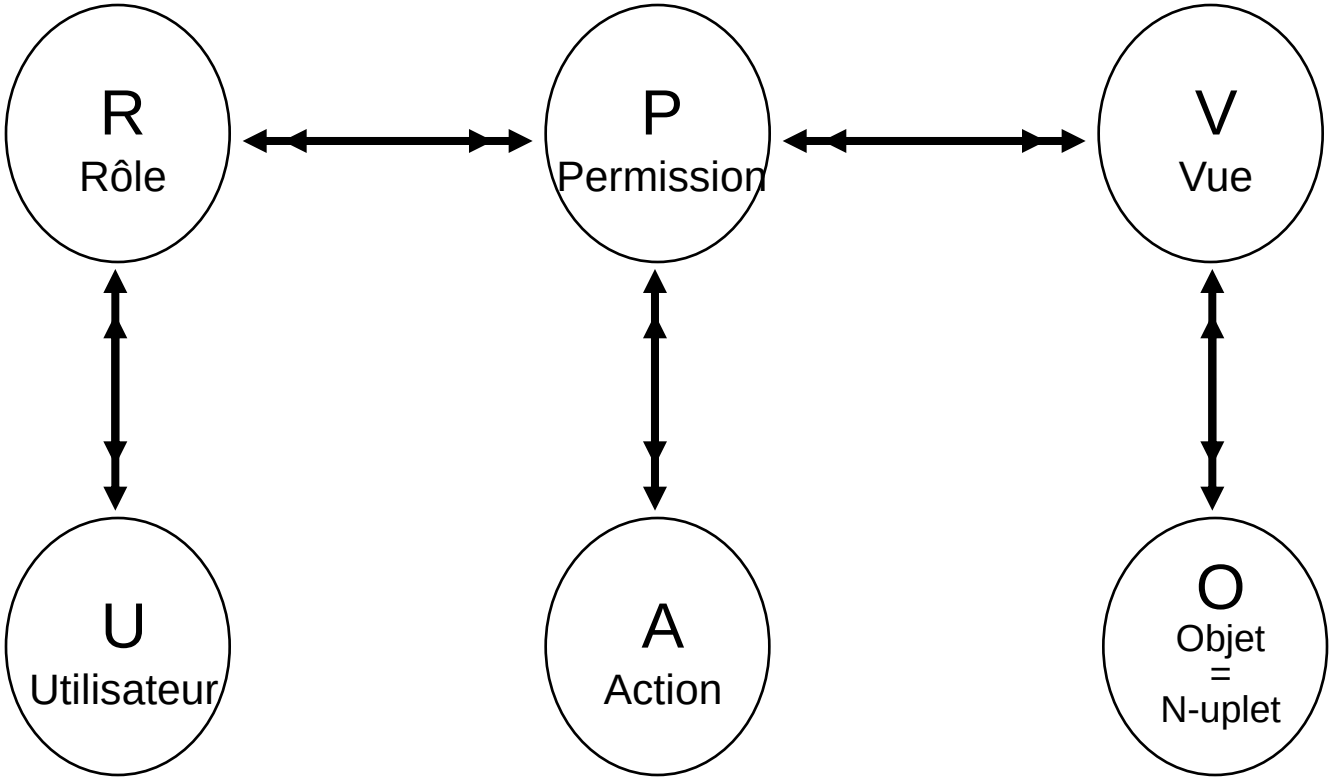
- R3 : les dossiers administratifs ne sont accessibles que pendant les heures ouvrables (règle dépendant du contexte)
  - CREATE VIEW dossier_ouvrable
    AS SELECT * FROM dossier_admin
    WHERE TO_CHAR(SYSDATE,'HH') BETWEEN '08' AND '18'

    en dehors de la période 8H – 18h le predicat est faux!
- R4 : les dossiers administratifs ne sont accessibles qu'à partir des terminaux du secrétariat (règle dépendant du contexte)
  - CREATE VIEW dossier_ouvrable
    AS SELECT * FROM dossier_admin
    WHERE sys_context('USERENV', 'IP_ADDRESS') IN ('T1', 'T2')

    le prédicat est faux pour tous les postes clients autres que T1 et T2 !

RBAC : Role-Based Access Control

- Rôle = ensemble de privilèges
- Les accès des utilisateurs sont gérés en fonction de leur rôle organisationnel
- Objectif = faciliter l'administration des droits



RBAC : Gestion des rôles dans SQL

- Le concept de rôle a été introduit dans SQL3
- Instructions de SQL3
  - CREATE ROLE <nom_role> ;
    - Création d'un nouveau rôle nom_role
  - DROP ROLE <nom_role> ;
    - Suppression du rôle nom_role
  - SET ROLE <liste_roles> ;
    - Permet à un utilisateur d 'activer un ensemble de rôles pendant la durée d'une session SQL

Adaptation de l'instruction GRANT

- Affectation des privilèges aux rôles

  GRANT <liste privileges>
  ON <table ou vue>
  TO <liste roles>
  [WITH GRANT OPTION] ;

- Affectation des rôles aux utilisateurs

  GRANT <liste roles>
  TO <liste utilisateurs>

- Rôle junior et rôle senior

  GRANT <role1> TO <role2>

  Le rôle role2 reçoit tous les privilèges du rôle role1

Limites des modèles DAC et RBAC

- Avec DAC, l'application s'exécutant pour le compte d'un utilisateur hérite des droits de ce dernier
- Avec RBAC, l'application s'exécutant pour le compte d'un utilisateur hérite des droits associés aux rôles activés dans la session ouverte par ce dernier
- Risque de programmes malveillants
  - Cheval de Troie : programme qui a une fonctionnalité apparente mais qui contient des fonctions cachées
  - Objectif : transmission illégale d'informations vers le bénéficiaire du piège
- Autres modèles de CA pour prendre en charge ces limites: Mandatory Access Control (MAC)

Synthèse sur les modèles de contrôle d'accès

- Principe fondateur

- DAC
  - Permet de structurer les Objets
- RBAC
  - Permet de structurer les Sujets
- MAC
  - Lutte contre les programmes malveillants
  - Mais permet peu de souplesse dans la définition des politiques

➔ **Mais tout cela suppose que l'utilisateur passe "par la porte d'entrée" !!**