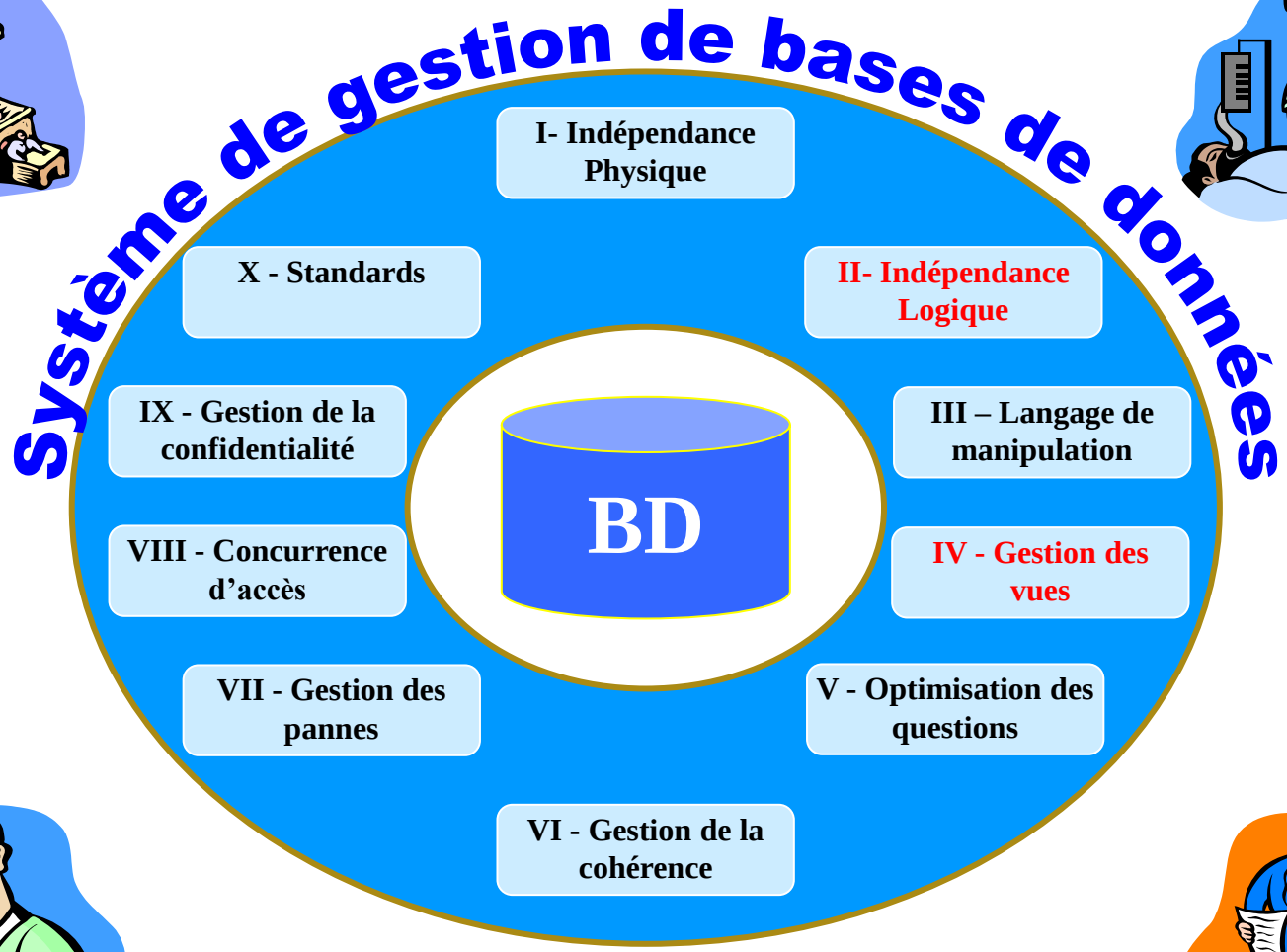
LA GESTION DES VUES

- Définition des vues
- Interrogation au travers des vues
- Mise à jours au travers des vues
- Vues matérialisées

Objectifs des SGBD



Indépendance logique

Les applications peuvent définir des **vues logiques** de la BD

Gestion des médicaments

Nombre_Médicaments

Id-M	Nom	Description	Nombre
1	Aspegic 1000	……..	30
2	Fluisédal	……..	20
3	Mucomyst	……..	230
….	……..	……..	…..

Cabinet du Dr. Masse

Visites

Id -D	Id -P	Id -V	Date	Prix
1	2	1	15 juin	250
2	3	4	1 mars	250

Prescription

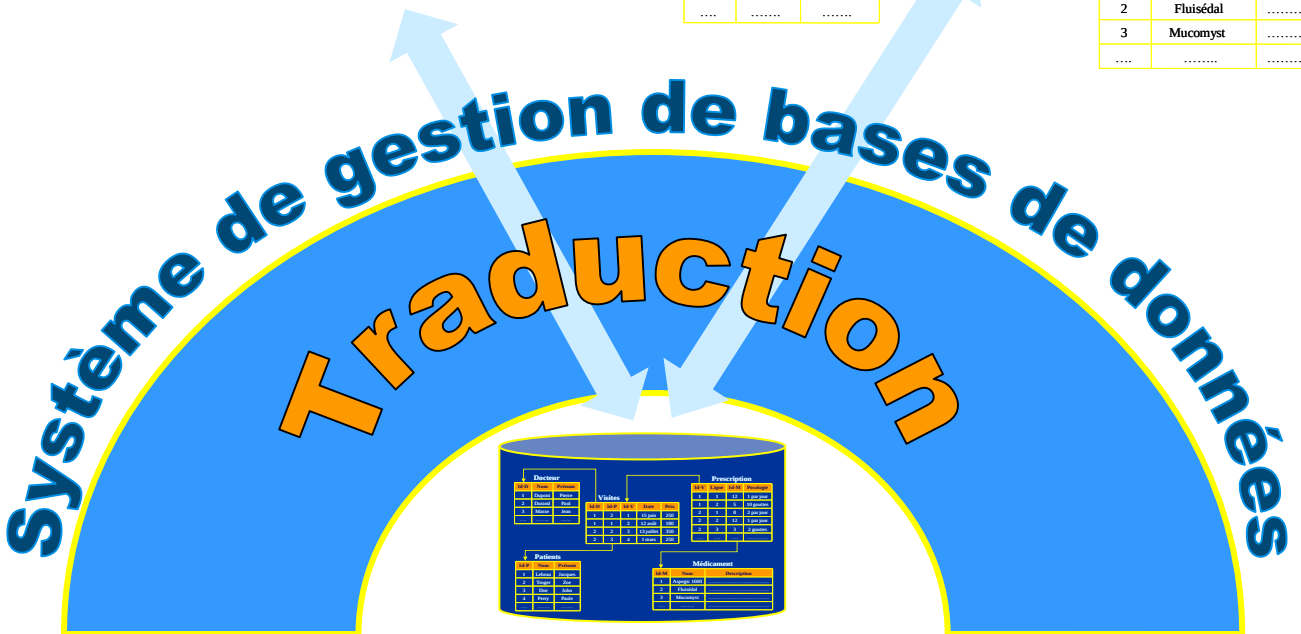
Id -V	Ligne	Id -M	Posologie
1	1	12	1 par jour
1	2	5	10 gouttes
….	….	….	……

Patients

Id -P	Nom	Prénom
1	Lebeau	Jacques
2	Troger	Zoe
….	…….	…….

Médicament

Id -M	Nom	Description
1	Aspegic 1000	……
2	Fluisédal	……
3	Mucomyst	……
….	……..	……



Avantages de l'indépendance logique

- Possibilité pour chaque application **d'ignorer** les besoins des autres (bien que partageant la même BD), i.e., **simplicité**.
- Possibilité **d'évolution de la base de données** sans réécriture des applications :
  - ajout de champs, ajout de relation, renommage de champs.
- Possibilité **d'intégrer des applications existantes** sans modifier les autres.
- Possibilité de limiter les conséquences du partage : **données confidentielles.**

Des vues multiples des données

- Les vues permettent d'implémenter l'indépendance logique en permettant de créer des **relations virtuelles**
- **Vue** = Question stockée
  - ***Table virtuelle*** *dont le schéma et le contenu sont dérivés de la base réelle par un ensemble de questions*

- Le SGBD stocke la **définition** et non le résultat
- Exemple :
  - la vue des patients parisiens
  - la vue des docteurs avec leurs patients
  - la vue des services statistiques

Syntaxe SQL

```
CREATE VIEW <nom de vue> [ (liste d'attributs) ]
    AS <question>
```

- nom de vue = nom de la table virtuelle
- liste d'attributs = les colonnes de la table virtuelle
- question = requête SELECT … FROM …
- si la liste d'attributs n'est pas spécifiée, la vue hérite directement les colonnes du SELECT

- Supprimer la vue :

```
DROP <nom de vue>
```

Exemples

- *TABLES*

RESPONSABLE (NR, NOM, PRENOM, DPT)
COURS (NC, CODE_COURS, INTITULE, ECTS, NR, DPT)
ETUDIANT (NE, NOM, PRENOM, VILLE, AGE)
INSCRIT (NE, NC, ANNEE)
RESULTAT (NE, NC, ANNEE, NOTE)

- *VUES*

COURS_DPT (DPT, NC, INTITULE)
COURS_INFO (NC, INTITULE, ECTS)
RESP_COURS (NOM, PRENOM, CODE_COURS, INTITULE)
ETUDIANTS_INFO (NE, NOM, PRENOM, VILLE, AGE)
ECTS_COURS_DEPARTEMENT (DEPARTEMENT, TOTAL_ECTS)
ADMIS (NE, NOM, PRENOM, ANNEE)
…

Vues simples (projection et restriction)

- La vue des cours et départements

```
CREATE  VIEW  COURS_DPT
AS SELECT DPT, NC, INTITULE
        FROM COURS
```

- La vue des cours du département INFO

```
CREATE  VIEW  COURS_INFO
AS SELECT NC, INTITULE, ECTS
        FROM COURS
        WHERE DPT LIKE « INFO »
```

Vue à partir de plusieurs tables

- La vue des étudiants qui suivent des cours d'info

```
CREATE VIEW ETUDIANTS_INFO
AS SELECT E.*
FROM ETUDIANTS E, INSCRIT I, COURS C
WHERE E.NE = I.NE AND I.NC = C.NC AND
        C.DPT LIKE « INFO »)

CREATE VIEW RESP_COURS
AS SELECT R.NOM, R.PRENOM,
                C.CODE_COURS, C.INTITULE
FROM RESPONSABLE R, COURS C
WHERE R.NR = C.NR
```

Vue de calcul

- La vue du nombre d'ECTS des cours proposés par chaque département

```
CREATE VIEW ECTS_COURS_DEPARTEMENT
                (DEPARTEMENT, TOTAL_ECTS)
AS SELECT DPT, SUM(ECTS)
FROM COURS
GROUP BY DPT
```

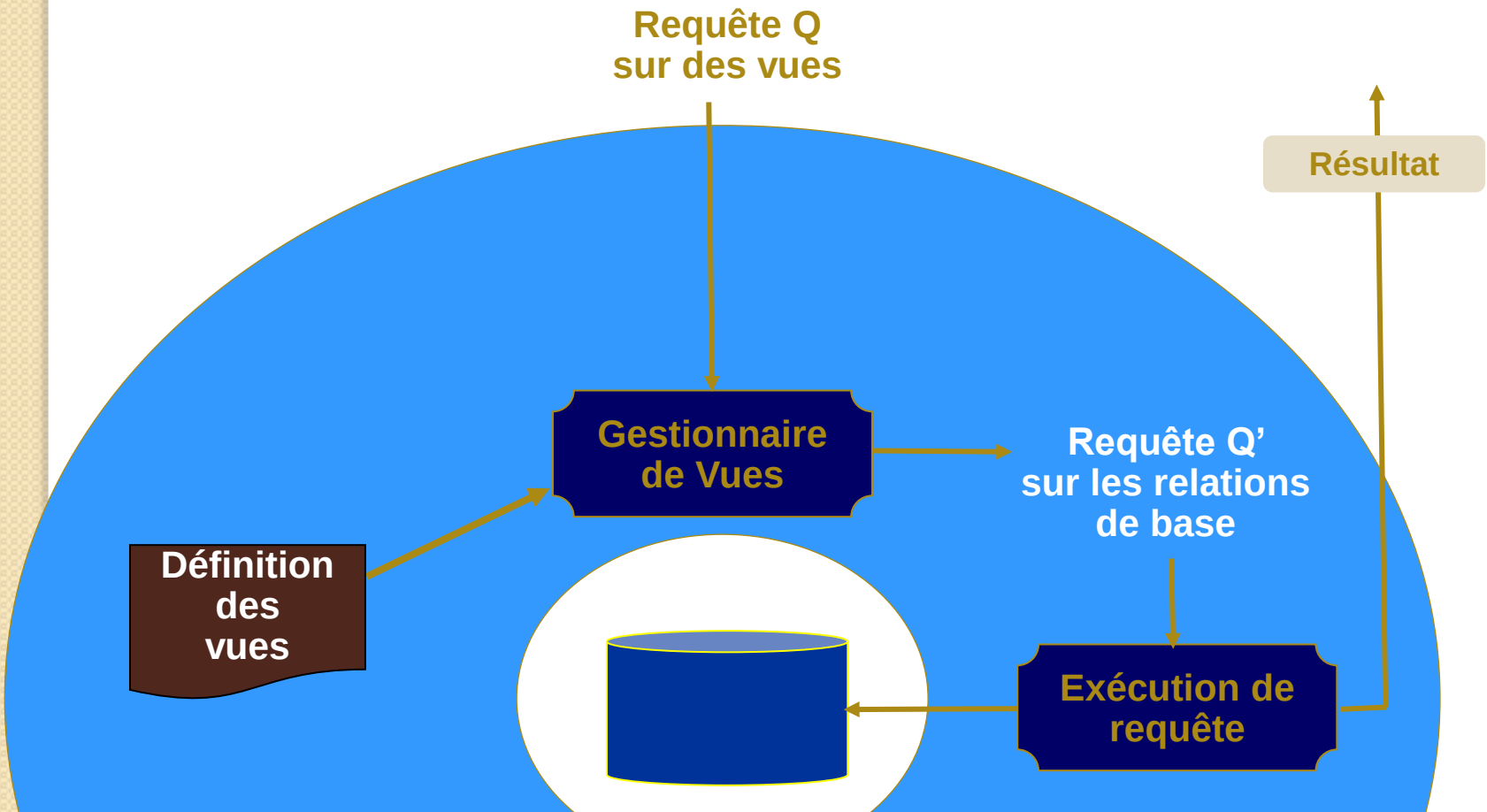
Vue « dynamique »

- La vue des étudiants admis chaque année

```
CREATE  VIEW ADMIS (NE, NOM, PRENOM,
ANNEE)
AS SELECT E.NE, E.NOM, E.PRENOM, R.ANNEE
FROM ETUDIANT E, RESULTAT R, COURS C
WHERE E.NE = R.NE AND R.NC = C.NC
GROUP BY E.NE, R.ANNEE
HAVING AVG(R.NOTE) >= 10
                AND SUM(C.ECTS) >= 60
```

Interrogation au travers des vues

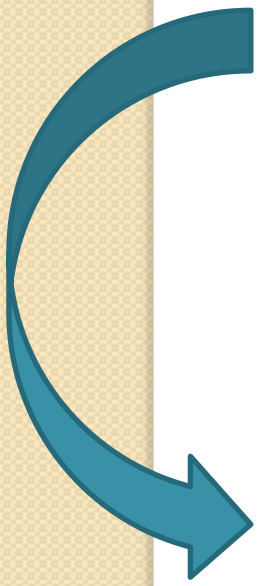
Le SGBD transforme la question sur les vues en question sur les relations de base



Interrogation au travers des vues

- Trouver les étudiants qui suivent des cours d'info et dont le nom commence par « G »

```
SELECT NOM, PRENOM
FROM ETUDIANTS_INFO
WHERE NOM LIKE « G% »
```



- Le SGBD transforme les requêtes qui comportent des vues en des requêtes sur des tables

```
SELECT NOM, PRENOM
FROM (SELECT *
        FROM ETUDIANTS E, INSCRIT I, COURS C
        WHERE E.NE = I.NE AND I.NC = C.NC AND
        C.DPT LIKE « INFO ») ) AS ETUDIANTS_INFO
WHERE NOM LIKE « G% »
```

Mise à jours au travers des vues

- Difficile d'implémenter car il faut répercuter la MAJ sur les tables dans la BD → risques d'ambiguïté
- Valeurs non-définies (ex., si la vue contient des projections)
  - Ex., insérer un nouveau cours d'info dans la vue COURS_INFO
- Risques d'incohérence (ex., si la vue contient des jointures)
  - Ex., supprimer un tuple de RESP_COURS
- Perte de signification (ex., si la vue contient des calculs d'agrégat)
  - Ex., insérer un nouveau tuple dans ECTS_COURS_DEPARTEMENT

Mise à jours au travers des vues

- Peu de systèmes acceptent les mise-à-jours des vues

- Restrictions à respecter pour éviter les conflits
  - Pas de jointure (une seule table dans le from !)
  - La vue contient les clés et les attributs « non nuls » de la table impliquée dans sa définition
  - Pas de distinct, d'agrégats, ni d'expression de calcul
  - Requêtes imbriquées possibles

- Certains SGBD permettent de MAJ multitables (ex., l'administrateur définit la stratégie de report)

Vues matérialisées

- **Vue matérialisée** (cliché, vue concrète, snapshot) : vue dont le contenu est matérialisé sur disque par le SGBD
- Intérêt :
  - Entrepôts de données : les vues facilitent l'analyse de données
  - Données de base peu dynamiques, mais gros volume et calculs complexes
- Difficulté : (auto-)maintenance efficace du contenu de la vue
  - Eviter la reconstruction totale → mise à jour de la vue en différentiel (si possible …)