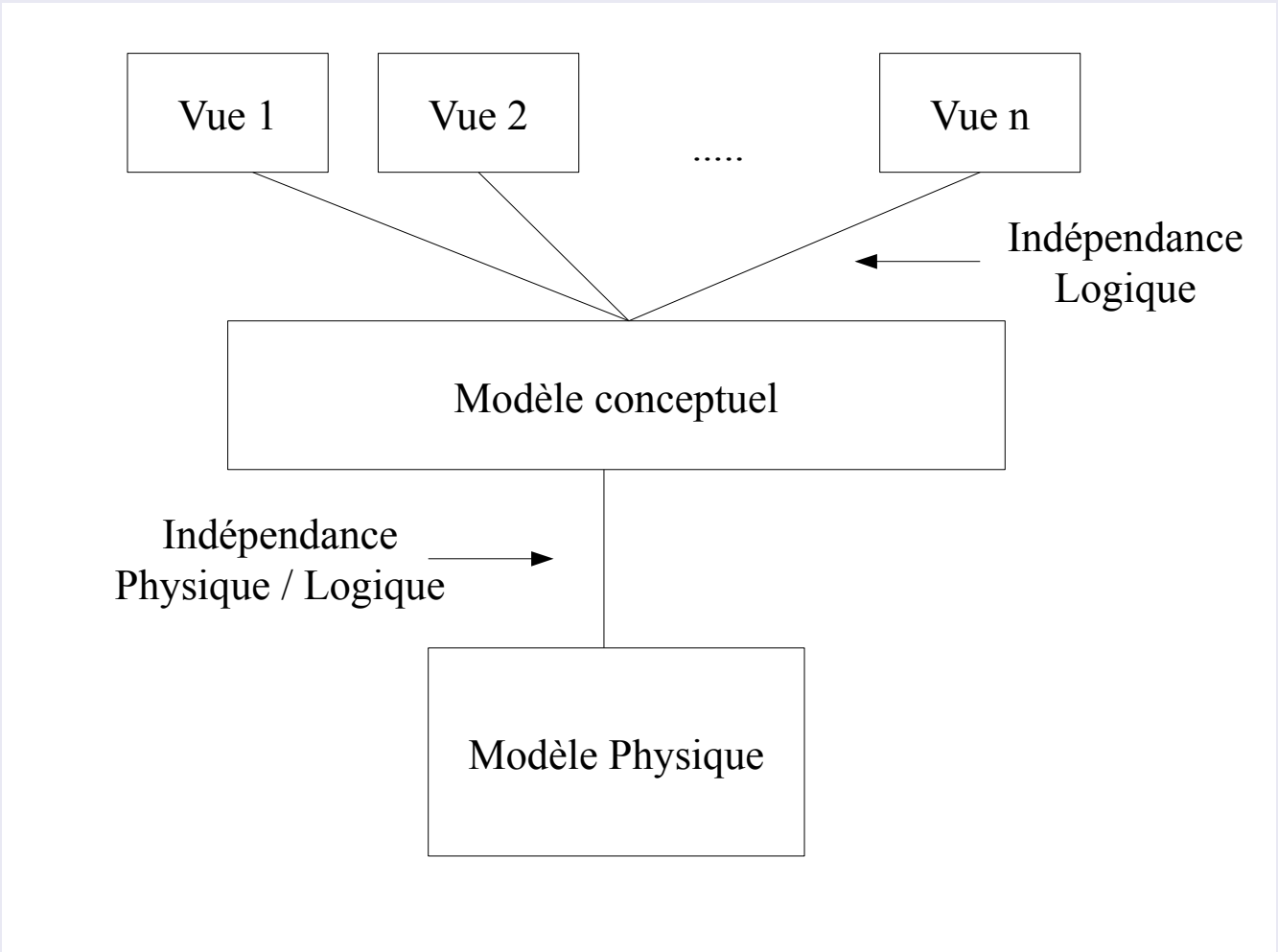
Indépendance



Offre d'un SGBD

- Niveau physique
  - Méthodes de placement
  - Index
  - Statistiques
  - Répartition à travers un réseau
- Niveau conceptuel
  - schéma relationnel
  - contrainte d'intégrité
  - procédure stockée / trigger
- Niveau externe
  - **Vues relationnelles**
  - Indépendance données / traitements : une application est indépendante des modifications apportées aux schémas conceptuels et physiques (en théorie).

Schéma EXEMPLE : Base COOPERATIVE

VINS (V) (NUM_VIN, CRU, MILLESIME)

VITICULTEURS (VT) (NUM_VITICULTEUR, NOM, PRENOM, VILLE)

PRODUCTIONS (P) (VIN, VITICULTEUR)

BUVEURS (B) (NUM_BUVEUR, NOM, PRENOM, VILLE)

COMMANDES (C) (NUM_COMMANDE, DATE, VIN, QUANTITE, BUVEUR)

EXPEDITION (E) (COMMANDE, DATE, QUANTITE)

Règle des vues

Principe Vues

Chaque (groupe d') utilisateur (ou application) possède sa propre vision de la base

- Certaines informations sont conservées
- Certaines informations du schéma conceptuel disparaissent (projection)
- Certaines informations sont re-structurées (jointure)
- Certaines informations n'existent pas et sont calculées (agrégat : DEGRE_MOYEN)

Exemples - Schéma

Gestion des vins

VINS (NUM_VIN, CRU, MILLESIME, DEGRE)

DEGRE_MOYEN_CRU (CRU, DEGRE_MOYEN)

Gestion des viticulteurs bordelais

VITIS_BORDEAUX (NUM_VITICULTEUR, NOM, PRENOM, VILLE)

Gestion des buveurs parisiens

BUVEURS_PARIS (NUM_BUVEUR, NOM, PRENOM)

COMMANDES_PARIS (NUM_COMMANDE, DATE, BUVEUR, VIN, QUANTITE)

QUANTITE_COMMANDEE_BUVEURS (BUVEUR, QUANTITE)

EXPEDITIONS_PARIS (COMMANDE, DATE, QUANTITE)

Vue

Principe VUE

- Relation dérivée (virtuelle) construite à partir de relations de base et/ou de vues relationnelles
- Une vue est spécifiée par une question : approche analytique
- Une vue est une fenêtre sur la base de données et elle permet de voir les **mises à jour** entre sessions de travail

EXEMPLE : viticulteurs de vins de Bordeaux

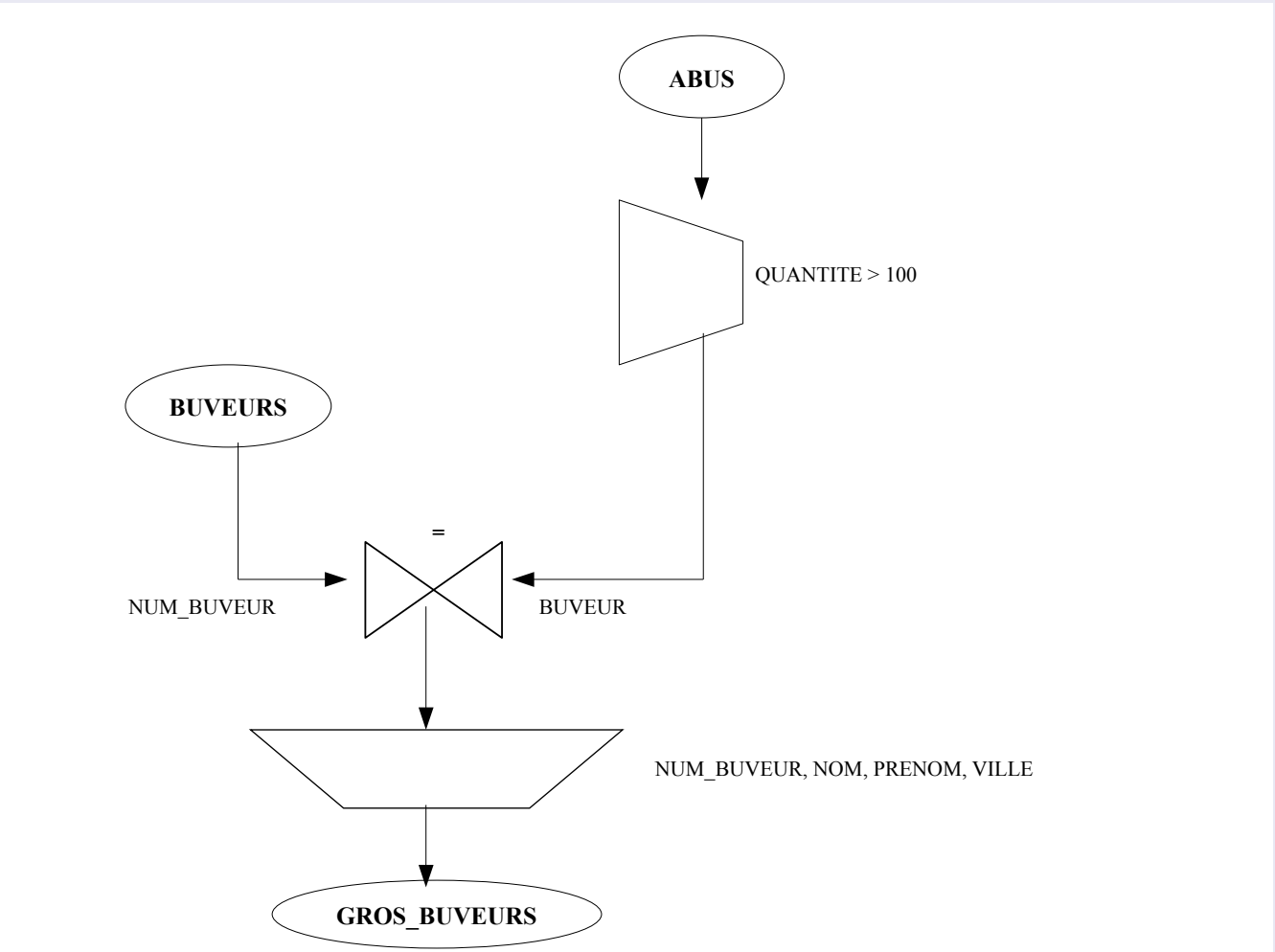
```
CREATE VIEW VITIS_BORDEAUX(NUM_VITICULTEUR,NOM,VILLE)
 AS SELECT VT.NUM_VITICULTEUR, VT.NOM, VT.VILLE
    FROM   VITICULTEURS VT, VINS V, PRODUCTIONS P
    WHERE  VT.NUM_VITICULTEUR = P.VITICULTEUR AND
           P.VIN = V.NUM_VIN AND
           V.CRU = 'Bordeaux';
```

Cliché

Principe Cliché

- Relation, matérialisée, dont les n-uplets sont définis par le résultat d'une requête sur des relations de base, vues ou clichés
- Un cliché **n'est pas mis à jour** entre sessions de travail - sauf demande expresse -

Vue / Cliché (Exemple)



Synthèse : Différence Vue / Relation de Base

- Les n-uplets d'une vue n'existe pas physiquement
- Les n-uplets sont calculables par la question définissant la vue
- La matérialisation peut être effectuée
- Le cliché n'est pas une vue mais une photo statique de la base à un moment donné

Syntaxe et principes

Création d'une vue

- **CREATE VIEW** <NOM> **AS** <Requête SQL>

Suppression d'une vue

- **DROP VIEW** <NOM>

Interrogation

- Toujours possible (sous réserve des droits)
- Syntaxe identique à une relation de base

Mise à jour

- Non définie dans de nombreux cas
- Syntaxe identique à une mise à jour d'une relation de base (si MAJ possible)

Exemples de vues mono et multi-relations

Mono relation

```
CREATE VIEW CRUS (NOM)
AS SELECT DISTINCT CRU
   FROM    VINS
```

Multi-relations

```
CREATE VIEW BUVEURS_BEAUJOLAIS
            (NUM_BUVEUR, NOM, QUANTITE_COMMANDEE)
AS SELECT    B.NUM_BUVEUR, B.NOM, SUM (C.QUANTITE)
   FROM      BUVEURS B, VINS V, COMMANDES C
   WHERE     B.NUM_BUVEUR = C.BUVEUR AND
             C.VIN = V.NUM_VIN AND
             V.CRU = 'Beaujolais'
   GROUP BY B.NUM_BUVEUR, B.NOM
```

Exemples de vues à partir d'une autre vue

A partir d'une autre vue

Relation de base : PARENTS (ASCENDANT, DESCENDANT)

➜ Vue : GRAND_PARENTS (ASCENDANT, DESCENDANT)

```
CREATE VIEW GRAND_PARENTS (ASCENDANT,
DESCENDANT)
AS SELECT P1.ASCENDANT, P2.DESCENDANT
FROM PARENTS P1, PARENTS P2
WHERE P1.DESCENDANT = P2.ASCENDANT;
```

➜ Vue : ARRIERE_GRAND_PARENTS (ASCENDANT, DESCENDANT)

```
CREATE VIEW ARRIERE_GRAND_PARENTS (ASCENDANT,
DESCENDANT)
AS SELECT P.ASCENDANT, GP.DESCENDANT
FROM PARENTS P, GRAND_PARENTS GP
WHERE P.DESCENDANT = GP.ASCENDANT;
```

- Impossibilité d'avoir une fermeture transitive (tous les ascendants d'une personne)

Mise à Jour

- Il faut retrouver la MAJ d'une session de travail sur l'autre
- Comment reporter la mise jour de la vue sur les relations de base ?

Exemples

'Monotable' avec **VINS** (NUM_VIN, CRU, MILLESIME, DEGRE)

```
CREATE VIEW VINS_BORDEAUX (NUM_VIN, MILLESIME,DEGRE)
AS SELECT   NUM_VIN, MILLESIME, DEGRE
   FROM     VINS
   WHERE    CRU = 'Bordeaux';

CREATE VIEW DEGRE_MOYEN_PAR_CRU (CRU, DEGRE_MOYEN)
AS SELECT   CRU, AVG(DEGRE)
   FROM     VINS
   GROUP BY CRU;
```

Exemple

Multi-relations

```
CREATE VIEW BUVEURS_BEAUJOLAIS
            (NUM_BUVEUR, NOM, QUANTITE_COMMANDEE)
AS SELECT   B.NUM_BUVEUR, B.NOM, SUM (C.QUANTITE)
   FROM     BUVEURS B, VINS V, COMMANDES C
   WHERE    B.NUM_BUVEUR = C.BUVEUR AND
            C.VIN = V.NUM_VIN AND
            V.CRU = 'Beaujolais'
   GROUP BY B.NUM_BUVEUR, B.NOM;
```

Principes des mises à jour sur une vue

- Génère éventuellement beaucoup de valeurs nulles
- Vérification : clause **WITH CHECK OPTION** à la création
- Système de règles avec trigger (do instead) chez PostgreSQL
- Dépend du SGBD pour l'implémentation des vues
- Matérialisation des vues dans les entrepôts de données

Traitement par le SGBD - Réécriture

- Modification de la question appliquée au niveau du code source de la requête
- La question posée sur une vue est transformée en une question posée sur les relations de base

Concepteur

```
CREATE VIEW VINS_05(NUM_VIN, CRU, DEGRE)
 AS SELECT NUM_VIN, CRU, DEGRE
 FROM VINS
 WHERE MILLESIME = 2005 ;
```

Utilisateur

```
SELECT *
FROM VINS_05
WHERE CRU <> 'Fronsac' ;
```

SGBD

```
SELECT NUM_VIN, CRU, DEGRE
FROM VINS
WHERE MILLESIME = 2005 AND CRU <> 'Fronsac' ;
```

Traitement par le SGBD - Restructuration algébrique

- Fusion des arbres algébriques de la vue et de la question utilisateur
- Optimisation de l'arbre

(Nécessite une compilation préalable des vues)

Synthèse sur les vues

Avantages

- Adaptation aux applications utilisateur
- Intégration d'application existantes
- Dynamique du schéma de la base
- Confidentialité des données
- Définition de contraintes d'intégrité

Inconvénient

- Mises à jour difficiles ou impossibles