BASES DE DONNÉES AVANCÉES 2

Partie 2

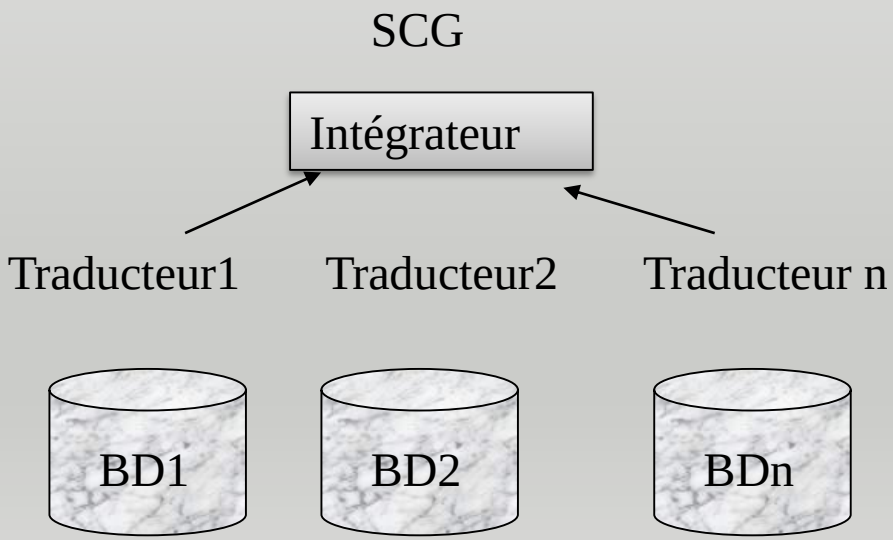
Approche Ascendante (bottom up design)

- Objectifs :
  - Donner aux utilisateurs une vue unique des données implémentées sur plusieurs systèmes a priori hétérogènes (plates-formes et SGBD)
  - Cas typique rencontré lors de la concentration d'entreprises* : faire cohabiter les différents systèmes tout en leur permettant d'inter opérer

- * La **concentration des entreprises** désigne le mouvement par lequel la taille des **entreprises** augmente

Étapes de Traduction

- L'intégration des bases de données peut être effectuée en deux étapes**: la traduction des schémas** et **l'intégration des schémas.**



Étapes de Traduction

- Transformer le schéma local en un autre schéma.
- **Exemple**: transformer un schéma en modèle réseau en un schéma en modèle relationnel

➢ **Pré-intégration**

Identification des éléments reliés (e.g. domaines équivalents) et établissement des règles de conversion (e.g. 1 inch =2.54 cm).

➢ **Comparaison**

Identification des conflits de noms (synonymes, homonymes) et des conflits structurels(clé, dépendances,…).

➢ **Conformance**

Résolution des conflits de noms (renommage) et des conflits structurels (changements des clés…)

➢ **Fusion et Restructuration**

Fusion des schémas intermédiaires et restructuration pour créer un schéma intégré optimal.

L’Intégration

- L’intégration peut être physique ou logique.
- Quand elle est physique on parle de Data Warehouses où la base de données intégrée est matérialisée.
- Dans le cas logique, on parle des systèmes Middleware où le schéma conceptuel global reste entièrement virtuel et n’est pas matérialisé.

La réplication

Réplication

- La réplication consiste à stocker une relation ou un fragment de relation en plusieurs copies.
- Par exemple, si une relation R est fragmentée en R1, R2 et R3. Le fragment R1 peut avoir une seule copie, le fragment R2 peut être répliqué sur deux autres sites et le fragment R3 peut être répliqué sur tous les sites.
- Une BD peut être entièrement répliquée sur chaque site comme elle peut être partiellement répliquée où les fragments sont distribués sur des sites de manière à ce que des copies d’un fragment soient stockées sur plusieurs sites.

Objectifs de la réplication

- **Résistance aux pannes** : puisque les données sont copiées sur plusieurs sites, elles peuvent être accessibles lorsque certains sites sont en panne.
- **Allègement du trafic réseau** : la multiplication des copies de données permet de les rapprocher de leur point d’accès ce qui améliore le temps de réponse.
- **Evolutivité** : La réplication permet de soutenir le développement des systèmes en termes de nombre de sites en garantissant des temps de réponse acceptables.

Types de réplication

- Le choix de l'endroit où les mises à jour sont d'abord effectuées détermine le type de la réplication.
- **Réplication centralisée** : La réplication est centralisée lorsque les mises à jour sont d'abord effectuées dans un site maître appelé primaire.
  - *La réplication centralisée à son tour peut être*
    - synchrone lorsque les mises à jour sont propagées vers les sites en temps réel, comme elle peut être
    - asynchrone lorsque les mises à jour sont propagées en différé.
- **Réplication distribuée** : Contrairement à la réplication centralisée, une réplication est dite distribuée lorsque les mises à jour sont autorisées sur toute réplication. De même, une réplication distribuée peut être synchrone comme elle peut être asynchrone

Traitement de la requête

Étapes de traitement de la requête

Décomposition de la requête utilisateur:

- *But:*

  transformer la requête utilisateur en une requête de l'algèbre relationnelle.

- *Décomposition:*
  - *Normalisation,*
  - *analyse,*
  - *élimination de redondance et*
  - *réécriture.*

Normalisation

- Transformer la requête en une forme permettant de faciliter son traitement.
- On peut distinguer deux formes: forme normale conjonctive et forme normale disjonctive.
- Cette transformation utilise les règles d’équivalence suivantes:

Normalisation

Règles d'équivalence

- $p1 \wedge p2 \Leftrightarrow p2 \wedge p1$.
- $p1 \vee p2 \Leftrightarrow p2 \vee p1$.
- $p1 \wedge (p2 \wedge p3) \Leftrightarrow (p1 \wedge p2) \wedge p3$.
- $p1 \vee (p2 \vee p3) \Leftrightarrow (p1 \vee p2) \vee p3$.
- $p1 \wedge (p2 \vee p3) \Leftrightarrow (p1 \wedge p2) \vee (p1 \wedge p3)$.
- $p1 \vee (p2 \wedge p3) \Leftrightarrow (p1 \vee p2) \wedge (p1 \vee p3)$.
- $\neg(p1 \wedge p2) \Leftrightarrow \neg p1 \vee \neg p2$
- $\neg(p1 \vee p2) \Leftrightarrow \neg p1 \wedge \neg p2$
- $\neg (\neg p) \Leftrightarrow p$

Analyse

- L’analyse de la requête permet d’identifier les requêtes incorrectes qui doivent être rejetées.
- Les raisons principales de rejection d’une requête sont : soit la requête est type incorrect soit elle est sémantiquement incorrecte.
- La requête est représentée par un graphe, nommé graphe de requête ou graphe de connexion. Si le graphe obtenu est non connexe alors la requête est sémantiquement incorrecte.

Analyse

Exemple1:

Soit la base de données suivante:

E (Enum, Enom, Titre, Salaire) /* E est la relation employé */

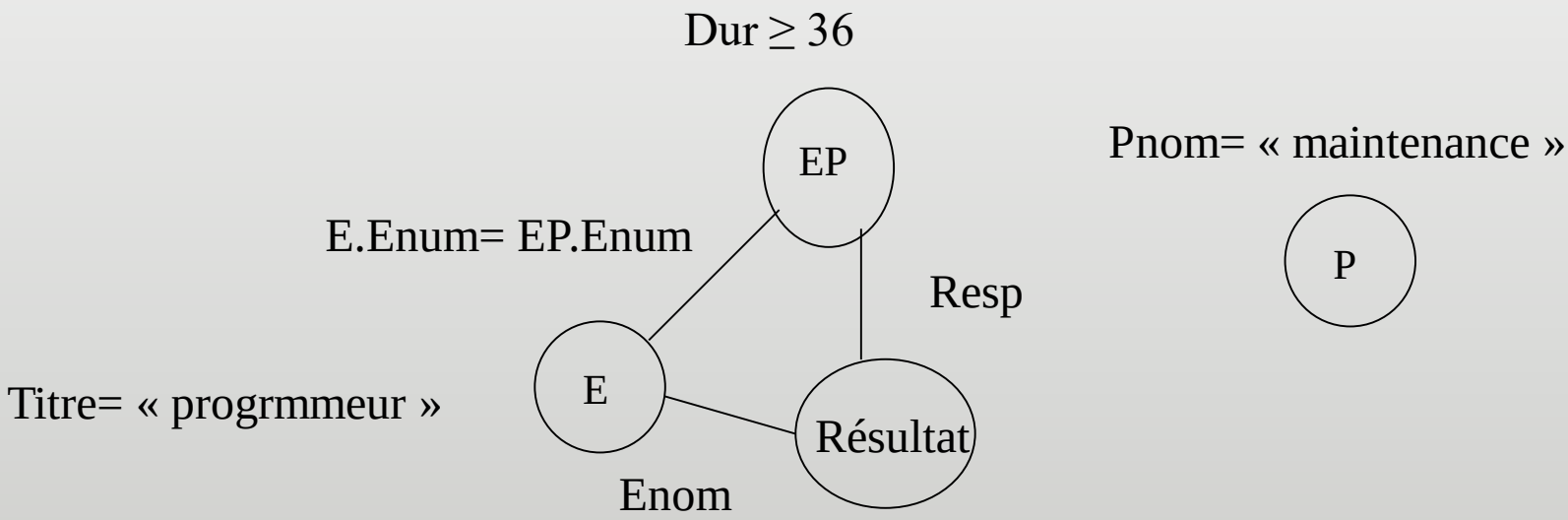
P (Pnum, Pnom, Budget) /* P est la relation projet */

EP(Enum, Pnum, Resp, Dur)

Soit la requête:

```
SELECT Enom, Resp
FROM    E, P, EP
WHERE  E.Enum = EP.Enum
 and         Pnom= « maintenance »
 and         Dur ≥ 36
 and         Titre =« programmeur »
```

Analyse




```
SELECT Enom, Resp
        FROM    E, P, EP
        WHERE  E.Enum = EP.Enum
         and         Pnom= « maintenance »
         and         Dur ≥ 36
         and         Titre =« programmeur »
```

- Cette requête est incorrecte car elle manque une condition de jointure:

  P.Pnum = EP.Pnum

Analyse

- **Exemple2**

  Soit la requête suivante:

  SELECT Enum

  FROM Employe

  WHERE Enom>200

- Cette requête est incorrecte car la condition « >200 » est incompatible avec le type varchar de l'attribut Enom.

Élimination de la redondance

- Les redondances peuvent être éliminées en simplifiant la requête par les règles suivantes
  - $p \wedge p \Leftrightarrow p.$
  - $p \vee p \Leftrightarrow p.$
  - $p \wedge true \Leftrightarrow p.$
  - $p \vee false \Leftrightarrow p.$
  - $p \wedge false \Leftrightarrow false.$
  - $p \vee true \Leftrightarrow true.$
  - $\neg p \wedge p \Leftrightarrow false$
  - $\neg p \vee p \Leftrightarrow true$
  - $p1 \vee (p1 \wedge p2) \Leftrightarrow p1$
  - $p1 \wedge (p1 \vee p2) \Leftrightarrow p1$

Réécriture

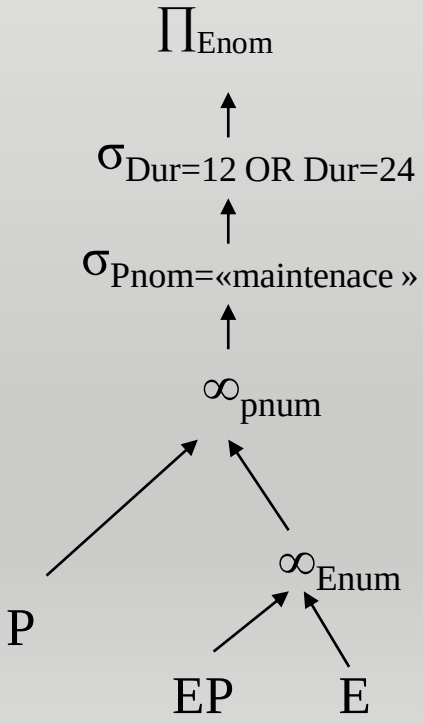
- Réécrire la requête en algèbre relationnelle
- La requête en algèbre relationnelle peut être représentée par un arbre où:
  - feuille = relation
  - noeud = opérateur relationnel

Réécriture

Exemple:

```
SELECT Enom
        FROM    E, P, EP
       WHERE  E.Enum = EP.Enum
        and         EP.Pnum= P.Pnum
        and         Pnom = « maintenance »
        and         (Dur = 12 OR Dur = 24)
```

- L' arbre de la requête en algèbre relationnelle :



Réécriture

- En appliquant les règles de transformation, on peut obtenir plusieurs arbres équivalents.
- Parmi les règles de transformation, on peut citer les règles suivantes:
- Soit les relations R, S et T:
- $R \infty S \Leftrightarrow S \infty R$
- $(R \infty S) \infty T \Leftrightarrow R \infty (S \infty T)$
- $\prod A'(\prod A''(R)) \Leftrightarrow \prod A'(R)$ si $A' \subseteq A''$.
- $\sigma_{P1(A1)}(\sigma_{P2(A2)}(R)) \Leftrightarrow \sigma_{P1(A1) \wedge P2(A2)}(R)$.

Réécriture

- LE SGBD réparti assure la réécriture des requêtes distribuées en plusieurs sous requêtes locales envoyées à chaque site.
- La réécriture de requête est une décomposition qui prend en compte les **règles de localisation**

Localisation des données

- Cette étape permet de translater une requête en algèbre relationnelle exprimée sur le schéma global de la base de données en une requête exprimée sur les schémas locaux (l'ensemble des fragments)

Localisation des données

Les requêtes utilisant des relations fragmentées horizontalement

- ***Exemple:***

On considère que la relation employé (E) est fragmentée comme suit:

$E_1 = \sigma_{Enum \leq 3}(E)$.

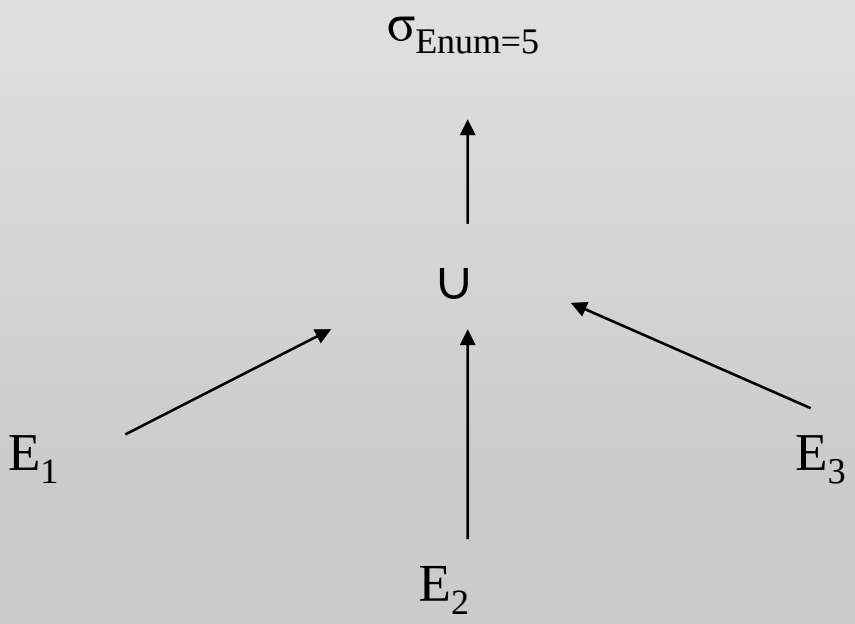
$E_2 = \sigma_{3 < Enum < 6}(E)$.

$E_3 = \sigma_{Enum > 6}(E)$.

La relation est obtenue par l'union $E_1 \cup E_2 \cup E_3$

Soit la requête :

```
SELECT *
FROM   E
WHERE Enum=5.
```


σEnum=5
∪
E1
E2
E3

Localisation des données

- Cette requête peut être simplifiée ⇒ requête réduite.
- Les règles de réduction permettent d'identifier les fragments générant des relations vides.

➢ **Sélection**: on considère une relation R fragmentée horizontalement en R_1, R_2,…R_n tel que $R_j = \sigma_{Pj}(R)$.

Règle 1: $\sigma_{Pi}(R_j) = \emptyset$ si $\forall$ x un tuple de R :$\neg(P_i(x) \wedge P_j(x))$

P_i et P_j sont des prédicats de sélection.

Exemple: on considère la requête:

SELECT * FROM E WHERE Enum=5.

D'après la règle1, si on applique le prédicat de sélection « Enum=5 » pour les fragments E_1 et E_3 on obtient des relation vides. Donc la requête est réduite comme suit

$\sigma_{Enum=5}$

↓

E_2

Localisation des données

Jointure: on peut détecter les jointures inutiles en se basant sur la règle suivante:

Règle 2: $R_i \infty R_j = \emptyset$ si $\forall$ x un tuple de R_j $\forall$ y un tuple de R_i :$\neg(P_i(x) \wedge P_j(y))$

R_i et R_j sont deux fragments définis respectivement par les prédicats P_i et P_j

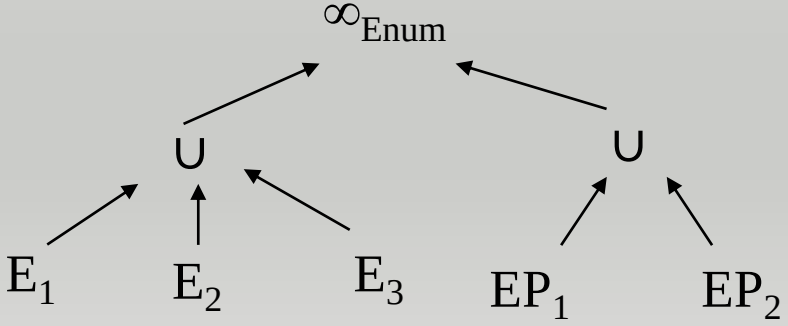
Exemple: on considère la requête:

SELECT * FROM E ,EP WHERE E.Enum=EP.Enum

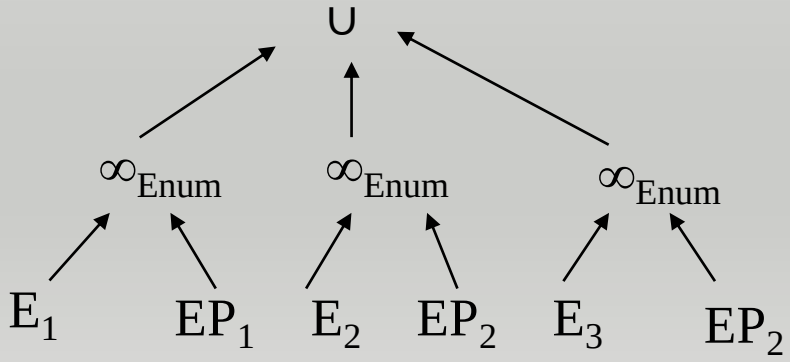
On considère que EP est fragmentée comme suit:

$EP_1 = \sigma_{Enum \leq 3}(EP)$

$EP_2 = \sigma_{Enum > 3}(EP)$



Requête Générique



Requête Réduite

Localisation des données

Les requêtes utilisant des relations fragmentées verticalement

On considère que la relation Employé E est fragmentée verticalement comme suit:

$E_1 = \prod_{Enum, Enom}(E)$

$E_2 = \prod_{Enum, Titre}(E)$

- La relation E est reconstruite par une opération de jointure.

$$E = E_1 \infty E_2$$

- Les requêtes utilisant des relations fragmentées verticalement peuvent être réduites en éliminant les opérations inutiles. Pour ce faire, on utilise la règle suivante:

Règle3: on pose $R_i = \prod_{A'}(R)$

$\prod_D(R_i)$ est inutile si l'ensemble des attributs D intersection A' égale au vide

Localisation des données

Exemple:

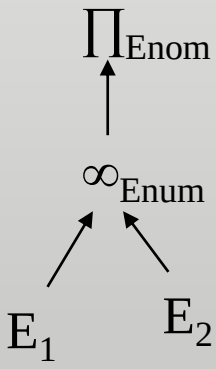
Soit la requête:

SELECT Enom

FROM E

$E_1 = \prod_{Enum,\ Enom}(E)$

$E_2 = \prod_{Enum,\ Titre}(E)$



Requête Générique

Requête réduite