Après les bases de données relationnelles ...

- **Besoins des applications BD avancées**
- **Objet-relationnel**
  - la norme SQL3
- **SGBD orienté objet**
  - la norme ODMG

1. BESOINS DES APPLICATIONS BD AVANCEES

- **CAO/FAO**
  - Objets structurés et volumineux
  - Objets partagés
  - VERSIONS/ALTERNATIVES
  - Base publique/Base privée
  - Transactions longues
- **Applications multi-media**
  - Objets volumineux et parfois non structurés : image, parole
    - 1 mn de son digitalisé 500 Ko
  - images bit-map sous forme de QUAD-TREE
  - Nombreux liens sémantiques entre objets :
    - "est-un", "est composé de", "est version de"
  - Langage d'interrogation associatif

1. BESOINS DES APPLICATIONS BD AVANCEES ...

- **Ateliers logiciels**
  - CAO/FAO +
  - Accès associatif ==> index
  - Manipulation de graphes:
    - (PERT, dépendances, références croisées …)
- **Systèmes d 'information géographiques**
  - Très gros volume de données (1 carte = qq Go)
  - Nouveaux types et opérations :
    - région, ligne, point
    - ∩, ∪, overlap, adjacence
  - parcours de réseau
  - Langage assertionnel étendu
  - Index spatial, liens topologiques

1. LES LIMITES DU RELATIONNEL

- **Support de domaines atomiques :**
  - 1ère forme normale de Codd
  - Pauvreté du système de typage (ens. types prédéfinis)
  - introduction de BLOB (binary large objects)
  - inadapté aux objets complexes (documents structurés)
- **Capture faible des liens sémantiques entre objets:**
  - jointure sur clé
- **Langage d'interrogation (SQL) non complet :**
  - Pas de récursion
  - Pas de structures de contrôle: if … then … else, for each

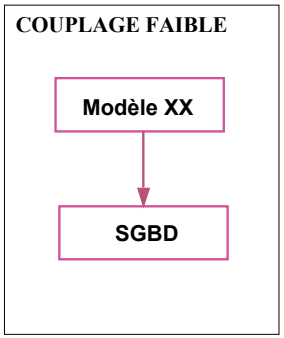
1. LES LIMITES DU RELATIONNEL ...

- **Opérations séparées des données :**
  - procédures stockées non intégrées dans le modèle
- **Ensemble fermé d'opérateurs :**
  - algèbre relationnelle
  - Critères d'optimisation liés à ces opérateurs
- **Index restreints aux types de base**
- **Gestion de transaction figée :**
  - atomicité
  - granule de verrouillage et de journalisation

2. QUE FAIRE ?

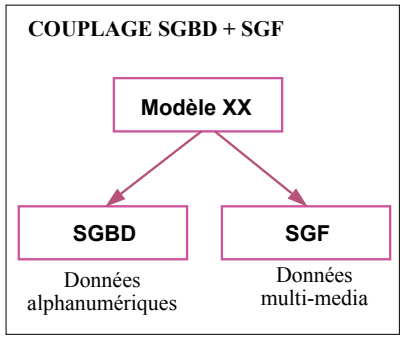
- **Définir un modèle de données plus riche**
- **Définir un système capable de le supporter**

2. COMMENT SUPPORTER DE NOUVEAUX MODELES ?



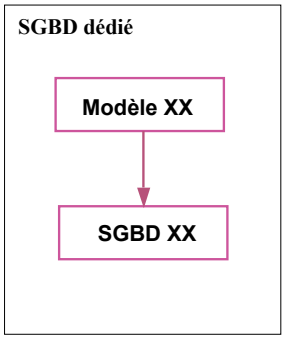
Conversions coûteuses

Limites inhérentes au SGBD sous-jacent



Perte d'une partie des avantages apportés par l'approche BD

2. COMMENT SUPPORTER DE NOUVEAUX MODELES ?



==> BESOIN DE PERFORMANCE

==> BESOIN D'EXTENSIBILITE

Très performant :

- coûteux à réaliser
- peu évolutif

2. L 'APPORT DES MODELES OBJETS

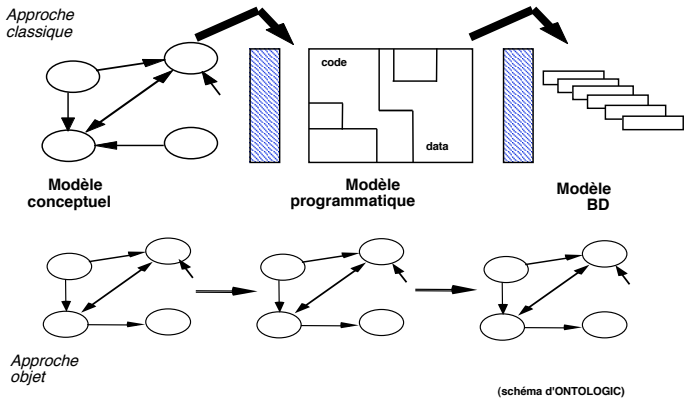
- **Identité d'objets**
  - introduction de pointeurs invariants
  - possibilité de chaînage
- **Encapsulation des données**
  - possibilité d'isoler les données par des opérations
  - facilite l'évolution des structures de données
- **Héritage d'opérations et de structures**
  - facilite la réutilisation des types de données
  - permet l'adaptation à son application
- **Possibilité d'opérations abstraites (polymorphisme)**
  - simplifie la vie du développeur

2. PROBLEMATIQUE

- **BESOINS :**
  - représentation et gestion d'objets complexes ou multi-media :
    - données structurées, textes, graphiques, cartes, images, son
  - mieux communiquer avec les utilisateurs :
    - réduire la distance sémantique existante entre les objets complexes du monde réel et leur représentation en 1ère forme normale
  - fournir un langage de manipulation de données compatible avec les langages de programmation :
    - réduire l'impédance mismatch
  - répondre aux besoins spécifiques des nouvelles applications:
    - restaurer la navigation, transactions longues faiblement concurrentes, le travail coopératif

2. OBJECTIFS

Le paradigme uniforme de spécification, de programmation et d'accès base



(schéma d'ONTOLOGIC)

2. CLASSES DE SYSTEMES

- **Les SGBD relationnels étendus :**
  - objet-relationnel : ORACLE (v8), ILLUSTRA, UniSQL
  - nécessité de conserver une compatibilité avec l 'existant
  - une approche partielle mais opérationnelle
- **Les langages de programmation persistants :**
  - VERSANT, OBJECTIVITY, OBJECSTORE, ONTOS, GEMSTONE
  - Langages orientés objet (C++, Smalltalk, Java) couplés à un gérant d'objets persistants
- **Les SGBD intégrés :**
  - O2, ORION, Matisse(GBASE), IRIS
  - Des systèmes complexes et ambitieux, vers de nouveaux langages et modèles

3 . EXTENSION DU RELATIONNEL

- **OBJECTIF :**
  - Conserver autant de concepts relationnels que possible

- **Deux principales extensions :**
  - Capacités d'inférence :
    - langage de règles
    - expression de la récursion (possible dans SQL3)
    - trigger

  ==> BD Déductives

  - Types de données et opérateurs associés définis par l'utilisateur:
    - domaine = ADT (Abstract Data Type)

  ==> BD objet-relationnel

3. LES NOTIONS DE BASE

1) Domaine :

Ensemble de valeurs.

2) Relation :

Sous-ensemble du produit cartésien d'une liste de domaines.

3) Attribut :

Colonne d'une relation caractérisée par un nom.

- **Modèle de données des ADTs :**
  - Extension du modèle relationnel par le support de domaines complexes définis par l'utilisateur.
  - ADT = sémantique opérationnelle
  - Données complexes construites à partir de types de données de base et de constructeurs auxquelles sont associées des propriétés spécifiques.

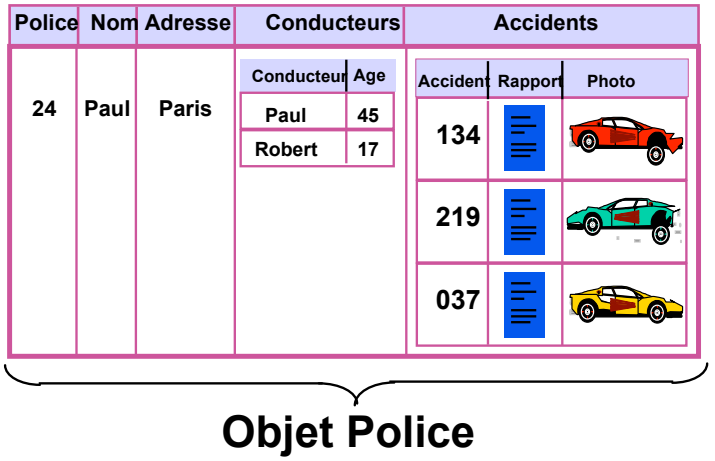
3. SGBD OBJET-RELATIONNEL

- **Extension du modèle relationnel par le support de domaines définis par l'utilisateur.**
  - Définition : (1) des domaines complexes, (2) des opérateurs associés, (3) du schéma de la BD
  - La définition des domaines et des fonctions associées est effectuée dans un langage propriétaire ou standard (C, lisp, java)

- **Langage d'interrogation (SQL) étendu aux types de données et aux fonctions utilisateurs**

- **Adaptation du coeur du SGBD:**
  - Nouvelles méthodes de placement et d'indexation
  - Prise en compte de nouveaux opérateurs et méthodes d'accès dans l'optimiseur de requêtes

3. Exemple de table et objet (Oracle8)



3. STOCKAGE D 'OBJETS COMPOSITES

- **Stockage relationnel dans des attributs longs :**
  - pas de partage de sous-composants : redondance, incohérence lors des mises à jour
  - pas de chargement d 'un sous-composant isolé : mal adapté aux objets volumineux
- **Stockage relationnel par décomposition :**
  - recomposition par jointure coûteuse
  - nécessité de contraintes d 'intégrité référentielle pour maintenir la cohérence des données
  - pas intuitif : forte distance sémantique entre l 'objet du monde réel perçu par l 'utilisateur et l 'objet stocké dans la BD
- **Stockage objet grâce à des identifiants d 'objets**
  - identité d 'objet, différents modes de stockage sont possibles

3. ARCHITECTURE

- **Différentes approches:**
  - Un interpréteur spécialisé est intégré au cœur du SGBD pour gérer le code utilisateur
    - ex : interpréteur PL/SQL (procédures stockées)
    - ex : intégration d 'une JVM (Java Virtual Machine) (Oracle8i)
  - Edition de liens et chargement dynamiques du code objet à l'exécution (attention : aux erreurs dans le code, arrêt du système)
  - Editions de liens statiques entre le programme SGBD et les programmes utilisateur (arrêt de la BD en cas de mise à jour)

3. CONCLUSION DE L 'APPROCHE

- **Avantages:**
  - approche ascendante bien intégrée au relationnel
  - effort de normalisation: définition de la norme SQL3
  - outil puissant et relativement facile à utiliser
  - Contrôle des erreurs dans le cas d'un langage de définition d'ADT interprété
- **Inconvénients :**
  - Dichotomie entre le modèle tout valeur et objet/valeur
  - Langage interprété plus lent
  - Difficulté de définir une nouvelle méthode d'accès
  - Intégration difficile au sein de l 'optimiseur de requêtes

4. DEFINITION D 'UN SGBDOO

The Object-Oriented Database System Manifesto

Atkinson, Bancilhon, Dewitt, Ditrich, Maier, Zdonik (Conf DOOD 89)

Les fonctionnalités BD :

- la persistance
- la concurrence
- la fiabilité
- la facilité d'interrogation

Optionnel :

- la distribution
- les modèles de transaction évolués
- les versions

Les fonctionnalités orientées objet :

- les objets atomiques ou complexes
- l'identité d'objet
- l'héritage simple
- le polymorphisme
- l'encapsulation

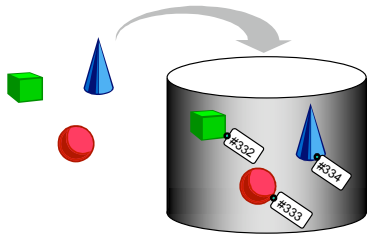
Optionnel :

- l'héritage multiple
- les messages d'exception ("error handling")

COUPLAGE LPOO - GERANT D'OBJETS

- **Extension du LPOO :**
  - identité d'objet
  - notion d'objet persistant et temporaire
  - méthodes de chargement des objets persistants
  - dictionnaire de type
  - notion d'associations (liens inverses)
- **Gérant d'objets :**
  - stockage unique des objets complexes
  - concurrence d'accès
  - méthode d'accès
  - fiabilité et reprises

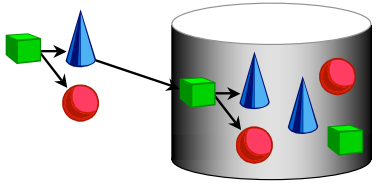
4. LE STOCKAGE DES OBJETS



- **Identifiant d'objet (OID)**
  - Identifie de manière unique (et automatique) un objet
- **Les relations**
  - Les OIDs permettent de définir les relations entre objets (plus de jointure)

VERSANT

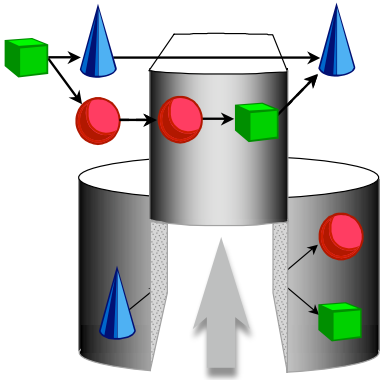
4. STOCKER DES OBJETS



- **Même modèle :**
  - Les objets dans la base sont les « mêmes » que ceux en mémoire
  - Pas de mapping => productivité et performance
  - Les relations deviennent des liens au sein de la base => performance

VERSANT

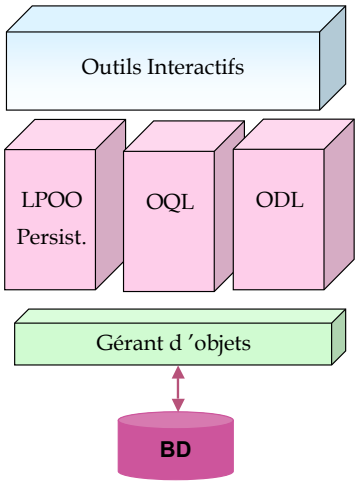
4. ACCES TRANSPARENT AUX OBJETS



- **A partir du LP :**
  - suppression de « l 'impedance mismatch »
  - Accès efficace par navigation, pas uniquement à partir de SQL
  - Productivité et maintenabilité accrues

VERSANT

4. ARCHITECTURE FONCTIONNELLE TYPE



- Editeur de classes
- Manipulateur d'objets
- Bibliothèques graphiques
- Débogueur, éditeur

OQL = Object Query Language

ODL = Object Defintion Language

LOO = Langage Orienté Objet

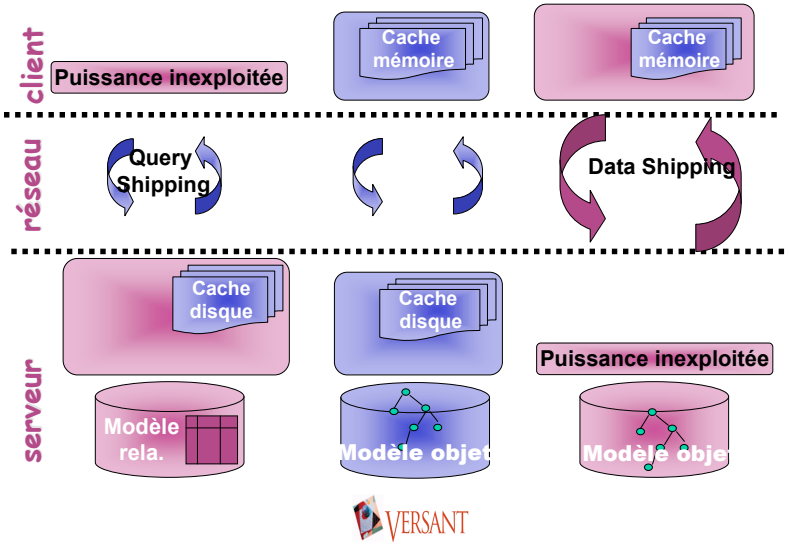
- Persistance
- Identification
- Accès
- Concurrence
- Fiabilité
- Sécurité

4. QUELLE ARCHITECTURE CHOISIR?

- **Comment distribuer au mieux les fonctionnalités d 'un SGBDOO?**
- **Constat :**
  - puissance croissante des postes de travail
    - puissance cumulée des clients très supérieures à celle des serveurs
  - débit croissant des réseaux locaux (ex: Gigabit Ethernet)
    - le réseau n 'est plus un goulot d 'étranglement
  - les applications BD sont, de par leur complexité, de plus en plus consommatrices de CPU

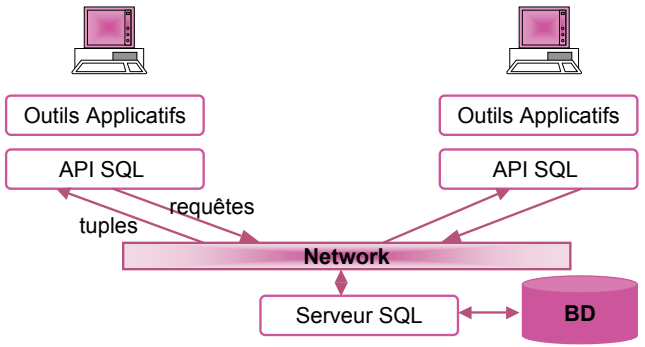
=>une grande partie des fonctions qui sont habituellement gérées par le SGBR doivent être déplacées au niveau du client

4.1. CLASSIFICATION DES ARCHITECTURES



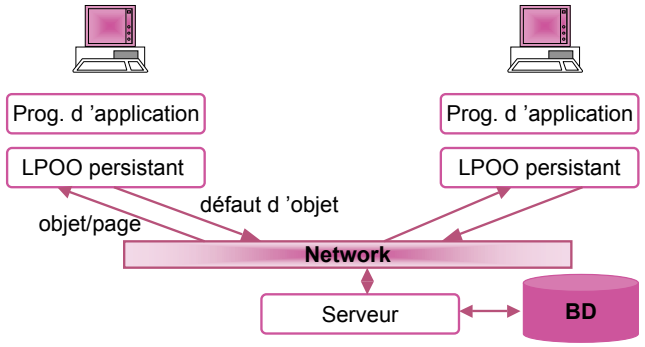
4.1. C/S BD RELATIONNEL

- **Architecture de type *Query (function) Shipping :***
  - requêtes et n-uplets transitent sur le réseau (1 RPC par requête SQL)
  - Réduction du trafic possible grâce à des procédures stockées
  - le traitement s 'effectue majoritairement sur le serveur



4.1. C/S BD ORIENTE OBJET

- **Architecture de type *Data Shipping :***
  - les requêtes transitent du serveur vers un cache maintenu sur le client
  - le traitement s 'effectue majoritairement sur le client
  - les échanges peuvent se faire avec un granule page ou objet



4.2. LA PERSISTANCE DES OBJETS

- **Objectifs :**
  - faire persister les objets du LPOO sans les démonter
  - assurer le plus possible la transparence d 'accès au programmeur?
  - garder des performances proches du travail en mémoire
  - récupérer les emplacements des objets détruits

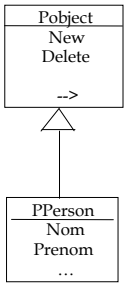
- **Moyens :**
  - définition d 'un modèle de persistance
  - offrir un gérant d'objets assurant la concurrence et la fiabilité
  - identifiant d'objets permettant de retrouver les objets en 1 ou 2 accès
  - ramasse-miettes périodique

4.2. MODELE DE PERSISTANCE : PAR HERITAGE

- **Seules les classes héritant de la propriété de persistance Pobject peuvent contenir des instances persistantes**
  - tout objet d'une sous-classe hérite des propriétés de persistance
  - les opérateurs : New, Delete et -> sont surchargés

- **Déclaration de la persistance :**
  - explicite :
    - PPerson p* = new Persistent PPerson(…);
    - Pperson t* = new PPerson(…); t->makePersistent();
  - si implicite :
    - il faut dupliquer les classes (persistantes et transientes)

- **Modèle de persistance non orthogonale au type :**
  - seuls les types héritants de PObject persistent



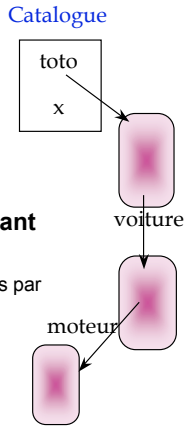
4.2. MODELE DE PERSISTANCE : PAR ATTEIGNABILITE

- **Définition d 'une racine de persistance :**
  - Name toto : Person;
  - toto = new Person(..);

- **Un objet racine de persistance est catalogué**
  - Person *p = (Person*) Lookup(« toto »);

- **Tout objet atteignable depuis la racine est persistant**
  - les références sont rendues persistantes :
    - lors de l'écriture d'un objet les références sont remplacées par des oid

- **Persistance orthogonale aux types**

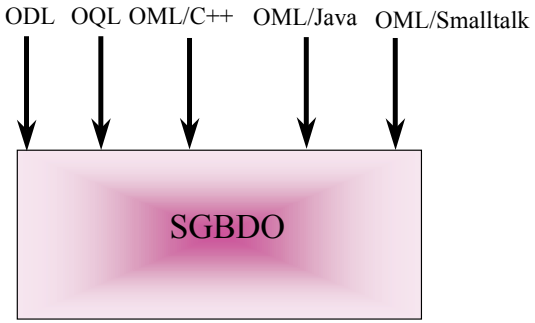


4.2. CHOIX D 'UN MODELE OBJET

- **Proposition d 'un "standard" pour les SGBDOO**
  - *Object Database Management Group* :
    - Fondé en septembre 91 par 5 constructeurs:
      - O2 Technology, Objectivity, Object Design, Ontos, Versant
    - Version 1.0 publiée en 1993
    - Version 2.0 en 1996 avec: 10 auteurs
      - POET Soft, Lucent , Barry &Ass., Amer Man. Syst., Windward Sol.,
  - *est renommé Object Data Management Group* en 1998
    - Sun rejoint l 'ODMG pour travailler sur le binding Java
    - *Version 3.0 en Janvier 2000*
- **Travaux de l'ODMG :**
  - définition d'un modèle objet de référence ODL
  - définition d'un langage déclaratif OQL
  - définition des différents binding avec C++, Smalltalk et Java

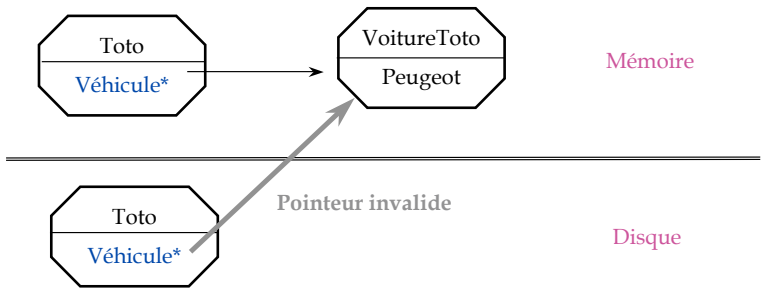
4.2. PROPOSITION DE L 'ODMG

- **Adaptation du modèle objet de l'OMG**
- **Interfaces d'accès à un SGBDO**



4.2. LA TRANSLATION DES POINTEURS

- **Les adresses mémoires doivent être translatées en adresses BD (oid) lors des écritures et vice versa**



4.2. APPROCHE DOUBLE POINTEUR

- **Tout pointeur sur un objet persistant est remplacé par un double pointeur**
  - **En écriture**
    - si Ad BD inconnue ALORS
      - { Ad BD = AllouerBD(Objet) ; Ecrire (Ad BD, Objet) } ;
  - **En lecture**
    - si Ad MC inconnue ALORS
      - {Ad MC = AllouerMC(Ad BD); Lire(AdMC, Ad BD) } ;

Ad MC
Ad BD

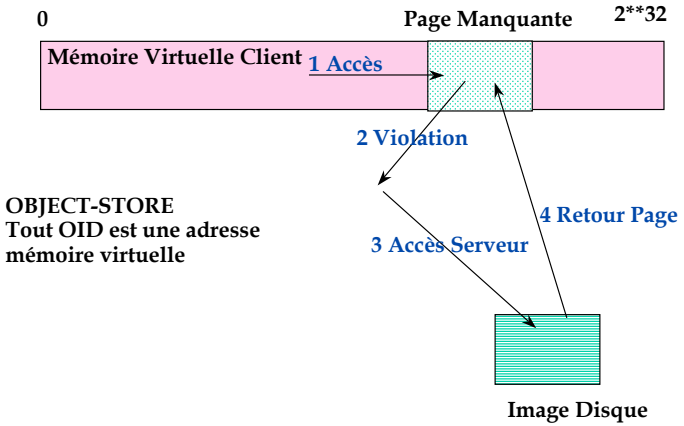
- **Inconvénients :**
  - objets persistants versus objets transients (doubles pointeurs)
  - lecture des objets pointés
  - faiblesse des performances

4.2. APPROCHE MEMOIRE VIRTUELLE (1)

- **Ecriture des objets sans modification**
  - Pointeurs disques = adresse mémoire virtuelle
  - Sauvegarde des pages dans des partitions BD images de la mémoire
- **Pré-allocation des objets en mémoire virtuelle**
  - Lorsqu'un objet est retrouvé, des pages inaccessibles sont allouées pour tous les objets référencés.
  - L'objet est chargé lors du premier accès par récupération de la violation mémoire virtuelle.

4.2. APPROCHE MEMOIRE VIRTUELLE (2)

- BD disque = image de l 'espace virtuel en mémoire
- mécanisme de défaut de page



4.3. CLUSTERING DES OBJETS

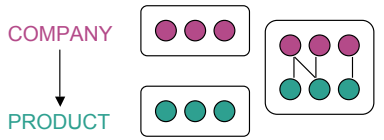
- **Clustering par classe**
  - Regroupement de toutes les instances d'une même classe dans un même fichier
- **Clustering par composition**
  - Regroupement d' un objet d'une classe avec un ou plusieurs de ses objets composants.
  - Placement adapté aux parcours de chemin
- **Clustering aléatoire**
  - les objets sont placés dans l'ordre de leur création, dans un espace unique.

4.3. TECHNIQUES DE CLUSTERING

Le clustering des objets sur disque permet de réduire le nombre d 'entrées-sorties

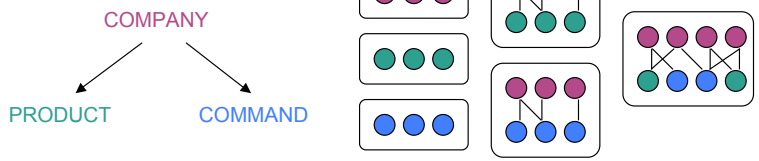
- **Clustering par classe**

- **Clustering par composition**

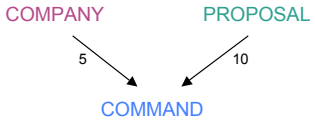


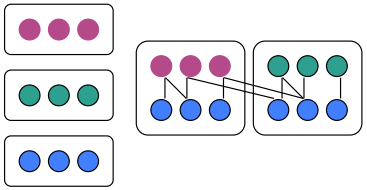
4.3. TECHNIQUES DE CLUSTERING AVANCEES

- **Clustering conjonctif**



- **Clustering dsijonctif**





4.4. ACCES AUX OBJETS PERSISTANTS

- **Par un langage de programmation - OML:**
  - accès à un objet (nommé) ou à une collection d'objets (itération)
  - opérations de navigation dans la base de données (déréférencer un pointeur)

  => compilation

- **Par un navigateur (browser)**
  - visualiser, créer et modifier le schéma et les instances de la base de données

  => interactif

4.4. ACCES AUX OBJETS PERSISTANTS ...

- **Par un langage déclaratif type SQL - OQL :**
  - Indépendance entre le niveau logique et le niveau physique
  - le programmeur n'a pas à spécifier l'enchaînement des opérations

- **Possibilités d'optimisation :**
  - choix des index, ordonnancement des prédicats

  => interprété (résolution de certaines méthodes à l 'exécution)

  => interactif ou extension d'un langage de programmation