Les systèmes de base de données à objets

C. Delobel, V. Benzaken, E. Waller

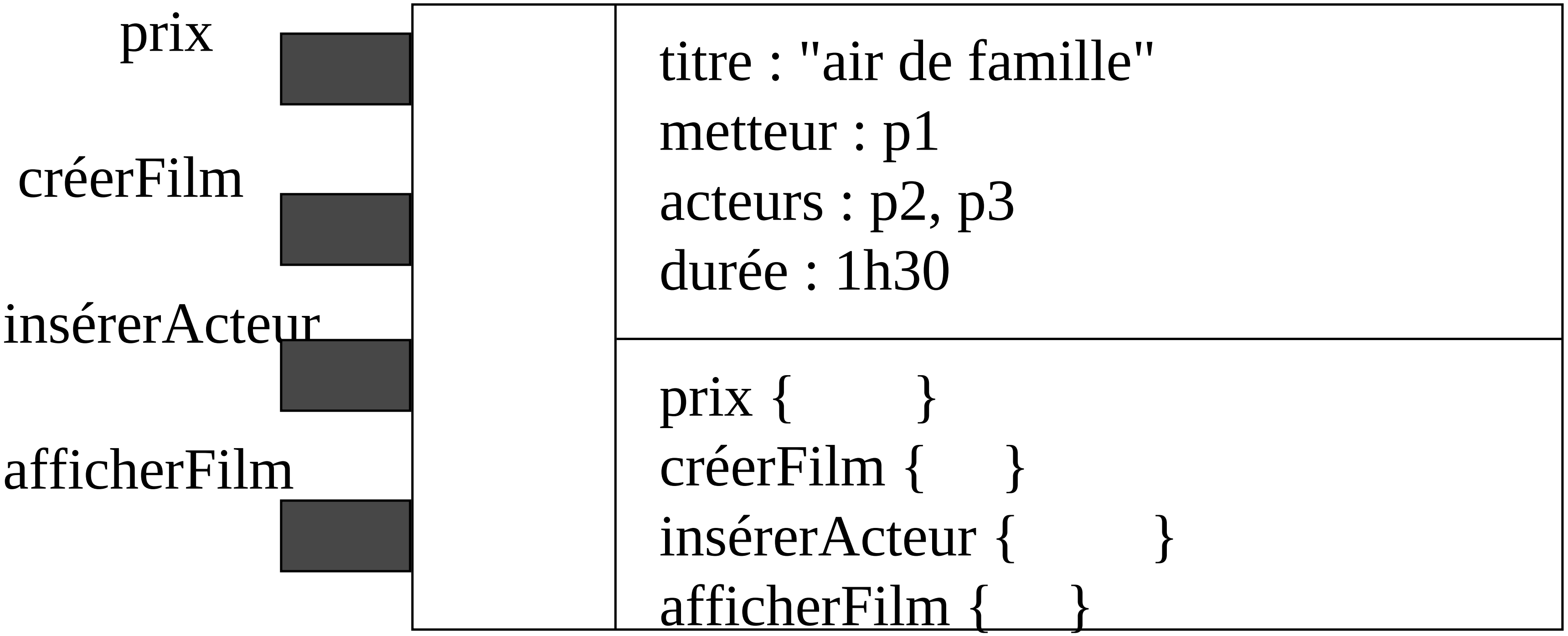
Principaux concepts

- Fusion entre la technologie base de données et objet
- Concepts base de données : schéma, instance, persistance, partage des données, fiabilité et sécurité, gestion du disque, langage de requêtes.
- Concepts objet : objet, encapsulation, classe, héritage, surcharge, extensibilité.

Notion d'objet

- Un objet est une chose tangible ou visible.
- Un objet est une chose qui peut être appréhendée intellectuellement.
- Un objet est une chose vers laquelle on peut diriger des actions.
- Un objet est une abstraction d'une entité du monde réelle.

Objet


prix
créerFilm
insérerActeur
afficherFilm
titre : "air de famille"
metteur : p1
acteurs : p2, p3
durée : 1h30
prix { }
créerFilm { }
insérerActeur { }
afficherFilm { }

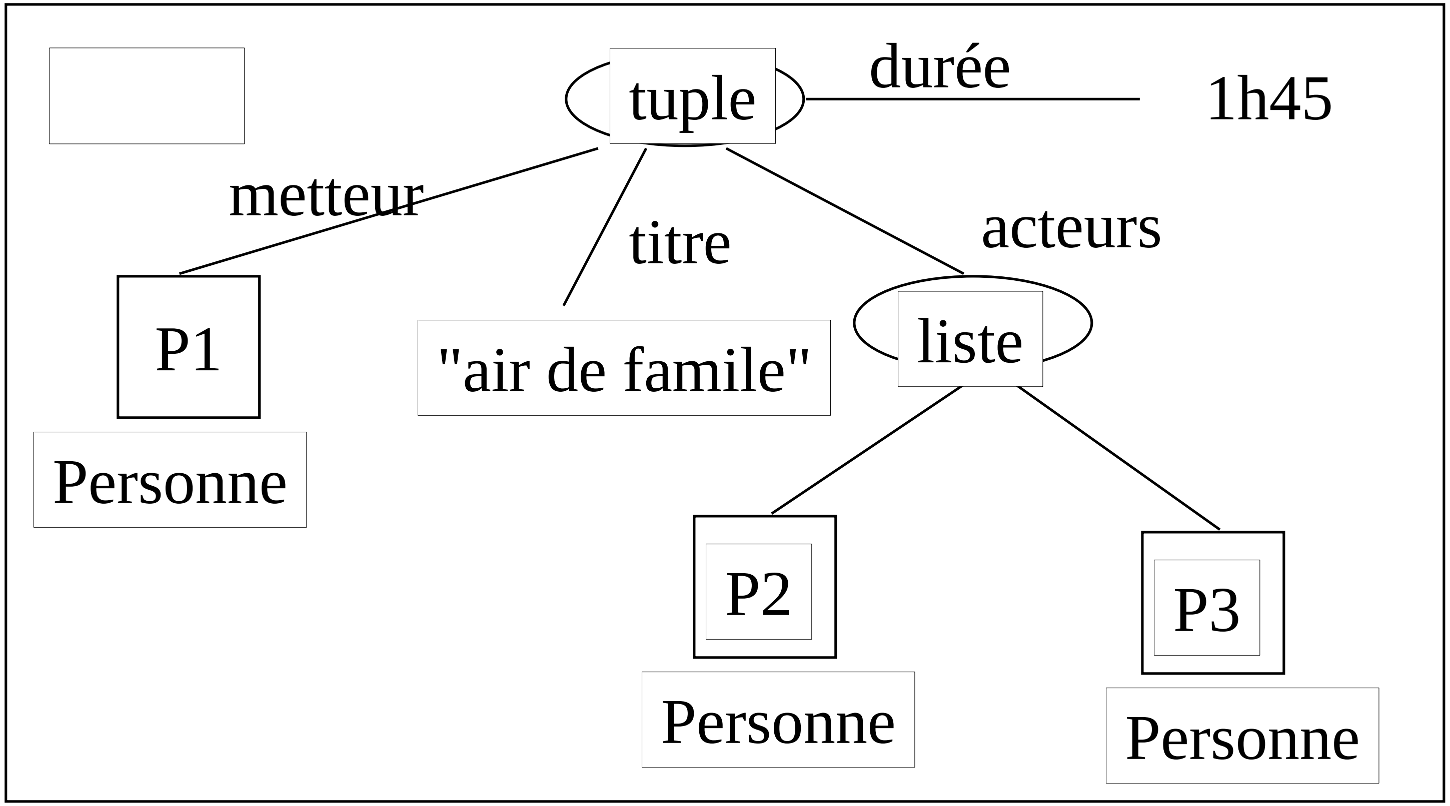
Objet et encapsulation : vision base de données

- Un objet encapsule la notion de données et de comportement.
- Les données sont la mémoire interne de l'objet.
- Les opérations sont associées à l'objet et peuvent lui être appliquées.

Identité d'objet

- Chaque objet peut recevoir une identité indépendante de son état interne (sa valeur).
- Les objets peuvent être partagés.
- Les objets peuvent se référencer mutuellement.
- Les identificateurs d'objet sont gérés par le système et ne sont pas connus du programmeur, celui-ci ne peut utiliser que des surnoms.

Objet complexe



C. Delobel, V. Benzaken, E. Waller

Objet complexe : avantages

- Auniveau conceptuel :
  - meilleure représentation des objets du monde réelle
  - objet composite (documents, textes, images,...)
- Au niveau des performances :
  - possibilité de calquer la représentation physique sur la représentation logique
  - évite la fragmentation de l'objet en plusieurs structures

Comportement d’un objet

- Le comportement d'un objet est la façon dont il agit et réagit en terme de changement d'état et de déclenchement des actions sur d'autres objets.
- Terminologie : *opérations, méthodes, fonctions membres*.
- Un message correspond au déclenchement d'une opération sur un objet.
- Le passage de message est le moyen par lequel les objets communiquent entre eux.

Les méthodes

- Un nom de méthode.
- Une signature : la liste des paramètres d'entrée et ou de sortie.
- Un corps de méthode qui constitue son implémentation. Celle-ci est dépendante d'un langage de programmation.

Classe

- Une classe décrit des objets qui ont la même structure et le même comportement.
- La description d'une classe comprend : la spécification (la liste des opérations associées à la classe) et l'implémentation (la structure des données ou le type, et l'implémentation des opérations).

Classe (exemple)

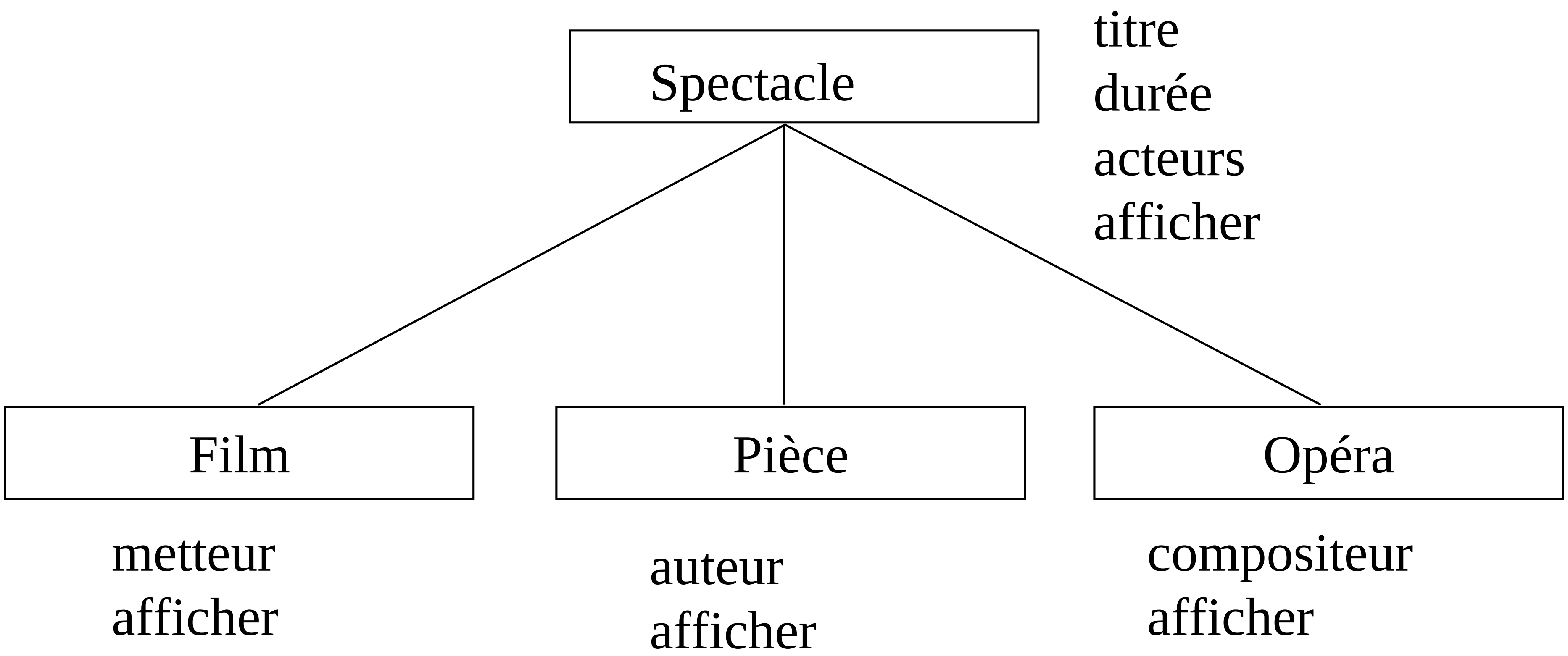
```
class Film type tuple (
        titre: String,
        metteur : Personne,
        acteurs : liste (Personne)
        durée : Time )
method créerFilm,
        prix : real,
        insérerActeur (Personne),
        afficherFilm

............
end;
```

Héritage

- Des structures voisines peuvent être décrites en utilisant le concept d'héritage.
- Par exemple, l'existence au sein d'un même serveur d'information de spectacles ayant des structures similaires ou des comportements voisins : films, pièces de théatre, concert, etc. peuvent être rapprochés.

Héritage (suite)



C. Delobel, V. Benzaken, E. Waller

Avantages de l’héritage

- Fournit une modélisation plus compacte et bien structurée du schéma.
- Evite la répétition du code en facilitant la ré-utilisation.
- Rend le code plus simple en plaçant chaque procédure à son niveau optimal.
- Permet une adaptabilité du schéma par l'insertion de nouvelles classes héritées.
- Attention à la définition rigoureuse du concept.

Concept de surcharge

- On a trois types de spectacle et si on veut afficher un type de spectacle il faut écrire une procédure spécifiques.
- Le programme d'affichage d'un ensemble de spectacles ressemble à :

```
pour chaque spectacle de EnsembledeSpectacle faire
        selon le cas de type de spectacle
                Film : afficher-Film (spectacle);
                Pièce : affciher-Pièce (spectacle);
                Opéra : affciher-Opéra (spectacle);
        fincas;
```

Surcharge (suite)

- Grâce au concept d'héritage et de surcharge on peut définir une seule opération d'affichage pour tous les types de spectacle.

pour tout spectacle de EnsembledeSpectacles faire
afficher (spectacle)

Envoi de message

- L'invocation d'une méthode m sur un objet o dans un programme est appelé envoi de message. L'objet o est le receveur.
- Certains langages (C++, Java, ..) n'utilisent, pour sélectionner l'implémentation, que le type du receveur.
- D'autres langages utilisent tous les paramètres invoqués.

Résolution tardive

- A cause de la redéfinition de l'opération d'affichage, à la compilation du programme on ne peut donner l'adresse du code correspondant, d'où la nécessité d'avoir un mécanisme de *résolution tardive*.
- L'association entre l'objet et le code exécuté ne peut se faire qu'à l'exécution.

Extensibilité

- Un système à objets est livré avec un certain nombre de classes prédéfinies. Ces classes peuvent être génériques (classe Date, Time) ou spécialisées pour un domaine applicatif (classe Point, Ligne, Surface, etc.)
- Le programmeur peut étendre cet ensemble, soit en définissant de nouvelles classes, soit en spécialisant certaines classes.

Persistance

- Dans les langages de programmation les données manipulées par un programme ne survivent pas à son exécution, sauf instruction particulière de stockage en mémoire secondaire.
- Cette action de persistance n'est pas indépendante du type de données. Pour le langage Cobol seul le contenu d'un type record peut être mis dans un fichier.
- Comment faire la distinction entre les données temporaires et persistantes ?

Persistance dans les bases à objets

- Persistance au niveau de la classe : tous les objets de la classe sont persistants.
- Persistance au niveau de l’objet :
  - ➤ à la création de l'objet
  - ➤ par une commande explicite (équivalent à un write)
- Par attachement à une racine de persistance.

Les collections

- Les langages de programmation ne possèdent pas en général un concept de collections. La structure de tableau est la plus proche pour représenter une collection.
- Les systèmes de base à objets permettent la définition et la manipulation de collections sous la forme d'ensemble, de liste et de multi-ensemble. Ces collections sont paramétrées par le type des éléments. Exemple : set (Personne), liste (tuple (texte : string, nom : string)).

Langage de requêtes à objets

- Les fonctionnalités : manipuler un objet, naviguer dans un objet complexe, appliquer des opérations, générer comme résultat des données complexes, fonctionner en mode interactif ou intégré à un langage de programmation, et être capable d'optimiser son interprétation.
- Le langage OQL défini par l'ODMG possède ces propriétés.

Propriétés de OQL

- modèle ODMG
- langage fonctionnel bien typé
- inclut des primitives pour manipuler des collections et des n-uplets
- permet l'invocation de méthodes
- les opérations de mise à jour ne sont pas incorporées à OQL
- peut être invoqué à partir d'un langage de programmation

Exemples de requête

```
select f.acteurs
from f in MesFilms
where f. titre = "air de famille"

select p. nom
from f in MesFilms
        p in f. acteurs
where f. titre = "air de famille"
```

```
select struct( nom : p. nom
                  pays : p. pays)
from f in MesFilms
        p in f. acteurs
where f. titre = "air de famille"
```


C. Delobel, V. Benzaken, E. Waller

Les systèmes existants

GemStone	Servio	USA
O2	O2 Technology	France
Objectivity	Objectivity	USA
ObjectStore	ODI	USA
Poet	Poet Software	Allemagne
Versant	Versant	USA

Conclusion : modèle

- Le modèle intègre les propriétes des langages à objets et des bases de données.
- Spécialisation, composition, encapsulation et identité d'objet sont des propriétés essentielles pour une modélisation fidèle des applications et des données.

Conclusion : systèmes

- Sur la base du benchmark 001 les systèmes à objets sont plus performants que les systèmes relationnels. Les systèmes à objets sont optimisés pour la navigation dans les objets complexes.
- La gestion de la concurrence sur des objets complexes dégrade les performances.
- Les techniques d'optimisation pour OQL sont plus complexes et la maturité des optimiseurs est entrain de rejoindre celle des systèmes relationnels.

Conclusion : développement

- L'ODMG est à sa deuxième version du standard et s'intègre dans la stratègie de l'OMG.
- La modélisation objet est largement utilisée par de nombreuses méthodologies.