# Journal de l'OSGeo

Le journal de la Fondation Open Source Geospatial

Volume 3 / Décembre 2007



## Compte rendu du FOSS4G 2007

**Intégration & Dévéloppement**

- Portable GIS : SIG sur une clé USB
- Génération Automatique d'Applications SIG / Base de données sur Internet
- db4o2D - Extension de Base de donnÃl'es Orientée Objet pour les types géospatiaux 2D
- Google Summer of Code pour la géomatique

**Intérêt thématique**

- Approche générique pour la gestion de standards de métadonnées
- Vers des services web dédiés à la cartographie thématique
- Interopérabilité pour les données géospatiales 3D
- Un Service Web orienté modèle pour une interopérabilité sèmantique améliorée
- Spatial-Yap : un système de base de données spatialement déductif

**Études de cas**

- Le Projet DIVERT : Développement de Télématiques Inter-Véhicules Fiables
- GRASS et la Modélisation des Risques Naturels
- Une Base de Données Spatiales pour lâĂŹIntégration des Données du Projet de Gestion des Ressources Naturelles du Rondonia
- GeoSIPAM : Logiciel libre et Open Source appliqué à la protection de lâĂŹAmazonie brésilienne
- Le Système de Suivi de la Déforestation Amazonienne

# db4o2D - Extension de Base de données Orientée Objet ...

**... pour les types géospatiaux 2D**

*Stefan Keller, traduit par Guillaume Sueur*

db4o signifie "database for objects", c'est-à-dire Base de données Orientée Objet. C'est un système de gestion de base de données orienté objet (OODBMS ou SGDBOO en français) écrit en Java et en .NET et de ce fait destiné à ces deux plateformes. Ce logiciel a été initialement publié en 2001 par db4Objects, Inc., et depuis lors s'est taillé une part majeure du marché de ce qu'on peut appeler les bases de données objet de deuxième génération. Il est disponible sous deux licences ("dual licensing model"), une licence OpenSource de type GPL pour l'utilisation personnelle et non-commerciale mais aussi une license commerciale.

Dans cet article, nous présenterons tout d'abord db4o. Dans le chapître *OODBMS et db4o*, nous en présenterons les avantages, limites et différences d'un point de vue conceptuel mais aussi selon une approche de développeur. Puis nous ferons une présentation de la thèse d'un étudiant nommée db4o2D. Enfin nous conclurons sur les perspectives du projet db4o2D.

## Premiers pas avec db4o

Commençons avec un peu de code source afin d'avoir une idée de la facilité avec laquelle on peut créer des objets persistants. Notre premier exemple met en oeuvre les quatre célèbres opérations de base en matière de bases de données : Créer, Lire, Mettre à jour, Supprimer (CRUD en anglais, pour Create, Read, Update and Delete). Cet exemple se base sur un objet Personne doté des attributs nom, prénom et date de naissance.

```
// 1: Initialisation d'un objet conteneur
ObjectContainer db= null;
try {
    // 2: Ouverture de la base db4o en mode integre
    db= Db4o.openFile("addressbook.yap");

    // 3: Creation et stockage de quelques 'Kellers'
    db.set(new Person("Brian", "Keller", 1960));
    db.set(new Person("Clara", "Keller"));
    db.set(new Person("Test"));
    db.commit();

    // 4: Recherche et lecture de tous les Keller
    // Avec une "Requete par l'exemple"
    ObjectSet result= db.get( \
        new Person(null, "Keller", 0));
    while (result.hasNext())
        System.out.println( \
            (Person) result.next());

    // 5: Mise a jour de l'age de Clara
    result= db.get(new Person("Clara", "Keller"));
    Person found= (Person) result.next();
    found.setYearOfBirth(1970);
    db.set(found);

    // 6: Suppression de la donnee de Test
    result= db.get(new Person("Test"));
    while (result.hasNext())
        db.delete(result.next());

    // 7: Enregistrement des modifications
    db.commit();
} finally {
    if (db != null)
        db.close();
}
```

*Fig. 1. Morceau de code montrant les opérations CRUD sur des objets personnes stockés dans un carnet d'adresses.*

Dans cet exemple, nous parcourons les étapes suivantes qui illustrent quelques opérations CRUD :

1. Cette instruction initialise un conteneur dans lequel db4o manipule l'ensemble des objets persistants de manière transactionnelle.
2. La deuxième instruction ouvre un fichier de base de données db4o. Nous proposons l'utilisation de l'extension .yap mais ce n'est pas standardisé. Il y a une explication officielle qui indique qu'il s'agit de l'abbréviation de 'yet another protocol' ('encore un nouveau protole' NdT). De manière non officielle, cette extension se réfère à Yap, une petite île de Micronésie. Comme indiqué, nous choisissons d'utiliser db4o en mode intégré, mais il existe aussi un mode client-serveur.
3. Avec la méthode `set()` d'un `ObjectContainer` trois personnes nouvellement créées sont stockées. Avec le `commit()` tous les objets créés,

modifiés ou supprimés sont alors synchronisés avec le fichier de base de données.
4. Dans cette étape nous parcourons tous les objets de la base de données dont le nom est égal à 'Keller', et bouclons ensuite sur le résultat. Dans cet exemple, la 'requête par l'exemple' est utilisée parmi les trois langages d'interrogation disponibles dans db4o.
5. Lors de la création de "Clara Keller" à l'étape 3 nous avons oublié la date de naissance lors de l'appel du constructeur (car il est poli de ne pas révéler à l'avance l'âge d'une dame). Maintenant nous désirons vraiment indiquer cette valeur, et donc nous devons trouver les objets tels que 'Clara Keller' et mettre à jour le premier que renverra la requête. Nous utilisons la méthode set() à nouveau et présumons qu'il existe une méthode `setYearOfBirth()` dans la définition de la classe Personne.
6. A des fins de démonstration, nous avons préalablement inséré des données de test qui vont maintenant être supprimées. Cette fois, tous les objets récupérés par la requête seront détruits par la méthode `delete()` de l'objet `ObjectContainer`.
7. Enfin nous terminons avec la méthode `commit()` et fermons la base de données.

Les seules définitions absentes de ce bout de code sont les instructions d'importation ainsi que celles de la classe Personne qui consiste en trois constructeurs, la méthode `setYearOfBirth()` et de manière optionnelle, une méthode `toString()` pour permettre d'écrire l'objet Personne proprement.

Cela suffit à démontrer la simplicité d'une application utilisant db4o. Maintenant nous allons présenter quelques exemples d'utilisation de SGBDOO, quelques fonctionnalités supplémentaires de db4o avant une présentation plus large de l'extension qui incorpore les types géométriques dans db4o.

# SGBDOO et db4o

## Exemples d'utilisation et fonctionnalités

Quand il n'y a qu'une seule application à la fois tournant sur une base de donnée et si c'est une application dédiée à la mobilité comme on en trouve dans les LBS (Location Based Systems, ou systèmes basés sur la localisation) ou dans les logiciels embarqués, cela peut valoir la peine d'essayer un SGBDOO de deuxième génération. Du fait de leur facilité à gérer les relations entre objets, les SGBDOO sont intrinsèquement bien adaptés lorsque des modèles objet complexes, des structures objet à plat ou de grandes arborescences objet sont de mise – ce qui est de fait souvent le cas dans les Systèmes d'Information Géographique (SIG) et les LBS.

Voici quelques-uns des fonctionnalités techniques de db4o :

- Mode intégré et mode client-serveur.
- Pas d'administration serveur lors de l'exécution. Les propriétés de la base de données sont controllées hors de l'application hôte.
- Besoin de peu d'espace disque pour les bibliothèques du programme, de même qu'en mémoire lors de l'exécution.
- Facilité d'utilisation : db4o utilise des interfaces de programmation d'application (API) réflectives à partir de Java et .NET. Il n'y a donc aucune annotation supplémentaire, aucun pré ou post-traitement (ingénierie de byte code), aucune sous-classe ou implémentation d'interface à effectuer.
- Méthodes de contrôle du lazy loading (NdT : technique qui consiste à ne charger que le strict nécessaire pour l'exécution du programme) des relations des objets incorporés.
- Outils de réplication en modules complémentaires.

Comme on pourrait s'y attendre de la part d'une base de données, db4o implémente des propriétés ACID (Atomicité, Consistence, Isolation et Durabilité) qui garantissent des transactions sûres : une transaction démarre quand on ouvre ou interroge la base, et se termine avec les méthodes `commit()` ou `rollback()`. Trois approches sont implémentées pour les requêtes : la requête par l'Exemple, les requêtes SODA (Simple Object Database Access, ou Objet Simple d'accès à la base de données) et les 'Requêtes natives'. La première a été mise en pratique plus haut. Les requêtes SODA sont plutôt comparables à celles utilisées dans les environnements ORM (Object Relational Mapping, ou correspondance Objet-Relationnel). Une partie de la théorie mise en oeuvre derrière les Requêtes natives provient d'un projet de Microsoft (LINQ 2007).

## Avantages

Parler d'environnement ORM nous amène à comparer les SGBDOO avec cette approche. db4o est très rapide si on en croit les bancs d'essai (Polepos 2007). Un argument conceptuel est qu'il n'y a pas d'erreur de correspondance objet/relationnel : aucun ajustement des types ou tailles des données (à moins de

le demander), aucune création d'un schéma relationnel séparé, pas de langage SQL et pas de requête textuelle SQL à gérer.

Il est donc important de mentionner que db4o propose une intéressante prise en charge des techniques de développement agile : c'est parce que les requêtes sont écrites dans le langage de l'application (Java, .NET) et sont donc sûres pour ce qui est du typage des données. Il y a aussi des fonctionnalités sympathiques d'évolution des schémas, et pas de SQL. Les ingénieurs logiciels ont la vie plus facile qu'avant car ils évoluent dans un monde objet à la différence des professionnels des bases de données.

### Limites

db4o n'est probablement pas adapté pour l'utilisation dans de grands entrepôts de données ou le data mining. Il est particulièrement non recommandé quand plusieurs applications accèdent à la base de données avec de nombreuses vues. Le fait que différents types de langages soient disponibles pour les requêtes montre qu'aucun langage standard et mature n'est disponible en comparaison au prédominant SQL du modèle relationnel. Les contraintes, telles que l'intégrité référentielle ne sont pas (encore) intégrées dans ces langages, sauf dans les Requêtes Natives qui implémentent simplement des fonctions callback. Pour finir, le manque de standardisation actuel a été reconnu. Des actions du Groupe de Management Objet (OMG) sont en cours de manière à avancer vers une nouvelle version de standard ODMG version 4.

## Projet db4o2D

PlaceLab (Intel 2006) — un projet Intel — propose des applications mobiles ou de bureau concernant le positionnement WLAN (Wifi). La base de données intégrée à ce logiciel est une base relationnelle open source embarquée. Un composant sensible ici est la gestion des points d'accès avec leur position respective. Cela nous sert de démonstration de base de données objet qui remplace la base relationnelle existante et stocke les géométries en tant qu'objets de première importance.

Dans un projet de thèse (db4o2D 2007), il a été décidé d'utilise db4o et d'adapter la très utilisée Java Topology Suite (JTS 2007). La JTS respecte le standard 'Simple Features (NdT : Objets simples)' (OGC 2006) qui définit quatre types d'attributs géométriques 2d (2.5D) : le point, la ligne, le polygones ainsi que des manipulations sûres sur ceux-ci.

Ainsi, dans la portion de code suivante (c.f. Fig 2.), il est montré comment les points d'accès sans fil sont créés (2), stockés (3) et relus (4). Un point contient une paire de valeur de coordonnées et comporte un système de références spatial et des unités de mesure par défaut.

```
// 1: Ouverture de la base de donnees db4o  (mode integre)
db= Db4o.openFile("poidb.yap");

// 2: Preparation de deux points JTS
GeometryFactory factory= new GeometryFactory();
double latitude1= 47.225571, longitude1= 8.822271;
Point pt1= factory.createPoint(new Coordinate(
        longitude1, latitude1));
double latitude2= 47.225582, longitude2= 8.822282;
Point pt2= factory.createPoint(new Coordinate(
        longitude2, latitude2));

// 3: Creation et stockage des points d'acces
db.set(new AccessPoint( \
    1001, "802.11g", "wep", 7, pt1));
db.set(new AccessPoint( \
    1002, "802.11b", "open", 11, pt2));
db.commit();

// 4: Parcours de tous les points d'acces
ObjectSet result= db.get(new AccessPoint());
while (result.hasNext()) {
    System.out.println( \
    (AccessPoint) result.next());
}
```

*Fig. 2. Portion de code avec des points d'accès sans fil qui contiennent une géométrie ponctuelle.*

## Conclusion

Le projet db4o2D deviendra un module complémentaire à db4o et est toujours en chantier. Bien que la couche existante d'accès à la base de données était correctement séparée, il est devenu évident que la maintenance du code était rendue plus courte, plus rapide et meilleure du fait de l'utilisation d'un pur SGBDOO.

Les travaux à venir incluent un index spatial et des tests de charge pour de grandes bases de données, ainsi qu'un mode client-serveur multitâche. Les requêtes SODA complexes sont un autre problème. Mais il faut noter que ce n'est pas un obstacle actuellement car les Requêtes Natives peuvent être utilisées d'ici là.

Dans le but de propager les technologies de base de données objet, un groupement à but non lucratif,

ODBMS.ORG (2007) a été créé. db4o est un des projets phares dans le domaine des bases de données objet et il tente de résoudre les problèmes bien connus dans le monde des logiciels embarqués, du développement logiciel, des LBS et des SIG. Sachant que ces problèmes sont des préoccupations depuis un certain temps, on pourrait dire que les SGBDOO de deuxième génération comme celui-ci sont en quelque sorte un retour vers le futur.

## Références

**db4o (2007) :** Page d'accueil de db4o et db4ojects, Inc. : www.db4o.com (dernière visite le 22 octobre 2007).

**db4o2D (2007) :** db4o2D project space, http://developer.db4o.com (dernière visite le 22 octobre 2007).

**Intel (2006) :** Le projet PlaceLab Project, PlaceLab — un système de localisation respectant la vie privée, www.placelab.org (dernière visite le 22 octobre 2007).

**JavaWPS (2007) :** Un système de positionnement en Java basé sur les signaux WLAN, www.gis.hsr.ch/wiki/JavaWPS (dernière visite le 22 octobre 2007).

**JTS (2007) :** Java Topology Suite, une API de fonctions et prédicats spatiaux 2D, www.vividsolutions.com/jts/JTSHome.htm (dernière visite le 22 octobre 2007).

**LINQ (2007) :** Le Projet LINQ, http ://msdn.microsoft.com/netframework/future/linq (dernière visite le 22 octobre 2007).

**ODBMS.ORG (2007) :** Un portail indépendant sur les technologies de bases de données objet, www.odbms.org, (dernière visite le 22 octobre 2007). OGC (2006) : Spécifications d'implémentation de l'information géographique de l'OpenGIS - Accès aux Objets Simples - Partie 1 : Architecture commune, www.opengeospatial.org/standards/sfa (dernière visite le 22 octobre 2007)

**PolePosition (2007) :** Un banc d'essai des bases de données OpenSource, www.polepos.org (dernière visite le 22 octobre 2007).

*Stefan Keller*
*Institute for Software and GISpunkt*
*University of Applied Sciences Rapperswil (UAS HSR)*
*CH-8640 Rapperswil, Switzerland*
*www.ifs.hsr.ch*
*Stefan Keller est un professeur en systèmes d'information enseignant les technologies de bases de données et la programmation.*
stefan.keller AT hsr.ch

The Open Source Geospatial Foundation, or OSGeo, is a not-for-profit organization whose mission is to support and promote the collaborative development of open geospatial technologies and data. The foundation provides financial, organizational and legal support to the broader open source geospatial community. It also serves as an independent legal entity to which community members can contribute code, funding and other resources, secure in the knowledge that their contributions will be maintained for public benefit. OSGeo also serves as an outreach and advocacy organization for the open source geospatial community, and provides a common forum and shared infrastructure for improving cross-project collaboration.

Publié par l'OSGeo, le Journal de l'OSGeo a pour objectif de publier les résumés des conférences, étude de cas et introduction, et les concepts lié Ãă l'open source et aux logiciels en géomatique.



Le *Jounal de l'OSGeo* est une publication de la *Fondation OSGeo*. La base de ce journal, le source des styles LaTeX 2ε a éé généreusement fournit par le bureau d'édition de GRASS et R.




**Journal en line :** http://www.osgeo.org/journal
**OSGeo Homepage :** http://www.osgeo.org
**Courrier postal :** OSGeo
PO Box 4844, Williams Lake,
British Columbia, Canada, V2G 2V8
**Téléphone :** +1-250-277-1621


**ISSN 1994-1897**