

Sistemi di persistenza alternativi, una overview

Ugo Landini
Senior Java Architect
Sun Microsystems



RDBMS

- RDBMS e programmazione ad oggetti sono basati su due paradigmi fondamentalmente differenti
 - Impedence mismatch
 - Paradigm cross
 - Paradigm shift
- La maggior parte delle applicazioni di business si scontrano con questo problema.

Paradigm shift

- Ma la verità è che non è assolutamente detto che si debba per forza usare un RDBMS, ad ogni problema il miglior tool
- Come evitare il paradigm shift:
 - Non usare un sistema di persistenza!
 - Usare un fw per il mapping
 - Usare un database ad oggetti
 - Usare un database gerarchico
 - Usare un sistema di persistenza di più basso livello

Non persistere

- Non usare un sistema di persistenza!
 - Usare solo le collection
 - ... vabbè ...
 - spesso si usano dei RDBMS per le configurazioni del software. Alternative migliori sono:
 - File di Properties
 - Preferences API
 - XML (Spring, ecc.)
 - LDAP

FW per mapping

- Usare un fw per il mapping
 - EJB Entity
 - Hibernate
 - JDO
 - ... e molti altri
- In teoria si dovrebbe beneficiare del meglio dei due mondi, in pratica ... no.

Object Database

- Usare un database ad oggetti
- Nessun impedance mismatch
- Dovevano essere la panacea di tutti i mali, ma...
- ...poco diffusi, molto costosi, poco conosciuti, unproven, poco usati.

Database gerarchico

- ... ma esistono ancora?
- LDAP!
- Perfetto per le configurazioni, anche e soprattutto a livello enterprise
- Ma non solo, da tenere in considerazione ogni volta che ci sono rapporti molto alti di letture / scritture

Sistemi alternativi

- Molto spesso quello che serve è avere una hash table (ma persistente), dove eventualmente costruire ricerche minimamente più sofisticate
- Tipicamente basati su meccanismi di serializzazione
- In ordine di semplicità (non necessariamente in ordine inverso di potenza) le alternative sono: File system, Prevayler, Swigom DB, JDBM, Berkeley DB
JE

File System

- Serializzare un oggetto (per esempio in XML, con xStream) e salvarlo in una directory (il database) con il nome uguale alla chiave
- Eccezionale con file system come ReiserFS
- Con file system limitati (NTFS, ma anche ext) ci sono diversi problemi, parzialmente mitigabili complicando un po' (hashing dei nomi)
- Si può espandere con l'uso di Lucene per l'indicizzazione di altri campi

Prevayler

- Salvare un oggetto in una normale collection tramite un command pattern, e salvare in un log il command stesso
- In questo modo si può replicare lo stato della collection dall'istante zero.
- Ovviamente ogni tanto si serializza l'intera collection per riportare all'istante zero i log
- Le prestazioni sono eccezionali, ma ha la limitazione di mantenere l'intera base dati in RAM
- Ci sono problemi di clustering
- <http://www.prevayler.org>

JDBM

- JDBM 1.0 (dopo tre lunghi anni a 0.12)
- Hashtable persistente, usa un btree per l'indexing
- Molto veloce
- Molto semplice da usare
- Zero amministrazione
- <http://jdbm.sourceforge.net/>

Berkeley DB JE

- Berkeley DB JE (Java Edition)
- Berkeley è un “database” storico, la versione Java è molto più recente
- La JE, porting diretto della versione C, ha più o meno le stesse prestazioni
- Semplice da usare, in più ci sono delle collection simili alle collection API
- Amministrazione minimale, ma non zero
- E' GPL!!! Si paga (e tanto) se usato in software commerciale, come l'LDAP di Sun
- <http://www.sleepycat.com>

Swigom DB

- Yet Another One Persistent Hash
- Usa un `RandomAccessFile` con gli oggetti serializzati sequenzialmente (compressione opzionale)
- Usa Lucene per indicizzare gli offset rispetto alle chiavi
- Semplice da usare, in più ci sono delle collection simili alle collection API
- Amministrazione zero (speriamo)
- Licenza Apache, prestissimo su <http://www.swigom.org>

Quando si può evitare un RDB?

- Se valgono tutte le seguenti:
 - Se non si ha del legacy
 - Se non servono moli di informazioni nella categoria HUGE (centinaia di Gb o Tera)
 - Se non serve fare delle query “libere” *
 - Se non serve partecipare a transazioni JTA**
 - Se si conosce la programmazione OO
 - Se nella vita non si è mai soddisfatti

* database ad oggetti e gerarchici hanno queste feature

** Berkeley è JTA compliant, ed anche diversi database ad oggetti