Simple Transactional Data Base (STDB)

Andreu Moreno, Alex Blanquer

Escola Universitària Salesiana de Sarrià (EUSS)
Passeig Sant Joan Bosco, 74
08017 Barcelona
amoreno@euss.es

Abril de 2005

Resum

Les bases de dades s'utilitzen de forma habitual en el món informàtic. Això no és així en el cas particular del programari pensat per sistemes encastats. La base de dades Simple Transactional Data Base (STDB) pretén millorar aquesta situació. Actualment l'opció llprogramatiure més destacada (Berkeley DB [1]) està implementada en una biblioteca compartida. El fet que estigui sota la llicència GPL condiciona la seva utilització. Aquest article presenta el projecte STDB com una base de dades simple amb suport transaccional pensada pel món encastat i sota llicència LGPL.

1 Introducció

La necessitat d'emprar una base de dades en un aplicatiu depèn de molts factors. Entre ells la quantitat d'informació que s'hagi de gestionar, les garanties d'integritat davant situacions adverses, les exigències en l'accés a la informació,..etc. En el cas del món dels sistemes encastats no hi ha massa tradició a utilitzar una base de dades. Els motius els podríem trobar en que les aplicacions no acostumen a tenir un volum molt gran d'informació a gestionar, les exigències mínimes de memòria no ho permeten o en que no hi tradició en la seva utilització, segurament en molts casos per manca de formació. Però, el que sí té habitualment un sistema encastat és la necessitat de garantir la integritat de la informació.

Fem una simple distinció entre les bases de dades relacionals basades en un servidor que centralitza les consultes realitzades en un llenguatge com SQL, o bé les bases de dades basades en fitxer, on es guarda la informació en format clau-valor. La manca de memòria de molts sistemes encastats fa que el primer tipus es descarti des d'un principi, i per tant l'elecció es quedi

en haver de triar si es fa servir una base de dades de fitxer o bé es realitza una gestió de la informació a mida per l'aplicació. Aquesta segona opció és molt costosa si es vol fer bé oferint totes les garanties d'integritat que són exigibles en la majoria de sistemes encastats.

El sector de les bases de dades de fitxer en el programari lliure està monopolitzat per Berkeley DB [1]. Berkeley DB és una base de dades lliure que ofereix tots els requeriments necessaris per un sistema encastat, però la dificultat sorgeix precisament en la seva llicència, la GPL. Resulta que la base de dades està continguda en una biblioteca compartida i per tant qualsevol aplicació desenvolupada amb aquesta base de dades ha de ser també GPL. El distribuïdor de Berkeley DB [1] ens ofereix una alternativa mitjançant un sistema de doble llicència per poder editar el codi de l'aplicació en llicència propietària. Aquesta opció és extremadament cara i per tant es descarta en la majoria d'ocasions.

En resum, la problemàtica està en que només tenim bàsicament una opció, Berkeley DB [1] i la seva utilització força que l'aplicació hagi de ser lliure. Pot xocar per una persona no introduïda en el món del programari encastat la preocupació per no editar el codi en format lliure. La veritat és que el codi desenvolupar per una aplicació encastada no gaudeix a la pràctica dels beneficis que la resta de programari pel fet de ser lliure. Cal pensar que es un codi totalment específic per un maquinari molt contret i per tant de difícil aplicació general. Per tant hi ha una manca d'incentius en quan a retorn per part d'altres programadors que s'hi vulguin involucrar, però alhora, es posa moltes facilitats a que la resta d'empreses del sector pugui duplicar el producte.

Nosaltres hem desenvolupat una base de dades totalment nova, inspirada en la interfície de Berkeley DB [1] però sota llicència LGPL. D'aquesta forma el programador té tota la llibertat alhora d'utilitzar-la.

2 Simple Transactional Data Base (STDB)

La base de dades Simple Transactional Data Base (STDB) s'ha desenvolupat partint dels següents requeriments:

- Emmagatzemen informació en format tupla clau - valor. Cada un dels components de la tupla pot ser qualsevol informació. La clau permet fer la recerca de la tupla.
- Ha de permetre introduir, modificar i esborrar tuples.
- Ha de tenir suport transaccional. Ha d'entrar dins dels paràmetres que es considera una base de dades ACID (*Atomicity Consistency Isolation Durability*)

  Atomicity. Garanties de les transaccions són atòmiques, és a dir, o hi són senceres o no hi són.

Consistency. Mai es deixa una feina a mitges, i en el cas que per raons de força major s'hagés de fer (per exemple per tall d'alimentació), en reiniciar-se ha de poder desfer el que ha deixat a mitges.

Isolation. Mantenir les transaccions aïllades unes de les altres. La STDB està pensada per ser utilitzada per només un procés.

Durability. Es registra tots els canvis per tal poder garantir la fiabilitat de les dades davant interrupcions inesperades.

En base a aquests requeriments s'ha afrontat un desenvolupament en llenguatge C++ i s'ha fer un us intensiu dels templates continguts a la llibreria STL del C++. En els següents apartats es fa una descripció més detallada.

2.1 Estructures de dades

Des d'un punt de vista estructural, la base de dades té un fitxer base, anomenat *fitxer de base de dades*, que conté una imatge de les tuples emmagatzemades l'últim cop que es va regenerar. També té un *fitxer de log*, que conté un registre de totes les operacions que s'ha realitzar sobre la base de dades des de l'última regeneració del *fitxer de base de dades*. Aquest fitxer és vital per poder implementar el suport transaccional. De forma periòdica es traspassa el registre del *fitxer de log* al *fitxer de bases de dades* (operació de regeneració), quedant el *fitxer de log* buit a l'espera d'enregistrar la propera operació. Altrament s'ha optat per tenir simultàniament tota la informació de la base de dades en memòria RAM. Això limita les possibilitats de volum d'informació però alhora accelera molt les consultes. Així en tot moment tenim tota la informació en memòria, i per tant les consultes es realitzen sense accés als fitxers. Quan hi ha canvis queden registrats al *fitxer de log* i periòdicament es regenera el *fitxer de base de dades*. Quan el sistema s'inicialitza es reconstrueix la imatge de la base de dades en memòria partint del *fitxer de base de dades* i aplicant tots el canvis registrats en el *fitxer de log*.

Internament el programa utilitza una llista i una pila de la llibreria STL per tal de poder oferir el suport transaccional. Puig en tot moment se'ns pot demanar que avortem voluntàriament la transacció actual o bé que la completem. En el primer cas una pila va bé per desfer les operacions en ordre invers al que foren realitzades. La llista s'utilitza per guardar la seqüència d'ordre que cal posar en el fitxer de log quan hi ha un final de transacció.

2.2 Robustesa

S'han esmerçat molts esforços en fer la base de dades el més robusta possible davant talls d'alimentació. En primer lloc el fet d'oferir suport transaccional ja en condiciona a que les transaccions incompletes han de ser eliminades

quan el sistema es torna a arrencar. En segon lloc preocupava una caiguda d'alimentació quan el sistema està executant l'interior del codi de la base de dades. S'han estudiat tots els punts conflictius i en cada cas s'ha donat al sistema la funcionalitat que pugui mantenir la garantia transaccional. Finalment s'ha posat especial èmfasi en el moment en que es regenera el *fitxer de base de dades*. Una interrupció en aquest instant podria fer perdre aquest fitxer. Per solucionar aquest problema, aquesta operació es fa sobre una copia del *fitxer de base de dades* per tal de discriminar en el moment de l'arrencada si venim d'un procés de regeneració interromput.

No s'han tingut en compte situacions com la pèrdua o corrupció d'alguns dels dos fitxers. Aquests casos es consideren propis d'un nivell superior de robustesa i s'ha deixat per una futura versió.

2.3 Antecedents

A part de Berkeley DB [1], els antecedents del projecte els trobem en els projectes *cdb* i *rdb*. *cdb* (*http://cr.yp.to/cdb.html*) és un projecte de base de dades constant. D'aquest projecte se n'ha agafat la idea de tenir com estructurada un fitxer de base de dades i alhora una imatge en RAM. El *rdb* (l'hipervincle a aquest projecte està perdut en els moments d'escriure aquest article) és precisament una millora de l'anterior amb capacitat de modificació amb l'ajut d'un fitxer de registre. Els dos projectes estan implementats en C amb un estil de programació a molt baix nivell però alhora maximitzant el rendiment.

El nostre projecte podriem dir que ha agafat la filosofia del *rdb* però s'hi ha afegit el suport transaccional i s'ha programat amb el llenguatge C++ orientat a objectes. Això ha facilitat que el codi sigui més mantenible a costa de rendiment.

2.4 Punt de vista del programador

Des del punt de vista del programador tot el codi de la base de dades està contingut en una biblioteca compartida. El programador disposa d'un fitxer capçalera amb les definicions dels objectes que ha d'utilizar i el prototipus de les funcions membres. La llicència és la LGPL i per tant el programador té total llibertat per triar la llicència de la seva aplicació particular.

3 Exemple

En el Figura 1 presentem un breu codi que exemplifica l'ús de la base de dades. El primer que cal fer és crear un objecte de la classe *Tstdb*. Aquest objecte és la base de dades i com a paràmetre en el constructor li hem de passar el *fitxer de base de dades*, el *fitxer de log* i el temps i el número de operacions màxims per forçar una regeneració. A continuació podem insertar

```
Tstdb db("prova.db","prova.log", 10000, 2);
long key1 = 1,data1 = 2;
long key2 = 2,data2 = 4;
long key3,data3;
long key4,data4;
unsigned int ldata3, ldata4;

db.Insert((char *)&key1, sizeof(key1), (char *)&data1, sizeof(data1));
db.Insert((char *)&key2, sizeof(key2), (char *)&data2, sizeof(data2));

db.Commit();

db.Reorganize();

db.Fetch((char *)&key3, sizeof(key3), &data3, &ldata3))
db.Fetch((char *)&key4, sizeof(key4), &data4, &ldata4))
```

Figura 1: Exemple d'utilització de la base de dades STDB

varies tuples (*Insert*), confirmar transacció (*Commit*), modifacar-ne (*Update*) d'insertades, eliminar-ne (*Delete*), forçar la regeneració (*Reorganize*),...

4 Conclusions

En aquest article hem presentat el desenvolupament d'una base de dades de fitxer amb suport transaccional. L'objectiu ha estat disposar en el món del programari lliure d'una base de dades que amb aquestes funcionalitats amb llicència LGPL. Les proves realitzades fins ara han estat força satisfactòries. Per tal de publicar-lo s'ha donat d'alta un projecte al portal SourceForge [3] per contenir tant el repositori dels fonts com una breu web [2].

Referències

[1] Berkeley DB. `http://www.sleepycat.com`.

[2] Simple transactional data base (STDB). `http://stdb.sourceforge.net`.

[3] SourceForge. `http://sourceforge.net`.