Tecnologie di Sviluppo per il Web

Programmazione su Basi di Dati: Tecniche di Programmazione

versione 3.0

G. Mecca – Università della Basilicata – mecca@unibas.it

Sommario

- Introduzione
- Identificatori
  - ⇨La Tecnica “High-Low”
- Connessioni
  - ⇨Pool di Connessioni
  - ⇨Caching dei Prepared Statements
- Riassumendo

Introduzione

- Finora
  - ⇨varie semplificazioni nella tecnica di sviluppo
- In particolare
  - ⇨semplificazioni nell'utilizzo delle connessioni
  - ⇨semplificazioni nella scelta delle chiavi
  - ⇨in questa lezione rimuoviamo le semplificazioni
  - ⇨progetto di riferimento: it.unibas.aci2

Identificatori

- Nell'esempio
  - ⇨tutti i componenti hanno un identificatore naturale (codice fiscale, targa)
  - ⇨può essere usato come valore per la chiave primaria nella tabella corrispondente
- In altri casi questo non succede
  - ⇨es: schedine, partite di calcio
  - ⇨in questi casi è il sistema che deve scegliere un valore per la chiave primaria

Identificatori

- Più in generale
  - ⇨anche quando gli identificatori naturali esistono, è spesso INOPPORTUNO usarli
- Infatti
  - ⇨gli identificatori naturali (es: codice fiscale, numero di telefono, targa) possono cambiare
  - ⇨cambiare le chiavi in un sistema informativo di grandi dimensioni ha spesso conseguenze negative

Identificatori

- Inoltre
  - ⇨i DBMS tipicamente sono ottimizzati per trattare chiavi intere e non stringhe
- Linea guida
  - ⇨è importante che le chiavi primarie nella base di dati non abbiano significato nella realtà
  - ⇨gli identificatori dovrebbero essere sintetici
  - ⇨e di tipo intero

Identificatori

- Come attribuire gli identificatori ?
  - ⇨varie possibilità
- Due approcci principali
  - ⇨approcci dipendenti dal DBMS (es: attributi autoincrementanti, attributi sequenza ecc.)
  - ⇨approcci indipendenti dal DBMS
  - ⇨è opportuno evitare i primi, per cui ci concentreremo sui secondi

Identificatori

- Problema
  - ⇨ogni volta che inserisco una nuova ennupla in una tabella (es: automobile), ottenere un numero intero che sia un identificatore unico per la ennupla nella tabella
  - ⇨ancora meglio se un identificatore unico in tutto il sistema informativo (surroga l'OID)
- Soluzione semplice
  - ⇨utilizzo un attributo intero
  - ⇨ogni volta calcolo il massimo della colonna e uso il valore calcolato aumentato di 1

Identificatori

```
create table proprietari (
  id integer not null primary key,
  codiceFiscale char(16) not null unique,
  nome varchar(50) not null,
  cittaDiResidenza varchar(50),
  annoPatente integer
)
```

- Esempio
  - start transaction;
  - select max(id) from proprietari for update;
  - nuovoid = max(id) + 1
  - insert into proprietari values (nuovoid, *cf, nome, cittares, anno*)
  - commit; (nota: il tutto da fare con JDBC)
- Attenzione
  - il livello di isolamento deve essere S2PL

Identificatori

- Svantaggi di questa tecnica
  - due accessi alla base di dati per ciascun inserimento
  - lo stesso svantaggio si ha con i campi autoincrementanti
  - inoltre, la colonna id diventa un collo di bottiglia (tutte le transazioni acquisiscono lock su tutte le ennuple della tabella)

La Tecnica High-Low

- Un miglioramento
  - ⇨la tecnica “High-Low” (Scott Ambler, 1998)
- Idea
  - ⇨per la generazione degli identificatori utilizzo una tabella a sé stante (idtable)
  - ⇨concedo gli identificatori in blocchi ai client in modo che una volta acquisito un blocco non sia necessario fare altre transazioni sulla tabella fino all’esaurimento del blocco

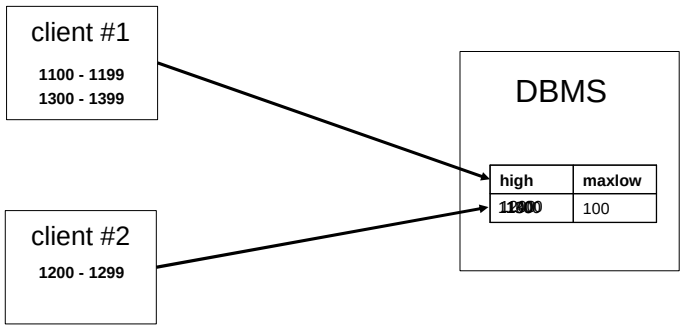
La Tecnica High-Low

- Struttura dell’identificatore
  - ⇨numero intero
  - ⇨somma (o concatenazione) di due numeri
  - ⇨parte alta (“high”): prelevata dalla tabella idtable in modo che sia unica per ogni transazione
  - ⇨parte bassa (“low”): assegnata dal client

La Tecnica High-Low

- Funzionamento
  - all'inizio di ogni sessione il client acquisisce da idtable un valore unico per la parte alta (es: 7200)
  - e il massimo valore utilizzabile per la parte bassa (maxlow, es: 100) – dim. del blocco
  - in questo modo ha acquisito in via esclusiva tutti gli id da 7200 fino a 7299
  - una volta esaurito il blocco chiede una nuova parte alta e di conseguenza acquisisce un nuovo blocco

La Tecnica High-Low



La Tecnica High-Low

- Vantaggi
  - ⇨il sistema è indipendente dal DBMS
  - ⇨è necessario effettuare una transazione per blocco e non una transazione per id
  - ⇨se maxlow = 100 una transazione ogni 100 inserimenti
  - ⇨idtable non è un collo di bottiglia
  - ⇨inoltre è facilmente possibile avere id unici per tutto il sistema informativo

La Tecnica High-Low

- In concreto, nel DBMS
  - ⇨è necessaria una tabella aggiuntiva idtable con un'unica ennupla
  - ⇨attributo high di tipo bigint
  - ⇨attributo maxlow di tipo int

```
create table idtable (
   high bigint not null primary key,
   maxlow integer not null
);

insert into idtable values (1000, 100);
```

La Tecnica High-Low

>> GeneratoreDiId

- Nell'applicazione
  - componente GeneratoreDiId ("IdBroker")
  - si tratta di un singleton
  - metodo public long getNuovoId()
- Ogni volta che viene richiesto un nuovo id
  - se ci sono id disponibili nel blocco restituisce this.high + this.low e incrementa this.low
  - se il blocco è esaurito (this.low = maxlow), richiede un nuovo blocco a idtable

La Tecnica High-Low

>> inserimento proprietario
>> inserimento automobile

- Utilizzo del generatore
  - tutti gli oggetti del modello hanno una proprietà in più: private long id;
  - ogni volta che c'è da fare un inserimento il metodo doInsert() del DAO richiede al generatore un nuovo id da assegnare all'oggetto
  - effettua l'inserimento usando l'id come chiave primaria e per le chiavi esterne
  - eventuali altre chiavi (es: codice fiscale) sono utilizzate solo per le ricerche

La Tecnica High-Low

- Attenzione
  - ⇨l'id viene attribuito ad un oggetto solo dopo l'inserimento
  - ⇨prima dell'inserimento è "indefinito"
  - ⇨di conseguenza l'oggetto è incompleto e non è possibile effettuare operazioni
  - ⇨es: assegnare un'automobile ad un proprietario senza id

La Tecnica High-Low

- Nota
  - ⇨il valore di high nella tabella idtable deve essere di tipo bigint
  - ⇨gli identificatori devono essere di tipo long
- Inoltre
  - ⇨l'id numerico è un identificatore puramente sintetico e di uso interno all'applicazione
  - ⇨non deve essere visualizzato sugli schermi, nè utilizzato per le ricerche degli utenti

La Tecnica High-Low

- Prestazioni
  - si tratta di un buon compromesso tra indipendenza dalla piattaforma ed efficienza
- Per ottenere le prestazioni migliori
  - è opportuno trovare il giusto compromesso nel valore di maxlow (dimensione dei blocchi concessi ai client)
  - un maxlow troppo alto “consuma” id se il client non utilizza integralmente un blocco
  - un maxlow troppo basso aumenta il numero di transazioni e peggiora le prestazioni

Connessioni

- Gestione delle connessioni
  - in generale sarebbe opportuno utilizzare un pool di connessioni
  - in aci2 viene fornita una versione del componente DataSource basato su un pool di connessioni
- Ma...
  - è opportuno avere la possibilità di utilizzare anche una DataSource semplice, senza pool

Connessioni

- Perchè due DataSource ?
  - ⇨alcuni DBMS tendono ad essere fragili rispetto all'apertura di pool di connessioni
  - ⇨in alcuni casi eccezioni non gestite possono costringere al riavvio del DBMS
  - ⇨durante la fase iniziale dello sviluppo, prima di avere effettuato il debugging dei DAO, può essere utile utilizzare una DataSource semplice, per abilitare il pool in produzione

Connessioni

- Problema
  - ⇨è necessario poter utilizzare due implementazioni diverse dello stesso componente nella stessa applicazione
- Soluzione
  - ⇨definire un'interfaccia comune
  - ⇨sviluppare le due implementazioni
  - ⇨centralizzare la logica della creazione della DataSource in una classe apposita

Connessioni

- In particolare, in aci2
  - ⇨viene definita una interfaccia DataSource
  - ⇨è possibile utilizzare una DataSource ordinaria (DataSourceSemplice)
  - ⇨oppure una DataSource basata su un pool di connessioni (DataSourcePool)
  - ⇨le due implementazioni implementano la stessa interfaccia DataSource

Connessioni

- Per la creazione della DataSource
  - ⇨una DataSourceFactory: una fabbrica di DataSource
  - ⇨si tratta di un singleton con un metodo public DataSource getDataSource()
- La selezione della implementazione
  - ⇨avviene attraverso un file di configurazione

Connessioni

>> DataSourceFactory
>> DataSource
>> DataSourceSemplice
>> pool.properties
+ esecuzione dei test

- File di configurazione
  - ⇨/risorse/pool.properties
- All'inizializzazione
  - ⇨la DataSourceFactory preleva i parametri dal file e li utilizza per inizializzare il pool
- Vantaggio
  - ⇨è possibile cambiare il DBMS, la base di dati o riconfigurare il pool senza ricompilare

Pool di Connessioni

- c3p0
  - ⇨un gestore di pool di connessioni open source molto usato
  - ⇨molto semplice e molto efficace
  - ⇨disponibile su www.mchange.com
  - ⇨package com.mchange.v2.c3p0
  - ⇨nel classpath deve essere disponibile c3p0.jar

Pool di Connessioni

- Scopo del pool
  - ⇨fornisce varie implementazioni di data source pronte per l'uso
- Attenzione
  - ⇨le implementazioni fornite da c3p0 implementano l'interfaccia javax.sql.DataSource
  - ⇨interfaccia del package javax.sql

Pool di Connessioni

- javax.sql.DataSource
  - ⇨interfaccia standard stabilita dalla Sun per DataSource evolute con pool di connessioni
- Metodo fondamentale
  - ⇨Connection getConnection()
- Altri metodi
  - ⇨get/setLogWriter
  - ⇨get/setTimeout

Pool di Connessioni

- Le implementazioni di c3p0
  - ⇨UnpooledDataSource: implementazione non basata su pool
  - ⇨PoolBackedDataSource: basata su pool
- Creazione di una datasource con pool
  - ⇨molto semplice; è sufficiente creare un oggetto di tipo ComboPooledDataSource
  - ⇨e fornire i valori (stringhe) di driver, databaseUri, nomeUtente e password

Pool di Connessioni

- Come tutti i pool di connessioni
  - ⇨c3p0 consente di specificare molti parametri di configurazione per il pool
  - ⇨es: numero minimo e massimo di connessioni, timeout, ecc.
- In effetti
  - ⇨tutti i parametri di c3p0 sono opzionali
  - ⇨il package fornisce valori di default accettabili nella maggior parte dei casi

Pool di Connessioni

- Specificare i parametri di configurazione
  - ⇨per sovrascrivere i valori di default è possibile creare un file c3p0.properties
  - ⇨accessibile in una cartella radice del classpath (/c3p0.properties)
  - ⇨NOTA: non è possibile metterlo nella cartella risorse (non sarebbe una cartella radice), per cui è necessario modificare leggermente il file di build di Ant

Pool di Connessioni

- I parametri di configurazione principali
  - ⇨maxIdleTime
  - ⇨minPoolSize
  - ⇨maxPoolSize
  - ⇨maxStatements
- maxStatements
  - ⇨abilita e disabilita la cache dei PreparedStatements (>>)

Pool di Connessioni

- maxIdleTime
  - ⇨tempo massimo prima che una connessione sia considerata scaduta e sia chiusa
  - ⇨serve ad evitare che il DBMS la chiuda unilateralmente allo scadere del timeout
- minPoolSize e maxPoolSize
  - ⇨dimensione minima e massime del pool di connessioni a seguito di aperture e chiusure successive

Pool di Connessioni

>> c3p0.properties

- Nel file di configurazione
  - ⇨è possibile anche specificare i parametri per il logging
- Logging in c3p0
  - ⇨c3p0 cerca log4j.jar nel classpath se non istruito diversamente; se non lo trova solleva eccezioni
  - ⇨per evitare di utilizzare log4j, è opportuno chiedere di reindirizzare il logging su System.err (FallbackMLog)
  - ⇨e abbassare il livello di logging da INFO a SEVERE

Pool di Connessioni

- Attenzione
  - ⇨nell'applicazione aci2 non viene direttamente utilizzata l'interfaccia javax.sql.DataSource per due ragioni
- Ragione n. 1
  - ⇨non prevede i metodi (comodi) di chiusura
- Ragione n. 2
  - ⇨evitare di implementare nella DataSource semplice i metodi per logWriter e timeout

Pool di Connessioni

- Di conseguenza
  - ⇨è necessario adattare le data source fornite da c3p0 all'interfaccia desiderata (it.unibas.aci2.persistenza.DataSource)
- Questo compito
  - ⇨è svolto da una classe "adapter"
  - ⇨it.unibas.aci2.persistenza.DataSourcePool

Pool di Connessioni

>> DataSourcePool

- Riassumendo
  - una DataSourceFactory con un file di configurazione /risorse/pool.properties
  - una interfaccia DataSource
  - una implementazione DataSourceSemplice
  - una implementazione DataSourcePool che incapsula una ComboPooledDataSource con il suo file di configurazione /c3p0.properties

Caching dei Prepared Statements

- Caching
  - c3p0 fornisce una cache di PreparedStatement
  - è possibile preparare uno statement su una connessione del pool
  - lo statement viene inserito nella cache
  - se viene richiesto di preparare di nuovo lo stesso statement sulla stessa connessione, viene restituito quello nella cache

Caching dei Prepared Statements

- Idea
  - ⇨nella fase iniziale il vantaggio di prestazioni è relativamente basso (gli statement vanno ripreparati sulle varie connessioni)
  - ⇨lo stesso metodo viene eseguito più volte di seguito con connessioni diverse
  - ⇨utilizzando sistemanticamente prepared statements, dopo un po' gli statements saranno stati preparati su TUTTE le connessioni del pool
  - ⇨da quel momento in poi il vantaggio di prestazioni è notevole

Caching dei Prepared Statements

- Nel caso di c3p0
  - ⇨il caching degli statement è disabilitato per default
  - ⇨maxStatements = 0
  - ⇨specificando un valore diverso da 0 il caching viene abilitato

Riassumendo

- Lo strato di persistenza
  - ⇨DAO per il componenti (con tutti i metodi raddoppiati, e in alcuni casi quadruplicati)
  - ⇨GeneratoreDiId
  - ⇨DataSourceFactory, DataSourceSemplice, DataSourcePool e relativi file di configurazione
  - ⇨DAOException
  - ⇨Test di regressione

Riassumendo

- In effetti
  - ⇨si tratta di una quantità notevole di codice da scrivere
  - ⇨peraltro ripetitivo e abbastanza insidioso per via delle istruzioni SQL e delle transazioni
- Sarebbe ideale
  - ⇨avere a disposizione un framework che riduca (idealmente azzeri) la necessità di scrivere questo codice

Riassumendo

- Introduzione
- Identificatori
  - ⇨La Tecnica “High-Low”
- Connessioni
  - ⇨Pool di Connessioni
  - ⇨Caching dei Prepared Statements
- Riassumendo

Termini della Licenza