Capitolo 1

Introduzione

Un esperto è una persona che ha fatto
tutti gli errori possibili
in un campo specifico.
– Niels Bohr

Ho rifiutato il mio primo lavoro in SQL.
Poco dopo essermi laureato in scienze dell'informazione presso l'università della California, sono stato contattato da un manager che lavorava presso l'università e mi conosceva per le mie attività nel campus. Possedeva una società di software in fase di avvio, che stava sviluppando un sistema di gestione di database utilizzabile da diverse piattaforme UNIX, con script di shell e relativi strumenti quali `awk` (in quel periodo linguaggi moderni e dinamici come Ruby, Python, PHP e persino Perl non erano ancora diffusi). Il manager mi contattò perché aveva bisogno di un programmatore che scrivesse il codice per riconoscere e utilizzare una versione limitata del linguaggio SQL.
Disse: "Non ho bisogno di supportare completamente il linguaggio – sarebbe troppo oneroso. Mi serve solo una istruzione SQL: `SELECT`".
Non avevo studiato SQL a scuola. I database non erano onnipresenti come lo sono oggi, e i marchi open source come MySQL e PostgreSQL non esistevano nemmeno. Ma avevo sviluppato applicazioni complete in shell, e sapevo qualcosa sui parser, avendo svolto progetti per le lezioni di progettazione di compilatori e linguistica computazionale. Così, cominciai a pensare se accettare il lavoro. Quanto poteva essere difficile effettuare il parsing di una singola istruzione di un linguaggio specializzato come SQL?

In questo capitolo

Trovai una guida di riferimento a SQL e mi accorsi immediatamente che si trattava di un linguaggio di tipo diverso rispetto a quelli che supportano istruzioni come `if()` e `while()`, assegnazioni di variabili e espressioni, e forse funzioni. Dire che `SELECT` è solo una istruzione di quel linguaggio è come dire che il motore è solo una parte di un'automobile. Entrambe le frasi sono vere, ma non tengono conto delle complessità e della profondità di entrambi gli argomenti. Per supportare l'esecuzione di quella sola istruzione SQL, mi sono accorto che dovevo sviluppare il codice per un sistema di database relazionale completo e un motore di query.
Rifiutai questa opportunità di programmare un parser SQL e un motore RDBMS in script di shell. Il manager aveva illustrato il suo progetto in maniera eccessivamente semplificata, forse perché non capiva quali fossero i compiti di un RDBMS.
La mia prima esperienza con SQL sembra abbastanza comune per gli sviluppatori di software, anche per coloro che hanno una laurea in informatica. Molte persone hanno imparato SQL per conto proprio, quando si sono trovati a lavorare su progetti che lo richiedevano, invece di studiarlo esplicitamente come farebbero per la maggior parte dei linguaggi di programmazione. Che si tratti di hobbisti o programmatori professionisti laureati, in molti casi SQL viene appreso senza seguire corsi specifici.
Una volta imparato qualcosa su SQL, fui sorpreso di quanto sia diverso dai linguaggi procedurali come C, Pascal, e shell, o dai linguaggi orientati agli oggetti come C++, Java, Ruby o Python. SQL è un *linguaggio di programmazione dichiarativo* come LISP, Haskell o XSLT. SQL utilizza i *set* come strutture dati fondamentali, mentre i linguaggi orientati agli oggetti utilizzano gli oggetti. Gli sviluppatori software di formazione classica si scontrano con la cosiddetta "discrepanza di impedenza" (*impedance mismatch*), quindi molti di essi sono attratti da librerie orientate agli oggetti per evitare la fatica di imparare a utilizzare SQL in modo efficace.
Dal 1992, ho lavorato molto con SQL. L'ho utilizzato nello sviluppo delle applicazioni, ho fornito supporto tecnico e sviluppato corsi e documentazioni per il prodotto RDBMS di Interbase, inoltre ho sviluppato librerie per la programmazione SQL in Perl e PHP. Ho risposto a migliaia di domande sulle mailing list e sui newsgroup di Internet. Noto molte cose che si ripetono – domande poste di frequente che mostrano che gli sviluppatori di software commettono continuamente gli stessi errori.

1.1 A chi si rivolge questo libro

Ho deciso di scrivere questo libro pensando agli sviluppatori di software che hanno bisogno di utilizzare SQL, per poterli aiutare a utilizzare questo linguaggio in modo più efficace, che siano apprendisti o esperti professionisti. Persone di qualsiasi livello di esperienza possono trovare utili i temi trattati in questo libro.
Forse avete già letto una guida di riferimento alla sintassi SQL. Conoscete quindi tutte le clausole dell'istruzione `SELECT`, e siete in grado di lavorarci. Gradualmente, potrete aumentare le vostre conoscenze di SQL esaminando altre applicazioni e leggendo articoli. Ma come potrete distinguere i buoni esempi da quelli scadenti? Come potrete essere sicuri di seguire i metodi migliori, e non quelli che vi porteranno in un pantano?
In questo libro potreste trovare alcuni argomenti che conoscete bene. Troverete nuovi modi di esaminare i problemi, anche se conoscete già le soluzioni. È utile confermare e rinforzare i propri metodi scoprendo quali sono le idee sbagliate più diffuse tra i

programmatori. Altri temi potrebbero risultare nuovi. Spero che possiate migliorare le vostre abitudini di programmazione in SQL leggendo queste pagine.
Un amministratore di database preparato forse conosce già i modi migliori per evitare le trappole di SQL che sono descritte qui. Questo libro può aiutare a introdurvi alla prospettiva degli sviluppatori di software. Non è raro che la relazione tra gli sviluppatori e i DBA (Database Administrator, o amministratori di database) sia difficile, ma il rispetto reciproco e il lavoro di squadra possono aiutare a lavorare insieme in modo più efficace. Utilizzate questo libro per spiegare i metodi migliori agli sviluppatori con cui lavorate e le conseguenze a cui si può andare incontro se ci si allontana dai percorsi suggeriti.

1.2 Contenuto del libro

Tutti i capitoli di questo libro si articolano attorno al concetto di *antipattern*, ma che cos'è un antipattern? Si tratta di una tecnica progettata per risolvere un problema, ma che talvolta porta ad altri problemi. Un antipattern viene utilizzato ampiamente e in modi diversi, ma c'è sempre un filo comune. Le idee di un antipattern a volte sono indipendenti, a volte sono il risultato della collaborazione con un collega, a volte sono prese da un libro o da un articolo. Molti antipattern di progettazione di software orientato agli oggetti e di gestione dei progetti sono documentati nel Portland Pattern Repository (`http://c2.com/cgi-bin/wiki?AntiPattern`) così come nel libro *AntiPatterns* del 1998 di William J. Brown *et al* [BMMM98].
Questo libro descrive gli errori che ho visto commettere ingenuamente e più frequentemente durante le mie attività di supporto tecnico, nelle sessioni di formazione, nel lavoro fianco a fianco e rispondendo alle domande poste nei forum di Internet. Anch'io sono caduto in molte di queste trappole; non c'è miglior modo di imparare che passare molte ore a notte fonda a lavorare per venire a capo dei propri errori.

Struttura del libro

Questo libro si compone di quattro parti, dedicate a diverse categorie di antipattern.

- **Antipattern nella progettazione logica di database**. Prima di cominciare a scrivere codice, dovreste decidere quali informazioni avete bisogno di mantenere sul vostro database e il miglior modo di organizzare e interconnettere i dati. Questo comprende l'attività di definire le tabelle, i campi e le relazioni del vostro database.
- **Antipattern nella progettazione fisica di database**. Dopo aver stabilito quali dati immagazzinare, si implementa la gestione dei dati nella maniera più efficiente possibile utilizzando le caratteristiche tecnologiche dell'RDBMS in uso. Questa fase prevede la definizione di tabelle e indici e la scelta dei tipi di dati. Si usa il *linguaggio di definizione dei dati* di SQL – istruzioni come `CREATE TABLE`.
- **Antipattern relativi alle query**. Avete la necessità di aggiungere dati al database e successivamente di recuperarli. Le query vengono scritte usando il *linguaggio di manipolazione dei dati* – istruzioni come `SELECT`, `UPDATE` e `DELETE`.
- **Antipattern nello sviluppo di applicazioni**. SQL viene solitamente utilizzato nel contesto di applicazioni scritte con altri linguaggi, come C++, Java, PHP, Python o Ruby. Esistono modi giusti e sbagliati di utilizzare SQL in un'applicazione, e questa parte del libro descrive le trappole più comuni.

Molti capitoli hanno titoli buffi ed evocativi, come "Alberi naif", "File fantasma" o "Indici dappertutto". Si usa assegnare nomi metaforici mnemonici sia ai pattern positivi, sia agli antipattern.
L'Appendice A fornisce descrizioni pratiche di alcuni elementi della teoria dei database relazionali. Molti degli antipattern descritti in questo libro sono il risultato di un'errata comprensione della teoria dei database.

Anatomia di un antipattern

Ogni capitolo dedicato agli antipattern è strutturato nei seguenti paragrafi.

- **Obiettivo**. È il problema che ci si può trovare a dover risolvere. Gli antipattern sono utilizzati con l'intenzione di fornire la soluzione, ma finiscono per causare più problemi di quelli che risolvono.
- **L'antipattern**. Questo paragrafo descrive la natura della soluzione comune e illustra le conseguenze non previste che la rendono un antipattern.
- **Come riconoscere l'antipattern**. Possono esserci alcuni indizi che aiutano a identificare quando si sta utilizzando un antipattern nel proprio progetto. Alcuni tipi di ostacoli che si incontrano, o frasi che si sentono pronunciare, possono indicare la presenza di un antipattern.
- **Uso legittimo dell'antipattern**. Le regole di solito hanno delle eccezioni. Possono esserci circostanze in cui un approccio considerato normalmente come un antipattern è tuttavia appropriato, o almeno rappresenta il minore dei mali.
- **Soluzione**. Questo paragrafo descrive la soluzione preferibile, che consente di raggiungere l'obiettivo originale senza incorrere nei problemi causati dall'antipattern.

1.3 Che cosa non c'è in questo libro

Non forniamo lezioni di sintassi e terminologia SQL. Ci sono moltissimi libri e riferimenti su Internet tramite i quali è facile reperire queste informazioni di base. Daremo per scontato che abbiate imparato abbastanza circa la sintassi di SQL e abbiate già lavorato con questo linguaggio.
Le performance, la scalabilità e l'ottimizzazione sono importanti per molte persone che sviluppano applicazioni legate ai database, specialmente sul Web. Ci sono libri che trattano nello specifico gli aspetti di performance relativi alla programmazione con database. Raccomando *SQL Performance Tuning* [GP03] e *High Performance MySQL, Second Edition* [SZT+08]. Alcuni dei temi trattati qui hanno attinenza con le performance, ma questo non è l'obiettivo principale del libro.
Nel testo si tenta di mostrare i problemi che possono capitare con database di tutti i produttori, e le soluzioni che dovrebbero funzionare in tutti i casi. Il linguaggio SQL è specificato come standard ANSI e ISO. Tutti i sistemi di database supportano questi standard, quindi cercheremo di descrivere un uso di SQL il più possibile indipendente, evidenziando chiaramente quando sono descritte estensioni di SQL specifiche di un produttore.
I framework di accesso ai dati e le librerie di mappatura da relazionale a oggetti sono strumenti utili, ma non costituiscono il punto di interesse centrale di questo libro. La

maggior parte degli esempi di codice è scritta in PHP, nel modo più semplice possibile. Gli esempi sono abbastanza semplici da essere ugualmente rilevanti per la maggior parte dei linguaggi di programmazione.
L'amministrazione di database e i compiti operativi come il dimensionamento del server, l'installazione e la configurazione, il monitoraggio, i backup, l'analisi dei log e la sicurezza sono importanti e meritano un libro a parte, mentre questo testo si rivolge agli sviluppatori che utilizzano il linguaggio SQL, più che agli amministratori di database.
Questo libro tratta di SQL e database relazionali, non di tecnologie alternative come database orientati agli oggetti, depositi chiave/valore, database *column-oriented* e *document-oriented*, database gerarchici, database di rete, *framework map/reduce* o depositi di dati semantici. Confrontare i punti di forza e i punti deboli, e descrivere l'uso appropriato di queste soluzioni alternative per la gestione dei dati potrebbe essere interessante, ma costituisce materia per altri libri.

1.4 Convenzioni adottate

Nel seguito sono descritte alcune convenzioni utilizzate in questo libro.

Tipografia

Le parole chiave SQL sono formattate in lettere maiuscole con un carattere non proporzionale, per evidenziarle nel testo. Per esempio: `SELECT`.
Le tabelle SQL, anch'esse riportate con un carattere non proporzionale, sono scritte con una maiuscola come iniziale di ogni parola che ne compone il nome. Esempi: `Accounts` o `BugsProducts`.
Le colonne SQL, anch'esse riportate con un carattere non proporzionale, sono scritte in minuscolo, e le parole sono separate da trattini underscore. Esempio: `account_name`.
Le stringhe letterali sono formattate in corsivo. Esempio: *bill@example.com*.

Terminologia

Nel contesto del suo utilizzo nell'ambito dei database, la parola *indice* si riferisce a una collezione ordinata di informazioni.
In SQL i termini *query* e *istruzione* sono a volte intercambiabili, indicando ogni comando SQL completo che può essere eseguito. Per maggiore chiarezza utilizziamo *query* per riferirci alle istruzioni `SELECT` e *istruzioni* per tutti gli altri comandi, tra i quali `INSERT`, `UPDATE` e `DELETE`, così come per le istruzioni di definizione dei dati.

Diagrammi entità-relazioni

Il modo più comune di rappresentare i database relazionali è attraverso l'uso dei diagrammi *entità-relazioni*. Le tabelle sono rappresentate come rettangoli ad angoli arrotondati e le relazioni come linee che li connettono, con simboli a entrambi gli estremi, che descrivono la cardinalità della relazione. Un esempio è visibile nella Figura 1.1.

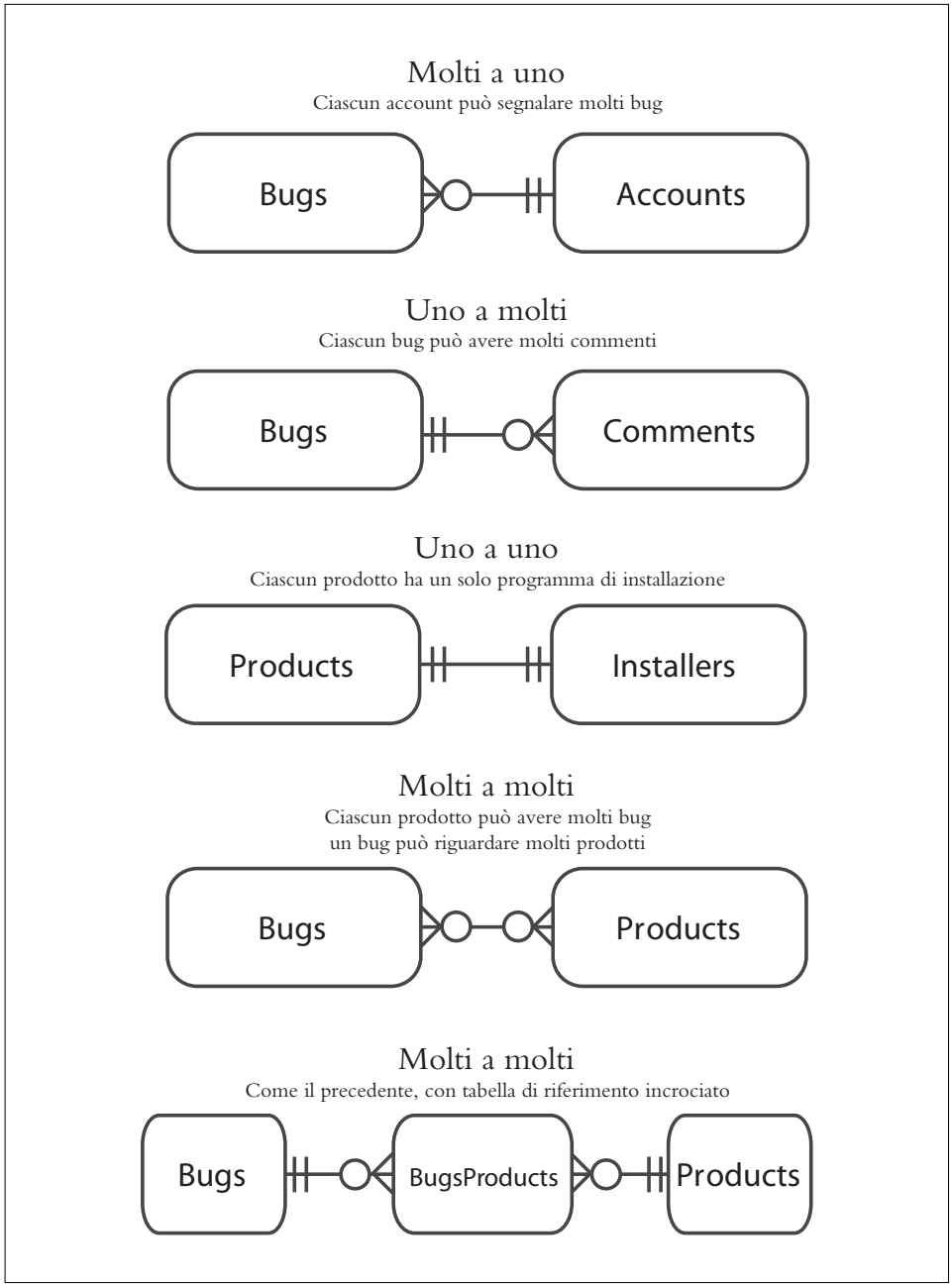

Figura 1.1 Esempi di diagrammi entità-relazioni.

1.5 Database di esempio

La maggior parte degli argomenti trattati in questo libro sono illustrati utilizzando un database per un'ipotetica applicazione di *bug-tracking*, ovvero per tracciare bug e segnalazioni. Il diagramma entità-relazione per questo database è mostrato nella Figura 1.2. Notate le tre connessioni tra la tabella `Bugs` e la tabella `Accounts`, che rappresentano tre diverse chiavi esterne.

Il seguente script in linguaggio di definizione dei dati mostra come sono state definite le tabelle. In alcuni casi sono state operate delle scelte a beneficio degli esempi presentati più avanti nel libro, quindi non sempre potrebbero essere le scelte da fare in un'applicazione reale. Abbiamo cercato di utilizzare solo SQL standard, in modo che l'esempio sia applicabile su qualsiasi tipo di database; sono tuttavia presenti alcuni tipi di dati di MySQL, come `SERIAL` e `BIGINT`.

Introduction/setup.sql

```
CREATE TABLE Accounts (
    account_id      SERIAL PRIMARY KEY,
    account_name    VARCHAR(20),
    first_name      VARCHAR(20),
    last_name       VARCHAR(20),
    email           VARCHAR(100),
    password_hash   CHAR(64),
    portrait_image  BLOB,
    hourly_rate     NUMERIC(9,2)
);
```

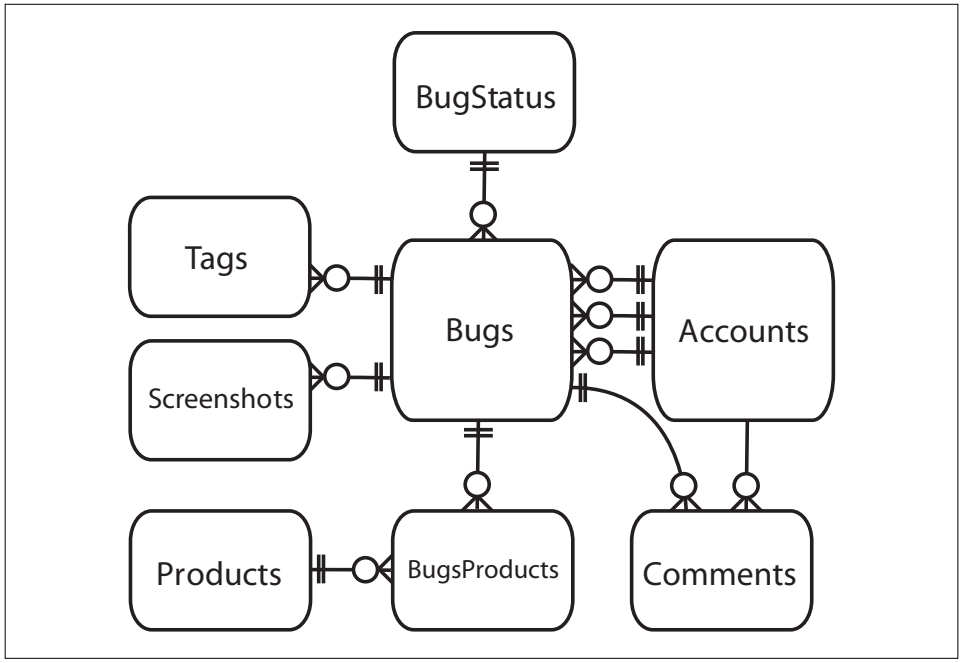

Figura 1.2 Diagramma per il database di esempio sui bug.

```
CREATE TABLE BugStatus (
    status            VARCHAR(20) PRIMARY KEY
);

CREATE TABLE Bugs (
    bug_id            SERIAL PRIMARY KEY,
    date_reported     DATE NOT NULL,
    summary           VARCHAR(80),
    description       VARCHAR(1000),
    resolution        VARCHAR(1000),
    reported_by       BIGINT UNSIGNED NOT NULL,
    assigned_to       BIGINT UNSIGNED,
    verified_by       BIGINT UNSIGNED,
    status            VARCHAR(20) NOT NULL DEFAULT 'NEW',
    priority          VARCHAR(20),
    hours             NUMERIC(9,2),
    FOREIGN KEY (reported_by) REFERENCES Accounts(account_id),
    FOREIGN KEY (assigned_to) REFERENCES Accounts(account_id),
    FOREIGN KEY (verified_by) REFERENCES Accounts(account_id),
    FOREIGN KEY (status) REFERENCES BugStatus(status)
);

CREATE TABLE Comments (
    comment_id        SERIAL PRIMARY KEY,
    bug_id            BIGINT UNSIGNED NOT NULL,
    author            BIGINT UNSIGNED NOT NULL,
    comment_date      DATETIME NOT NULL,
    comment           TEXT NOT NULL,
    FOREIGN KEY (bug_id) REFERENCES Bugs(bug_id),
    FOREIGN KEY (author) REFERENCES Accounts(account_id)
);

CREATE TABLE Screenshots (
    bug_id            BIGINT UNSIGNED NOT NULL,
    image_id          BIGINT UNSIGNED NOT NULL,
    screenshot_image  BLOB,
    caption           VARCHAR(100),
    PRIMARY KEY (bug_id, image_id),
    FOREIGN KEY (bug_id) REFERENCES Bugs(bug_id)
);

CREATE TABLE Tags (
    bug_id            BIGINT UNSIGNED NOT NULL,
    tag               VARCHAR(20) NOT NULL,
    PRIMARY KEY (bug_id, tag),
    FOREIGN KEY (bug_id) REFERENCES Bugs(bug_id)
);

CREATE TABLE Products (
    product_id        SERIAL PRIMARY KEY,
    product_name      VARCHAR(50)
);

CREATE TABLE BugsProducts(
    bug_id            BIGINT UNSIGNED NOT NULL,
    product_id        BIGINT UNSIGNED NOT NULL,
    PRIMARY KEY (bug_id, product_id),
```

```
    FOREIGN KEY (bug_id) REFERENCES Bugs(bug_id),
    FOREIGN KEY (product_id) REFERENCES Products(product_id)
);
```

In alcuni capitoli, particolarmente nella parte dedicata agli antipattern nella progettazione logica di database, sono riportate diverse definizioni di database, per mostrare l'antipattern o una soluzione alternativa che consenta di evitarlo.

1.6 Ringraziamenti

Voglio innanzitutto esprimere la mia gratitudine a mia moglie Jan. Non avrei potuto scrivere questo libro senza l'ispirazione, l'amore e il supporto che mi dai, per non parlare dei calci nel sedere di tanto in tanto...
Desidero ringraziare i miei revisori per avermi dedicato molto del loro tempo; i loro suggerimenti hanno migliorato moltissimo il libro. Marcus Adams, Jeff Bean, Frederic Daoud, Darby Felton, Arjen Lentz, Andy Lester, Chris Levesque, Mike Naberezny, Liz Nealy, Daev Roehr, Marco Romanini, Maik Schmidt, Gale Straney, e Danny Thorpe.
Ringrazio il mio editor Jacquelyn Carter e i publisher di Pragmatic Bookshelf, che hanno creduto in questo libro.