

## Sicurezza di un database

Una delle classiche problematiche che un DBMS deve gestire è l'accesso simultaneo ai dati da parte di diversi utenti, sia in lettura che in aggiornamento. Una situazione tipica è il caso in cui due utenti leggono lo stesso dato con l'intenzione di aggiornarlo: evidentemente uno dei due lo farà per primo, e a quel punto il secondo utente, quando tenterà a sua volta un aggiornamento, troverà una situazione variata rispetto al momento in cui aveva letto i dati, col rischio di creare situazioni incongruenti. Le soluzioni per questi problemi sono, nella forma più semplice, i lock sulle tabelle, e in quella più avanzata le transazioni. Queste ultime sono disponibili solo su tabelle InnoDB e BDB.

### Lock

Un Lock può essere considerato come un vincolo di "uso esclusivo" che un utente può ottenere su determinate tabelle per il tempo necessario a svolgere le operazioni che gli sono necessarie. Può essere richiesto in lettura o in scrittura: nel primo caso nessuno può effettuare modifiche alla tabella, ma sia l'utente con i privilegi e sia gli altri possono leggere la tabella in questione. Il lock in scrittura invece impedisce agli altri utenti qualsiasi tipo di accesso alla tabella, e consente all'utente che l'ha ottenuto operazioni di lettura e scrittura.

Vediamo la sintassi delle operazioni di lock, ricordando che esse richiedono il privilegio **LOCK TABLES**:

```
lock tables nome_tabella1 READ|WRITE,  
nome_tabella2 READ|WRITE;
```

Per sbloccare i lock su tutte le tabelle, si usa il comando seguente:

```
unlock tables;
```

### Transazioni

L'uso delle transazioni permette di "consolidare" le modifiche alla base dati solo in un momento ben preciso: dal momento in cui avviamo una transazione, gli aggiornamenti rimangono sospesi (e invisibili ad altri utenti) fino a quando non li confermiamo (**commit**); in alternativa alla conferma è possibile annullarli (**rollback**). Queste transazioni possono essere effettuate solo su tabelle InnoDB.

Innanzitutto va segnalato che MySQL gira di default in **autocommit mode**: questo significa che tutti gli aggiornamenti vengono automaticamente consolidati nel momento in cui sono eseguiti. Se siamo in autocommit, per iniziare una transazione dobbiamo usare l'istruzione **START TRANSACTION**; da questo punto in poi tutti gli aggiornamenti rimarranno sospesi. La transazione può essere chiusa con l'istruzione **COMMIT**, che consolida le modifiche, oppure con **ROLLBACK**, che annulla tutti gli aggiornamenti effettuati nel corso della transazione. Possiamo utilizzare anche **COMMIT AND CHAIN** o **ROLLBACK AND CHAIN**, che provocano l'immediata apertura di una nuova transazione, oppure **COMMIT RELEASE** o **ROLLBACK RELEASE**, che oltre a chiudere la transazione chiudono anche la connessione al server.

Con l'istruzione **SET AUTOCOMMIT=0** possiamo disattivare l'autocommit: in questo caso non è più necessario avviare le transazioni con **START TRANSACTION**, e tutti gli aggiornamenti rimarranno sospesi fino all'uso di **COMMIT** o **ROLLBACK**.

All'interno di una transazione è anche possibile stabilire dei **savepoint**, cioè degli stati intermedi ai quali possiamo ritornare con una **ROLLBACK**, invece di annullare interamente la transazione.

Vediamo un esempio:

```
START TRANSACTION  
...istruzioni di aggiornamento (1)...  
SAVEPOINT sp1;  
...istruzioni di aggiornamento (2)...  
ROLLBACK TO SAVEPOINT sp1;  
...istruzioni di aggiornamento (3)...  
COMMIT
```

In questo caso, dopo avere avviato la transazione abbiamo eseguito un primo blocco di aggiornamenti, seguito dalla creazione del savepoint col nome 'sp1'; in seguito abbiamo eseguito un secondo blocco di aggiornamenti; l'istruzione ROLLBACK TO SAVEPOINT sp1 fa sì che "ritorniamo" alla situazione esistente quando abbiamo creato il savepoint: in pratica solo il secondo blocco di aggiornamenti viene annullato, e la transazione rimane aperta; una semplice ROLLBACK invece avrebbe annullato tutto e chiuso la transazione. La COMMIT effettuata dopo il terzo blocco fa sì che vengano consolidati gli aggiornamenti effettuati nel primo e nel terzo blocco.