

## Tipi di Tabelle

MySQL permette di utilizzare diversi tipi di tabelle, ovvero diversi "storage engine" per la memorizzazione dei dati. Si dividono in transazionali e non transazionali.

I motori transazionali offrono alcuni importanti vantaggi: sono più sicuri (permettono di recuperare i dati anche in caso di crash di MySQL o di problemi hardware) e consentono di effettuare più modifiche e convalidarle tutte insieme o, al contrario, ripristinare la situazione preesistente se qualcosa va male.

Dal canto loro, i motori non transazionali hanno il vantaggio di una maggior velocità, minore utilizzo di spazio su disco e minor richiesta di memoria per gli update. È anche possibile combinare tabelle transazionali e non nelle stesse istruzioni, anche se, in questo caso, le modifiche fatte sulle tabelle non transazionali divengono comunque effettive nel momento in cui sono eseguite.

Quando si crea una tabella si specifica a MySQL di che tipo si tratta attraverso l'opzione **ENGINE**:

```
CREATE TABLE tabella (a INT) ENGINE = INNODB;
```

Nel caso in cui la dichiarazione venga omessa, MySQL utilizzerà il tipo di default, che normalmente è MyISAM o INNODB (a seconda della versione di MySQL). Potete modificare questo valore all'avvio di MySQL digitando:

```
mysqld --default-table-type = MySAM;
```

### MySAM

È lo storage engine di default di MySQL. Ogni tabella MyISAM utilizza tre file: un file `.frm` che contiene la definizione della tabella, un file `.MYD` per i dati e un file `.MYI` per gli indici.

È possibile indicizzare le colonne di tipo BLOB e TEXT, e si possono utilizzare valori NULL nelle colonne indicizzate. Inoltre MyISAM può gestire una colonna di tipo AUTO\_INCREMENT per ogni tabella.

### InnoDB

È uno storage engine transazionale dotato di capacità di commit, rollback e crash recovery. È ottimizzato per l'uso concorrente dei dati fra molti utenti e per essere molto performante anche su grandi quantità di dati. Inoltre supporta le FOREIGN KEY.

Se non usate le tabelle InnoDB, potete avviare il server con l'opzione `--skip-innodb`. Se invece le utilizzate, dovrete fornire al server le indicazioni sui file da utilizzare per i dati.

InnoDB gestisce il valore AUTO\_INCREMENT per una tabella in modo particolare: questo viene infatti calcolato la prima volta che si rende necessario dopo l'avvio del server (ad esempio per una INSERT), selezionando il valore massimo esistente sulla tabella e incrementandolo di 1. A quel punto il valore viene conservato in memoria ma non scritto su disco, per cui al riavvio successivo sarà ricalcolato. Questo significa che se cancellate gli ultimi valori della tabella senza fare nuovi inserimenti, al successivo riavvio il server riutilizzerà quei valori.

Con le tabelle InnoDB possiamo definire le foreign key, cioè collegare i valori delle colonne che contengono chiavi di altre tabelle alle tabelle stesse. In questo modo è possibile verificare automaticamente quando i valori della tabella madre vengono modificati o eliminati in modo da impedire queste modifiche o, al contrario, modificare di conseguenza anche i valori sulla tabella figlia. Inoltre non è possibile inserire nella tabella figlia valori che non hanno un corrispondente nella tabella madre.

Eccone un esempio:

```
shell> CREATE TABLE madre (id INT NOT NULL,  
                           PRIMARY KEY (id)  
                           ) ENGINE=INNODB;  
  
shell> CREATE TABLE figlia (id INT, madre_id INT,  
                            INDEX par_ind (madre_id),  
                            FOREIGN KEY (madre_id) REFERENCES madre(id)  
                            ) ENGINE=INNODB;
```

In questo codice vediamo definite due tabelle nelle quali la tabella figlia ha una foreign key sulla tabella madre. Quando viene cancellata una riga dalla tabella madre, se il valore di id è presente in un campo madre\_id della tabella figlia la riga

corrispondente viene anch'essa eliminata. Entrambe le tabelle devono essere di tipo InnoDB; entrambe le colonne devono figurare come primo campo di un indice.

## MERGE

Una tabella MERGE è formata da un insieme di tabelle MyISAM identiche fra loro nella struttura. Questo significa che devono avere le stesse colonne e gli stessi indici; anche l'ordine in cui colonne e indici sono dichiarate deve essere lo stesso. Eccone un esempio:

```
shell> CREATE TABLE totali (  
      a INT NOT NULL AUTO_INCREMENT,  
      messaggio CHAR(20), INDEX(a))  
ENGINE=MERGE UNION=(t1,t2) INSERT_METHOD=LAST;
```

La clausola **UNION** specifica i nomi delle tabelle che costituiscono la tabella totali. Tali tabelle dovranno avere definiti gli stessi campi e gli stessi indici; l'unica eccezione sono le primary key che non vengono replicate nella tabella MERGE in quanto l'unicità delle chiavi può essere garantita solo all'interno della singola tabella.

Con **INSERT\_METHOD=LAST** si specifica che eventuali inserimenti devono essere effettuati sull'ultima delle tabelle elencate; l'alternativa sarebbe l'opzione **FIRST** per effettuare gli inserimenti sulla prima. Nel caso non venga specificata la clausola, non sarà possibile inserire record sulla tabella. Da notare che se si esegue il **DROP TABLE** su una tabella MERGE sarà solo quest'ultima ad essere eliminata, non le tabelle sottostanti.

## MEMORY

Le tabelle MEMORY hanno una definizione che viene salvata su disco, ma i loro dati sono conservati solo in memoria. Questo significa che ad ogni riavvio del server tali tabelle saranno vuote. Bisogna però stare attenti a non generare tabelle di dimensioni troppo grandi, altrimenti si può mandare in crash il server stesso.

## BDB (BerkeleyDB)

Le tabelle BDB non funzionano su sistemi Linux Alpha, AMD64, IA-64, s390, nonché su Mac OS X. Funzionano invece su piattaforme Linux Intel, nonché Sun Solaris, FreeBSD, AIX 4.3.x, SCO (OpenServer e UnixWare 7.1.x), Windows (da architetture NT in poi). Sono di tipo transazionali.

Le tabelle BDB sono formate da un file .frm con la definizione della tabella e da un file .db che contiene dati e indici. Tali file non possono essere spostati perchè il file .db contiene anche il proprio percorso.

## ARCHIVE

Le tabelle di tipo ARCHIVE possono essere utilizzate per archiviare grosse quantità di dati senza indici, e senza la possibilità di modificarli ma solo di aggiungere righe alla tabella. Su queste tabelle è possibile infatti compiere esclusivamente **SELECT** e **INSERT**. I dati vengono compressi con *zlib*, e per questo la loro occupazione di spazio è molto limitata.

## CSV

Questo storage engine consente di memorizzare i dati su file di testo i cui valori sono separati da virgole.

Queste sono le tipologie più usate; per gli altri tipi di tabelle, rimando alla guida ufficiale.