

Guía del Motor de Almacenamiento Falcon

Guía del Motor de Almacenamiento Falcon

Resumen

Este documento provee información para el uso de la entrega especial alfa del Motor de Almacenamiento Falcon, que es parte de la base de datos MySQL.

Fecha de generación del documento: 2008-11-01 (revision: 516)

Copyright 2007 MySQL AB

This documentation is NOT distributed under a GPL license. Use of this documentation is subject to the following terms: You may create a printed copy of this documentation solely for your own personal use. Conversion to other formats is allowed as long as the actual content is not altered or edited in any way. You shall not publish or distribute this documentation in any form or on any media, except if you distribute the documentation in a manner similar to how MySQL disseminates it (that is, electronically for download on a Web site with the software) or on a CD-ROM or similar medium, provided however that the documentation is disseminated together with the software on the same medium. Any other use, such as any dissemination of printed copies or use of this documentation, in whole or in part, in another publication, requires the prior written consent from an authorized representative of MySQL AB. MySQL AB reserves any and all rights to this documentation not expressly granted above.

Please email [<docs@mysql.com>](mailto:docs@mysql.com) for more information or if you are interested in doing a translation.

Tabla de contenidos

Acerca de la entrega del motor de almacenamiento Falcon	vi
1. El Motor de Almacenamiento Falcon	1
1.1. Características de Falcon	1
2. Instalación de Falcon	2
2.1. Instalación de Falcon desde una distribución binaria	2
2.2. Instalación de Falcon desde una distribución de código fuente	2
2.3. Cómo obtener y generar Falcon desde el código fuente del árbol de desarrollo	4
3. Cómo utilizar Falcon	7
3.1. Parámetros de configuración	7
3.2. Creación de espacios de tablas de Falcon	9
3.3. Creación de tablas e índices dentro de Falcon	10
3.4. Diagnósticos de Rendimiento	10
3.5. Principios y Terminología	11
3.5.1. Ficheros de datos y estructuras de datos de Falcon	12
3.5.2. Registro serial de Falcon	12
3.5.3. Recuperación de fallos de Falcon	13
3.5.4. Cachés de memoria de Falcon	14
3.5.5. Subprocesos de Falcon (Threads)	14
3.5.6. Compresión de Datos	14
3.5.7. Espacio de Registro	14
3.6. Límites de Falcon	14
A. Plan de Desarrollo (Roadmap) de Falcon	16
B. Cambios en MySQL 6.0 Falcon	17
B.1. Cambios en la entrega 6.0.2 (04 septiembre 2007)	17
B.2. Cambios en la entrega 6.0.1 (No entregado)	18
B.3. Cambios en la entrega 6.0.0 (30 Abril 2007)	18
B.4. Cambios en MySQL 5.2 Falcon	20
B.4.1. Cambios en la entrega 5.2.3 (15 de febrero 2007)	20
B.4.2. Cambios en la entrega 5.2.2 (no entregado)	20
B.4.3. Cambios en la entrega 5.2.1 (no entregado)	20

Lista de tablas

3.1. Tablas de diagnóstico de rendimiento de Falcon en INFORMATION_SCHEMA	11
---	----

Acerca de la entrega del motor de almacenamiento Falcon

El motor de almacenamiento Falcon es parte del SGBD (DBMS) MySQL, y, como tal, depende del servidor MySQL. Está siendo entregado dentro de una generación especial alfa de MySQL y será actualizado regularmente durante el proceso alfa. La versión de producción (GA, Generally Available) del motor de almacenamiento Falcon será incorporada en la próxima versión significativa de MySQL junto con otras nuevas características.

Falcon es actualmente una entrega Alfa, y no debería ser utilizado en ambientes de producción. Por el momento, Falcon solo puede obtenerse dentro de una ramificación especial de la entrega de MySQL, llamada MySQL-6.0-falcon. Esta entrega no se considera lista para producción. El árbol mysql-6.0-falcon se provee solo para pruebas y evaluación del motor de almacenamiento Falcon.

Nota

Puede ser que el árbol mysql-6.0-falcon no incluya todas las características o correcciones de bugs que han sido aplicados a entregas previas de MySQL.

Capítulo 1. El Motor de Almacenamiento Falcon

Falcon es un motor de almacenamiento para el servidor de base de datos MySQL. Un motor de almacenamiento provee el funcionamiento para almacenar, indexar y obtener información desde el almacén físico de los datos. Existen otros motores de almacenamiento disponibles para MySQL, incluyendo [El motor de almacenamiento InnoDB](#) y [El motor de almacenamiento MyISAM](#). Para más información sobre motores de almacenamiento vea [Motores de almacenamiento de MySQL y tipos de tablas](#). Para conocer más detalles sobre MySQL, vea el [MySQL 5.1 Reference Manual](#).

El motor de almacenamiento Falcon ha sido diseñado teniendo en mente las exigencias de las bases de datos modernas, particularmente para el uso en servicios de web de alto volumen de datos u otros ambientes que requieren de alto rendimiento, a la vez que soporta la funcionalidad transaccional y de registro (logging) que se requieren en estos ambientes.

Por el momento, Falcon está disponible solamente para los sistemas operativos Windows de 32 bits y Linux de 32 ó 64 bits. Otras plataformas será añadidas después de la entrega alfa.

1.1. Características de Falcon

Falcon ha sido desarrollado especialmente para sistemas con soporte para arquitecturas de grandes cantidades de memoria y ambientes multiproceso (multi-threaded) o de procesadores de múltiples núcleos (multi-core CPUs). La mayoría de arquitecturas de 64 bits son plataformas ideales para el motor Falcon, ya que disponen de un espacio de memoria mayor y generalmente disponen de procesadores de 2, 4 u 8 núcleos (cores). Pero también puede ser implementado dentro de un ambiente estándar de 32 bits.

El motor de almacenamiento Falcon está diseñado para trabajar dentro de aplicaciones transaccionales de alto tráfico de datos. Posee varias características clave que hacen esto posible, las cuales son:

- El Control de Concurrencia Multi Versión (Multi Version Concurrency Control, MVCC) permite que los registros y las tablas sean actualizadas sin la sobrecarga asociada con los mecanismos de bloqueo (locking) al nivel de filas. La implementación de MVCC prácticamente elimina la necesidad de bloquear tablas o filas durante el proceso de actualización.
- Bloqueo (locking) flexible, que permite flexibilidad en los niveles de bloqueo y detección inteligente de interbloqueo. Esto protege los datos y permite que las transacciones y operaciones fluyan a toda velocidad.
- Está optimizado para CPUs y ambientes modernos para permitir múltiples procesos, lo que permite múltiples transacciones y un manejo rápido de ellas.
- Transacciones seguras, ya que cumple a cabalidad las propiedades ACID, y es capaz de manejar múltiples transacciones concurrentes.
- Registro (Log) serial, provee capacidad para un alto rendimiento y recuperación sin sacrificar el rendimiento.
- Índices de árbol B (B-Tree) avanzados.
- Compresión de datos. Falcon almacena la información en el disco en un formato comprimido, comprimiendo y descomprimiendo los datos en el instante. El resultado es datos físicos más pequeños y eficientes.
- Administración Inteligente de Discos. Automáticamente administra los ficheros de datos y sus extensiones. El espacio dentro de los ficheros de datos y de registro (log) se reclama automáticamente y se reutiliza.
- Utilización de cachés para datos e índices para un acceso rápido a los datos sin la necesidad de cargar un índice de datos desde el disco.
- Puntos de resguardo (savepoints) implícitos aseguran la integridad de los datos durante las transacciones.

Si desea probar el motor de almacenamiento Falcon, puede usar el [MySQL Query Browser](#).

Capítulo 2. Instalación de Falcon

La instalación de Falcon está disponible en formato binario (para instalación directa), formato en código fuente (para generaciones personalizadas (custom builds)) y directamente desde el árbol de desarrollo (para generaciones personalizadas de características aún no entregadas).

- Para obtener instrucciones sobre la instalación de la Inspección Previa de MySQL Falcon desde una instalación binaria vea [Sección 2.1, “Instalación de Falcon desde una distribución binaria”](#).
- Para obtener instrucciones sobre la generación de la Inspección Previa de MySQL Falcon desde una distribución de código fuente, vea [Sección 2.2, “Instalación de Falcon desde una distribución de código fuente”](#).
- Para obtener instrucciones sobre la generación de la Inspección Previa de MySQL Falcon desde el código fuente del árbol de desarrollo, vea [Sección 2.3, “Cómo obtener y generar Falcon desde el código fuente del árbol de desarrollo”](#).

Para obtener más información sobre las características y la funcionalidad de Falcon, vea [Capítulo 3, *Cómo utilizar Falcon*](#). Si desea intercambiar información sobre su experiencia con otros, puede usar el [Foro de Falcon](#). Si piensa que ha encontrado un bug, puede reportarlo a [MySQL Bugs](#).

2.1. Instalación de Falcon desde una distribución binaria

Se puede instalar la inspección previa de Falcon en distribución binaria, tanto en Linux como Windows, siguiendo las mismas instrucciones para la instalación estándar de MySQL.

Para instalar la inspección previa de Falcon sobre Linux utilizando una distribución binaria empaquetada como un `tar .gz`, vea [Instalación de MySQL en otros sistemas similares a Unix](#). Para instalar utilizando un RPM, vea [Instalar MySQL en Linux](#).

Para instalar la inspección previa de Falcon sobre Windows utilizando una distribución binaria, vea [Instalar MySQL en Windows](#).

2.2. Instalación de Falcon desde una distribución de código fuente

Los comandos básicos que debe ejecutar para instalar una distribución de código fuente de MySQL son:

```
shell> groupadd mysql
shell> useradd -g mysql mysql
shell> gunzip < mysql-VERSIÓN.tar.gz | tar -xvf -
shell> cd mysql-VERSIÓN
shell> ./configure --prefix=/usr/local/mysql
shell> make
shell> make install
shell> cp support-files/my-medium.cnf /etc/my.cnf
shell> cd /usr/local/mysql
shell> chown -R mysql .
shell> chgrp -R mysql .
shell> bin/mysql_install_db --user=mysql
shell> chown -R root .
shell> chown -R mysql var
shell> bin/mysqld_safe --user=mysql &
```

Nota

Este procedimiento no configura ninguna contraseña para las cuentas de MySQL. Después de ejecutar el procedimiento, continúe con [Puesta en marcha y comprobación de la instalación](#), para la configuración post-instalación y pruebas.

Una versión más detallada de la descripción anterior para instalar MySQL desde una distribución de código fuente se presenta a continuación:

1. Agregue un usuario de ingreso y un grupo para ejecutar `mysqld` de la siguiente manera:

```
shell> groupadd mysql
shell> useradd -g mysql mysql
```

Estos comandos agregan el grupo `mysql` y el usuario `mysql`. La sintaxis para `useradd` y `groupadd` puede variar un poco en las diferentes versiones de Unix, o pueden tener diferentes nombres, como `adduser` y `addgroup`.

Si desea poner otro nombre al usuario y grupo `mysql`, puede hacerlo. Si lo hace, sustituya el nombre que corresponda en los pasos siguientes.

2. Siga los siguientes pasos ejecutando los comandos con el usuario `mysql`, excepto cuando se indique lo contrario.
3. Extraiga la distribución al directorio actual:

```
shell> gunzip <
/ruta/de/mysql-VERSIÓN.tar.gz | tar xvf -
```

Este comando crea un directorio con el nombre `mysql-VERSIÓN`.

Si usa GNU `tar`, no es necesaria una llamada separada a `gunzip`. Puede usar el siguiente comando alternativo para decomprimir y extraer la distribución:

```
shell> tar zxvf /ruta/de/mysql-VERSIÓN-SO.tar.gz
```

4. Cambie al directorio de primer nivel de la distribución que decomprimió:

```
shell> cd mysql-VERSIÓN
```

Note que actualmente debe configurar y generar MySQL desde el directorio de primer nivel. No se puede generar en otro directorio.

5. Configure la entrega y compile todo:

```
shell> ./configure --prefix=/usr/local/mysql
shell> make
```

Cuando ejecute `configure`, quizá quiera especificar otras opciones. Ejecute `./configure --help` para ver una lista de opciones. La sección [Opciones tÃ-picas de configure](#), discute algunas de las opciones mÃ-Ãs Ãtiles.

Si la compilaciÃ³n falla, vea [Problemas en la compilaciÃ³n de MySQL](#), para obtener ayuda.

6. Instale la distribuciÃ³n:

```
shell> make install
```

Puede que tenga que ejecutar este comando como `root`.

Si desea configurar un fichero de opciones, use uno de los presentes en el directorio `support-files` como plantilla. Por ejemplo:

```
shell> cp support-files/my-medium.cnf /etc/my.cnf
```

Puede que tenga que ejecutar este comando como `root`.

Si desea configurar soporte para tablas `InnoDB`, debe editar el fichero `/etc/my.cnf`, eliminar el caracter `#` que aparece antes de las opciones que empiezan con `innodb_...`, y modificar los valores de las opciones como usted desee. Vea [Usar ficheros de opciones](#), y [ConfiguraciÃ³n de InnoDB](#).

7. Cambie hacia el directorio de instalaciÃ³n:

```
shell> cd /usr/local/mysql
```

8. Si ejecutÃ³ el comando `make install` como `root`, los ficheros instalados serÃ¡n propiedad de `root`. Para asegurarse de que la instalaciÃ³n es accesible al usuario `mysql`, puede ejecutar los siguientes comandos como `root` en el directorio de instalaciÃ³n:

```
shell> chown -R mysql .
shell> chgrp -R mysql .
```

El primer comando cambia el atributo de propietario de los ficheros al usuario `mysql`. El segundo cambia el atributo de grupo al grupo `mysql`.

9. Si no ha instalado MySQL antes, debe crear el directorio de datos de MySQL e inicializar las tablas de permisos:

```
shell> bin/mysql_install_db --user=mysql
```

Si ejecuta el comando como `root`, incluya la opciÃ³n `--user` como se muestra arriba. Pero si ejecuta el comando habiendo iniciado sesiÃ³n como `mysql`, puede omitir la opciÃ³n `--user`.

El comando debe crear el directorio de datos y su contenido con el usuario `mysql` como propietario.

Después de usar `mysql_install_db` para crear las tablas de permisos para MySQL, debe reiniciar el servidor manualmente. El comando para hacer esto, `mysqld_safe`, se muestra en un paso posterior.

10. Si usted así lo desea, la mayoría de los ficheros de instalación de MySQL pueden ser propiedad de `root`. La excepción es el directorio de datos, que debe ser propiedad de `mysql`. Para cumplir con esto, ejecute los siguientes comandos como `root` en el directorio de instalación:

```
shell> chown -R root .
shell> chown -R mysql var
```

11. Si desea que MySQL inicie automáticamente cuando arranca su ordenador, puede copiar el fichero `support-files/mysql.server` al lugar donde su sistema tiene sus ficheros de inicio. Puede encontrar más información en el mismo script, `support-files/mysql.server`; también puede ver [Arrancar y parar MySQL automáticamente](#).

Luego que todo haya sido instalado, es necesario probar la distribución. Para iniciar el servidor MySQL, use el siguiente comando:

```
shell> /usr/local/mysql/bin/mysqld_safe --user=mysql &
```

Si ejecuta el comando como `root`, debe usar la opción `--user` como se muestra arriba. El valor de la opción es el nombre de la cuenta de usuario que creó en el primer paso para usar al ejecutar el servidor. Si ejecuta el comando habiendo iniciado sesión como ese usuario, puede omitir la opción `--user`.

Si el comando falla inmediatamente e imprime `mysqld ended`, puede encontrar más información en el fichero `nombre_equipo.err` en el directorio de datos.

Puede encontrar más información sobre el comando `mysqld_safe` en [El script de arranque del servidor mysqld_safe](#).

Nota

Las cuentas que se listan en las tablas de permisos de MySQL inicialmente no tienen contraseñas. Después de iniciar el servidor, debe configurar contraseñas para esas cuentas siguiendo las instrucciones en [Puesta en marcha y comprobación después de la instalación](#).

2.3. Cómo obtener y generar Falcon desde el código fuente del árbol de desarrollo

Falcon ha estado disponible como un árbol especial de MySQL llamado `mysql-6.0-falcon`. Para poder probar Falcon debe descargar el código fuente y generar la versión especial del árbol de MySQL.

Para generar la entrega de Falcon se necesita:

- GNU `make`
- GNU `autoconf` 2.58 (o más reciente)
- GNU `automake` 1.8.1 (o más reciente)
- GNU `libtool` 1.5

Puede obtener el código fuente de Falcon en <http://mysql.bkbits.com>. Para poder descargar el código fuente, debe usar el cliente BitKeeper, ya sea el cliente gratuito o el comercial.

Para obtener un árbol de código fuente, primero descargue e instale el cliente gratuito BitKeeper si todavía no lo tiene. Este lo puede obtener en <http://www.bitmover.com/bk-client2.0.shar>. Su sistema debe tener instalado `gcc`, `make`, `patch` y `tar`. *Note que las versiones anteriores del cliente gratuito BitKeeper como la 1.1 no funcionarán.*

Para instalar el cliente BitKeeper en Unix, use estos comandos:

```
shell> /bin/sh bk-client2.0.shar
shell> cd bk-client2.0
shell> make
```

Si recibe el error `cc: command not found`, invoque el comando `make` de esta manera:

```
shell> make CC=gcc
```

Este creará la herramienta `bkf`, que es el cliente gratuito de BitKeeper. Para obtener más información sobre el cliente gratuito BitKeeper ejecute:

```
shell> bkf --help
```

Para instalar el cliente BitKeeper en Windows, siga estas instrucciones:

1. Descargue e instale Cygwin desde <http://cygwin.com>.
2. Asegúrese de que `gcc` y `make` se han instalado en Cygwin. Puede hacer una prueba ejecutando los comandos `which gcc` y `which make`. Si alguno de estos no está instalado, ejecute el administrador de paquetes de Cygwin (Cygwin package manager) para instalarlos, seleccionando `gcc`, `make`, o ambos.
3. Para la instalación del cliente gratuito BitKeeper, siga las mismas instrucciones que para sistemas tipo Unix que ya se dieron anteriormente.

Nota

Para obtener más información sobre cómo utilizar los códigos fuentes de MySQL a través de BitKeeper, así como el cliente gratuito BitKeeper, vea [Instalar desde el Árbol de código fuente de desarrollo](#).

Después de haber instalado el cliente BitKeeper, ya puede acceder al árbol de desarrollo de MySQL:

1. Cambie al directorio en el que quiere trabajar, y entonces ejecute el siguiente comando para hacer una copia local del árbol `mysql-6.0-falcon`:

```
shell> bkf clone bk://mysql.bkbits.net/mysql-6.0-falcon mysql-6.0-falcon
```

En este ejemplo, se configura el árbol de código fuente en el subdirectorio `mysql-6.0-falcon/` del directorio de trabajo actual.

Por favor note que la primera descarga del árbol de código fuente puede tardar un poco, dependiendo de la velocidad de su conexión. Habrá que ser paciente y esperar.

Una vez que haya descargado el árbol de Falcon:

1. Cambie al directorio que contiene el árbol de Falcon:

```
shell> cd mysql-6.0-falcon
```

2. Ejecute el script de generación apropiado a su sistema para iniciar el proceso de generación. Para CPUs Pentium, ejecute `compile-pentium-debug-falcon`, para CPUs AMD 64, use `compile-amd64-debug-falcon`. Los scripts de generación se encuentran en `BUILD`, pero deben ser ejecutados desde el directorio raíz del árbol:

```
shell> ./BUILD/compile-pentium-debug-falcon
```

3. El script de generación creará los ficheros (stubs) necesarios para la ejecución del proceso de configuración. Luego, ejecute `configure`, y después `make` para iniciar el proceso de compilación.
4. Es mejor crear un paquete binario que puede ser distribuido y reubicado al lugar apropiado, que instalar directamente desde esta generación. Para crear una distribución así, use el script `make_binary_distribution`:

```
shell> ./scripts/make_binary_distribution
```

El comando anterior creará un archivo con todos los ficheros que necesita configurados para comportarse como si estuvieran en un directorio con el nombre de acuerdo a la versión y etiqueta para esta entrega. Por ejemplo, en un ordenador Pentium con Linux, la entrega Falcon creará un archivo con el nombre `mysql-6.0.0-falcon-alpha-linux-i686.tar`, que contiene el directorio `mysql-6.0.0-falcon-alpha-linux-i686`.

Para instalar los ficheros en un lugar nuevo, debe extraerlos especificando el nuevo directorio base:

```
shell> tar zxf mysql-6.0.0-falcon-alpha-linux-i686 -C /usr/local
```

Puede usar ya sea un directorio o un vínculo a otro lugar. Por ejemplo, se puede crear un vínculo a /usr/local/mysql-falcon:

```
shell> cd /usr/local
shell> ln -s mysql-6.0.0-falcon-alpha-linux-i686 mysql-falcon
```

5. Es necesario configurar la nueva instalación, lo que incluye crear las bases de datos básicas. Al ejecutar `configure` en este directorio se hará esta configuración, y se iniciará una instancia de `mysqld`:

```
shell> cd mysql-falcon
shell> ./configure
```

Si `mysqld` falla al iniciar, hay que asegurarse de que los ficheros en su directorio sean propiedad del usuario `mysql`:

```
shell> chown -R mysql:mysql *
```

6. Puede confirmar que `mysqld` se está ejecutando obteniendo la versión del servidor:

```
shell> ./bin/mysqladmin version
./bin/mysqladmin Ver 8.42 Distrib 6.0.0-falcon-alpha, for pc-linux-gnu on i686
Copyright (C) 2000 MySQL AB & MySQL Finland AB & TCX DataKonsult AB
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL license

Server version          6.0.0-falcon-alpha-debug
Protocol version        10
Connection              Localhost via UNIX socket
UNIX socket             /tmp/mysql.sock
Uptime:                 3 min 24 sec

Threads: 1  Questions: 1  Slow queries: 0  Opens: 14  Flush tables: 1  »
Open tables: 7  Queries per second avg:
```

Puede que haya necesidad de actualizar `LD_LIBRARY_PATH` con las bibliotecas de MySQL:

```
shell> export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/mysql-falcon/lib
```

Una vez que Falcon se ha instalado, se puede probar creando tablas con el motor de almacenamiento Falcon. Por ejemplo, para crear una tabla simple con Falcon se puede hacer lo siguiente:

```
shell> ./bin/mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 6.0.0-falcon-alpha-debug Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use test
Database changed
mysql> CREATE TABLE falcon_basic (id int, fname varchar(20)) ENGINE=Falcon;
Query OK, 0 rows affected (1.07 sec)
```

Capítulo 3. Cómo utilizar Falcon

Este capítulo contiene información de cómo utilizar Falcon, lo que incluye sus parámetros de configuración, los detalles del funcionamiento interno de Falcon y las limitaciones que tiene la entrega actual.

3.1. Parámetros de configuración

Los parámetros se configuran a través de los ficheros estándares de MySQL, `my.cnf` o `my.ini`. Los parámetros se pueden configurar especificando el nombre del parámetro y su valor correspondiente, separado por un espacio. Los valores de memoria se pueden especificar en bytes, o por un número seguido de `kb`, `mb` o `gb`.

Se puede usar `SHOW VARIABLES` para ver una lista de variables relativas a Falcon:

```
mysql> show variables like "%falcon%";
```

Variable_name	Value
falcon_checkpoint_schedule	7 * * * * *
falcon_debug_mask	0
falcon_debug_server	OFF
falcon_disable_fsync	OFF
falcon_index_chill_threshold	4
falcon_initial_allocation	0
falcon_max_transaction_backlog	150
falcon_page_cache_size	4194304
falcon_page_size	4096
falcon_record_chill_threshold	5
falcon_record_memory_max	262144000
falcon_record_scavenge_floor	50
falcon_record_scavenge_threshold	67
falcon_scavenge_schedule	15,45 * * * * *
falcon_serial_log_buffers	10
falcon_serial_log_dir	
have_falcon	YES

- `falcon_checkpoint_schedule` — la programación de los puntos de verificación (checkpoint), es decir, la frecuencia con la que se llama a `fsync()` para sincronizar los datos en la memoria con los del disco. La especificación es una serie de valores al estilo de `crontab`, separados por espacios. Cada especificación tiene seis campos, idénticos a los de `crontab`, pero con uno adicional para los segundos. Estos campos, de izquierda a derecha, son:

- segundos (0-59)
- minutos (0-59)
- horas (0-23)
- día del mes (1-31)
- mes (1-12)
- día de la semana (0-7, donde 0 y 7 son domingo)

Los valores especificados pueden ser absolutos, o bien, puede especificar un rango o una lista de valores separados por comas. Por ejemplo, la especificación:

```
7,37 * * * * *
```

hará un punto de verificación (checkpoint) cada 7 y 37 segundos de cada minuto, de cada hora de cada día. La siguiente especificación hará un checkpoint solamente de 6am a 6pm de cada día al segundo 0 y 30 de cada minuto.

```
0,30 * 6-17 * * *
```

- `falcon_debug_mask` — (antes `falcon_log_mask`) establece la información de registro que se produce como salida en la salida (output) estándar de `mysqld` cuando ocurre un error. El valor es una máscara de bits; y debe combinar varios valores para habilitar diferentes combinaciones de tipos de mensajes de error. Los valores que se pueden establecer son:

Valor	Nombre	Descripción
1	LogLog	Produce salida de errores menores, de índices, registros y otras faltas.
2	LogDebug	Produce salida de información de estado y de avance detallada como ayuda para errores de depuración.

Valor	Nombre	Descripción
4	LogInfo	Genera mensajes de información general y de estado.
8	Sin utilizar	Actualmente no se está utilizando.
16	Sin utilizar	Actualmente no se está utilizando.
32	LogGG	
64	LogPanic	
128	LogScrub	
256	LogException	Registra excepciones y errores de SQL.
512	LogScavenge	Informa de estadísticas del recolector de registros que ya no son útiles.

- `falcon_debug_server` — especifica si el servidor de depuración debe estar habilitado.

El valor predeterminado es deshabilitado (OFF).

- `falcon_disable_fsync` — si su valor es verdadero, la operación `fsync`, que se ejecuta periódicamente para sincronizar los datos al disco se deshabilita. Configurar este valor a verdadero puede resultar en pérdida de datos, pero puede incrementar el rendimiento.

El valor predeterminado es falso (`fsync` habilitado).

- `falcon_index_chill_threshold` — el número de megabytes de datos de índice pendientes que deben ser almacenados durante una transacción grande antes de que los cambios del índice sean guardados al registro serial. Si el índice es de datos únicos, o si la transacción lee repetidas veces los datos del índice, entonces éstos son almacenados en memoria (para un acceso más rápido). El guardado de los datos del índice al registro serial es llamado "enfriamiento" (chilling). El enfriamiento de los índices pendientes ayuda a Falcon a cargar conjuntos de datos grandes en una sola transacción sin agotar la memoria.

El valor mínimo es 1, el máximo es 1024 y el valor predeterminado es 4.

Esta opción de configuración está disponible dentro de `mysqld` como una variable de servidor.

- `falcon_initial_allocation` — la cantidad de espacio, en MB, que debe ser reservada en disco cuando un nuevo fichero de espacio de tablas (tablespace) de Falcon se ha creado.

El valor predeterminado es 0 (no reservar). El valor mínimo es 10.

- `falcon_max_transaction_backlog` — el número máximo de transacciones pendientes que estarán activas antes que el proceso de actualización se bloquee, hasta que el número de transacciones pendientes disminuya.

El valor predeterminado es 150.

- `falcon_page_cache_size` — (Tamaño de la caché de páginas) establece la cantidad de memoria que será reservada para el uso de las cachés de páginas del fichero del espacio de tablas.

El valor predeterminado es 4MB.

- `falcon_page_size` — (Tamaño de página) controla el tamaño de las páginas utilizadas para almacenar información dentro del espacio de tablas. Los tamaños válidos son 1, 2, 4, 8, 16 y 32 KB.

El valor predeterminado es 4KB

- `falcon_record_chill_threshold` — el número de Mbytes de datos de registros pendientes que Falcon mantendrá en memoria durante una transacción grande antes de guardar estos registros al registro serial. A este guardado se le llama "enfriamiento" porque hace que los datos no estén disponibles inmediatamente. Si los registros enfriados son accedidos nuevamente durante la transacción, éstos son restaurados inmediatamente ("derretidos") desde el registro serial. El enfriamiento de registros pendientes ayuda a Falcon a realizar transacciones muy grandes sin quedarse sin memoria.

El valor mínimo aceptado es 1, el máximo es 1024 y el valor predeterminado es 5.

- `falcon_record_memory_max` — establece la máxima cantidad de memoria que se reservará para la caché de registros de datos.

El valor predeterminado es 20MB.

- `falcon_record_scavenge_floor` — el porcentaje de `falcon_record_scavenge_threshold` que se retendrá en la caché de registros después de que el proceso recolector ha completado su ejecución.

Puede determinar el tamaño mínimo de la caché de registros con esta fórmula:

```
min(falcon_record_memory_max
    * (falcon_record_scavenge_threshold/100)
    * (falcon_record_scavenge_floor/100))
```

El valor predeterminado es 50. El mínimo aceptado es 10, y el máximo es 90.

- `falcon_record_scavenge_threshold` — el porcentaje de `falcon_record_memory_max` que causará que el proceso recolector empiece a eliminar las generaciones más antiguas de registros en la caché de registros.

El valor predeterminado es 67. El valor mínimo aceptado es 10, y el máximo es 100.

- `falcon_scavenge_schedule` — la programación de recolección de registros, especificado al estilo de la programación de `crontab`. Vea `falcon_checkpoint_schedule`.
- `falcon_serial_log_buffers` — el número de ventanas de memoria reservadas para el registro serial de Falcon. Cada ventana es de 1 MByte. Anteriormente era `falcon_log_windows`.

El mínimo valor aceptado es 10, el máximo es 32768 y el valor predeterminado es 10.

- `falcon_serial_log_dir` — (Ubicación del fichero de Registro) establece el directorio que almacenará el registro serial. Los nombres de los ficheros usados por el registro serial (dos ficheros se crean para almacenar los datos seriales) se crean de acuerdo al nombre del espacio de tablas. La ubicación predeterminada es en el mismo directorio que el fichero de espacio de tablas. Anteriormente era `falcon_log_dir`.

La relación entre la caché de registros de la caché de páginas depende de la información que almacena cada caché. Los registros que están activos (los que están siendo leídos o actualizados) se almacenan dentro de la caché de registros; sin embargo, los datos `BLOB` son almacenados únicamente dentro de la caché de páginas.

La caché de páginas se utiliza para almacenar metadatos de la base de datos, datos `BLOB` e índices de tablas.

Los parámetros de Falcon también pueden configurarse en la línea de comando de `mysqld` usando las siguientes opciones de línea de comandos:

- `--falcon-max-record-memory=#`
- `--falcon-min-record-memory=#`
- `--falcon-page-cache-size=#`

También se puede habilitar y deshabilitar el motor de almacenamiento Falcon al inicio, proveyendo estas opciones a `mysqld`, siempre que el fichero binario de `mysqld` incluya el Motor de Almacenamiento Falcon.

- `--falcon` habilita el motor de almacenamiento Falcon.
- `--skip-falcon` deshabilita el motor de almacenamiento Falcon.

3.2. Creación de espacios de tablas de Falcon

En Falcon, todos los datos de una base de datos son almacenadas dentro de un fichero dentro de la estructura de directorios de MySQL. El fichero contiene los datos de Falcon y será almacenado con el nombre de la base de datos Falcon con la extensión `.fts`. Por ejemplo, las tablas Falcon definidas dentro de la base de datos `test` serán almacenados dentro del fichero `test.fts` dentro del directorio de datos principal de MySQL.

Falcon también permite espacios de tablas con nombre, que permiten almacenar tablas dentro de ficheros específicos que pueden ser diferentes al fichero de almacenamiento predeterminado de Falcon para esa base de datos. Cuando el motor de almacenamiento Falcon es habilitado en el servidor, se crean automáticamente tres espacios de tablas. Estas tablas son:

- Un espacio de tablas sin nombre que se usa para mantener las tablas del sistema.
- `falcon_user`, usado como el lugar predeterminado para tablas definidas por el usuario.
- `falcon_temporary`, usado para mantener tablas temporales.

Todos los espacios de tablas comparten los mismos ficheros de registro, memoria y procesos. Las transacciones se ejecutan transparentemente entre todos los espacios de tablas. No hay ninguna relación inherente entre un espacio de tabla y la base de datos/esquema al cual se relaciona.

Para crear un nuevo espacio de tablas, use la sentencia `CREATE TABLESPACE`:

```
CREATE TABLESPACE espacio_de_tablas
ADD DATAFILE 'fichero'
ENGINE [=] Falcon
```

Dos ficheros adicionales son creados por Falcon, y contienen una copia en disco del registro serial de Falcon. Estos ficheros también son creados dentro del dominio de la base de datos correspondiente. En una entrega futura, será posible especificar una ubicación alterna para estos ficheros de registro. Así, con el ejemplo anterior del fichero de datos `test.fts` los ficheros de registro tendrán los nombres `test.f11` y `test.f12`.

Las definiciones de las tablas, al igual que con otros motores de almacenamiento de MySQL, se almacenan dentro de un fichero `.frm` en un directorio específico de la base de datos. Por ejemplo, la tabla `falcontest` dentro de la base de datos `test` creará el fichero de definición de tabla `falcontest.frm` dentro del directorio `test`.

3.3. Creación de tablas e índices dentro de Falcon

Falcon puede utilizar todos los tipos de datos de columnas estándares que soporta MySQL.

Para crear una tabla que utilice el motor Falcon, use la opción `ENGINE = Falcon` dentro de la sentencia `CREATE TABLE`:

```
CREATE TABLE names (id INT, fname VARCHAR (20), lname VARCHAR (20)) ENGINE=Falcon
```

Los índices se pueden crear usando los métodos estándares; por ejemplo, puede especificar explícitamente un índice en una columna:

```
CREATE TABLE ids (id int, index (id)) ENGINE=Falcon
```

Generar uno como parte de una clave primaria:

```
CREATE TABLE ids (id int),PRIMARY KEY (id) ENGINE=Falcon
```

O puede crear índices de múltiples claves y múltiples índices:

```
CREATE TABLE t1 (id int NOT NULL,id2 int NOT NULL,id3 int NOT NULL,
name CHAR(30),primary key (id,id2),index index_id3 (id3)) ENGINE=Falcon
```

Para crear una tabla dentro de un espacio de tablas específico, agrague la definición `TABLESPACE`:

```
CREATE TABLE names (id INT, fname VARCHAR (20), lname VARCHAR (20))
TABLESPACE my_big_tables ENGINE=Falcon
```

Puede usar `ALTER TABLE` para cambiar el espacio de tablas de una tabla; Falcon moverá los datos de la tabla al nuevo espacio de tablas:

```
ALTER TABLE names TABLESPACE my_small_tables
```

Puede eliminar un espacio de tablas cuando está vacío (es decir, que ya no contiene ninguna tabla) usando la sentencia `DROP TABLESPACE`:

```
DROP TABLESPACE my_big_tables ENGINE=Falcon
```

Actualmente no se puede alterar un espacio de tablas (usando `ALTER TABLESPACE`).

Nota

Actualmente, re-utilizar un espacio de tablas con el mismo nombre producirá un error. De la misma manera, cuando se elimina un fichero asociado con un espacio de tablas, el fichero en sí no se elimina físicamente en el sistema de ficheros. En cualquier caso, el error que se devuelve será: `ERROR 65433 (HY000): Unknown error -103`.

3.4. Diagnósticos de Rendimiento

Falcon exporta información de diagnóstico de rendimiento interno a las tablas globales de `INFORMATION_SCHEMA`. Actualmen-

te, Falcon genera información a las siguientes tablas:

```

+-----+
| Tables_in_information_schema |
+-----+
| FALCON_RECORD_CACHE_SUMMARY |
| FALCON_SYSTEM_MEMORY_DETAIL |
| FALCON_SYSTEM_MEMORY_SUMMARY |
| FALCON_SYNCOBJECTS |
| FALCON_RECORD_CACHE_DETAIL |
| FALCON_TRANSACTION_SUMMARY |
| FALCON_DATABASE_IO |
| FALCON_TRANSACTIONS |
| FALCON_SERIAL_LOG |
+-----+
    
```

Tabla 3.1. Tablas de diagnóstico de rendimiento de Falcon en `INFORMATION_SCHEMA`

Tabla I_S	Descripción
<code>FALCON_SYSTEM_MEMORY_DETAIL</code>	Detalle de memoria del sistema; provee una cuenta detallada del uso de memoria y de objetos a través de las diferentes instancias de clases dentro de Falcon.
<code>FALCON_SYSTEM_MEMORY_SUMMARY</code>	Resumen de la memoria del sistema; provee un resumen del uso de memoria en Falcon, incluyendo el total de memoria reservada, el espacio libre y la fragmentación.
<code>FALCON_RECORD_CACHE_DETAIL</code>	Detalle de la caché de registros; muestra el número de registros activos que contiene la caché de registros y el espacio que están consumiendo.
<code>FALCON_RECORD_CACHE_SUMMARY</code>	El resumen de la caché de registros muestra el espacio reservado y el disponible para almacenamiento de registros, incluyendo una indicación de la fragmentación de la caché de registros.
<code>FALCON_TRANSACTIONS</code>	Transacciones; muestra las transacciones actualmente activas y su estado y dependencias, incluyendo el número de registros afectados y la edad de la transacción.
<code>FALCON_TRANSACTION_SUMMARY</code>	Resumen de transacciones de las transacciones activas.
<code>FALCON_SYNCOBJECTS</code>	SyncObjects; muestra el detalle de la utilización interna de objetos de Falcon. Note que puesto que existen conjuntos separados de objetos de sincronización para cada base de datos activa, puede que obtenga filas de información duplicada en la tabla generada.
<code>FALCON_SERIAL_LOG</code>	Información de estado del registro serial. Muestra las transacciones y la utilización de objetos del registro serial por base de datos.
<code>FALCON_DATABASE_IO</code>	Estadísticas de E/S, que muestran el tamaño de página, tamaño de búfer y las lecturas/escrituras por cada base de datos.
<code>FALCON_TABLES</code>	Muestra una lista de todas las tablas de Falcon.

Para obtener información de diagnóstico puede hacerlo ejecutando una sentencia `SELECT`. Dependiendo de la tabla de I_S que utilice, la información provista será por base de datos o por tabla. Si la información es sobre una tabla, y esa tabla está almacenada dentro de un espacio de tablas único, entonces el nombre del espacio de tablas se presentará junto con el nombre de la tabla. Por ejemplo, se puede obtener estadísticas de E/S (I/O) de bases de datos Falcon desde la tabla `falcon_database_io`:

```

mysql> select * from information_schema.falcon_database_io;
+-----+
| DATABASE | PAGE_SIZE | BUFFERS | READS | WRITES | FETCHES | FAKES |
+-----+
| RMS      | 4096      | 2560    | 0     | 109    | 98687   | 615   |
| GIMF     | 4096      | 2560    | 565   | 28     | 56870   | 3     |
+-----+
    
```

También puede unir información con `JOIN` entre diferentes tablas para obtener estadísticas más específicas. Por ejemplo, la sentencia que sigue mostrará la lista de sentencias sobre tablas de Falcon que estén bloqueando durante las transacciones:

```

mysql> select a.id thread, a.user,b.id txn_id,b.database,a.time, b.waiting_for, statement
-> from information_schema.processlist a, information_schema.falcon_transactions b
-> where a.id = b.thread_id;
+-----+
| thread | user | txn_id | database | time | waiting_for | statement |
+-----+
| 2      | root | 8      | GIMF     | 0    | 0           |           |
| 3      | root | 9      | GIMF     | 76   | 8           | update rms set c1=5 where c1=1 |
+-----+
    
```

3.5. Principios y Terminología

Para aprovechar al máximo el motor Falcon, debe entender los principios básicos y terminología de Falcon.

La arquitectura de MySQL Falcon combina técnicas avanzadas con una estructura simplificada que resulta en una base de datos transaccional de alto rendimiento que requiere poco mantenimiento y reparación por parte del administrador de la base de datos.

- **Fichero de datos del usuario** — almacena los datos de Falcon.
- **Registro serial de Falcon** — contiene los cambios de datos confirmados recientemente, los cambios de índices e información transaccional. También provee facilidades para la recuperación de datos.
- **Caché de páginas** — mantiene las páginas de la base de datos que están siendo leídas o escritas.
- **Caché de registros** — mantiene copias de los registros activos y sin confirmar.
- **Memoria del sistema** — contiene información de contexto de transacciones, aceleradores de índices y los metadatos del sistema.
- **Procesos de trabajo** — son procesos de fondo. Hay dos procesos, el proceso "mandadero" ("gopher") mueve datos desde el registro serial de Falcon hacia la caché de páginas de la base de datos y desde la caché de páginas al disco. El segundo es el proceso escritor de páginas, el cual escribe páginas blob.

3.5.1. Ficheros de datos y estructuras de datos de Falcon

Un solo fichero de base de datos de Falcon almacena todos los datos de registros, índices, estructura de la base de datos y otra información. La información individual se almacena dentro de una serie de páginas.

Las páginas son la unidad interna con la que el motor de almacenamiento Falcon reserva espacio de almacenamiento. Las páginas se utilizan para almacenar información de datos e índice. Entre los factores que afectan el rendimiento del motor están el tamaño de página, la manera como el motor Falcon guarda en cachés y cómo reserva páginas cuando almacena la información, aunque también depende de los registros que están siendo almacenados, así como de la complejidad de los índices utilizados.

Las páginas almacenadas en la caché de memoria se utilizan para almacenar índices, blobs y los datos estructurales para un espacio de tablas dado. Los registros activos (aquellos que se están leyendo o actualizando) son almacenados dentro de una caché de registros separada.

Todas las transacciones en la base de datos son registradas y almacenadas dentro de un fichero de registro separado. El fichero de registros es guardado automáticamente y los cambios se escriben al disco cuando hay un comando `COMMIT`, cuando auto-commit está habilitado, o automáticamente cada 30 segundos cuando las transacciones no están siendo utilizadas.

3.5.2. Registro serial de Falcon

Falcon utiliza un registro serial para mantener ciertos tipos de información antes que los datos sean confirmados a la base de datos. El registro es utilizado para almacenar los siguientes tipos de información:

- Registros de datos durante la fase de confirmación (commit phase).
- Los cambios físicos de una base de datos necesarios para la recuperación de datos luego de una falla.
- Los cambios lógicos de una base de datos necesarios para la recuperación de recursos luego de una falla.
- Los cambios de estado de transacción de todas las transacciones activas (activa a confirmada, activa a cancelada (rolled back), activa a limbo).

Todas las transacciones dentro de Falcon son escritas al registro serial y luego se confirman a la base de datos. Esto se hace automáticamente si `AUTOCOMMIT` está habilitado, o manualmente cuando se usa el comando `COMMIT`.

La información de registro se almacena en la memoria, y los cambios que no se hayan escrito se guardan periódicamente al disco. Un proceso de fondo es el que se encarga de procesar el contenido del registro, confirmando los cambios del registro a la base de datos. Así, el proceso de confirmación (commit) es el que establece el estado final de todos los registros y las páginas, a pesar de cualquier estado intermedio. Esto quiere decir, que sólo el estado final se escribe al disco.

Sin embargo, nótese que el proceso de confirmación del registro serial sólo actualiza los datos de registro a través de la caché de páginas en memoria. Los datos de registros en sí, serán escritos hacia el disco cuando ocurra el proceso de punto de verificación (checkpoint). La excepción a esta regla son las entradas de índices y registro (log), que son escritas inmediatamente al disco durante el proceso de confirmación (commit).

Falcon crea dos ficheros de registro serial. El primero es utilizado para almacenar los datos del registro serial hasta que el registro

(log) llegue a un tamaño determinado. Una vez el fichero llega a ese tamaño, el registro se almacena al segundo fichero de registro (log) serial. El proceso de confirmación (commit) continúa leyendo del primer fichero de registro (log) hasta que todas las transacciones hayan sido escritas a la base de datos. Entonces el primer fichero de registro se libera y se vuelve a crear, listo para volver a ser utilizado.

Luego de esto, las entradas del registro (log) en el segundo fichero se procesan hasta que todas las transacciones en el registro (log) hayan terminado. Entonces, ese fichero se libera y se vuelve a crear, listo para ser utilizado nuevamente cuando el primer fichero de registro se llene o cuando esté bloqueado por las confirmaciones (commits).

3.5.2.1. Proceso de Deshacer (Rollback)

Cuando una transacción se debe deshacer (rollback), es el mismo proceso de la transacción el que se encarga de deshacer la transacción. La acción de deshacer (rollback) implica lo siguiente:

- Revertir las actualizaciones de índices.
- Revertir cualquier dato binario (blob) creado por la transacción.
- Liberar espacios reservados de registros.
- Revertir las versiones de los registros creados en la memoria.

3.5.2.2. Confirmaciones en grupo (Group Commits)

Por razones de rendimiento, Falcon utiliza un sistema de confirmación en grupo (group commit) que se encarga de que todas las actualizaciones pendientes en el registro serial (serial log) sean escritas al disco al mismo tiempo. Falcon puede tener múltiples transacciones activas, pero sólo una escribe todos los cambios pendientes al registro serial en el disco, reduciendo así el número de escrituras en el disco y al mismo tiempo mejora el rendimiento general del registro serial.

Por ejemplo:

1. La transacción 1 es confirmada (commit), y crea todas las entradas necesarias del registro (log), y comienza a escribir el registro (log) al disco.
2. Mientras la confirmación de la transacción 1 está siendo escrita, las transacciones 2 y 3 escriben sus entradas de registro (log) al registro serial (serial log).
3. Cuando la transacción 1 termina de escribir al disco, la transacción 2 ó la 3 (pero no ambas) escribirá la porción de datos del registro en memoria, que todavía no ha sido escrita, al disco. Puesto que ambas transacciones han ocurrido desde la última vez que se escribió al disco el registro serial (log), la información de las dos transacciones es escrita al disco al mismo tiempo.
4. Mientras las transacciones 2 y 3 están escribiendo, las transacciones 4, 5 y 6 están siendo escritas al registro (log) en memoria. Cuando las transacciones 2 y 3 han terminado de escribir al disco, las entradas de las transacciones 4, 5 y 6 se escriben al disco.

El resultado de este proceso es que solamente se escribe al disco físico tres veces, a pesar de que realmente hay seis transacciones en la secuencia:

- Transacción 1
- Transacciones 2 y 3
- Transacciones 4, 5 y 6

El proceso continúa, solamente con una transacción escribiendo todas las entradas en memoria del registro serial (log) al disco desde la última escritura al disco. Todo el sistema se encarga de que los registros (logs) en memoria y en disco estén sincronizados, con la menor cantidad posible de escrituras al disco.

3.5.3. Recuperación de fallos de Falcon

Cuando la primera tabla de una base de datos de Falcon se abre, se examina el Registro Serial (serial log). Si el estado del registro indica que hay transacciones sin confirmar, el proceso de recuperación comienza automáticamente y actualiza la base de datos. Los cambios y transacciones que se escriben en el registro serial (log) incluyen entradas que indican cambios en todas las áreas de la base de datos, incluyendo índices, cambios a datos binarios `BLOB`, y cualquier cambio estructural de la base de datos.

Durante la recuperación de fallos, Falcon examina el registro serial (log) e identifica la primera entrada que no se ha confirmado a la base de datos. El proceso de recuperación escribe todos los datos sin escribir, cambios en los índices y datos blob, libera cualquier espacio de registro (de registros eliminados) y confirma cualquier cambio estructural.

3.5.4. Cachés de memoria de Falcon

Falcon fue diseñado para obtener buen rendimiento en sistemas con cantidades generosas de memoria. Las cachés de memoria utilizadas por Falcon son similares en algunos aspectos con los utilizados en otros SGBDRs (RDBMS) y motores de MySQL; sin embargo, las estructuras de las cachés de Falcon tienen algunas mejoras con respecto a las estrategias de las cachés de memoria tradicionales. Las cachés de memoria utilizadas por Falcon son las siguientes:

- **Caché de Registro (Log Cache)** — la información de registro (log) se mantiene en memoria hasta que se vacía hacia el registro de Falcon (Falcon Log) cuando las transacciones son confirmadas. Falcon mantiene ocho ventanas de 1 MB en el fichero de registro para lectura y escritura.
- **Caché de Sistema e Índice** — los datos que Falcon utiliza (definiciones de tablas y campos, estados de las transacciones, etc.) también se mantiene en la memoria para referencia rápida. Además, los aceleradores de índice locales, que representan segmentos de índice creados por las transacciones, también se almacenan en la memoria del sistema. Cuando una transacción cambia campos indexados, se genera una sección de acelerador de índice en la memoria del sistema, que representa los cambios del índice. Al confirmar la transacción (commit), todos los cambios del índice en la transacción se escriben al registro serial (serial log) en el orden del índice y luego el proceso de fondo se encarga de fusionarlo con el índice permanente.
- **Caché de páginas** — las páginas de datos leídas desde el disco para una base de datos en particular. El tamaño de la caché de páginas se controla por medio del parámetro `falcon_page_cache_size`, cuyo valor predeterminado es 4MB, y se configura en el fichero `my.cnf`. Aunque los cambios de registros (record) e índices van al registro serial (serial log) antes de ser escritos a las páginas de bases de datos, los datos blob se escriben directamente a la caché de páginas. Esto evita registrar elementos grandes de datos que raras veces se les hace referencia o se cambian por las transacciones que los crea.
- **Caché de registros** — la caché de registros es una región de memoria dedicada a mantener filas que han sido pedidas por consultas del usuario final para una base de datos particular, o que han sido creadas por transacciones activas. Nótese que esta caché difiere de las cachés de datos tradicionales, ya que solo almacena las filas específicas que necesita la aplicación, y no páginas de datos enteras (de las cuales sólo un subconjunto es la información que se necesita). Esta caché puede almacenar varias versiones de los registros que han sido modificados o eliminados. Esta técnica garantiza que los datos que se necesitan para satisfacer las peticiones del usuario están en la memoria, acorta el tiempo de acceso a la fila, y reduce el tamaño de la caché al no incluir información no solicitada. La caché de registros también sirve para los mecanismos del Control de Concurrencia Multi Versión (Multi Version Concurrency Control, MVCC) del motor Falcon. La caché de registros es controlada por dos parámetros. El parámetro `falcon_min_record_memory` (valor predeterminado 10MB) determina la cantidad mínima de RAM para la caché de registros, y `falcon_max_record_memory` (valor predeterminado 20MB) indica el límite para la cantidad total de memoria disponible para la caché de registros.
- Debido al soporte que la caché de registros significa para las transacciones, el proceso recolector se utiliza para asegurar que sólo existen datos "calientes" en la caché. Cuando se alcanza el límite definido por `falcon_max_record_memory`, Falcon muestrea la demográfica de los datos generacionales en la caché, y elimina las generaciones más antiguas. Este proceso, aunque es más complicado que el algoritmo estándar LRU utilizado por muchos sistemas de bases de datos, es más eficiente y rápido.

3.5.5. Subprocesos de Falcon (Threads)

Falcon utiliza dos procesos de trabajo para procesar la información dentro de las estructuras de Falcon. Un proceso, el proceso "mandadero" ("gopher"), se dedica a mover cambios de datos confirmados desde el registro (log) de Falcon a las páginas de datos y de fusionar cambios de los índices con los datos de índice permanentes. El segundo proceso maneja el guardado periódico de la caché de páginas y recolecta el espacio reservado dentro de la caché de registros.

3.5.6. Compresión de Datos

Los datos almacenados en los espacios de tablas de Falcon están comprimidos en el disco, pero se almacenan sin comprimir en la memoria. La compresión de los datos ocurre automáticamente cuando los datos se confirman al disco duro.

3.5.7. Espacio de Registro

Un espacio de registro es un identificador de registros (record) interno que se utiliza para encontrar los registros en memoria y en disco. Básicamente es un apuntador a las páginas que contienen los datos para un registro en particular. Un espacio de registro nuevo se crea para cada registro durante la existencia de ese registro. El espacio de registro se libera solamente cuando el registro es borrado de la base de datos.

3.6. Límites de Falcon

Existen ciertos límites en la entrega alfa de Falcon. Estos serán trabajados en las siguientes entregas:

- Actualmente, Falcon solo está disponible para los sistemas operativos Windows y Linux, ambos de 32 y 64 bits.
- La longitud máxima de clave está limitada a 1100 bytes.
- No se soportan niveles de insulación serializable.
- Falcon se comporta como si la opción `lower_case_table_names` estuviera habilitada sin importar la plataforma actual.
- No se soporta la configuración de tiempo de espera de bloqueo.
- No se soportan las transacciones distribuidas.
- Existe un límite de 2^{32} (4.29 mil millones) de filas para una sola tabla. Puede tener más registros usando múltiples tablas dentro del mismo espacio de tablas. En entregas futuras este límite será removido.
- Cada espacio de tablas tiene un límite de 2^{32} páginas dentro de un espacio de tablas. Al combinar el tamaño de página y el número máximo de páginas en un espacio de tablas, se obtiene un límite de 140,737,488,355,328 bytes (128 TB) para un espacio de tablas.
- No se soporta la copia de respaldo en línea (Online backup), pero está planeado para una entrega futura.
- Soporte para claves foráneas actualmente no está disponible.

Aunque la cantidad máxima de almacenamiento disponible dentro de un espacio de tablas es 128TB, el número real de registros y de datos que puede almacenar depende de varios factores:

- Los requerimientos de almacenaje de los registros
- Los requerimientos de almacenaje de los índices
- Radio de compresión de los datos almacenados

Debido a la compleja relación entre el almacenaje, indexación y las utilidades de compresión es imposible calcular con precisión el espacio de almacenamiento en disco necesario para un conjunto de datos específico.

Apéndice A. Plan de Desarrollo (Roadmap) de Falcon

Las siguientes características serán agregadas a Falcon antes de que llegue a GA (General Availability). Esta sección está sujeta a cambios mientras el desarrollo de Falcon de MySQL esté en sus primeras etapas.

- Transacciones XA, incluyendo ejecución de dos fases (two phase commit) permanente
- Optimización para la cláusula `LIMIT`
- Adición y eliminación de índices en línea
- Soporte para Mac OS X
- Truncamiento del fichero de registro (log)

Apéndice B. Cambios en MySQL 6.0 Falcon

Nota

Antes de MySQL 6.0.0 Falcon Alpha (del 13 de abril de 2007), todas las entregas de Falcon eran basadas del árbol MySQL 5.1. Las entregas hechas desde esta versión, incluyendo los bugs y otros cambios, aplican al árbol MySQL 6.0 Falcon.

B.1. Cambios en la entrega 6.0.2 (04 septiembre 2007)

Funcionalidad agregada o cambiada:

- Soporte para Windows de 64 bits.
- Se ha añadido soporte para Mac OS X (Intel). Para generar sobre Mac OS X a partir del código fuente del repositorio debe tener las versiones más recientes de `bison`, `automake`, `autoconf` y `libtool` instalados.

Existen problemas conocidos con la generación de Falcon sobre Mac OS X. ([Bug#30564](#))

- Soporte para espacios de tablas.
- Nuevos parámetros de configuración de rendimiento, `falcon_log_windows`, `falcon_index_chill_threshold`, y `falcon_record_chill_threshold`.
- Los parámetros para la caché de registros de Falcon se han cambiado. Los parámetros `falcon_max_record_memory` y `falcon_min_record_memory` ya no se soportan.

En su lugar, los parámetros `falcon_record_memory_max`, `falcon_record_scavenge_threshold`, `falcon_record_scavenge_floor` y `falcon_inital_allocation` ahora se usan para controlar la caché de registros en memoria dentro de Falcon. Consulte [Sección 3.1, "Parámetros de configuración"](#). ([Bug#30083](#))

- La opción `falcon_initial_allocation` se ha añadido para controlar el tamaño inicial de un espacio de tablas de Falcon en el disco.
- La opción `falcon_disable_fsync` se ha añadido. Si se establece como verdadera, la operación periódica `fsync` se deshabilitará.

Bugs corregidos:

- El cambio de nombre de tablas a o desde espacios de tablas de Falcon generaba un error. ([Bug#22155](#))
- Algunas variables de Falcon estaban marcadas como variables de estado. ([Bug#29169](#))
- En ciertas circunstancias las tablas y el registro de Falcon podían corromperse y prevenir la recuperación a partir de los ficheros posteriores a una falla. ([Bug#28351](#))
- Actualizar una tabla grande sin índice podía bloquear todos los registros durante una transacción, y desbloquearlos individualmente. ([Bug#30124](#))
- Falcon podía permitir incorrectamente la creación de dos tablas con el mismo nombre, pero diferente "case sensitivity", sin generar un error, pero trataba las dos tablas como la misma en las consultas subsiguientes. ([Bug#30210](#)).
- Cuando se cargaban conjuntos de datos grandes a una tabla de Falcon, `mysqld` podía fallar. Ahora, un error de Falta de Memoria (Out of memory) se generará en esta situación. ([Bug#30251](#))
- Se lanzaba una aserción cuando había una cantidad grande de actualizaciones de campos `BLOB`. ([Bug#30463](#))
- Accesar una tabla de `INFORMATION_SCHEMA` generada por Falcon, cuando Falcon no estaba habilitado podía hacer que `mysqld` fallara. ([Bug#29014](#))
- En ocasiones, Falcon podía reportar un problema con un error de clave duplicada durante un `INSERT` cuando ingresaba los mismos datos a una columna de datos únicos en dos o más conexiones simultáneamente. ([Bug#29240](#))
- Para generaciones de depuración (debug builds), el servidor fallaba cuando se ingresaba un valor negado `DECIMAL` de máxima precisión (65 dígitos), por ejemplo en el comando `INSERT INTO ... SELECT -col_val ...` ([Bug#28810](#))
- Al crear un espacio de tablas con un nombre único pero usando el mismo fichero de datos que un espacio de tablas existente re-

sultaba en la re-inicialización del espacio de tablas y la pérdida de los datos contenidos en él. Ahora Falcon reporta un error si el fichero de datos ya existe. ([Bug#29511](#))

- Las sentencias `DELETE` podían causar un fallo cuando muchos procesos simultáneos se estaban ejecutando. ([Bug#26475](#))
- Hacer un `SELECT` sobre una tabla que usaba dos columnas `INT` con un solo índice podía fallar al devolver filas que consultaban ambas columnas y con operadores complejos de comparación. ([Bug#29319](#))
- Columnas anchas de tipo `DECIMAL` mostraban errores de redondeo durante un `SELECT`. ([Bug#29201](#))
- Ingresar registros en una tabla con índice único simultáneamente en dos conexiones de tal manera que causara un interbloqueo podía causar que MySQL colgara. Ahora, se indentifica la situación de interbloqueo y se genera un error. ([Bug#29206](#))
- Accesar datos dentro de columnas tipo `DECIMAL` de ancho mayor que 18 dígitos podía causar una falla. ([Bug#28725](#))
- Falcon podía no ser generado bajo Mac OS X/Intel. Un parche preliminar está disponible para permitir la generación bajo Mac OS X/Intel solamente (Todavía no hay soporte para PowerPC). Nótese que Mac OS X/Intel todavía sigue siendo una plataforma sin soporte. ([Bug#26466](#))
- En ciertas situaciones, detener MySQL utilizando `mysqladmin` podía causar que Falcon corrompiera las tablas de la base de datos y que no pudiera reiniciar apropiadamente. ([Bug#26296](#))
- Las consultas podían fallar con el error `Can't find record in ...` ([Bug#26328](#))
- Columnas de tipo `DECIMAL` de ancho grande no funcionaban bien, ni durante `INSERT` o `SELECT`. ([Bug#26607](#)).
- Renombrar una base de datos podía generar un error `ERROR 1030 (HY000): Got error 157 from storage engine`. ([Bug#22182](#))
- La carga de ciertos conjuntos de datos, importándolos directamente podían provocar problemas con los índices y fallar. ([Bug#26930](#))
- Agregar grandes cantidades de columnas idénticas en una tabla, seguido por un `SELECT` o `UPDATE` podía causar que colgara o fallara. ([Bug#27277](#))
- El valor de `FALCON_SYSTEM_MEMORY_SUMMARY.TOTAL_SPACE` reportada en `INFORMATION_SCHEMA` podía ser incorrecta. ([Bug#28197](#))
- Los índices únicos en columnas tipo `VARCHAR` no se identifican correctamente. ([Bug#28500](#))
- `mysqld` podía fallar después de un alto número de sentencias `ALTER TABLE`, `INSERT` y `UPDATE`. ([Bug#22154](#), [Bug#28515](#))
- La búsqueda de filas dentro de una tabla con algunos conjuntos de caracteres occidentales podía no devolver los resultados correctos si el `SELECT` dependía de un índice. ([Bug#27697](#))
- Las búsquedas de caracteres acentuados en una tabla UTF8 fallaban si existía un índice para la columna. ([Bug#26057](#))
- Las búsquedas usando `LIKE` en una tabla UTF8 fallaban si la búsqueda dependía de una columna indexada. ([Bug#24921](#))
- Búsquedas de datos sobre un índice parcial para una columna que usaba el conjunto de caracteres UTF8 podían fallar. ([Bug#24858](#))
- Búsquedas de datos que usaban conjuntos de datos o colación exóticos fallaban si la búsqueda dependía de una columna indexada. ([Bug#23689](#))
- Agregar filas a una tabla con un índice único donde el valor del índice único es idéntico en dos conexiones separadas podía bloquear la segunda transacción. ([Bug#22847](#))
- Grandes inserciones a una tabla dentro de una sola transacción disparaba un alto uso de memoria y finalmente fallaba. ([Bug#22169](#))

B.2. Cambios en la entrega 6.0.1 (No entregado)

Nota

Este fue una entrega interna, y no se publicaron ficheros binarios.

B.3. Cambios en la entrega 6.0.0 (30 Abril 2007)

Funcionalidad agregada o cambiada:

- Ahora se soporta la sentencia `SELECT ... FOR UPDATE`.
- Se ha implementado la recolección de registros sin confirmar.
- Diagnósticos de rendimiento están disponibles a través del `INFORMATION_SCHEMA`.

Bugs corregidos:

- Usar `ALTER` with transacciones intermedias produce una falla en `mysqld`. (Bug#22165)
- Actualizar columnas `BLOB` puede resultar en una falla. (Bug#26324)
- Usar `SELECT ... FOR UPDATE` y `ROLLBACK` podía hacer que `mysqld` colgara indefinidamente. (Bug#28165)
- Actualizaciones concurrentes en dos conexiones diferentes podía llevar a la falla de una aserción. (Bug#28090)
- Al actualizar una fila en una tabla con índice compuesto de valores únicos a un valor no único, no se generaba un error. (Bug#27997)
- Manejadores "Continue" dentro de procedimientos almacenados podían causar una falla. (Bug#26433)
- Deshacer la inserción de una fila mientras se accedía a la misma en una conexión diferente podía causar una falla. (Bug#27993)
- Crear una tabla con una columna `DECIMAL` de 19 dígitos podía causar que se almacenaran datos incorrectos. Esto es debido a una limitación actual en Falcon por la que no se puede crear una tabla con una columna de precisión mayor de 18 dígitos (p.ej. `DECIMAL(18,9)`). Crear una columna con una especificación mayor que esta ahora fallará y levantará un error. (Bug#27962)
- Ejecutar `INSERT INTO ... SELECT FROM` podía causar una falla en conjuntos de datos grandes. (Bug#27951)
- Ingresar datos en la misma tabla en dos conexiones diferentes con autocommit deshabilitado podía causar una falla. (Bug#27895)
- Crear una tabla Falcon inmediatamente después de crear una base de datos nueva podía causar una falla. (Bug#27768)
- Ejecutar `SELECT ... FOR UPDATE` en una segunda conexión sobre una tabla recién creada y populada podía causar una falla. (Bug#27767)
- Eliminar una gran cantidad de filas en una tabla podía resultar en el `ERROR 1020`. (Bug#26055)
- Ejecutar `SELECT` después de cambiar el contenido de una tabla no resulta en un nuevo conjunto de datos. (Bug#22181)
- Actualizar continuamente una columna `BLOB` podía causar que el servidor MySQL fallara. (Bug#27719)
- Utilizar un trigger sobre un comando `UPDATE` a una tabla Falcon cuando autocommit estaba deshabilitado podía causar que el servidor MySQL server fallara. (Bug#27574)
- Utilizar `ROLLBACK` después de `DELETE` no restablecía la fila eliminada. (Bug#27357)
- Inserciones de filas a una tabla con columnas `VARCHAR` grandes, y con índices compuestos grandes podía causar que MySQL fallara. (Bug#26850)
- Falcon podía consumir grandes cantidades de memoria en grandes cantidades de sentencias `INSERT` continuas. (Bug#26843)
- Actualizar una tabla particionada en dos sesiones simultáneamente podía causar que MySQL fallara. (Bug#26828)
- Bloquear entre sesiones utilizando `SELECT ... FOR UPDATE` podía no funcionar. (Bug#26826)
- En búsquedas de registros en tablas con una columna tipo `decimal(6,6)` el valor podía no ser encontrado. (Bug#26469)
- En filas con una columna numérica, podía no encontrar registros con valor cero. (Bug#26468)
- Tablas con un índice grande de múltiples columnas podía no encontrar un registro para la sentencia `UPDATE`. (Bug#26420)
- Tablas con restricciones de clave tipo `UNIQUE` no eran respetadas. (Bug#26803)
- Al abrir la misma base de datos con tablas Falcon en diferentes conexiones podía causar una falla. (Bug#27428)

- Interrumpir un procedimiento almacenado durante su ejecución podía causar una falla. ([Bug#27539](#))
- Dos sentencias `SELECT ... FOR UPDATE` simultáneas con nivel de insulación `READ-COMMITTED` podía resultar en que se regresara un mensaje de error incorrecto. ([Bug#26871](#))
- Cuando se actualizaba una tabla con una restricción de clave única, la restricción no se respetaba. ([Bug#26802](#))
- Ejecutar `DROP TABLE` sobre una tabla que fue creada usando `CREATE TABLE ... SELECT`. ([Bug#25564](#))
- Obtener filas desde una tabla que utiliza un índice podía a veces no obtener la fila. ([Bug#26452](#))
- Actualizaciones aleatorias de columnas tipo `LONG VARCHAR` podían fallar. ([Bug#23818](#))

B.4. Cambios en MySQL 5.2 Falcon

Un resumen de los cambios en la secuencia de las entregas de Falcon en MySQL 5.2.

B.4.1. Cambios en la entrega 5.2.3 (15 de febrero 2007)

Bugs corregidos:

- MySQL podía fallar con una aserción en el inicio durante una recuperación haciendo referencia a `tc.log`. ([Bug#26161](#))
- Una falla podía ocurrir al ejecutar `UPDATE` dentro de un ciclo en un procedimiento almacenado. ([Bug#25537](#))

B.4.2. Cambios en la entrega 5.2.2 (no entregado)

Bugs corregidos:

- MySQL podía fallar con una aserción al inicio. ([Bug#25835](#))

B.4.3. Cambios en la entrega 5.2.1 (no entregado)

Funcionalidad agregada o cambiada

- Mejoras de rendimiento: se redujeron los cuellos de botella en procesos cuando una gran cantidad de procesos auto-commit en paralelo ejecutaban una consulta trivial en un bucle.

Bugs corregidos:

- Between falla con un campo Unicode. ([Bug#24511](#))
- Falla si se crea un índice sobre una columna que permite valores nulos (nullable) `utf8`. ([Bug#25555](#))
- MySQL falla durante la sentencia `DROP TABLE` después de un `CREATE TABLE ... SELECT *`. ([Bug#25564](#))
- Hay aserción cuando se termina una sentencia `CREATE TABLE ... SELECT`. ([Bug#25565](#))
- Problema de clave primaria compuesta en Falcon. ([Bug#25828](#))