Atitit mysql insert perf enhance 批量插入数据库性能

目录

[1.1. 案一：使用ignore关键字 1](#_Toc16296)

[2. 异步插入 2](#_Toc28192)

[2.1. 其它关键：DELAYED  做为快速插入，并不是很关心失效性，提高插入性能。 IGNORE  只关注主键对应记录是不存在，无则添加，有则忽略。 2](#_Toc20906)

[2.2. 13.2.5.3 INSERT DELAYED语句 异步 2](#_Toc29479)

[2.3. 多线程 批量 java级别 5](#_Toc8079)

[3. other 5](#_Toc18479)

[3.1. Archil引擎 5](#_Toc25462)

[3.2. Mem引擎 5](#_Toc13848)

[3.3. 严禁批量更新默认是一个长事务 6](#_Toc25853)

[3.4. 检查移除多进程，防止多个进程同时更改同一记录 6](#_Toc26463)

[4. Other 6](#_Toc26805)

[4.1. 方案三：ON DUPLICATE KEY UPDATE 6](#_Toc27185)

## ****案一：使用ignore关键字****

如果是用主键primary或者唯一索引unique区分了记录的唯一性,避免重复插入记录可以使用：

INSERT IGNORE INTO `table\_name` (`email`, `phone`, `user\_id`)

VALUES ('test9@163.com', '99999', '9999');

这样当有重复记录就会忽略,执行后返回数字0

# 异步插入

## 其它关键：DELAYED  做为快速插入，并不是很关心失效性，提高插入性能。 IGNORE  只关注主键对应记录是不存在，无则添加，有则忽略。

[MySQL 5.6参考手册](https://dev.mysql.com/doc/refman/5.6/en/)  /  [...](https://dev.mysql.com/doc/refman/5.6/en/insert-delayed.html)  /  INSERT DELAYED语句

## 13.2.5.3 INSERT DELAYED语句 异步

INSERT DELAYED ...

该语句的DELAYED选项 [INSERT](https://dev.mysql.com/doc/refman/5.6/en/insert.html" \o "13.2.5 INSERT语句)是对标准SQL的MySQL扩展，可用于某些类型的表（例如MyISAM）。当客户端使用时 [INSERT DELAYED](https://dev.mysql.com/doc/refman/5.6/en/insert-delayed.html" \o "13.2.5.3 INSERT DELAYED语句)，它将立即从服务器获得许可，并且当该表未被任何其他线程使用时，该行将排队插入。

**注意**

[INSERT DELAYED](https://dev.mysql.com/doc/refman/5.6/en/insert-delayed.html" \o "13.2.5.3 INSERT DELAYED语句)[INSERT](https://dev.mysql.com/doc/refman/5.6/en/insert.html" \o "13.2.5 INSERT语句)如果不使用该表，则它比正常速度慢。服务器还要为每个有延迟行的表处理一个单独的线程，这会产生额外的开销。这意味着[INSERT DELAYED](https://dev.mysql.com/doc/refman/5.6/en/insert-delayed.html" \o "13.2.5.3 INSERT DELAYED语句)仅在确实确定需要时才应使用 。

从MySQL 5.6.6开始，[INSERT DELAYED](https://dev.mysql.com/doc/refman/5.6/en/insert-delayed.html" \o "13.2.5.3 INSERT DELAYED语句)已弃用，并将在以后的版本中删除。使用[INSERT](https://dev.mysql.com/doc/refman/5.6/en/insert.html" \o "13.2.5 INSERT语句) （不带DELAYED）代替。

排队的行仅保留在内存中，直到将它们插入表中为止。这意味着，如果强行终止[mysqld](https://dev.mysql.com/doc/refman/5.6/en/mysqld.html" \o "4.3.1 mysqld-MySQL服务器)（例如，使用 kill -9）或[mysqld](https://dev.mysql.com/doc/refman/5.6/en/mysqld.html" \o "4.3.1 mysqld-MySQL服务器)意外终止 ，则*所有尚未写入磁盘的排队行都将丢失*。

使用以下限制 DELAYED：

[INSERT DELAYED](https://dev.mysql.com/doc/refman/5.6/en/insert-delayed.html" \o "13.2.5.3 INSERT DELAYED语句)只有工作MyISAM，MEMORY， ARCHIVE，和BLACKHOLE 表格。对于不支持的引擎， DELAYED会发生错误。

[INSERT DELAYED](https://dev.mysql.com/doc/refman/5.6/en/insert-delayed.html" \o "13.2.5.3 INSERT DELAYED语句)如果将其与已锁定的表一起使用， 则会发生错误，LOCK TABLES因为插入必须由单独的线程而不是由持有锁的会话处理。

对于MyISAM表，如果数据文件中间没有空闲块，则支持并发 [SELECT](https://dev.mysql.com/doc/refman/5.6/en/select.html" \o "13.2.9 SELECT语句)和 [INSERT](https://dev.mysql.com/doc/refman/5.6/en/insert.html" \o "13.2.5 INSERT语句)语句。在这种情况下，你很少需要使用[INSERT DELAYED](https://dev.mysql.com/doc/refman/5.6/en/insert-delayed.html" \o "13.2.5.3 INSERT DELAYED语句)带 MyISAM。

[INSERT DELAYED](https://dev.mysql.com/doc/refman/5.6/en/insert-delayed.html" \o "13.2.5.3 INSERT DELAYED语句)仅应用于[INSERT](https://dev.mysql.com/doc/refman/5.6/en/insert.html" \o "13.2.5 INSERT语句)指定值列表的语句。服务器忽略 DELAYEDfor [INSERT ... SELECT](https://dev.mysql.com/doc/refman/5.6/en/insert-select.html" \o "13.2.5.1 INSERT ... SELECT语句)或 [INSERT ... ON DUPLICATE KEY UPDATE](https://dev.mysql.com/doc/refman/5.6/en/insert-on-duplicate.html" \o "13.2.5.2 INSERT ... ON DUPLICATE KEY UPDATE语句)语句。

因为该[INSERT DELAYED](https://dev.mysql.com/doc/refman/5.6/en/insert-delayed.html" \o "13.2.5.3 INSERT DELAYED语句) 语句立即返回，所以在插入行之前，您不能使用它 [LAST\_INSERT\_ID()](https://dev.mysql.com/doc/refman/5.6/en/information-functions.html" \l "function_last-insert-id)来获取AUTO\_INCREMENT该语句可能生成的 值。

DELAYED在[SELECT](https://dev.mysql.com/doc/refman/5.6/en/select.html" \o "13.2.9 SELECT语句)实际插入之前，行对语句不可见 。

[INSERT DELAYED](https://dev.mysql.com/doc/refman/5.6/en/insert-delayed.html" \o "13.2.5.3 INSERT DELAYED语句)每当is 或的值时，将作为简单的[INSERT](https://dev.mysql.com/doc/refman/5.6/en/insert.html" \o "13.2.5 INSERT语句)（即不带DELAYED选项）进行处理。（在后一种情况下，该语句不会触发切换到基于行的日志记录，因此将使用基于语句的格式进行记录。） [binlog\_format](https://dev.mysql.com/doc/refman/5.6/en/replication-options-binary-log.html" \l "sysvar_binlog_format)STATEMENTMIXED

当使用基于行的二进制日志记录模式（[binlog\_format](https://dev.mysql.com/doc/refman/5.6/en/replication-options-binary-log.html" \l "sysvar_binlog_format)设置为 ROW）时，该方法不适用，在该模式下， [INSERT DELAYED](https://dev.mysql.com/doc/refman/5.6/en/insert-delayed.html" \o "13.2.5.3 INSERT DELAYED语句)始终使用DELAYED指定的选项执行语句并将其记录为行更新事件。

DELAYED在从属复制服务器上被忽略，因此在从属服务器上[INSERT DELAYED](https://dev.mysql.com/doc/refman/5.6/en/insert-delayed.html" \o "13.2.5.3 INSERT DELAYED语句)被视为正常 [INSERT](https://dev.mysql.com/doc/refman/5.6/en/insert.html" \o "13.2.5 INSERT语句)。这是因为DELAYED可能导致从机拥有与主机不同的数据。

[INSERT DELAYED](https://dev.mysql.com/doc/refman/5.6/en/insert-delayed.html" \o "13.2.5.3 INSERT DELAYED语句) 如果表被写锁定并[ALTER TABLE](https://dev.mysql.com/doc/refman/5.6/en/alter-table.html" \o "13.1.7 ALTER TABLE语句)用于修改表结构，则 待处理语句将丢失 。

[INSERT DELAYED](https://dev.mysql.com/doc/refman/5.6/en/insert-delayed.html" \o "13.2.5.3 INSERT DELAYED语句) 视图不支持。

[INSERT DELAYED](https://dev.mysql.com/doc/refman/5.6/en/insert-delayed.html" \o "13.2.5.3 INSERT DELAYED语句) 不支持分区表。

以下详细描述了使用DELAYED选项 [INSERT](https://dev.mysql.com/doc/refman/5.6/en/insert.html" \o "13.2.5 INSERT语句)或 时会发生什么 [REPLACE](https://dev.mysql.com/doc/refman/5.6/en/replace.html" \o "13.2.8更换声明)。在本说明中， “ 线程 ”是接收[INSERT DELAYED](https://dev.mysql.com/doc/refman/5.6/en/insert-delayed.html" \o "13.2.5.3 INSERT DELAYED语句)语句 的线程， “ 处理程序 ”是处理[INSERT DELAYED](https://dev.mysql.com/doc/refman/5.6/en/insert-delayed.html" \o "13.2.5.3 INSERT DELAYED语句)特定表的所有语句的线程 。

当线程执行DELAYED 表的DELAYED语句时，如果尚不存在处理程序线程，则会创建一个处理程序线程来处理该表的所有语句。

线程检查处理程序以前是否已获取DELAYED锁；如果不是，它告诉处理程序线程这样做。DELAYED 即使其他线程在表上具有READ或WRITE锁，也可以获得该锁 。但是，处理程序将等待所有 [ALTER TABLE](https://dev.mysql.com/doc/refman/5.6/en/alter-table.html" \o "13.1.7 ALTER TABLE语句)锁或 [FLUSH TABLES](https://dev.mysql.com/doc/refman/5.6/en/flush.html" \l "flush-tables)语句完成，以确保表结构是最新的。

该线程执行该 [INSERT](https://dev.mysql.com/doc/refman/5.6/en/insert.html" \o "13.2.5 INSERT语句)语句，但不是将行写入表中，而是将最后一行的副本放入由处理程序线程管理的队列中。该线程会发现任何语法错误，并将其报告给客户端程序。

客户端无法从服务器获取重复的行数或AUTO\_INCREMENT 结果行的值，因为 [INSERT](https://dev.mysql.com/doc/refman/5.6/en/insert.html" \o "13.2.5 INSERT语句)插入操作完成之前的返回。（如果使用C API，则[mysql\_info()](https://dev.mysql.com/doc/refman/5.6/en/mysql-info.html" \o "23.7.6.35 mysql_info（）)由于相同的原因，该函数不会返回任何有意义的信息。）

当将行插入表中时，处理程序线程会更新二进制日志。如果是多行插入，则在插入第一行时更新二进制日志。

每次 [delayed\_insert\_limit](https://dev.mysql.com/doc/refman/5.6/en/server-system-variables.html" \l "sysvar_delayed_insert_limit)写入行时，处理程序都会检查是否还有任何 [SELECT](https://dev.mysql.com/doc/refman/5.6/en/select.html" \o "13.2.9 SELECT语句)语句尚未处理。如果是这样，则允许它们在继续之前执行。

当处理程序的队列中没有更多行时，该表将被解锁。如果[INSERT DELAYED](https://dev.mysql.com/doc/refman/5.6/en/insert-delayed.html" \o "13.2.5.3 INSERT DELAYED语句)在[delayed\_insert\_timeout](https://dev.mysql.com/doc/refman/5.6/en/server-system-variables.html" \l "sysvar_delayed_insert_timeout) 几秒钟内未收到新语句 ，则处理程序终止。

如果[delayed\_queue\_size](https://dev.mysql.com/doc/refman/5.6/en/server-system-variables.html" \l "sysvar_delayed_queue_size)在特定处理程序队列中有多个 行挂起，则请求线程 [INSERT DELAYED](https://dev.mysql.com/doc/refman/5.6/en/insert-delayed.html" \o "13.2.5.3 INSERT DELAYED语句)等待，直到队列中有空间为止。这样做是为了确保 [mysqld](https://dev.mysql.com/doc/refman/5.6/en/mysqld.html" \o "4.3.1 mysqld-MySQL服务器)不会将所有内存用于延迟的内存队列。

处理程序线程显示在MySQL进程列表 delayed\_insert中的 Command列中。如果执行一条[FLUSH TABLES](https://dev.mysql.com/doc/refman/5.6/en/flush.html" \l "flush-tables) 语句或使用杀死它，它将被杀死。但是，在退出之前，它首先将所有排队的行存储到表中。在此期间，它不接受来自其他线程的任何新 语句。如果在此之后执行语句，则会创建一个新的处理程序线程。 KILL ***thread\_id***[INSERT](https://dev.mysql.com/doc/refman/5.6/en/insert.html" \o "13.2.5 INSERT语句)[INSERT DELAYED](https://dev.mysql.com/doc/refman/5.6/en/insert-delayed.html" \o "13.2.5.3 INSERT DELAYED语句)

这意味着，如果有 处理程序正在运行，则[INSERT DELAYED](https://dev.mysql.com/doc/refman/5.6/en/insert-delayed.html" \o "13.2.5.3 INSERT DELAYED语句)语句的优先级高于普通[INSERT](https://dev.mysql.com/doc/refman/5.6/en/insert.html" \o "13.2.5 INSERT语句)语句[INSERT DELAYED](https://dev.mysql.com/doc/refman/5.6/en/insert-delayed.html" \o "13.2.5.3 INSERT DELAYED语句)。其他更新语句必须等到[INSERT DELAYED](https://dev.mysql.com/doc/refman/5.6/en/insert-delayed.html" \o "13.2.5.3 INSERT DELAYED语句)队列为空，有人终止处理程序线程（带有 ）或有人执行。 KILL ***thread\_id***[FLUSH TABLES](https://dev.mysql.com/doc/refman/5.6/en/flush.html" \l "flush-tables)

以下状态变量提供有关[INSERT DELAYED](https://dev.mysql.com/doc/refman/5.6/en/insert-delayed.html" \o "13.2.5.3 INSERT DELAYED语句)语句的信息 。

| **状态变量** | **含义** |
| --- | --- |
| [Delayed\_insert\_threads](https://dev.mysql.com/doc/refman/5.6/en/server-status-variables.html" \l "statvar_Delayed_insert_threads) | 处理程序线程数 |
| [Delayed\_writes](https://dev.mysql.com/doc/refman/5.6/en/server-status-variables.html" \l "statvar_Delayed_writes) | 写入的行数 [INSERT DELAYED](https://dev.mysql.com/doc/refman/5.6/en/insert-delayed.html" \o "13.2.5.3 INSERT DELAYED语句) |
| [Not\_flushed\_delayed\_rows](https://dev.mysql.com/doc/refman/5.6/en/server-status-variables.html" \l "statvar_Not_flushed_delayed_rows) | 等待写入的行数 |

您可以通过发出[SHOW STATUS](https://dev.mysql.com/doc/refman/5.6/en/show-status.html" \o "13.7.5.36 SHOW STATUS语句)语句或执行[mysqladmin extended-status](https://dev.mysql.com/doc/refman/5.6/en/mysqladmin.html" \o "4.5.2 mysqladmin-用于管理MySQL服务器的客户端) 命令来查看这些变量 。

## 多线程 批量 java级别

# other

## Archil引擎

## Mem引擎

去除插入触发器，改为update触发器模式 手动刷新

在没有更新update之前，加个fullfld标识。。可以不在select显示，只有update过了，才在ui显示

插入触发器改用代码实现 一次更新一条

## 严禁批量更新默认是一个长事务

可用代码拆分细分

## 检查移除多进程，防止多个进程同时更改同一记录

# Other

## ****方案三：ON DUPLICATE KEY UPDATE****

如‍上所写，你也可以在INSERT INTO…..后面加上 ON DUPLICATE KEY UPDATE方法来实现。如果您指定了ON DUPLICATE KEY UPDATE，并且插入行后会导致在一个UNIQUE索引或PRIMARY KEY中出现重复值，则执行旧行UPDATE。

您可以在UPDATE子句中使用VALUES(col\_name)函数从INSERT…UPDATE语句的INSERT部分引用列值。换句话说，如果没有发生重复关键字冲突，则UPDATE子句中的VALUES(col\_name)可以引用被插入的col\_name的值。本函数特别适用于多行插入。VALUES()函数只在INSERT…UPDATE语句中有意义，其它时候会返回NULL。

INSERT INTO `table` (`a`, `b`, `c`) VALUES (1, 2, 3), (4, 5, 6)

ON DUPLICATE KEY UPDATE `c`=VALUES(`a`)+VALUES(`b`);

本语句与以下两个语句作用相同：

INSERT INTO `table` (`a`, `b`, `c`) VALUES (1, 2, 3) ON DUPLICATE KEY UPDATE `c`=3;

INSERT INTO `table` (`a`, `b`, `c`) VALUES (4, 5, 6) ON DUPLICATE KEY UPDATE c=9;