mysql[数据库](http://111cn.cn/database/database.html)程序优化方法  
　1、选取最适用的字段属性

　　MySQL可以很好的支持大数据量的存取，但是一般说来，数据库中的表越小，在它上面执行的查询也就会越快。因此，在创建表的时候，为了获得更好的性能，我们可以将表中字段的宽度设得尽可能小。例如，在定义邮政编码这个字段时，如果将其设置为CHAR(255),显然给数据库增加了不必要的空间，甚至使用VARCHAR这种类型也是多余的，因为CHAR(6)就可以很好的完成任务了。同样的，如果可以的话，我们应该使用MEDIUMINT而不是BIGIN来定义整型字段。

　　另外一个提高效率的方法是在可能的情况下，应该尽量把字段设置为NOT NULL，这样在将来执行查询的时候，数据库不用去比较NULL值。

　　对于某些文本字段，例如“省份”或者“性别”，我们可以将它们定义为ENUM类型。因为在MySQL中，ENUM类型被当作数值型数据来处理，而数值型数据被处理起来的速度要比文本类型快得多。这样，我们又可以提高数据库的性能。

　　2、使用连接（JOIN）来代替子查询(Sub-Queries)

　　MySQL从4.1开始支持SQL的子查询。这个技术可以使用SELECT语句来创建一个单列的查询结果，然后把这个结果作为过滤条件用在另一个查询中。例如，我们要将客户基本信息表中没有任何订单的客户删除掉，就可以利用子查询先从销售信息表中将所有发出订单的客户ID取出来，然后将结果传递给主查询，如下所示：

　　DELETE FROM customerinfo WHERE CustomerID NOT in (SELECT CustomerID FROM salesinfo )

　　使用子查询可以一次性的完成很多逻辑上需要多个步骤才能完成的SQL操作，同时也可以避免事务或者表锁死，并且写起来也很容易。但是，有些情况下，子查询可以被更有效率的连接（JOIN）.. 替代。例如，假设我们要将所有没有订单记录的用户取出来，可以用下面这个查询完成：

　　SELECT \* FROM customerinfo WHERE CustomerID NOT in (SELECT CustomerID FROM salesinfo )

　　如果使用连接（JOIN）.. 来完成这个查询工作，速度将会快很多。尤其是当salesinfo表中对CustomerID建有索引的话，性能将会更好，查询如下：

　　SELECT \* FROM customerinfo LEFT JOIN salesinfoON customerinfo.CustomerID=salesinfo. CustomerID WHERE salesinfo.CustomerID IS NULL

　　连接（JOIN）.. 之所以更有效率一些，是因为 MySQL不需要在内存中创建临时表来完成这个逻辑上的需要两个步骤的查询工作。

　　3、使用联合(UNION)来代替手动创建的临时表

　　MySQL 从 4.0 的版本开始支持 UNION 查询，它可以把需要使用临时表的两条或更多的 SELECT 查询合并的一个查询中。在客户端的查询会话结束的时候，临时表会被自动删除，从而保证数据库整齐、高效。使用 UNION 来创建查询的时候，我们只需要用 UNION作为关键字把多个 SELECT 语句连接起来就可以了，要注意的是所有 SELECT 语句中的字段数目要想同。下面的例子就演示了一个使用 UNION的查询。

　　SELECT Name, Phone FROM client UNION SELECT Name, BirthDate FROM author  
　　UNION  
　　SELECT Name, Supplier FROM product

　　4、事务

　　尽管我们可以使用子查询（Sub-Queries）、连接（JOIN）和联合（UNION）来创建各种各样的查询，但不是所有的数据库操作都可以只用一条或少数几条SQL语句就可以完成的。更多的时候是需要用到一系列的语句来完成某种工作。但是在这种情况下，当这个语句块中的某一条语句运行出错的时候，整个语句块的操作就会变得不确定起来。设想一下，要把某个数据同时插入两个相关联的表中，可能会出现这样的情况：第一个表中成功更新后，数据库突然出现意外状况，造成第二个表中的操作没有完成，这样，就会造成数据的不完整，甚至会破坏数据库中的数据。要避免这种情况，就应该使用事务，它的作用是：要么语句块中每条语句都操作成功，要么都失败。换句话说，就是可以保持数据库中数据的一致性和完整性。事物以BEGIN 关键字开始，COMMIT关键字结束。在这之间的一条SQL操作失败，那么，ROLLBACK命令就可以把数据库恢复到BEGIN开始之前的状态。

　　BEGIN;

　　INSERT INTO salesinfo SET CustomerID=14;

　　UPDATE inventory SET Quantity=11

　　WHERE item='book';

　　COMMIT;

　　事务的另一个重要作用是当多个用户同时使用相同的数据源时，它可以利用锁定数据库的方法来为用户提供一种安全的访问方式，这样可以保证用户的操作不被其它的用户所干扰。

　　5、锁定表

　　尽管事务是维护数据库完整性的一个非常好的方法，但却因为它的独占性，有时会影响数据库的性能，尤其是在很大的应用系统中。由于在事务执行的过程中，数据库将会被锁定，因此其它的用户请求只能暂时等待直到该事务结束。如果一个数据库系统只有少数几个用户

　　来使用，事务造成的影响不会成为一个太大的问题；但假设有成千上万的用户同时访问一个数据库系统，例如访问一个电子商务网站，就会产生比较严重的响应延迟。

　　其实，有些情况下我们可以通过锁定表的方法来获得更好的性能。下面的例子就用锁定表的方法来完成前面一个例子中事务的功能。

　　LOCK TABLE inventory WRITE  
　　SELECT Quantity FROM inventory  
　　WHEREItem='book';  
　　...

　　UPDATE inventory SET Quantity=11  
　　WHEREItem='book';  
　　UNLOCK TABLES

　　这里，我们用一个 SELECT 语句取出初始数据，通过一些计算，用 UPDATE 语句将新值更新到表中。包含有 WRITE 关键字的 LOCK TABLE 语句可以保证在 UNLOCK TABLES 命令被执行之前，不会有其它的访问来对 inventory 进行插入、更新或者删除的操作。

　　6、使用外键

　　锁定表的方法可以维护数据的完整性，但是它却不能保证数据的关联性。这个时候我们就可以使用外键。例如，外键可以保证每一条销售记录都指向某一个存在的客户。在这里，外键可以把customerinfo 表中的CustomerID映射到salesinfo表中CustomerID，任何一条没有合法CustomerID的记录都不会被更新或插入到salesinfo中。

　　CREATE TABLE customerinfo  
　　(  
　　　CustomerID INT NOT NULL ,  
　　　PRIMARY KEY ( CustomerID )  
　　) TYPE = INNODB;  
　　CREATE TABLE salesinfo  
　　(  
　　　SalesID INT NOT NULL,  
　　　CustomerID INT NOT NULL,  
　　　PRIMARY KEY(CustomerID, SalesID),  
　　　FOREIGN KEY (CustomerID) REFERENCES customerinfo  
　　　(CustomerID) ON DELETECASCADE  
　　) TYPE = INNODB;

　　注意例子中的参数“ON DELETE CASCADE”。该参数保证当 customerinfo 表中的一条客户记录被删除的时候，salesinfo 表中所有与该客户相关的记录也会被自动删除。如果要在 MySQL 中使用外键，一定要记住在创建表的时候将表的类型定义为事务安全表 InnoDB类型。该类型不是 MySQL 表的默认类型。定义的方法是在 CREATE TABLE 语句中加上 TYPE=INNODB。如例中所示。

　　7、使用索引

　　索引是提高数据库性能的常用方法，它可以令数据库服务器以比没有索引快得多的速度检索特定的行，尤其是在查询语句当中包含有MAX(), MIN()和ORDERBY这些命令的时候，性能提高更为明显。那该对哪些字段建立索引呢？一般说来，索引应建立在那些将用于JOIN, WHERE判断和ORDER BY排序的字段上。尽量不要对数据库中某个含有大量重复的值的字段建立索引。对于一个ENUM类型的字段来说，出现大量重复值是很有可能的情况，例如customerinfo中的“province”.. 字段，在这样的字段上建立索引将不会有什么帮助；相反，还有可能降低数据库的性能。我们在创建表的时候可以同时创建合适的索引，也可以使用ALTER TABLE或CREATE INDEX在以后创建索引。此外，MySQL

　　从版本3.23.23开始支持全文索引和搜索。全文索引在MySQL 中是一个FULLTEXT类型索引，但仅能用于MyISAM 类型的表。对于一个大的数据库，将数据装载到一个没有FULLTEXT索引的表中，然后再使用ALTER TABLE或CREATE INDEX创建索引，将是非常快的。但如果将数据装载到一个已经有FULLTEXT索引的表中，执行过程将会非常慢。

　　8、优化的查询语句

　　绝大多数情况下，使用索引可以提高查询的速度，但如果SQL语句使用不恰当的话，索引将无法发挥它应有的作用。下面是应该注意的几个方面。首先，最好是在相同类型的字段间进行比较的操作。在MySQL 3.23版之前，这甚至是一个必须的条件。例如不能将一个建有索引的INT字段和BIGINT字段进行比较；但是作为特殊的情况，在CHAR类型的字段和VARCHAR类型字段的字段大小相同的时候，可以将它们进行比较。其次，在建有索引的字段上尽量不要使用函数进行操作。

　　例如，在一个DATE类型的字段上使用YEAE()函数时，将会使索引不能发挥应有的作用。所以，下面的两个查询虽然返回的结果一样，但后者要比前者快得多。

　　SELECT \* FROM order WHERE YEAR(OrderDate)<2001;  
　　SELECT \* FROM order WHERE OrderDate<"2001-01-01";

　　同样的情形也会发生在对数值型字段进行计算的时候：

　　SELECT \* FROM inventory WHERE Amount/7<24;  
　　SELECT \* FROM inventory WHERE Amount<24\*7;

　　上面的两个查询也是返回相同的结果，但后面的查询将比前面的一个快很多。第三，在搜索字符型字段时，我们有时会使用 LIKE 关键字和通配符，这种做法虽然简单，但却也是以牺牲系统性能为代价的。例如下面的查询将会比较表中的每一条记录。

　　SELECT \* FROM books  
　　WHERE name like "MySQL%"

　　但是如果换用下面的查询，返回的结果一样，但速度就要快上很多：

　　SELECT \* FROM books  
　　WHERE name>="MySQL"and name<"MySQM"

1:调节服务器参数  
　　  
　　用shell>mysqld-help这个命令声厂一张所有mysql选项和可配置变量的表.输出以下信息:  
　　  
　　possible variables for option--set-variable(-o) are:  
　　  
　　back\_log current value:5 //要求mysql能有的连接数量.back\_log指出在mysql暂停接受连接的时间内有多少个连接请求可以被存在堆栈中  
　　  
　　connect\_timeout current value:5 //mysql服务器在用bad handshake(不好翻译)应答前等待一个连接的时间  
　　  
　　delayed\_insert\_timeout current value:200 //一个insert delayed在终止前等待insert的时间  
　　  
　　delayed\_insert\_limit current value:50 //insert delayed处理器将检查是否有任何select语句未执行,如果有,继续前执行这些语句  
　　  
　　delayed\_queue\_size current value:1000 //为insert delayed分配多大的队  
　　  
　　flush\_time current value:0 //如果被设置为非0,那么每个flush\_time 时间,所有表都被关闭  
　　  
　　interactive\_timeout current value:28800 //服务器在关上它之前在洋交互连接上等待的时间  
　　  
　　join\_buffer\_size current value:131072 //用与全部连接的缓冲区大小  
　　  
　　key\_buffer\_size current value:1048540 //用语索引块的缓冲区的大小,增加它可以更好的处理索引  
　　  
　　lower\_case\_table\_names current value:0 //  
　　  
　　long\_query\_time current value:10 //如果一个查询所用时间大于此时间,slow\_queried计数将增加  
　　  
　　max\_allowed\_packet current value:1048576 //一个包的大小  
　　  
　　max\_connections current value:300 //允许同时连接的数量  
　　  
　　max\_connect\_errors current value:10 //如果有多于该数量的中断连接,将阻止进一步的连接,可以用flush hosts来解决  
　　  
　　max\_delayed\_threads current value:15 //可以启动的处理insert delayed的数量  
　　  
　　max\_heap\_table\_size current value:16777216 //  
　　  
　　max\_join\_size current value:4294967295 //允许读取的连接的数量  
　　  
　　max\_sort\_length current value:1024 //在排序blob或者text时使用的字节数量  
　　  
　　max\_tmp\_tables current value:32 //一个连接同时打开的临时表的数量  
　　  
　　max\_write\_lock\_count current value:4294967295 //指定一个值(通常很小)来启动mysqld,使得在一定数量的write锁定之后出现read锁定  
　　  
　　net\_buffer\_length current value:16384 //通信缓冲区的大小--在查询时被重置为该大小  
　　  
　　query\_buffer\_size current value:0 //查询时缓冲区大小  
　　  
　　record\_buffer current value:131072 //每个顺序扫描的连接为其扫描的每张表分配的缓冲区的大小  
　　  
　　sort\_buffer current value:2097116 //每个进行排序的连接分配的缓冲区的大小  
　　  
　　table\_cache current value:64 //为所有连接打开的表的数量  
　　  
　　thread\_concurrency current value:10 //  
　　  
　　tmp\_table\_size current value:1048576 //临时表的大小  
　　  
　　thread\_stack current value:131072 //每个线程的大小  
　　  
　　wait\_timeout current value:28800 //服务器在关闭它3之前的一个连接上等待的时间