选择优化的数据类型：

1. 尽量使用正确存储数据的最小数据类型。更小的数据类型通常更快，因为它们占用更少的磁盘、内存和CPU缓存，并且处理时需要的CPU周期也更少。
2. 简单数据类型的操作通常需要更少的CPU周期。例如，整型比字符操作代价更低，因为字符集和校对规则使字符比较比整型比较更复杂。一个是使用Mysql内建的类型（date、time、datetime）而不是字符串来存储日期和时间，另外一个应该用整型存储IP地址。
3. 尽量避免用NULL

有两种类型的数字：整数和实数。如果存储整数，可以使用这几种整数类型：TINYINT,SMALLINT,MEDIUMINT,INT,BEGINT。分别使用8，16，24，32，64位存储空间。

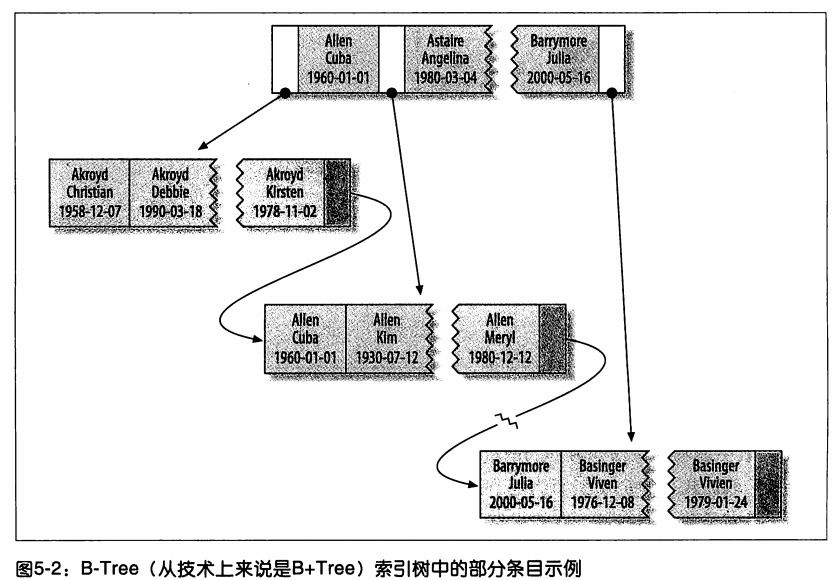
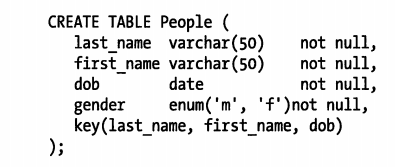
MYSQL可以为整数类型指定宽度，例如INT（11）,对于大多数应用这是没有意义的：它不会限制值得合法范围，只是规定了MYSQL得一些交互工具用来显示字符得个数。对于存储和计算来说，INT（1）和INT（20）是相同的。

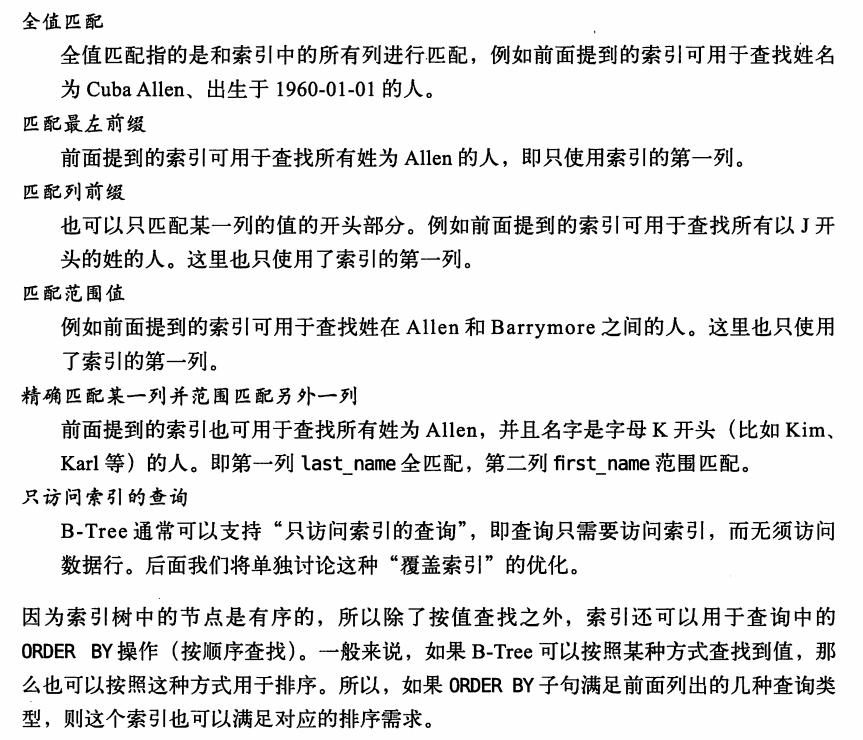
VARCHAR和CHAR是两种最主要的字符串类型。

VARCHAR类型用于存储可变长字符串，是最常见的字符串数据类型。它比定长类型更节省空间，因为它仅使用必要的空间。但会容易产生碎片。适合的情况：字符串列的最大长度比平均长度大很多；列的更新很少，所以碎片不是问题。VARCHAR需要使用1或2个额外字节来记录字符串的长度：如果列的最大长度小于或等于255字节，则只会使用1个字节表示，否则使用2个字节。

CHAR类型是定长的：MYSQL总是根据定义的字符串长度分配足够的空间。当存储CHAR值时，Mysql会删除所有的末尾空格。CHAR适合存储很短的字符串，或者所有值都接近同一个长度。对于经常变更的数据，CHAR也比VARCHAR更好，因为定长的CHAR类型不容易产生碎片。对于非常短的列，CHAR也比VARCHAR在存储空间上也更有效率。

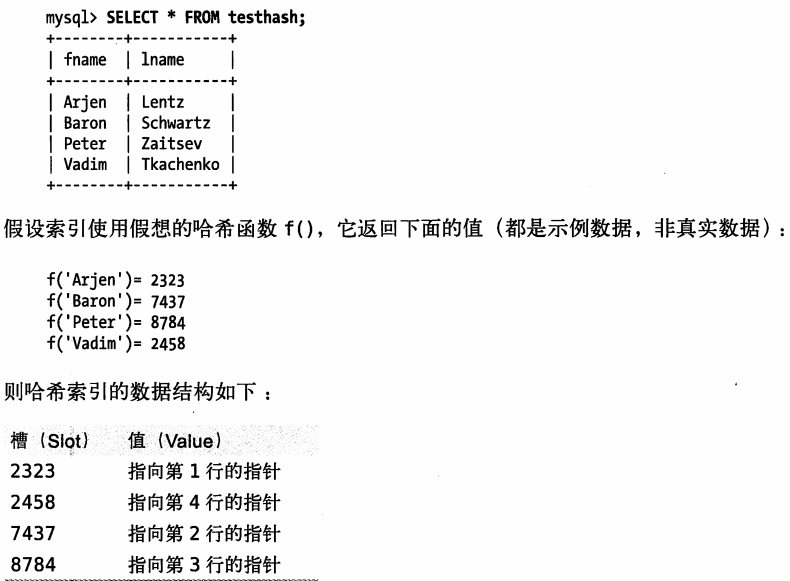
BLOB和TEXT都是为存储很大的数据而设计的字符串数据类型，分别采用二进制和字符方式存储。与其他类型不同，MySQL把每个BLOB和TEXT值当作一个独立的对象来处理。存储引擎在存储时会做特殊处理。当BLOB和TEXT值太大时，InnoDB会使用专门的“外部”存储区域来进行存储，此时每个值在行内需要1~4个字节存储一个指针，然后在外部存储区域存储实际的值。





B-Tree索引的限制：

1. 如果不是按照索引的最左列开始查找，则无法使用索引。
2. 不能跳过索引中的列。
3. 如果查询中有某个列的查询范围，则其右边所有列都无法使用索引优化查找。例如有查询WHERE last\_name=’Smith’ AND first\_name LIKE ‘J%’ AND dob = ‘1976-12-23’，这个查询只能使用索引的前两列，因为这里LIKE是一个范围条件。



1. 哈希索引只包含哈希值和行指针，而不存储字段值，所以不能使用索引中的值来避免读取行。不过，访问内存中的行的速度很快，所以大部分情况下这一点对性能的影响并不明显。
2. 哈希索引数据并不是按照索引值顺序存储的，所以也就无法用于排序。
3. 哈希索引也不支持部分索引列匹配，因为哈希索引始终是使用索引列的全部内容计算哈希值的。例如，在数据列（A，B）上建立哈希索引，如果查询只有数据列A，则无法使用该索引。
4. 哈希索引只支持等值比较查询。

查询性能优化：

