```
    }
    rows.Close()
    return
}
```

Posts 函数使用了 sql.DB 结构的 Query 方法来执行查询，这个方法会返回一个 Rows 接口。Rows 接口是一个迭代器，程序可以通过重复调用它的 Next 方法来对其进行迭代并获得相应的 sql.Row；当所有行都被迭代完毕时，Next 方法将返回 io.EOF 作为结果。

Posts 函数在每次进行迭代的时候都会创建一个 Post 结构，并将行包含的数据扫描到结构里面，然后再将这个结构追加到 posts 切片的末尾。当所有行都被迭代完毕之后，Posts 函数就会将这个包含了多个 Post 结构的 posts 切片返回给调用者。

6.4 Go 与 SQL 的关系

关系数据库之所以能够成为一种流行的数据存储手段，其中一个原因就是它可以在表与表之间建立关系，从而使不同的数据能够以一种一致且易于理解的方式互相进行关联。基本上，有 4 种方法可以把一项记录与其他记录关联起来：

- 一对一关联，也被称为“有一个”（has one）关系，比如一个用户必然会拥有一个个人简介；
- 一对多关联，也被称为“有多个”（has many）关系，比如一个用户可能会拥有多篇论坛帖子；
- 多对一关联，也被称为“属于”（belongs to）关系，比如多篇论坛帖子可能会同属于某一个用户；
- 多对多关联，比如一个用户可能会参与论坛里面多篇帖子的讨论，而一篇帖子里面也会有多个用户在发表评论。

在前面的内容中，我们已经学习了如何对单个数据库表执行标准的 CRUD 操作，但我们还不知道如何才能对两个相关联的表执行相同的操作。因此，在这一节，我们将要学习如何通过一对多关系为一篇论坛帖子构建多篇评论。与此同时，因为一对多关系跟多对一关系实际上就是一体两面的两个东西，所以除了一对多关系之外，我们还会学习如何使用多对一关系。

6.4.1 设置数据库

在正式开始之前，我们需要再次对数据库进行设置，不过跟上次只创建一个表的做法不同，这一次我们将会创建两个表。此外，这次设置需要用到的命令跟上一次设置使用的命令完全一样，只是被执行的 setup.sql 脚本跟之前的有所不同，代码清单 6-13 展示了新脚本的具体定义。

代码清单 6-13 创建两个相关联的表

```
drop table posts cascade if exists;
drop table comments if exists;
```