第二部分 MySQL 编程接口

第5章 MySQL 程序设 计介绍

在本书的这部分中，我们将讨论编写自己的访问 MySQL 数据库的程序所需要知道的内容。MySQL 有一组实用程序。例如， mysqldump 导出表的上下文和结构定义， mysqlimport 将数据文件加载到表中， mysqladmin 实现管理w操作， mysql 可以使用户与服务器交互来执行任意的查询。每个标准的 MySQL 实用程序都倾向于小巧，重点放在程序可完成特定的、有限的功能。即使 mysql 也是如此，从感觉上说， mysql 比其他实用程序更灵活，因此可以用它来执行任何数量的各种查询，即它就是为允许向服务器直接发布 SQL 查询，并可查看查询结果这一单一目的而设计的。

MySQL 客户机这种有限的特性并不是缺点，而是特意设计的。程序是具有通用目的的实用程序；它们并不试图预料您所想做的所有可能的需要。 MySQL 的开发者们不赞成编写大型的、臃肿的程序来试图做可能想去做的每件事情（而且这样做的结果将使程序中包括大量的您根本不关心的事情的代码）。然而，有时有些应用确实有常规客户机的能力所无法处理的需求。为了处理这些情况， MySQL 提供一个客户机编程库。这允许您编写自己的程序，满足您的应用程序可能具有的任何特定需求。通过允许您对 MySQL 服务器的访问，客户机的开放程度只受您自己想象力的限制了。

编写自己的程序可以获取如何特殊的能力呢？让我们比较一下 mysql 客户机和其没有附加代码的接口对 MySQL 服务器的访问：

- 可以定制输入处理。用 mysql可以输入原始的 SQL 语句。用自己的程序，可以为用户提供使用起来更直观、更容易的输入方法。用程序可使用户不必知道 SQL —— 甚至不必知道在完成的任务中数据库承担的角色。

  输入信息的采集可能是像命令行风格的提示和值读取这样基本的方式，或者可能是使用屏幕管理程序包（如 curses 或 S-Lang）、使用 Tcl/Tk 的 X 窗口或 Web 浏览器格式实现的基于屏幕输入那样复杂的方式。

  对大多数人来说，通过填写一定的格式来指定搜索参数的形式比通过发布 SELECT 语句更容易。例如，一位房地产经纪人，要寻找一定价格范围、风格或位置的房屋，只要将查寻参数输入到表格中，就可以最小的代价得到符合条件的内容。输入新记录或更新已有记录也类似地考虑这种应用。在数据输入部门的键盘操作员应该不需要知道像 INSERT、REPLACE 或 UPDATE 这样的 SQL 语法。

  在最终用户和 MySQL 服务器之间提出输入采集层的另一个原因是可以校验用户提供的输入。例如，可以检查数据，确认它们是符合 MySQL 要求的格式，或可以要求填写特定的区域。
- 可以定制输出。mysql 的输出基本上是无格式的；可以选择以制表符为分隔符或以表格

形式输出。如果想要使输出结果看起来比较好，则必须自己对它进行格式化。这些需求可能像打印“ Missing ”而不是 NULL 这样简单，也可能更复杂。考虑下面的报告：

```
State  City         Sales
-----------------------------
AZ     Mesa          $94,384.24
       Phoenix       $17,328.28
       ----------------------
       subtotal     $117,712.52
-----------------------------
CA     Los Angeles $118,198.18
       Oakland       $38,838.36
       ----------------------
       Subtotal     $157,036.54
=============================
       TOTAL        $274,749.06
```

这个报告包括几个特定的元素：

- 定制标题。
- 在 State 列中重复值的抑制以便只在更改时才将这些值打印出来。
- 小计和总计的计算。
- 数字格式，如 94384.24，打印为美元数量为$94,384.24。

对于一些任务，甚至可能不需要任何输出。您可能正在对计算向后插入到另一个数据库表中的结果进行简单地检索信息。除了用户运行这个查询以外，甚至可能还想将这个结果输出到其他地方。例如，如果正在提取姓名和电子邮件地址以自动地填入为批量电子邮件生成信件格式的过程中，则程序产生输出。但是该输出由邮件接受者的信息组成，而没有运行程序人员的信息。

- **可以在SQL自身施加的约束条件的环境下工作**。SQL 不是一种带有条件选择、循环和子例程的流程控制结构的过程语言。 SQL 脚本包括一组从开始到结束一次一个的可执行语句，具有最低限度的错误检查。

  如果在批处理模式中使用mysql 执行 SQL 查询的一个文件，则mysql 在出现第一个错误后退出，或者，如果指定 --force 选项，则不管出现多少错误，都不加选择地执行所有查询。程序可以围绕语句提供流程控制，以便可以有选择地适应查询的成功或失败。可以根据另一个查询的成功或失败来执行一个查询，或根据前一个查询的结果来决定下一步要做的事情。

  SQL 具有非常有限的语句间的连续性，这点也被带到mysql 中。使用一个查询的结果，并将它们应用于另一个查询中，或将多个查询结果联系在一起是困难的。LAST_INSERT_ID() 可用于获取由前一个语句最新生成的AUTO_INCREMENT 值，仅仅是关于它的。

  更一般的情况是，要想检索一组记录，然后使用每一条记录作为一系列复杂的进一步操作的基础是困难的。例如，检索一个消费者列表然后查询每个消费者的详细信用历史，对每个客户来说可能要包括若干个查询。在某些情况下，可能想开发票，在发票头写上需要联系的客户信息，然后按次序列出每项条目。mysql 不适合这些类型的任务，因为可能需要依赖于前几个查询结果的若干查询，并且这些任务超出了mysql 的布局设计的能力。

  一般来说，除了 mysql 外，还需要工具来执行包括主细目关系和具有复杂输出格式需求的任务。程序提供将查询连接在一起的“ 胶 ”，并可用一个查询的输出作为另一个查询的输入。

- **可以将 MySQL 集成到任何应用程序中**。许多程序都利用数据库的能力提供信息。通

过发布一个快速查询，应用程序可以校验消费者号或检查一项条目是否在产品清单中。假设一个客户要寻找某些作者的所有书，则 Web 应用程序可以在数据库中查找它们，然后将结果显示在该客户的浏览器上。

通过使用调用带有包含 SQL 语句的输入文件的 mysql 的外壳脚本（shell script），可以实现一种初步的“集成”，然后，再使用其他UNIX实用程序加工这些输出。但是这可能变得很难看，特别是当任务变得更复杂时。当应用程序不断增长成为杂乱的修补工作时，它也可能产生一种“在工作，但觉得有错误”的感觉。此外，运行其他命令的外壳脚本的创建过程的开销可能超过您的预想。但它可能更有效率地与 MySQL 服务器直接交互，当在应用程序执行的每个阶段需要它的时候，都可以精确地提取想要的信息。

针对我们在第1章“MySQL 和 SQL 介绍”中安装的样例数据库 samp_db，我们已经列举了若干需要自己编写与MySQL服务器交互的程序的目标。这些目标中的一些显示在下面的列表中：

- 为打印而格式化 Historical League 目录。
- 考虑外观和联机目录的寻找。
- 通过电子邮件向成员发送补充通知。
- 使用 Web 浏览器很容易地将分数输入到学分册中。

在一些细节方面，我们将考虑的一个方面是将 MySQL 的能力与 Web 环境结合起来。MySQL 不直接提供对 Web 应用程序的支持，但通过组合带有适当的工具的 MySQL，通过Web可以很容易地访问数据库。使用 Web 服务器可以指定查询，向客户的浏览器报告结果。

将 MySQL 和 Web 结合可能有两个想法：

- 主要的兴趣在于数据库，只是想使用 Web 作为工具来获取对数据更容易的访问。在这样的想法下，数据库的位置是清楚且明显的，因为它是兴趣的焦点。例如，可以编写 Web 页来允许查看数据库所包含的表、表的结构，及表的内容。您打算使用 Web 服务器来提高对 MySQL 的访问能力。这可能也是 MySQL 管理者的观点。
- 主要的兴趣可能是 Web 站点，为了使站点的内容对访问者更有价值，您可能想使用 MySQL 作为一个工具。例如，如果为站点的访问者运行信息板或讨论清单，则可以使用 MySQL 保留信息的轨迹。在这种情况下，数据库的角色更微妙，访问者甚至可以不关心您必须提供给他在服务器中执行的部分。您打算使用 MySQL 提高 Web 服务器的能力。这可能也是 Web 站点开发者的一个观点。

这些想法并不矛盾。例如，在 Historical League 情况下，我们想通过允许联机输入来作为成员获取访问成员目录内容的一种方法而使用 Web。提供对数据库的访问是 Web 的一个用法。同时，League 的 Web 站点在某些方面有些不完全，所以向站点增加目录内容，以便为成员提高站点的价值。增强站点所提供的服务是数据库的一种用法。

无论您如何看待 MySQL 与 Web 的结合，实现方法都是类似的，即将前台的 Web 站点与后台的 MySQL 连接，使用 Web 服务器作为媒介。Web 服务器将查询从用户发送到 MySQL 服务器，检索查询结果，然后将它们传送给客户，在浏览器上显示。

当然，不一定要联机处理数据，但这样做往往有好处，特别是与经过标准的 MySQL 客户机程序访问数据做比较时：

- 通过 Web 访问数据，人们可以使用他们喜欢的浏览器，在他们喜欢的平台上运行。他们不限制 MySQL 客户机程序所运行的系统。Web 浏览器更是这样，无论 MySQL 客

户机分布如何广泛。

- Web 界面的使用比独立命令行的 MySQL 客户机程序的使用更简单。
- Web 界面可以根据特殊应用程序的要求来定制。而 MySQL客户机程序是用固定接口来完成基本功能的工具。
- 动态 Web 页面扩充了 MySQL 的能力，它可以做到用 MySQL 客户机很难做到或根本不可能做到的事情。例如，仅用 MySQL 客户机程序不可能真正地使一体化购买车辆的应用程序组合成整体。

任何编程语言都可用来编写基于 Web 的应用程序，但是，有些语言比其他语言更适合一些。请参阅5.2节“选择 API”，我们将看到这点。

5.1 MySQL 可用的 API

为了方便应用程序的开发，MySQL 提供了用 C 编程语言编写的客户机库，它允许从任何 C 程序的内部访问 MySQL 数据库。客户机库实现应用程序编程接口（API），API 定义客户机程序如何建立和执行与服务器的通信。

然而，使用 C 来编写 MySQL 程序并不受限制。许多其他语言处理器本身也是由 C 编写的，或具有使用C库的能力，所以 MySQL 客户机库提供了这个方法，由此，MySQL 对这些语言的约束可以建立在 C API 的上面。这就为与 MySQL 服务器通信而编写应用程序提供了许多选择。客户机程序的 API 是用 Perl、PHP、Java、Python、C++、Tcl 和其他一些语言编写的。有关最新的清单，请查看 MySQL 参考指南或 MySQL Web 站点，因为有时会增加用新语言编写的 API。

每种语言约束都定义自己的接口，特别是访问 MySQL 的规则。这里没有足够的时间来讨论 MySQL 可使用的每种 API，我们只讲述最流行的三种：

- **C 客户机库** API。这是 MySQL 的基本编程接口。
- Perl **通用目标脚本语言的 DBI（数据库接口）** API。DBI 是作为与其他模块在 DBD（数据库驱动程序）级接口的 Perl 模块来实现的，每个模块都提供对特定类型的数据库引擎的访问（当然，我们将讨论的特定的 DBD 模块也提供对 MySQL 的支持）。DBI 对 MySQL 的最普遍用法是编写由命令行来调用的独立的客户机，以及试图由 Web 服务器调用的脚本来提供 Web 对 MySQL 的访问。
- PHP API。PHP 是一种脚本语言，它提供了在 Web 页中嵌入程序的一种便利的方法。在发送以前，这样的页面由 PHP来处理，它允许这些脚本生成动态的内容，如在页面中包括 MySQL 查询的结果。“PHP”原始的意思是个人主页（Personal Home Page），但是 PHP 的成长已经远远超过它简单的原始功能。PHP Web 站点现在使用的这个名称表示“PHP：超文本预处理程序（Hypertext Preprocessor）”，它像GNU(是GUN而不是UNIX)一样以同样的方式自我引用。

使用他人成果

当标准的MySQL客户机不能满足需要时，您并不总是需要编写自己的程序。其他一些人一直编写程序，而这些程序中有许多是可共享得到的。请参阅附录 I 中的一些样例。只要找到几个就能节省您的许多工作。

以上这三种API都有专门章节详细说明。本章只提供对 API 比较的概述，用来说明它们的基本特征，并给出对特定的应用程序可能选择某个而不是其他 API 的原因。

当然，不必只考虑某个API，应了解每个 API，并用可以明智选择适合自己的 API。在包括若干组件的大项目中，可能使用多个 API，多种语言，这取决于每个子任务适合哪一种语言。

对于试图使用的任何一种 API，如果需要得到必需的软件，请参阅附录 A。

5.1.1 C API

C API 用于编译 C 程序上下文环境内部。它是一种客户机库，提供可用来与 MySQL服务器对话的最低级别的接口——具有创建与服务器通信所需的能力。

> **DBI 和 PHP 的前身**
>
> DBI 的 Perl 前身是 Mysqlperl 模块 Mysql.pm。这个模块不再被支持，而且不应该用于新的 MySQL 的开发。有一件事需要明白，Mysqlperl 是依赖于 MySQL 的，但 DBI 不是。如果编写 MySQL 的 Perl 应用程序，然后，决定想用另外一种数据库引擎来使用它们，则移植 DBI 脚本比 Mysqlperl 脚本更容易一些，因为它们很少依赖于特定的数据库引擎。
>
> 如果获取了访问 MySQL 的一段 Perl 脚本，并发现它是用Mysqlperl 而不是 DBI 编写的，则仍然可以使用 DBI。DBI 包括了对 Mysqlperl 的仿真支持，因此不需要安装两个程序包。
>
> PHP 3 的前身是 PHP/FI 2.0（FI代表“ form interpreter，即格式解释程序 ”）。像 Mysqlperl 一样，PHP/FI也是过时的，所以我们就不再进一步讨论它了。

> **MySQL C API 的起源**
>
> 如果已经有编写 mSQL RDBMS 程序的经验,那么将注意到 MySQL C API 类似于 mSQL 相应的 C API。当 MySQL 的开发者们开始实现他们的 SQL 引擎时，许多有用的共享实用程序可用于 mSQL。要想花费最小的难度将那些 mSQL 实用程序移植为 MySQL 的实用程序是可能的，可有意地将 MySQL API设计为与 mSQL API 类似（MySQL 甚至带有与 mSQL API 函数名称相应的 MySQL 名称的简单的文本替代品的 msql2mysql 脚本。这个操作相对烦琐，实际上也照顾了许多涉及为使用 MySQL 而转换 mSQL 程序的工作）。

MySQL 分发包提供的 C 客户机是基于这个 API 的。C 客户机库也作为 MySQL 对其他语言约束的基础来提供服务，但 Java API 是一个例外。例如，通过连接 MySQL C 客户机库代码（这个过程在附录A中通过 DBI 和 PHP 安装指导来举例说明），MySQL可用Perl DBI 模块专有的 MySQL 驱动程序和 PHP 代码。

5.1.2 Perl DBI API

DBI API 用于 Perl 脚本语言编写的应用程序的上下文环境内部。这种 API 在我们考虑的这三种 API 结构中是最高的，因为它可与许多数据库工作，而同时在脚本中可忽略许多特定数据库的细节。

DBI 经过使用两级结构的 Perl 模块来实现（请参阅图5-1）：

■ DBI(**数据库接口）级**。为客户机脚本提供接口。这个级别提供的是抽象接口，并不是

指特定数据库引擎。

- DBD(**数据库驱动器）级。**在这个级别由特定引擎的驱动程序来提供对各种数据库引擎的支持。

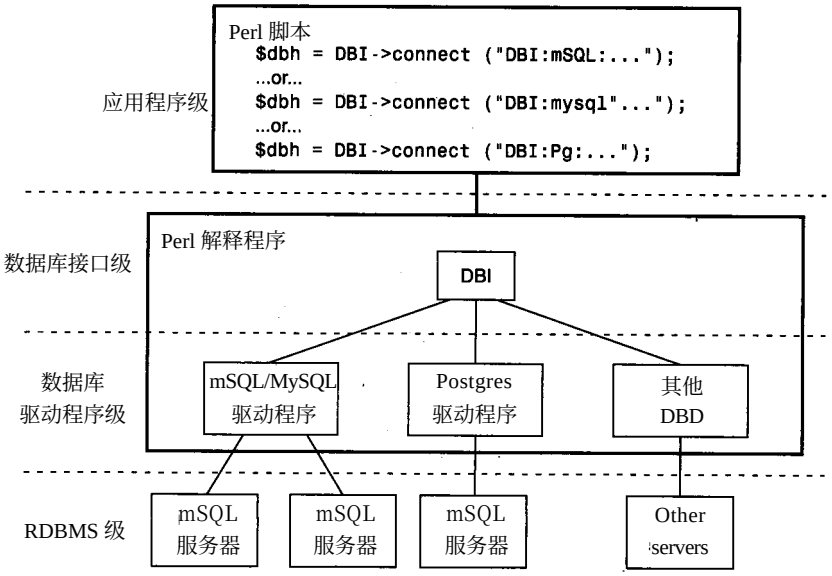

图5-1 DBI 体系结构

MySQL 对 DBI 的支持环境由 Msql-Mysql-modules分发包提供。这个模块在 DBD 级操作。可以从分发包名称及图 5-1中分辨它，一个驱动程序可以提供对一个以上的 RDBMS 的支持。Msql-Mysql-Modules 最初是为 mSQL 而编写的，后来扩展到 MySQL。这种影响类似于对 mSQL 和 MySQL 的 C API。由于设计的 MySQL C API 类似于 mSQL C API，所以将 mSQL DBD（使用 mSQL C API）扩展到对 MySQL 的使用很有意义。

DBI 体系结构编写应用程序的风格相对普通。当编写 DBI 脚本时，可使用一组标准的调用。DBI 级在 DBD 级调用适当的驱动程序来处理请求，对于想使用的特定数据库服务器通信中包括的特定问题，由驱动程序处理。DBD 级传送从服务器返回的数据，备份到 DBI 级，使数据出现在应用程序中。数据的格式与数据库的数据来源一致。

其结果得到这样一个接口，该接口从应用程序的编写者的观点隐藏了数据库引擎之间的差异，这样可使用多种不同的引擎——和驱动程序一样多。 DBI 通过允许以统一风格访问每个数据库来提供一致性客户接口以增加可移植性。

当打开数据库时，出现由脚本编写的数据库专有的界面。当创建连接时，应指出使用哪个驱动程序。例如，要想使用 MySQL 数据库，应这样连接：

```
$dbh = DBI->connect ("DBI:mysql:...");
```

而要想使用 Postgres 或 mSQL，应这样连接：

```
$dbh = DBI->connect ("DBI:Pg:...");
$dbh = DBI->connect ("DBI:mSQL:...");
```

连接以后，对该驱动程序不需要再做任何做特殊的引用。让 DBI 和该驱动程序解决数据库专有的细节。

无论如何这都是理论问题。然而，至少有两个因素与 DBI 脚本的可移植性矛盾：

- 在 RDBMS 引擎之间 SQL 的实现不同，为一个引擎编写的 SQL 另一个引擎根本不理解是完全可能的。如果 SQL 相当通用，则脚本可在引擎之间作相应的移植。但如果 SQL 依赖于引擎，则脚本也是这样。例如，如果使用 MySQL 指定的 SHOW TABLES 语句，则该脚本不能用其他数据库执行。
- DBD 模块通常提供引擎专有类型的信息来允许脚本编写者使用特定数据库系统的特定功能。例如，MySQL DBD 提供访问查询结果中列属性的方法，如每列的最大长度、列是否是数值型的，等等。而这些属性在其他数据库中没有任何相似物。DBD 专有的特性与可移植性相背，通过使用它们，将 MySQL 编写的脚本用于其他数据库系统是困难的（然而，在第 7 章中，您将发现，笔者毫不费力地避免了由 MySQL DBD 提供的 MySQL 专有的结构。那是因为您应该知道那些结构是什么，以便可以决定自己是否使用它们）。

尽管存在数据库专有脚本的这两个因素，但以抽象方式提供数据库访问的 DBI 机制是完成可移植性的合理方式，只要您决定利用它多少次即可。

5.1.3 PHP API

像 Perl 一样，PHP 也是一种脚本语言。但它与 Perl 不同，PHP 很少作为通用目标语言来设计，而是作为编写 Web 应用程序的一种语言。PHP API 主要作为在 Web 页面中嵌入可执行脚本的一种方法来使用。这使 Web 的开发者们很容易用动态生成上下文环境来编写页面。当客户浏览器向 Web 服务器发送 PHP 页面的请求时，PHP 执行在该页面中它所发现的任何脚本，并用脚本的输出来替换它。该结果再送回浏览器。这就使浏览器中实际出现的页面根据请求的页面环境的不同而有所不同。例如，当在 Web 页面中嵌入下面简短的 PHP 脚本时，它出现所请求页面的主机 IP 地址：

```
<?php echo $REMOTE_ADDR; ?>
```

可以使用脚本为访问者提供基于数据库上下文环境的最新信息。下面的样例说明可用于 Historical League Web 站点的一个简单脚本。该脚本发布一个请求来确定当前的 League 的成员数目，并将该数目报告给访问该站点的人（如果出现错误，则该脚本不报告任何数目）：

```
<HTML>
<HEAD>
<TITLE>US Historical League</TITLE>
</HEAD>
<BODY>
<P>Welcome to the US Historical League Website.
<?php
$link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
    or exit ();
mysql_select_db ("samp_db")
    or exit ();
$result = mysql_query ("SELECT COUNT(*) FROM member")
    or exit ();
if ($row = mysql_fetch_array ($result))
    echo "<P>The League currently has " . $row[0] . " members";
mysql_free_result ($result);
?>
</BODY></HTML>
```

DBI和DBD的含义

尽管 DBI 级是独立于数据库的，而 DBD 级是依赖于数据库的，但那并不是“ DBI ”和“ DBD ”所代表的意义。它们的意思是“ 数据库接口 ”和“ 数据库驱动程序 ”。

PHP 脚本通常看起来像是带有嵌入在“ <？php ”和“ ？> ”标识符中的脚本的 HTML 页面。一个页面可能包括若干个脚本。这为脚本的开发提供了一种非常灵活的方法。例如，如果您喜欢，可以编写一个正常的 HTML 页面来创建通用的页面框架，然后再增加脚本的内容。

对于不同的数据库引擎， PHP 对统一的接口不再作任何事情， DBI 也用这种方法。取而代之，每个引擎的接口看起来非常像相应的实现该引擎低级 API 的 C 库接口。例如，用于从 PHP 脚本内部访问 MySQL 的 PHP 函数的名称非常类似于 MySQL C 客户库中函数的名称。

5.2 选择API

本节介绍根据各种类型的应用程序选择 API的方法，比较C、DBI 和 PHP API 的能力，并给出它们相对的优点和缺点，并指出什么时候应选择哪一个。

首先应该指出，笔者不认为任一种语言优于其他语言。尽管笔者的确有自己的喜好，但还是统统使用它们。您也会有自己的喜好，像我的评论家一样。一个评论家会感觉应该强调 C 对 MySQL 编程的重要性，应将这种重要性上升到更重要的程度，而另一个评论家会认为 C 编程相当困难，应放弃使用它！您应当权衡本节中讨论的这些因素，得出自己的结论。

在对特定任务选择哪个 API 时，要考虑以下问题：

- **预期的执行环境**。期望使用应用程序的上下文环境。
- **性能。**当在 API 语言中编写时，如何使应用程序高效地执行。
- **开发的容易性。**如何便于 API 和它的语言编写应用程序。
- **可移植性。**除 MySQL 以外，应用程序是否还将用于其他数据库系统。

下面进一步分析每个问题。要注意这些因素的相互影响。例如，您想要一个运行良好的应用程序，但使用一个可快速开发该应用程序的语言也同等重要，即使该应用程序不能非常有效地运行也同样。

5.2.1 执行环境

当编写应用程序时，通常应考虑使用哪种环境。例如，该应用程序可能是从外壳程序中调用的报告生成器程序，或一个应付账目概要程序，在每月的月底作为 cron job 进行运行。从外壳程序或 cron 程序中运行的命令通常依赖它们自己，而且很少需要运行环境。另外，可以编写一个应用程序来试图由 Web 服务器调用。这样的程序期望能从它的运行环境中抽出非常特殊类型的信息：客户正在使用什么浏览器？在邮件清单订阅请求格式中输入什么参数？客户提供正确的口令访问我们个人信息了吗？

每种 API 语言都以它在这些不同的环境中适于编写应用程序而变化：

- C 是通用目标的语言，从理论上讲任何任务都可使用它。在实际中， C 倾向于用于更频繁的独立程序而不是对 Web 的编程。其原因可能是在 C 中不像在 Perl 或在 PHP 中那样容易地实现文本处理和内存管理，并且这些处理和管理在 Web 应用程序中大量地使用。

- Perl，像 C 一样，适合于编写独立的程序。然而，对于 Web 站点的开发，Perl 也是非常有用的，例如通过使用 CGI.pm 模块。这使 Perl 成为编写连接 MySQL 和 Web 的应用程序的便利的语言。这样的应用程序可以经 CGI.pm 模块与 Web 接口，并可以使用 DBI 与 MySQL 相互作用。
- PHP 是设计用来编写 Web 应用程序的语言，所以这个环境显然是最适合的。而且，数据库访问是 PHP 最大的优势之一，所以它是实现与 MySQL 相关的任务的 Web 应用程序最自然的选择。也可以将 PHP 作为一个独立的解释程序（例如，从外壳程序中运行脚本），但不能非常频繁地使用它。

根据以上这些需要考虑的问题，对于独立的应用程序，C 和 Perl 是最佳语言。对于 Web 应用程序，Perl 和 PHP 是最合适的。如果需要编写这两种类型的应用程序，但又不会使用这些语言的任何一种，并想用尽可能少的精力来学习，则 Perl 可能是您最佳的选择。

5.2.2 性能

我们通常喜欢应用程序尽可能快地运行。然而，实际上性能的重要性取决于所使用的程序的频率。对于一个月运行一次晚上定时工作的程序，性能可能不是非常重要的。而对于在 Web 站点上一秒钟运行若干次的程序，则每当排除一点无效性都会带来巨大的不同。后一种情况下，在站点的有效性和请求中，性能发挥着重要的作用。一个缓慢的站点是令用户苦恼的，无论站点的内容如何，如果您依靠站点作为一项收入来源，则性能的降低直接影响收入。如果不能一次为多个连接提供服务，访问者只会产生厌烦情绪而去其他的站点。

性能评价是一个复杂的问题。当编写特定的 API 时，应用程序完成得好坏的最好指标是在这个 API 环境下编写并进行测试。而且最好的比较测试是在不同的 API 环境下多次运行该应用程序，来比较每个版本。当然，那不是一般的工作。一般来说，您只想获取编写的应用程序。一旦它工作了，如果它需要运行得更快，您就可以考虑优化它，使用更少的内存，或有某些需要用其他方法提高的方面。但是，至少有如下两个因素会影响性能：

- 编译的程序比解释的程序运行得更快。
- 对于在 Web 上下文环境中使用的解释语言，在解释程序作为 Web服务器自身的一部分而不是单独的过程模块被调用时，性能更好。

1. 相对于解释语言的编译语言

编译的应用程序比用脚本语言编写的程序的同样版本效率更高、使用的内存更少，并且执行得更快，这是基本规律。这是由于执行脚本的语言的解释程序的开销问题。因为 C 是编译的，而 Perl 和 PHP 是解释的，所以 C 程序通常比 Perl 或 PHP 脚本运行得更快一些。

对于大量使用的程序，通常用 C 是最好的选择。在 MySQL 分发包中包括的 mysql 命令行客户机程序就是最好的样例。

当然，有一些因素能使这种明显的差别减小。对于一项任务，可用 C 编写出更快的程序，但也很有可能编写出低效率的 C 程序。用编译语言编写的程序并不自动地保证更好的性能。所以需要不断地考虑所做的事情。此外，如果一个脚本化的应用程序需花费大部分时间来执行连接到解释程序引擎的 MySQL 客户机库例程的代码，则编译程序和解释程序之间的差别将有所减少。

2. 相对于语言解释程序模块版本的独立程序

对于基于 Web 的应用程序，脚本语言解释程序通常以两种形式之一来使用，至少对 Apache 是这样，当编写 Web 应用程序时，Apache 是我们将使用的 Web 服务器：

- 可以安排 Apache 去调用这个解释程序作为单独的过程。当 Apache 需要运行 Perl 或 PHP 脚本时，它启动相应的程序，并告知它来执行该脚本。在这种情况下，Apache 使用该解释程序作为 CGI 程序，也就是说，它使用公共网关接口（Common Gateway Interface，CGI）协议与它们通信。
- 解释程序可用作直接连接到 Apache 二进制程序和作为其过程自身的一部分运行的模块。在 Apache 条件下，Perl 和 PHP 解释程序获得 mod_perl 和 mod_php3 模块的形式。

Perl 和 PHP 的提倡者们极力宣扬解释程序有速度优势，但所有的人都同意之所以喜欢解释程序是因为其运行的形式比语言本身有更大的诱惑力。在这两者中，解释程序作为模块运行比作为独立的 CGI 应用程序运行更快。

对于独立的应用程序，每当运行一个脚本时都必须启动该解释程序，所以将导致重大的创建过程的开销。当在已经运行 Apache 过程的内部作为模块使用时，解释程序可以立即从 Web 页面中访问。通过减少开销显著地提高了性能，并直接转换为快速处理获取的请求并发送它们的能力的增加。

独立解释程序启动的性能比模块解释程序的性能至少差一个数量级。当考虑 Web 页面服务包括少量处理的快速事务处理而不是具有许多处理时，解释程序启动的开销特别重要。如果花费许多时间只是为了启动而不是用于实际执行该脚本，则大部分资源一直处于等待状态。一天中的大部分时间可能花费在准备工作上，4 点到达，然后 5 点回家。

您可能想知道，为什么解释程序的模块版本由于必须一直启动 Apache 而能节省时间呢？。这个原因是，当 Apache 启动时，它立即产生一些子过程，用于处理收到的请求。当包括脚本执行的请求到达时，已经有 Apache 进程在准备等待去处理它。同样，Apache 的每个实例可服务于多个请求，所以该进程启动的开销只导致每组请求一次，而不是每个请求一次。

当 Perl 和 PHP 以模块形式安装时（像 mod_perl 和 mod_php3），哪一个完成得更好一些？那就是争论的主题，以下是适用于一般性的指南：

- Perl 将脚本转换为内部可编译的形式；而 PHP 却不这样。因此，一旦该脚本通过语法分析，则 Perl 可更快地执行它，特别是对于具有大量迭代的循环。
- mod_perl 可以运行脚本高速缓存来提高重复执行的脚本的性能。如果脚本在高速缓存中，则Perl 可更快地开始执行脚本，因为它不需要再一次分析。否则，PHP 开始执行的该脚本的速度更快。
- mod_perl 比 PHP 有更大的内存覆盖区；利用 mod_perl 连接的 Apache 进程比利用 mod_php3 的更大。PHP 被假定必须在另一个进程的内部协同存在，并且在那个进程内部可以多次激活或撤消。Perl 被设计成从命令行作为独立程序运行，而不是作为被嵌入在 Web 服务器进程中的一个语言进行运行。这可能使它付出较大的内存覆盖区；Perl是模块，因此它不运行在自身环境中。脚本的高速缓存和该脚本使用的附加 Perl 模块是付出较大的内存覆盖区的另外的因素。在这两种情况下，有更多的代码使用内存，并将运行的 Apache 进程保留在内存中。

在脚本运行速度方面，Perl 无论有什么可超过 PHP 的优势，都被 PHP 4 除掉了。PHP 4 在

它的能力和接口方面类似于 PHP 3，但它合并了 Zend，Zend 是一种更高性能的解释程序的引擎。

无论如何，所有的这些因素只导致 Perl 和 PHP 的模块版本之间性能比不同。无论您选择哪种语言，最重要的是尽可能地避免独立的解释程序。

解释程序的独立版本的确有一个优点超过它的模块版本，即可以安排它在不同的用户 ID 下运行。模块版本始终运行在与 Web 服务器相同的用户 ID 下，出于安全原因，该用户是一个典型的具有很少权限的账号。对于需要特殊权限的脚本来说不能很好地运行（例如，如果您需要能读写受保护的文件）。如果愿意，可以将模块方法和独立方法结合起来：缺省情况下使用模块版本，而在具有特定用户的权限的脚本的情况下使用独立版本。

降低 mod_perl 的内存需求

有些技术允许您只能对 mod_perl 使用某些的 Apache 进程。这样，只对运行 Perl 脚本的那些进程产生额外的内存开销。Apache Web 站点的 mod_perl 部分有可选择的各种策略的讨论（有关的详细信息，请参阅 http://perl.apache.org/guide/）。

综上考虑，也就是说，选择 Perl 还是 PHP，您应该试着从 Apache 模块中而不是通过调用一个单独的解释程序过程来使用它。只对不能由模块处理的那些情况使用独立的解释程序，例如需要特殊权限的脚本。对于这些实例，可以通过使用 Apache 的 suEXEC 机制在给定的用户 ID 下启动解释程序来处理脚本。

5.2.3 开发时间

刚才描述的这些因素影响应用程序的性能，但不能只考虑运行效率。时间及编程的简易性也是重要的，所以为 MySQL 编程选择 API 时要考虑的另一个因素是如何很快地开发出自己的应用程序。如果开发同样程序，用 Perl脚本只需用 C 语言一半的时间，那您可能宁愿使用 Perl DBI API，而非 C API，即使开发出的应用程序运行速度不是非常快也如此。考虑程序的运行时间比考虑编写程序时花的时间更少一些通常是合理的，特别是对不经常运行的应用程序更是如此。您的一小时比机器的一小时要值钱得多！

一般来说，脚本语言编写程序更快，特别是得到应用程序的原型更快，这是由于以下两个原因：

第一，脚本语言提供更高级别的结构。这允许您在抽象的更高级别上进行思考，以便集中考虑要做些什么，而不是考虑如何做。例如，Perl 的关联数组（散列）对于维护具有键 /值关系（如学生 ID/学生姓名对）的数据节省了大量时间。C 没有这样的构造。如果想在 C 中实现这样的事情，则可能需要编写代码来处理许多低级的细节，其中包括内存管理和串操纵的问题，并且需要调试它，这也要花时间。

第二，脚本语言的开发周期的步骤较少。用 C 开发应用程序时，要经历通常的编辑—编译—测试的循环周期。每次修改程序时，在测试之前都必须重新编译它。而用 Perl 和 PHP，由于每次修改后不用编译就可以立即运行脚本，因此，开发周期可简化为编辑—测试。另一方面，编译程序对程序在严格的类型检查形式方面有更多的约束条件。编译程序施加的更多约束条件有助于避免在松散语言 (如 Perl 和 PHP)中不易捕获的错误。在 C 中，如果您错误地拼写了变量的名称，则编译程序将警告您。PHP 不这样，Perl 也不这样，除非向它询问。当应用程序变得更大且更难于维护时，这些更严格的约束条件可能特别有用。

一般来说，在编译和解释语言之间权衡的是开发时间与性能的折衷：是想要使用编译语言开发程序，以便在运行时可以更快，但花费更多的时间来编写它？还是想要用脚本语言编写程序，以便在缩短编程时间，但要损失一些运行速度？

将两种方法合并起来也是可能的。编写一个脚本作为“第一个草案”来快速地开发出一个应用程序原型以测试其逻辑性，确定算法的可用性。如果这个程序有用，并且被频繁使用，则性能成为关心的焦点，这时可作为编译的应用程序重新对它编写代码。这样做给您带来两种方法的优点：快速得到应用程序的初始开发原型，同时得到最终产品的最佳性能。

从某种严格的意义来说，Perl DBI 和 PHP API 并没有给您在 C 客户机库中没有的能力。这是因为这两种 API 通过 MySQL C 库连接到 Perl 和 PHP 解释程序来获取对 MySQL 的访问。然而，对于嵌入 MySQL 的环境，C 与 Perl 或 PHP 有很大的不同。应考虑在与 MySQL 服务器相互作用时要做的事情并提问每个 API 语言如何帮助您完成这些事情。下面有一些样例：

- **内存管理。**在C中，您发现自己对任何任务，包括动态分配数据结构都使用 malloc() 和 free() 来工作。Perl 和 PHP 可为您处理这些任务。例如，数组的大小自动地增加，并且可以使用动态长度的字符串而不用考虑内存管理。
- **文本处理。**在这点 Perl 具有最大的开发能力，而 PHP 位于第二。比较起来，C 是非常初级的。

当然，可以用C编写自己的库，将内存管理和文本处理这样一些任务封装成更容易工作的函数。但是，然后还必须调试它们，并且您也想使自己的算法效率更高。在这两个方面，基本上可以断定，由于它们已经具有了通过许多双眼睛检查过的好处，对这些事情 Perl 和 PHP 中的算法一般是易于调试并且有合理的效率。通过利用其他人的工作可以节省您的时间（另一方面，如果解释程序偶尔有一个错误，您可能必须携带它，直到这个问题纠正为止。而用 C 编写时，可以更细地控制程序性能）。

这些语言的不同还在于它们的“安全性”。C API 提供对服务器最低级别的接口，而且强制的原则最少。从这种意义上说，它提供最低级的安全性网络。如果您超出正常顺序执行 API 函数，则可能获得一个“超出同步”的错误，或使程序崩溃。Perl 和 PHP 都提供了很好的保护。如果您没有以适当的顺序进行，则脚本失败，但是解释程序并不崩溃。在 C 程序中，出现崩溃错误的另一个非常可能的来源是动态分配内存和与它们相关的指针的使用。Perl 和 PHP 为您处理内存管理，所以您的脚本很少可能因为内存管理的错误而瘫痪。

开发时间受语言可用的外部支持的影响。C 可用的外部支持是将 MySQL C API 函数封装为更容易使用的实例的包装库的形式。这些库对于 C 和 C++ 都可用。Perl 无疑具有最大数量的附加软件，都是 Perl 模块的形式（与在 Apache 模块中具有的概念类似）。甚至有一个基础结构被设计来容易地定位并获取这些模块（即综合 Perl 归档网络 Comprehensive Perl Archive Network, CPAN）。使用 Perl 模块，不用编写代码就可以获取对所有类型的函数的访问。想要编写从数据库中生成报告的脚本，然后作为一个附件发送给某人吗？只要获取 MIME 模块中的一个，就立即具有附件生成的能力。

PHP 没有同样程度的外部支持（这并不令人惊奇，因为它是较新的语言）。也许所知道的最佳的附加软件是PHP基本库（PHP Base Library，PHPLIB）。根据名称和口令机制的一些排序，假设您正在编写需要限定只有经授权的用户才可以对某个 Web 页面访问的 Web 应用程

序。可以用任意语言编写对它的支持程序，但是如果使用 PHPLIB，则不必花费时间重新做这件事情。PHPLIB提供确认并且允许通过会话跟踪经授权的用户（从作为单个逻辑访问部分的给定客户机中连续页面的命中）。还可以分配给用户许可权，这允许您进行像定义具有更多权限的管理用户的工作。

5.2.4 可移植性

可移植性的问题与为访问 MySQL 引擎所编写的程序怎样才能容易地修改为使用不同引擎的程序有关。您可能不担心这个事情。然而，除非可以预知未来，否则，说“除了 MySQL 以外，我永远都不会将这个程序用在任何其他的数据库上”可能有些冒险：假设您找到另一份工作，并想使用自己的旧程序，但您的新老板使用不同的数据库系统呢？

如果可移植性是需要优先考虑的事情，则应该考虑在 API 之间的区别：

- DBI API的可移植性最好，因为它独立于数据库是 DBI 设计的一个明确目标。
- PHP 的可移植性稍差，因为它不提供对 DBI 提供的各种数据库引擎的同样类型的统一接口。对每个支持数据库的 PHP 函数的调用类似于在相应的基础 C API 中的那些。稍有一些不同，但很少，需要更改您调用的数据库相关函数的名称。还有可能要修改一点应用程序的逻辑，因为不同数据库的借口并不都是以同样的方式工作的。
- C API 提供的数据库之间的可移植性最差。因为它生来就是为 MySQL 设计的。

当需要在同一个应用程序中访问多个数据库系统时，独立于数据库的可移植性特别重要。这可能包括像将数据从一个 RDBMS 移动到另一个 RDBMS 中的简单任务，或更复杂的任务，如基于从许多数据库系统中得到的信息生成报告。

DBI 和 PHP 都提供对访问多个数据库引擎的支持，所以对不同的数据库，甚至在不同的主机上，都可以很容易地同时连接到服务器上。然而， DBI 和 PHP 在对于从多个异构数据库系统中检索和处理数据的任务的适宜性方面有所不同。而 DBI 更好些，因为无论使用哪种数据库，它都使用一组单独的访问调用。假设想在 MySQL、mSQL 和 Postgres 数据库之间传送数据。使用 DBI，则使用这三种数据库的惟一不同之处在于用于连接到每个服务器的 DBI->connect()调用。而用 PHP时，可能需要有更复杂的脚本，该脚本将含有三组读取调用和三组写入调用。

多数据库应用程序的一个极好的例子是 MySQL 分发包中的 crash-me 脚本，它可测试许多不同的数据库服务器的能力。该脚本是用 DBI编写的，对于这样的应用程序，这种选择是很明显的，因为您可以同样的方式访问所有的数据库。