第 1 章　不同数据库的差异

目前市场上流行的数据库有很多种，虽然这些数据库在开发时遵循同样的一个标准，但是，每种数据库都有一些与其他同类产品不一样的地方。本章将主要介绍不同种数据库间的差异，并对 PHP 与不同数据库的操作上的差异作一下介绍。

1.1　ANSI SQL 以及常见关系式数据库的 SQL 扩展

在前面的章节中介绍了如何使用 SQL 语言来操作数据库。SQL 语言是目前流行的一款专门用于操作关系型数据库的语言。现在流行的 SQL 语言标准为 ANSI 制定的 ANSI SQL-92。ANSI 是美国工商业集团组织，是 ISO 和 IEC 的成员之一。1992 年 ISO 和 IEC 共同发布了 SQL-92，随后，ANSI 也发布了 ANSI SQL-92，通常称为 ANSI SQL。尽管市面上流行的各种关系型数据库与 ANSI SQL 版本有一些差异，但是基本上都是遵循 ANSI SQL 的标准。

本节将介绍几种常见的关系型数据库中的 ANSI SQL 扩展。

1.1.1　ANSI SQL

ANSI SQL 主要包括两种数据操作语言，一个是用于数据库元素定义的数据定义语言（DDL），一个是用于数据库元素管理的数据管理语言（DML）。

1．数据定义语言（DDL）

DDL 是用于定义和管理数据库元素的语言，主要用于数据库中的数据表、索引、视图的创建、修改和删除等。以下代码在数据库中创建了一个表。

```
CREATE TABLE MyTable
(user_id smallint,
username char(22),
password char(22),
email char(30));
```

以下代码对上面创建的表进行了修改操作，增加了一个新列。

```
ALTER TABLE MyTable
ADD tel_no char(22);
```

以下代码将删除上面创建的表。

```
DROP TABLE MyTable;
```

可以看到，上面的三条 SQL 语句实现了对数据库中元素的创建、修改和删除。对于其他数据库元素均可以使用相似的方法来完成。

2．数据管理语言（DML）

DML 是用于数据库中数据管理的语言，主要用于数据库中数据的查询、插入、修改和删除等。例如，以下代码向数据库中的 MyTable 中插入了一条数据。

```
INSERT INTO MyTable
VALUES(100001, 'Simon', '123456', 'pch1982cn@yahoo.com.cn');
```

以下代码对这条数据进行了修改。

```
UPDATE MyTable
SET username = "Elaine"
WHERE user_id = 100001;
```

以下代码删除了这条数据。

```
DELETE FROM MyTable
WHERE user_id = 100001;
```

以下代码从数据库中查询这条数据。

```
SELECT * FROM MyTable
WHERE user_id = 100001;
```

上面的例子均为 ANSI SQL 的典型例子，在前面的章节中已经介绍过了。对于大多数关系型数据库产品来说，均可以使用 ANSI SQL 对数据库进行一般性操作。

1.1.2　MySQL 对 ANSI SQL 扩充

MySQL 包含了一些在 ANSI SQL 中找不到的功能。本小节将简要介绍其中的几项。

1．注释

由于使用 MySQL 专门的 SQL 语句将导致 SQL 语句不再与其他的数据库管理系统兼容。因此，MySQL 提供了一种注释形式，可以使用“/*! … */”形式的注释将 MySQL 专门的代码写在注释内。这样，在其他的数据库管理系统上运行 SQL 时将不会被运行，而在 MySQL 中，注释内的代码也将被 MySQL 执行。例如以下代码将 TEMPORARY 关键字放入了注释中。

```
CREATE /*! TEMPORARY */ TABLE MyTable (col1 int);
```

这时，在 MySQL 中实际上执行的 SQL 语句是没有注释的 SQL，如下所示。

```
CREATE TEMPORARY TABLE MyTable (col1 int);
```

而在其他的数据库管理系统中，实际上执行的 SQL 语句如下所示。

```
CREATE TABLE MyTable (col1 int);
```

对于 MySQL 的不同版本，也可以通过注释的方法来实现区分。其方法是在叹号“!”后面写上版本号，这样，注释内的代码将只在该版本号以上的 MySQL 版本下运行，如以下代码所示。

```
CREATE /*!40102 TEMPORARY */ TABLE MyTable (col1 int);
```

上面的 SQL 语句在 MySQL 版本大于等于 4.1.2 下运行时的 SQL 如下所示。

```
CREATE TEMPORARY TABLE MyTable (col1 int);
```

而在更早期版本中，实际上执行的 SQL 语句如下所示。

```
CREATE TABLE MyTable (col1 int);
```

2．SELECT 语句中的逻辑判断

MySQL 提供了在 SELECT 语句中进行逻辑判断的功能。其逻辑判断操作符主要包括=、<>、<=、<、>=、>、AND、OR 和 LIKE 等。当逻辑成立时，其返回值为 1，否则为 0。

以下代码首先查询出 mytable 表上的全部记录，然后判断工资是否大于 6,000。

```
mysql> select * from mytable;
+-----------+--------+-----+------------+---------+---------+
| serial_no | name   | age | birthday   | salary  | bonus   |
+-----------+--------+-----+------------+---------+---------+
|    100001 | Simon  |  24 | 1982-11-06 | 5000.00 | 1000.00 |
```

```
|      100002 | Elaine |   24 | 1982-01-01 | 5000.00 | 1000.00 |
|      100003 | Susan  |   31 | 1975-01-01 | 9000.00 | 2200.00 |
+-----------+--------+-----+------------+---------+---------+
3 rows in set (0.00 sec)

mysql> select serial_no, name, salary, salary>6000 from mytable;
+-----------+--------+---------+-------------+
| serial_no | name     | salary   | salary>6000 |
+-----------+--------+---------+-------------+
|      100001 | Simon   | 5000.00 |                  0 |
|      100002 | Elaine | 5000.00 |                 0 |
|      100003 | Susan   | 9000.00 |                  1 |
+-----------+--------+---------+-------------+
3 rows in set (0.00 sec)
```

3．Limit 语句

MySQL 提供了一个 Limit 子句用于选择查询结果中的记录，该子句一般放在 SELECT 或者 DELETE 语句的末尾，如以下代码所示。

```
mysql> select * from mytable
    -> limit 1,2;
+-----------+--------+-----+------------+---------+---------+
| serial_no | name     | age | birthday     | salary   | bonus     |
+-----------+--------+-----+------------+---------+---------+
|      100002 | Elaine |   24 | 1982-01-01 | 5000.00 | 1000.00 |
|      100003 | Susan  |   31 | 1975-01-01 | 9000.00 | 2200.00 |
+-----------+--------+-----+------------+---------+---------+
2 rows in set (0.00 sec)
```

该 SQL 语句仅查询出 mytable 表中的第 2 和第 3 条记录。需要注意的是使用 limit 子句时，第一条记录为的编号为 0。

1.1.3　SQL Server 的 T-SQL

T-SQL 是 SQL Server 提供的基于 SQL 语言的一个增强型 SQL 语言。T-SQL 提供了 ANSI SQL 的全部功能，并增加了扩展的函数、系统预存、程序设计结构等更多功能。下面将介绍几个常见的 T-SQL 增强特点。

1．T-SQL 提供了更丰富的数据类型

T-SQL 提供了多种额外的数据类型，可以在数据库的一个属性中最多保存 2GB 的数据。并且，还提供了 XML 数据类型用于专门存储 XML 数据。

2．错误处理机制

T-SQL 提供了类似 PHP 钟 try…catch 语句的方法来捕获 SQL 代码运行时的异常，其语法格式如下所示。

```
BEGIN TRY
--SQL 代码
END TRY
BEGIN CATCH TRAN_ABORT
--错误处理
```

```
END CATCH
```

1.1.4　Oracle 的 PL/SQL

PL/SQL 与 T-SQL 类似，是 Oracle 提供了 SQL 扩展。PL/SQL 使用块语法来完成一个功能。PL/SQL 的块语法格式如下所示。

```
[DECLARE]
---定义部分
BEGIN
---执行部分
[EXCEPTION]
---异常处理部分
END
```

PL/SQL 中的每条语句均以分号结束，可以在其中嵌入一条或者多条 SQL 语句。

Oracle 提供四种类型的 PL/SQL 程序。

- 函数：函数可以接受若干个参数，并将处理后的变量返回，其语法格式如下所示。

```
FUNCTION name [{parameter1[,parameter2,...])] RETURN data_type IS
---定义部分
BEGIN
---执行部分
[EXCEPTION
---异常处理部分
END [name]
```

- 过程：过程与函数类似，其区别在于没有返回值，其语法格式如下所示。

```
PROCEDURE name [{parameter1[,parameter2,...])] IS
---定义部分
BEGIN
---执行部分
[EXCEPTION
---异常处理部分
END [name]
```

- 包：包是一个各种相关对象的集合，可以包括多个函数和过程。
- 触发器：用于接收来自数据库中的操作，并触发一段 PL/SQL 块。

1.2　PHP 数据库应用的差异

前面几章介绍了 PHP 与 MySQL、PostgreSQL、SQL Server 和 Oracle 的操作。本节将对 PHP 与各种数据库的连接和操作进行比较。事实上，虽然 PHP 与各种数据库的连接方法不尽相同，但是其思想是一致的。掌握了 PHP 与其中一种数据库的操作方法后，对于其他数据库的操作都会很容易地掌握。

1.2.1　MySQL 与 PHP 的应用

对于 MySQL 数据库，PHP 提供了 php_mysql 扩展来实现与 MySQL 数据库的操作。要操作 MySQL，首先需要修改 PHP 安装目录下的 php.ini 文件，将 php_mysql 扩展前面的注释去掉，如下所示。

```
extension=php_mysql.dll
```

修改后重新启动 Apache 服务器即可。

以下代码在 MySQL 上执行了一个查询操作。

```
<?php
@mysql_connect("localhost", "root")                    //连接数据库服务器
or die("数据库服务器连接失败");
@mysql_select_db("mydb")                               //选择数据库 mydb
or die("数据库不存在或不可用");
$query = @mysql_query("select * from mytable")         //执行 SQL 语句
or die("SQL 语句执行失败");
echo "<table border=1>";
while($row = mysql_fetch_row($query))                  //通过循环的方式输出从第 0 行到最大的一行的所有记录
{
    $serial_no = $row[0];                              //输出第$i 行的 serial_no 列
    $name = $row[1];                                   //输出第$i 行的 name 列
    $salary = $row[4];                                 //输出第$i 行的 salary 列
    echo "<tr>";
    echo "<td>$serial_no</td>";
    echo "<td>$name</td>";
    echo "<td>$salary</td>";
    echo "</tr>";
}
echo "</table>";
?>
```

1.2.2　PostgreSQL 与 PHP 的应用

对于 PostgreSQL 数据库，PHP 提供了 php_pgsql 扩展来实现与 PostgreSQL 数据库的操作。要操作 PostgreSQL，首先需要修改 PHP 安装目录下的 php.ini 文件，将 php_pgsql 扩展前面的注释去掉，如下所示。

```
extension= php_pgsql.dll
```

修改后重新启动 Apache 服务器即可。

以下代码在 PostgreSQL 上执行了一个查询操作。

```
<?php
$pg = @pg_connect("host=localhost user=postgres password=postgres dbname=employees")
or die("Can't connect to database.");                                    //连接服务器
$query = "SELECT * FROM employees ORDER BY serial_no";
$result = @pg_query($pg, $query) or die("Can't run query to table.");    //执行 SQL
echo "<table border=1>";                                                 //输出表头
echo "<tr>";
echo "<td>serial_no</td>";
echo "<td>name</td>";
echo "<td>birthday</td>";
echo "</tr>";
for($i=0; $i<pg_num_rows($result); $i++)                                 //循环输出所有记录
{
    $row = @pg_fetch_row($result) or die("Can't fetch row from table.");
    $serial_no = $row[0];                                                //输出第$i 行的 serial_no 列
```

```
    $name = $row[1];                                    //输出第$i 行的 name 列
    $birthday = $row[4];                                //输出第$i 行的 salary 列
    echo "<tr>";
    echo "<td>$serial_no</td>";
    echo "<td>$name</td>";
    echo "<td>$birthday</td>";
    echo "</tr>";
}
echo "</table>";
pg_close($pg);                                          //关闭连接
?>
```

可以看到，上面的代码使用与操作 MySQL 数据库类似的方法实现了同样的功能，不同之处只在于数据库类型。

1.2.3　SQL Server 与 PHP 的应用

对于 SQL Server 数据库，PHP 提供了 php_mssql 扩展来实现与 SQL Server 数据库的操作。要操作 SQL Server，首先需要修改 PHP 安装目录下的 php.ini 文件，将 php_mssql 扩展前面的注释去掉，如下所示。

```
extension=php_mssql.dll
```

修改后重新启动 Apache 服务器即可。

以下代码在 SQL Server 上执行了一个查询操作。

```
<?php
$conn=mssql_connect("localhost","sa","");                   //连接 SQL Server
$dbname=mssql_select_db("MyDB")                             //选择要使用的数据库
or die("Cannot connect to SQL Server.");
$sql="select * from MyTable";                               //定义要执行的 SQL 语句
$result=mssql_query($sql);                                  //执行 SQL 语句
echo "<table border=1>";
//通过循环的方式输出从第 0 行到最大的一行的所有记录
while($row = mssql_fetch_row($result))
{
    $serial_no = $row[0];                                   //输出当前行的 serial_no 列
    $name = $row[1];                                        //输出当前行的 name 列
    $salary = $row[4];                                      //输出当前行的 salary 列
    echo "<tr>";
    echo "<td>$serial_no</td>";
    echo "<td>$name</td>";
    echo "<td>$salary</td>";
    echo "</tr>";
}
echo "</table>";
?>
```

1.2.4　Oracle 与 PHP 的应用

对于 Oracle 数据库，PHP 提供了 php_oci8 扩展来实现与 Oracle 8 数据库的操作。要操作 Oracle，首

先需要修改 PHP 安装目录下的 php.ini 文件，将 php_oci8 扩展前面的注释去掉，如下所示。

```
extension=php_oci8.dll
```

修改后重新启动 Apache 服务器即可。

以下代码在 Oracle 上执行了一个查询操作。

```
<?php
$conn = ocilogon("username","password","OracleDB");          //连接 Oracle 数据库
$result = OCIParse($conn,"select * from MyTable");          //执行 SQL 语句
OCIDefineByName($result,"SERIAL_NO",&$serial_no);           //定义执行结果中的每个属性
OCIDefineByName($result,"NAME",&$name);
OCIDefineByName($result,"SALARY",&$salary);
OCIExecute($result);                                        //执行 SQL 语句
echo "<table border=1>";
//通过循环的方式输出从第 0 行到最大的一行的所有记录
while(OCIFetch($result))
{
    echo "<tr>";
    echo "<td>$serial_no</td>";                             //输出当前行的 serial_no 列
    echo "<td>$name</td>";                                  //输出当前行的 name 列
    echo "<td>$salary</td>";                                //输出当前行的 salary 列
    echo "</tr>";
}
echo "</table>";
OCIFreeStatement($result);
OCILogoff($conn);
?>
```

使用 PHP 操作 Oracle 数据库的函数与操作 MySQL 或 SQL Server 的差别比较大，但是其基本思想是一样的。这里需要注意的是 OCIDefineByName 函数，该函数用于将结果集中的每条记录中的一列存放到一个变量中，在每次使用 OCIFetch 函数获取当前行时，数组中的属性将被直接放置到该变量中。

1.2.5　SQLite 与 PHP 的应用

对于 SQLite 数据库，PHP 默认提供了对于 SQLite 的支持。SQLite 是一个轻量级的数据库，该数据库直接将数据库放置在一个文件中。对于小型应用，这种设计对于系统的移植有很大好处。

以下代码在 SQLite 上执行了一个查询操作。

```
<?php
$db=sqlite_open ("mydb.db");                                //连接 SQLite 数据库文件
$result = @sqlite_query($db, 'select * from MyTable');      //执行 SQL 语句
echo "<table border=1>";
//通过循环的方式输出从第 0 行到最大的一行的所有记录
while($row = sqlite_fetch_array($result))
{
    $serial_no = $row[0];                                   //输出当前行的 serial_no 列
    $name = $row[1];                                        //输出当前行的 name 列
    $salary = $row[4];                                      //输出当前行的 salary 列
    echo "<tr>";
    echo "<td>$serial_no</td>";
    echo "<td>$name</td>";
```

```
    echo "<td>$salary</td>";
    echo "</tr>";
}
echo "</table>";
?>
```

可以看到，PHP 与 SQLite 的连接方法与 MySQL 的连接方法几乎完全相同，不同的是函数的前缀由 mysql 变成了 sqlite。需要注意的是由于 SQLite 将数据库文件直接存放在磁盘上的文件中，而不是贮存在系统中。因此，PHP 代码的第一步并不是连接数据库管理系统，而是打开这个数据库文件。

1.2.6　Access 与 PHP 的应用

Access 是 Microsoft 推出了一个轻量级数据库，与 SQLite 类似，Access 也是将数据库文件存放在一个磁盘的一个文件中。PHP 并不提供操作 Access 的扩展，但是可以通过 COM 组件来完成。

以下代码在 Access 上执行了一个查询操作。

```
<?php
$conn=new com("adodb.connection");                                          //创建 COM 对象
$conn->open("driver=microsoft access driver (*.mdb);dbq=d:\mytable.mdb");   //连接数据库文件
$rs=$conn->execute("select * from tablename");                              //执行 SQL 语句
echo "<table border=1>";
//通过循环的方式输出从第 0 行到最大的一行的所有记录
while(!$rs->eof)
{
    echo "<tr>";
    echo "<td>$rs->fields["serial_no"]</td>";                               //输出当前行的 serial_no 列
    echo "<td>$rs->fields["name"]</td>";                                    //输出当前行的 name 列
    echo "<td>$rs->fields["salary"]</td>";                                  //输出当前行的 salary 列
    echo "</tr>";
    $rs->movenext();
}
echo "</table>";
$rs->close();
?>
```

上面的操作方式与 PHP 对于其他数据库的操作略有不同，如果读者有 ASP 基础，则可以发现，上面的代码与 ASP 下操作数据库的方式很类似。

1.2.7　ODBC 与 PHP 的应用

ODBC 是 Windows 提供的一个通用数据库引擎，很多类型的数据库都可以通过 ODBC 建立一个对象，应用程序对于数据库的操作即可以通过 ODBC 来完成。

首先介绍如何建立一个 ODBC 连接。双击 Windows 的控制面板下的管理工具文件夹中的“数据源（ODBC）”图标，可以看到如图 1-1 所示的界面。

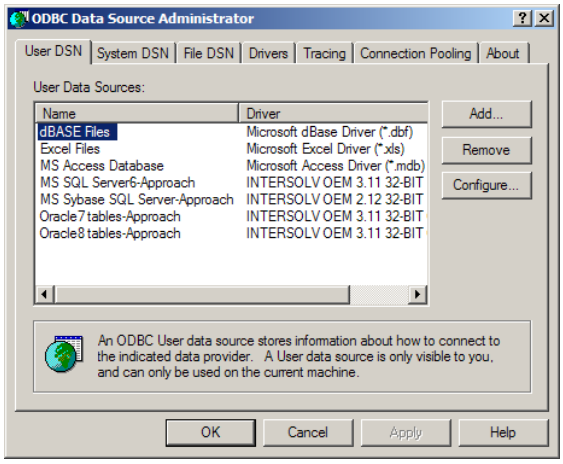

图 1-1　数据源配置界面

从图 1-1 上可以看到多个用于定义 ODBC 连接的面板，要创建用于程序调用的 ODBC 的连接，一般使用“System DSN”来创建。单击“System DSN”标签，然后单击“Add”按钮。这时要求用户选择一个要连接的数据库类型，如图 1-2 所示。

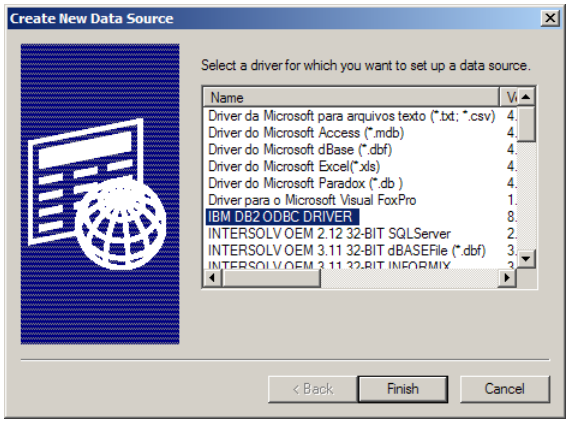

图 1-2　选择连接的数据库类型

这里，选择“IBM DB2 ODBC DRIVER”将建立一个与 IBM DB2 的连接。单击“Finish”按钮完成选择。这时会弹出“ODBC IBM DB2 驱动程序 – 添加”对话框，此时可以选择一个已经存在的数据库并输入相应的数据源名称。数据源名称将在今后应用程序对数据库调用时使用，如图 1-3 所示。

图 1-3　添加新的 ODBC 驱动

单击“确定”按钮后，将会在 ODBC 主面板上看到新的数据源名称。

PHP 提供了直接访问 ODBC 的函数。以下代码通过上面建立的 ODBC 执行了一个查询操作。

```
<?php
$conn=odbc_connect("db2test","db2admin","password");          //建立与 ODBC 的连接
```

```
$sql="select * from mytable";                          //定义 SQL
$result=odbc_do($conn,$sql);                           //执行 SQL 语句
while(odbc_fetch_row($result))                         //通过循环读取数据内容
{
   $serial_no = odbc_result($result,0);                //输出当前行的 serial_no 列
   $name = odbc_result($result,1);                     //输出当前行的 name 列
   $salary = odbc_result($result,2);                   //输出当前行的 salary 列
   echo "<tr>";
   echo "<td>$serial_no</td>";
   echo "<td>$name</td>";
   echo "<td>$salary</td>";
   echo "</tr>";
}
echo "</table>";
odbc_close($myconn);                                   //关闭连接
?>
```

这样，使用 ODBC 将数据库封装起来，直接使用 PHP 操作 ODBC 的方法可以使用相同的程序代码来操作不同的数据库。但是，由于 ODBC 的介入，这种方法的数据库连接往往效率较低，对于大型系统不建议使用这种数据库连接方式。

1.3　小结

本章介绍了几种不同的数据库管理系统中的 SQL 语言，以及 PHP 与几种数据库之间连接方法的不同。在实际应用中，读者可能需要使用 PHP 与各种各样的数据库进行连接。本章仅简要介绍了 PHP 连接其他数据库的方法，读者如果需要与其他数据库建立连接，可以参考 PHP 的编程手册并使用与 PHP 连接 MySQL 相似的方法完成所需的数据库操作。