Классификация СУБД

В качестве основных классификационных признаков СУБД можно использовать следующие:

- вид программы
- характер использования
- модель данных

В общем случае под СУБД можно понимать любой программный продукт, поддерживающий процессы создания, ведения и использования БД.

К СУБД относятся следующие основные виды программ:

- полнофункциональные СУБД
- серверы БД
- клиенты БД
- средства разработки программ работы с БД

Полнофункциональные СУБД

Полнофункциональные СУБД (ПФСУБД) — традиционные СУБД, которые сначала появились для больших машин, затем для мини-машин и для ПЭВМ. Из числа всех СУБД современные ПФСУБД являются наиболее многочисленными и мощными по своим возможностям. К ПФСУБД относятся, например, такие пакеты как: Clarion Database Developer, DataBase, Dataplex, dBase IV, Microsoft Access, Microsoft FoxPro и Paradox R:BASE.

- Развитый интерфейс (позволяет создавать и модифицировать структуры таблиц, вводить данные, формировать запросы, разрабатывать отчеты, выводить их на печать и т.п.)
- Для создания запросов и отчетов не обязательно программирование, а удобно пользоваться языком QBE (Query By Example — формулировки запросов по образцу).
- Многие ПФСУБД включают средства программирования для профессиональных разработчиков.

Некоторые системы имеют в качестве вспомогательных и дополнительные средства проектирования схем БД или CASE-подсистемы. Для обеспечения доступа к другим БД или к данным SQL-серверов полнофункциональные СУБД имеют факультативные модули.

Серверы БД

Серверы БД предназначены для организации центров обработки данных в сетях ЭВМ. Серверы БД реализуют функции управления базами данных, запрашиваемые другими (клиентскими) программами обычно с помощью операторов SQL. Примерами серверов БД являются следующие программы: MySQL (TcX), NetWare SQL (Novell), MS SQL Server (Microsoft), InterBase (Borland), SQLBase Server (Gupta), Intelligent Database (Ingress).

Клиенты БД

В роли ***клиентских программ*** для серверов БД в общем случае могут использоваться различные программы: ПФСУБД, электронные таблицы, текстовые процессоры, программы электронной почты и т. д. При этом элементы пары клиент–сервер могут принадлежать одному или разным производителям программного обеспечения.

Например, для сервера БД SQL Server (Microsoft) в роли клиентских (фронтальных) программ могут выступать многие СУБД, такие как: dBASE IV, Blyth Software, Paradox, DataBase, Focus, 1-2-3, MDBS III, Revelation и другие.

Средства разработки программ работы с БД

Средства разработки программ работы с БД могут использоваться для создания разновидностей следующих программ:

- клиентских программ
- серверов БД и их отдельных компонентов
- пользовательских приложений

К средствам разработки пользовательских приложений относятся системы программирования, например Clipper, разнообразные библиотеки программ для различных языков программирования, а также пакеты автоматизации разработок (в том числе систем типа клиент-сервер). В числе наиболее распространенных можно назвать следующие инструментальные системы: Visual Basic (Microsoft), SILVERRUN (Computer Advisers Inc.), S-Designor (SDP и Powersoft) и ERwin (LogicWorks).

Кроме перечисленных средств, для управления данными и организации обслуживания БД используются различные дополнительные средства, к примеру, мониторы транзакций.

- По характеру использования СУБД делят на ***персональные*** и ***многопользовательские***.

Персональные СУБД обычно обеспечивают возможность создания персональных БД и недорогих приложений, работающих с ними. Персональные СУБД или разработанные с их помощью приложения зачастую могут выступать в роли клиентской части многопользовательской СУБД. К персональным СУБД, например, относятся Visual FoxPro, Paradox, Clipper, dBase, Access и др.

Многопользовательские СУБД включают в себя сервер БД и клиентскую часть и, как правило, могут работать в неоднородной вычислительной среде (с разными типами ЭВМ и операционными системами). К многопользовательским СУБД относятся, например, СУБД Oracle, MySQL и Informix.

- По используемой модели данных СУБД (как и БД), разделяют на ***иерархические***, ***сетевые***, ***реляционные***, ***объектно-ориентированные*** и ***другие типы***. Некоторые СУБД могут одновременно поддерживать несколько моделей данных.
- С точки зрения пользователя, СУБД реализует функции ***хранения***, ***изменения*** (пополнения, редактирования и удаления) и ***обработки информации***, а также ***разработки*** и ***получения*** различных выходных документов.

Для работы с хранящейся в базе данных информацией СУБД предоставляет программам и пользователям следующие два типа языков:

- язык описания данных — высокоуровневый непроцедурный язык декларативного типа, предназначенный для описания логической структуры данных;
- язык манипулирования данными — совокупность конструкций, обеспечивающих выполнение основных операций по работе с данными: ввод, модификацию и выборку данных по запросам.

Названные языки в различных СУБД могут иметь отличия. Наибольшее распространение получили два стандартизованных языка:

- QBE (Query By Example) — язык запросов по образцу,
- SQL (Structured Query Language) — язык структурированных запросов.

QBE в основном обладает свойствами языка манипулирования данными, SQL сочетает в себе свойства языков обоих типов — описания и манипулирования данными.

Модели архитектуры «клиент-сервер»

В качестве основных классификационных признаков СУБД можно использовать следующие:

- вид программы
- характер использования
- модель данных

В общем случае под СУБД можно понимать любой программный продукт, поддерживающий процессы создания, ведения и использования БД.

К СУБД относятся следующие основные виды программ:

- полнофункциональные СУБД
- серверы БД
- клиенты БД
- средства разработки программ работы с БД

Связь с базами данных на примере MySQL

Подключение к серверу базы данных

mysqli_connect() - данная функция имеет три аргумента, в которых указывается имя компьютера, имя пользователя и пароль. Если вы опустите эти необязательные аргументы, то функция будет считать, что требуется компьютер localhost, а имя пользователя и его пароль не требуются, т.е. не установлены в таблице mysqluser.

В следующем примере приводится фрагмент, выполняющий подключение к серверу базы данных:

```
$mysqli_user = "pGG"; // здесь GG – номер группы
$mysqli_password = "pGG";
$conn = mysqli_connect("localhost", $mysqli_user, $mysqli_password);
if (!$conn ) die("Нет соединения с MySQL");
```

Если же вы хотите установить постоянное соединение с базой данных, то для подключения к базе данных можете перед именем хоста дописать p:

```
$conn = mysqli_connect("p:localhost", $mysqli_user, $mysqli_password);
```

Если вы воспользуетесь функцией установки постоянного соединения, то соединение с сервером не исчезает после завершения работы программы или вызова функции mysqli_close()

Выбор базы данных

После того как соединение с сервером MySQL установлено, вам нужно выбрать базу данных, с которой собираетесь работать. ***mysqli_select_db()*** - этой функции нужно передать идентификатор подключения к серверу и имя базы данных. Функция возвращает *true*, если указанная база данных существует и доступ к ней возможен. В следующем примере мы выбираем базу данных с именем *sample*.

```
$database = "sample";
mysqli_select_db($conn, $database)
or die ("Нельзя открыть $database");
```

Обработка ошибок

Номер ошибки можно получить используя функцию ***mysqli_errno()***, а строку с описанием — с помощью функции ***mysqli_error()***, которая возвращает строку с описанием ошибки, если такая произойдет. Функция ***mysqli_error()*** требует соединение с базой данных как аргумент – его вам возвращает функция ***mysqli_connect***. Если программа будет пытаться открыть несуществующую базу данных (db_2), то в таком случае функция ***die()*** выведет примерно такое сообщение:

```
Нельзя открыть db_2: Access denied for user: 'pGG@localhost' to database 'db_2'
```

Добавление данных в таблицу

```
create table domains_brNN
(id INT NOT NULL AUTO_INCREMENT,
 PRIMARY KEY(id),
 domain VARCHAR(20),
 myname VARCHAR(10),
 mail VARCHAR(20));
```

Для добавления данных в эту таблицу нам нужно сконструировать и выполнить запрос SQL. Для этого в языке PHP есть функция ***mysqli_query()***. Этой функции нужно передать идентификатор подключения и строку с запросом. Функция возвращает положительное число, в случае успешного выполнения запроса, и false, если в запросе содержится ошибка или если вы не имеет права на выполнение такого запроса.

```
$query = "create table domains_brNN
        (id INT NOT NULL AUTO_INCREMENT,
         PRIMARY KEY(id),
         domain VARCHAR(20),
         myname VARCHAR(10),
         mail VARCHAR(20)
        )";// здесь NN – номер бригады

$result = mysqli_query($conn, $query)
        or die ("<p>Ошибка: ".mysqli_error());
```

Успешное выполнение запроса не обязательно подразумевает изменение данных в таблице.

Подключение к базе данных и добавление записи в таблицу

```
<html> <head>
<title> Листинг 1. Добавление записи в таблицу </title> </head> <body>
<?php
$user = "pGG"; // здесь GG – номер группы
$pass = "pGG";
$db = "sample"; $table = "domains_brNN";
$conn = mysqli_connect("localhost", $user, $pass);
if (!$conn ) die("Нет соединения с MySQL");
mysqli_select_db($conn, $db)
    or die ("Нельзя открыть $db: ".mysqli_error($conn));
// вставить создание таблицы
$query = "INSERT INTO $table (domain, myname, mail)
  VALUES('123abc.com', 'abc', 'abc@mail.ru')";
mysqli_query($conn, $query)
    or die ("Нельзя добавить данные в таблицу
          $table: " .mysqli_error($conn));
print "<p>Данные в таблицу $table дбавлены";
mysqli_close($conn);
?> </body> </html>
```

Добавление в базу данных информации, введенной пользователем

```
<html> <head> <title> Листинг 2. Добавление в базу данных информации, введенной пользователем
</title> </head> <body> <?php
function Add_to_database($domain, $myname, $mail, &$dberror)
   { $user = "pGG";  // здесь GG – номер группы
    $pass = "pGG"; $db = "sample";
    $table = "domains_brNN";
     $conn = mysqli_connect("localhost", $user, $pass);
    if (!$conn) {$dberror = "Нет соединения с MySQL сервером";
        return false; }
    if (!mysqli_select_db($conn, $db)) {
        $dberror = mysqli_error($conn);
        return false;  }
    $query = "INSERT INTO $table (domain, myname, mail) VALUES('$domain', '$name', '$mail')";
 if (!mysqli_query($conn, $query)) {
        $dberror = mysqli_error($conn);
        return false;
}
    return true;
}
function Write_form()
{
    print "<form action='{$_SERVER['PHP_SELF']}' method='post'>";
    print "<p>Введите имя домена: \n";
    print "<input type='text' name='domain'> ";
    print "<p>Введите ваш e-mail: \n";
    print "<input type='text' name='mail'> ";
    print "<p>Введите вашу фамилию: \n";
    print "<input type='text' name='myname'> ";
    print "<p><input type='submit' value='Записать! '>\n
          </form>\n";
}
```

Добавление в базу данных информации, введенной пользователем

```
// Ввод данных в таблицу
if (isset($_POST['domain']) &&
isset($_POST['myname']) &&
isset($_POST['mail'])) {  #3
    // Обязательно проверить, что вводит
пользователь!
    $dberror = "";
    $ret = Add_to_database($_POST['domain'],
$_POST['myname'], $_POST['mail'],
$dberror);
    if (!$ret) {
        print "Ошибка: $dberror<br>";
    } else {
        print "Спасибо";
    }
}
else Write_form();
?> </body>
</html>
```

В программе из листинга 2 мы для простоты и краткости опустили один очень важный момент — нами не проверялись данные, введенные пользователем. А так поступать, вообще-то, не следует. Нельзя доверять пользователю в вопросах достоверности данных. Все данные, введенные им в форму, нужно тщательно проверять.

Нами проверяется существование переменных $domain, $myname и $mail. Если эти переменные определены, то мы можем быть уверены, что форма передана, и вызываем функцию Add_to_database().

Функция Add_to_database() имеет 4 аргумента: переменные $domain, $myname, $mail, которые переданы пользователем, и строку $dberror. В эту строку будет записано сообщение об ошибке, если такая произойдет. Поэтому мы передаем эту переменную по ссылке. Все изменения, произошедшие с данной переменной внутри функции, повлияют на саму переменную, а не на ее копию.

Нами создается подключение к серверу MySQL, и если оно не происходит, то прекращаем выполнение программы, вернув значение false. Мы выбираем базу данных, содержащую таблицу domains и строим SQL-запрос, добавляющий переданные пользователем данные в таблицу. Этот запрос нами передается функции ***mysqli_query()***.

Относительно функции ***mysqli_query()*** нужно помнить следующее: ее аргумент, т.е. строка запроса к базе данных, не должна содержать символа «;». Фактически это означает, что аргументом может быть только один SQL-запрос. Следовательно, если требуется последовательно выполнить несколько SQL-запросов, необходимо столько же раз использовать mysqli_query(), как в следующем примере:

```
$query = "INSERT INTO domains_brNN (domain, myname, mail) values('aaa.com', 'aaa', 'aaa@mail.ru’)”;
$result = mysqli_query($conn, $query);
$query = "INSERT INTO domains_brNN (domain, myname, mail) values('bbb.com', 'bbb', 'bbb@mail.ru')";
$result = mysqli_query($conn, $query);
$query = "INSERT INTO domains_brNN (domain, myname, mail) values('ddd.com', 'ddd', 'ddd@mail.ru')";
$result = mysqli_query($conn, $query);
```

Получение значения автоматически изменяемого поля

В языке PHP есть функция ***mysqli_insert_id()***, возвращающая значение ключа последней добавленной записи. Этой функции нужно передать идентификатор подключения, а если его опустить, то будет использовано последнее подключение.

```
$query = "INSERT INTO domains_brNN (domain, myname, mail) values('$domain', '$myname', '$mail')";
mysqli_query($conn, $query);
$id = mysqli_insert_id($conn);
print "Спасибо. Ваш номер - $id.
Пожалуйста, используйте его в других запросах.";
```

Доступ к информации

Для этого нужно сделать SQL-запрос типа SELECT, но как воспользоваться результатами этого запроса, т.е. возвращенными записями? При успешном выполнении запроса функция ***mysqli_query()*** возвращает идентификатор результата запроса, и данный идентификатор мы можем передавать другим функциям для обработки результата.

Число записей, найденных в запросе

Узнать число записей, возвращенных в результате выполнения запроса SELECT, можно с помощью функции ***mysqli_num_rows().*** Этой функции нужно передать идентификатор запроса, а возвращает она число записей, содержащихся в этом результате.

Результат запроса

После выполнения запроса SELECT и получения его идентификатора вы можете в цикле просмотреть все записи, найденные в результате запроса. PHP создает для вас внутренний указатель, в котором записана позиция в наборе записей результата. Этот указатель автоматически перемещается на следующую позицию после обращения к текущей записи.

С помощью функции ***mysqli_fetch_row()*** можно для каждой записи получить массив, состоящий из ее полей. Этой функции нужно передать идентификатор запроса, а вернет она массив. По достижении конца запроса функция ***mysqli_fetch_row()*** вернет значение false.

Чтение отдельных полей

```
$result = mysqli_query($conn, "SELECT * from domains_brNN");
$num_fields = mysqli_num_fields($result);
```

- Для выяснения свойств поля передайте идентификатор результата и номер поля функции ***mysqli_fetch_field_direct***. Данная функция принимает идентификатор результата и число, обозначающее порядковый номер поля, а возвращает вам объект – переменную, хранящую в себе все свойства данного поля.

- Чтобы получить имя поля, обратитесь к свойству name:

```
$result = mysqli_query($conn, "SELECT * from domains_brNN");
$num_fields = mysqli_num_fields($result);
for ($x=0; $x<$num_fields; $x++) {
    $properties = mysqli_fetch_field_direct($result, $x);
    print ($properties->name)."<br>\n";
}
```

- Для того чтобы узнать максимальную длину поля, обратитесь к свойству length.

```
$result = mysqli_query("SELECT * from domains_brNN");
$num_fields = mysqli_num_fields($result);
for ($x=0; $x<$num_fields; $x++) {
    $properties = mysqli_fetch_field_direct($result, $x);
    print ($properties->length)."<br>\n";
}
```

- Для того чтобы узнать флаги, обратитесь к свойству flags.

```
$result = mysqli_query("SELECT * from domains_brNN");
$num_fields = mysqli_num_fields($result);
for ($x=0; $x<$num_fields; $x++) {
    $properties = mysqli_fetch_field_direct($result, $x);
    print ($properties->flags)."<br>\n";
}
```

- Точно так же можно выяснить тип поля.

```
$result = mysqli_query("SELECT * from domains_brNN");
$num_fields = mysqli_num_fields($result);
for ($x=0; $x<$num_fields; $x++) {
    $properties = mysqli_fetch_field_direct($result, $x);
    print ($properties->type)."<br>\n";
}
```

Обратите внимание, что для типа поля и флагов возвращается не их строковое название (пример – INT, CHAR, BOOL), а число, которое обозначает их название. Например, типу BOOL соответствует число 1.

Вывод всех записей таблицы

В листинге 3 приведен пример выполнения запроса SELECT, который запрашивает все строки таблицы domains_brNN, затем определяет размер этой таблицы с помощью функции mysqli_num_rows() и выводит на экран всю таблицу domains_brNN.

```
<html> <head>
<title> Листинг 3. Вывод всех записей таблицы
</title> </head> <body>
<?php
$user = "pGG"; // здесь GG – номер группы
$pass = "pGG"; $db = "sample";
$table = "domains_brNN";
$conn = mysqli_connect("localhost", $user, $pass);
if (!$conn) die("Нет соединения с MySQL");
mysqli_select_db($conn, $db)
or die ("Нельзя открыть $db: " . mysqli_error($conn));
$result = mysqli_query($conn, "SELECT * FROM
$table");
$num_rows = mysqli_num_rows($result);
 // количество записей в запросе
print "<P>В таблице $table содержится $num_rows строк";
$num_fields = mysqli_num_fields($result);
// количество столбцов в запросе
print "<p><table style='border: 1px'>\n";
print "<tr>\n";
for ($x = 0; $x < $num_fields; $x++) {
    $name = mysqli_fetch_field_direct($result, $x)->name;
    print "\t<th>$name</th>\n";
    // печатаем имя $x-того столбца
}
print "</tr>\n";
while ($a_row = mysqli_fetch_row($result)) {      // печатаем
содержимое столбцов
    print "<tr>\n";
    foreach ($a_row as $field)  // $a_row – массив
        print "\t<td>$field</td>\n";
    print "</tr>\n";
}
print "</table>\n";
mysqli_close($conn); ?> </body> </html>
```

Функциия ***mysqli_fetch_row()*** возвращает массив. После подключения к серверу базы данных и выбора базы мы с помощью функции ***mysqli_query()*** посылаем запрос на сервер базы данных. Возвращенный результат запроса сохраняем в переменной $result.

Потом нами будет использоваться эта переменная для обращения к результату запроса: в переменной $num_rows мы сохраняем количество записей в запросе, а в переменной $num_fields — количество полей (столбцов) в запросе. Затем, в цикле, в переменную $name заносим название $x–того поля таблицы и выводим это название в виде ячеек-заголовков <th>.

В условном выражении цикла while мы присваиваем переменной $a_row значение, возвращенное функцией ***mysqli_fetch_row()***.

Результатом операции присваивания является значение ее правого операнда, поэтому данное условное выражение имеет значение true, если функция ***mysqli_fetch_row()*** вернет положительное число.

Внутри цикла while мы просматриваем массив, записанный в переменной $a_row, и выводим каждый элемент этого массива на экран, обрамляя его тегами ячейки таблицы. Кроме того, к полям записи можно обратиться по имени: функция ***mysqli_fetch_array()*** возвращает ассоциативный массив, в котором в качестве ключа используются имена полей. В следующем примере используется эта функция.

```
print "<table style='border: 1px >\n";
while ($a_row = mysqli_fetch_array($result))
print "<tr>\n";
print "<td>$a_row[mail]</td>
    <td>$a_row[domain]</td>\n";
print "</tr>\n";
print "</table>\n";
```