Базы данных

База данных - это информационная модель, позволяющая упорядоченно хранить данные о группе объектов, обладающих одинаковым набором свойств. Пожалуй, одним из самых банальных примеров баз данных может быть записная книжка с телефонами ваших знакомых. Этот список фамилий владельцев телефонов и их телефонных номеров, представленный в вашей записной книжке в алфавитном порядке, представляет собой, вообще говоря, проиндексированную базу данных. Использование индекса - в данном случае фамилии (или имени) позволяет вам достаточно быстро отыскать требуемый номер телефона.

	Поле ↓ ФИО	Поле ↓ Номер телефона
Запись →	Алексеев Алексей	111-11-11
	Иванов Иван	222-22-22
	Борисов Борис	333-33-33
	Сергеева Елена	444-44-44

СУБД

Программное обеспечение, предназначенное для работы с базами данных, называется система управления базами данных (СУБД). СУБД используются для упорядоченного хранения и обработки больших объемов информации.

СУБД организует хранение информации таким образом, чтобы ее было удобно:

- просматривать,
- пополнять,
- изменять,
- искать нужные сведения,
- делать любые выборки,
- осуществлять сортировку в любом порядке.

Реляционная база данных

По структуре организации данных базы данных бывают иерархическими и реляционными. *В иерархической базе данных* записи упорядочиваются в определенную последовательность, как ступеньки лестницы, и поиск данных может осуществляться последовательным «спуском» со ступени на ступень.

***Реляционная* база данных**, по сути, представляет собой двумерную *таблицу*. Столбцы таблицы называются **полями**: каждое поле характеризуется своим именем и типом данных. **Поле БД** – это столбец таблицы, содержащий значения определенного свойства.

Типы полей реляционной БД

- ✓**Числовой**
- ✓**Символьный** (слова, тексты, коды и т.д.)
- ✓**Дата** (календарные даты в форме «день/месяц/год»)
- ✓**Логический** (принимает два значения: «да» - «нет» или «истина» - «ложь»)

Числовые типы данных БД

Целые числа:

TINYINT - Может хранить числа от -128 до 127
SMALLINT- Диапазон от -32 768 до 32 767
MEDIUMINT - Диапазон от -8 388 608 до 8 388 607
INT - Диапазон от -2 147 483 648 до 2 147 483 647
BIGINT - Диапазон от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807

Дробные числа:

FLOAT - Число с плавающей точкой небольшой точности
DOUBLE - Число с плавающей точкой двойной точности
REAL - Синоним для DOUBLE.
DECIMAL - Дробное число, хранящееся в виде строки.
NUMERIC - Синоним для DECIMAL.

Строковые и временные типы данных БД

Строковы типы данных:

VARCHAR - может хранить не более 255 символов.

TEXT - может хранить не более 65 535 символов.

MEDIUMTEXT - может хранить не более 16 777 215 символов

Дата и время:

DATE - дата в формате ГГГГ-ММ-ДД

TIME - время в формате ЧЧ:ММ:СС

DATETIME - дата и время в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС

Основные понятия БД

Поле базы данных - это столбец таблицы, содержащий значения определенного свойства.

Запись базы данных - это строка таблицы, содержащая набор значений свойств, размещенный в полях базы данных.

Каждая таблица должна содержать, по крайней мере, одно *ключевое поле*, содержимое которого уникально для каждой записи в этой таблице. Ключевое поле позволяет однозначно идентифицировать каждую запись в таблице.

- ***Ключевое поле*** - *это поле, значение которого однозначно определяет запись в таблице*.

Таблицы

- Когда вы храните информацию в шкафу для документов, вы стараетесь не перемешивать их. Напротив, все документы хранятся в соответствующих папках. В мире баз данных такая папка называется таблицей. Таблица — это структурированный файл, в котором могут храниться данные определенного типа. В таблице может находиться список клиентов, каталог продукции и любая другая информация.

Первичный ключ

В каждой строке таблицы должно быть несколько столбцов, которые уникальным образом идентифицируют ее. В таблице с клиентами для этого может использоваться столбец с номером клиента, тогда как в таблице, содержащей заказы, таким столбцом может быть идентификатор заказа. В таблице со списком служащих может использоваться номер служащего или столбец с номерами карточек социального страхования.

Первичный ключ - столбец (или набор столбцов), значения которого уникально идентифицируют каждую строку таблицы.

Первичный ключ используется для определения конкретной строки. Без него выполнять обновление или удаление строк таблицы было бы очень затруднительно, так как не было бы никакой гарантии, что мы изменяем нужные строки.

Пример базы данных

Рассмотрим, например, базу данных "Компьютер", которая содержит перечень объектов (компьютеров), каждый из которых имеет **имя (название)**. В качестве характеристик (свойств) можно рассмотреть **тип установленного процессора** и **объем оперативной памяти**. Поля *Название* и *Тип процессора* являются текстовыми, *Оперативная память* - числовым, а поле *№ п/п* - счетчиком
При этом каждое поле обладает определенным набором свойств. Например, для поля *Оперативная память* задан формат данных *целое число*.

№ п/п	Название	Тип процессора	Оперативная память (Мбайт)
1	Compaq	Celeron	64
2	Dell	Pentium III	128
3	IBM	Pentium 4	256

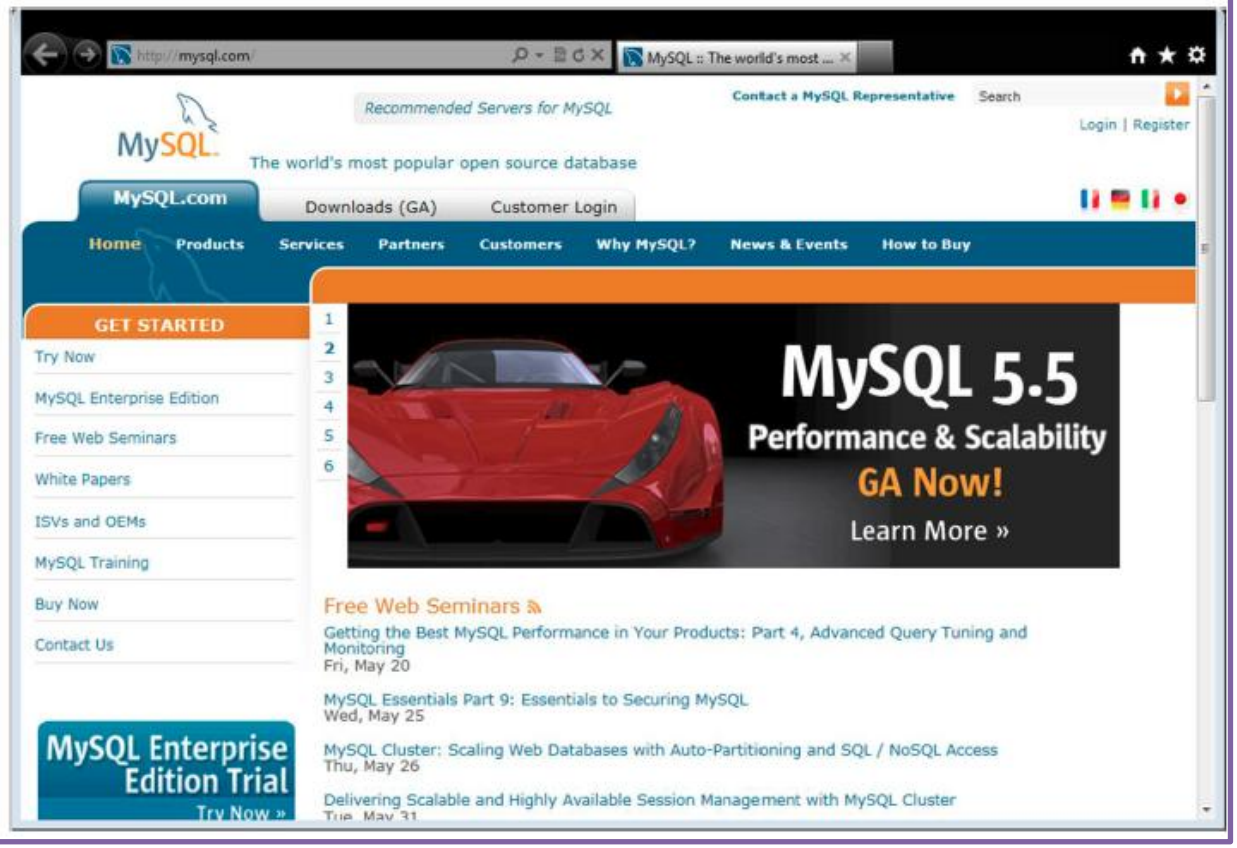
1. В чем заключается разница между записью и полем в табличной базе данных?
2. Поля каких типов полей могут присутствовать в базе данных?
3. Чем отличается ключевое поле от остальных полей?

Практика-1

Создать базу данных магазина и таблицы, которые описывают сферу деятельности торговли

СУБД MySQL

Сервер баз данных MySQL – один из самых распространенных серверов баз данных http://mysql.com



Язык запросов SQL

SQL (ˈɛsˈkjuˈɛl; англ. *structured query language* — «структурированный язык запросов») — формальный непроцедурный язык программирования, применяемый для создания, модификации и управления данными в произвольной реляционной базе данных, управляемой соответствующей системой управления базами данных (СУБД)

Язык запросов SQL

- **select** - оператор выбора данных из таблицы
- **where** - оператор условия для фильтрации данных
- **UPDATE** - оператор обновления таблицы
- **INSERT**- оператор вставки новой записи
- **DELETE**- удаление записей
- **like** - поиск значения в соответствии с шаблоном
- **distinct** - получение уникальных значений
- **count()** - получение количества записей
- **max()** - поиск максимального элемента в столбце
- **min()** - поиск минимального значения в столбце
- **sum()** - вычисление суммы столбца
- **ORDER BY** - сортировка данных

Выборка данных

Для получения данных из таблицы базы данных используется оператор «**SELECT**». Чтобы с помощью данного оператора извлечь данные, нужно указать - что мы хотим выбрать и откуда.

SELECT name_field FROM name_table

Чтобы выбрать все столбцы из таблицы, вместо названий столбцов указывается «*»

Сортировка выбранных данных

По умолчанию выбранные данные выводятся в том порядке, в котором они находятся в таблице. Для точной сортировки данных используется оператор ORDER BY. После оператора указывается имя одного или нескольких столбцов, по которым и сортируются результаты

```
SELECT prod_name FROM Products ORDER BY prod_name;
```

Указание направления сортировки

Сортировка данных не ограничена порядком по возрастанию (от А до Я). Несмотря на то что этот порядок является порядком по умолчанию, в предложении ORDER BY также можно использовать порядок по убыванию (от Я до А). Для этого необходимо указать **ключевое слово DESC**. Если столбец содержит числовое значение то аналогично ***с помощью ключевого слова DESC – сортируем данные по убыванию, а с помощью ключ. слова ASC – по возрастанию***.

```
SELECT prod_id, prod_price, prod_name FROM
Products ORDER BY prod_price DESC;
```

Фильтрация данных

В операторе SELECT данные фильтруются путем указания критерия поиска оператора **WHERE**. Оператор ***WHERE указывается сразу после названия таблицы*** (оператора **FROM**) следующим образом:

SELECT prod_name, prod_price **FROM** Products **WHERE** prod_price = 3.49**;**

Расширенная фильтрация БД

Чтобы увеличить уровень контроля над фильтром, можно использовать несколько предложений WHERE, которые можно использовать в виде логических операторов AND или OR.

```
SELECT * FROM table WHERE count=5
```

Вернутся записи, в которых поле "**count**" имеет значение **5**. Теперь усложним запрос:

```
SELECT * FROM table WHERE count=5 AND id < 100
```

Таким образом, вернутся записи, у которых поле "**count**" имеет значение **5 И** поле "**id**" имеет значение меньше **100**.

Разумеется, Вы можете использовать и другие логические операции. Их полный список:

- **!** (**отрицание**)
- **AND** (**И**)
- **OR** (**ИЛИ**)
- **XOR** (**ИСКЛЮЧАЮЩЕЕ ИЛИ**, иногда ещё называют **МОНТАЖНОЕ ИЛИ**, но такое название встречается в основном в микропроцессорной литературе)

Ключевое слово DISTINCT

Ключевое слово **DISTINCT** используется, когда нужно получить только различающиеся значения.

SELECT DISTINCT имя(имена)_колонки FROM имя_таблицы

Company	OrderNumber
Sega	3412
W3Schools	2312
Trio	4678
W3Schools	6798

```
SELECT DISTINCT Company FROM Orders
```

Вставка новых рядов

Выражение INSERT INTO вставляет в таблицу новые ряды:

```
INSERT INTO имя_таблицы
VALUES (значение_1, значение_2,....)
```

Кроме того, вы можете конкретизировать колонки, в которые вы хотите вставить данные:

```
INSERT INTO имя_таблицы (колонка_1, колонка_2,...)
VALUES (значение_1, значение_2,....)
```

LastName	FirstName	Address	City
Pettersen	Kari	Storgt 20	Stavanger

А вот SQL-выражение:

```
INSERT INTO Persons
VALUES ('Hetland', 'Camilla', 'Hagabakka 24', 'Sandnes')
```

Мы получим вот такой результат:

LastName	FirstName	Address	City
Pettersen	Kari	Storgt 20	Stavanger
Hetland	Camilla	Hagabakka 24	Sandnes

Вставка данных в определенные колонки

Вот таблица Persons:

LastName	FirstName	Address	City
Pettersen	Kari	Storgt 20	Stavanger
Hetland	Camilla	Hagabakka 24	Sandnes

А вот SQL-выражение:

```
INSERT INTO Persons (LastName, Address)
VALUES ('Rasmussen', 'Storgt 67')
```

Мы получим вот такой результат:

LastName	FirstName	Address	City
Pettersen	Kari	Storgt 20	Stavanger
Hetland	Camilla	Hagabakka 24	Sandnes
Rasmussen		Storgt 67	

Обновление рядов БД

Выражение UPDATE обновляет или изменяет ряды:

```
UPDATE имя_таблицы SET имя_колонки = новое_значение
WHERE имя_колонки = некоторое_значение
```

Таблица Person

LastName	FirstName	Address	City
Nilsen	Fred	Kirkegt 56	Stavanger
Rasmussen		Storgt 67	

Обновление одной колонки в ряде

Мы хотим добавить имя персоны, фамилия которой "Rasmussen":

```
UPDATE Person SET FirstName = 'Nina'
WHERE LastName = 'Rasmussen'
```

Мы получим вот такой результат:

LastName	FirstName	Address	City
Nilsen	Fred	Kirkegt 56	Stavanger
Rasmussen	Nina	Storgt 67	

Удаление рядов из таблицы БД

Выражение **DELETE** позволяет удалить ряды(т.е. строки таблицы) из таблицы

```
DELETE FROM имя_таблицы WHERE имя_колонки = значение
```

LastName	FirstName	Address	City
Nilsen	Fred	Kirkegt 56	Stavanger
Rasmussen	Nina	Stien 12	Stavanger

Удаление ряда

Удалим запись о персоне Nina Rasmussen:

```
DELETE FROM Person WHERE LastName = 'Rasmussen'
```

Функция в SQL для подсчета записей

Для подсчета количества записей в базе данных используется функция **COUNT.** Данная функция возвращает число выбранных строк(рядов) в результате запроса.

```
SELECT COUNT(колонка) FROM таблица
```

Name	Age
Hansen, Ola	34
Svendson, Tove	45
Pettersen, Kari	19

Этот запрос возвратит число рядов в таблице:

```
SELECT COUNT(*) FROM Persons
```

Практика-1

- *Создайте базу данных продуктового магазина. Смоделируйте следующие варианты:*

1. Получить товар, который имеет стоимость более 100 руб, но менее 500 руб. В результате не должно быть повторяющихся строк
2. Поступление товара
3. Модификация товара
4. Определенный вид товара продан
5. Подсчитать количество оставшегося определенного товара

Избыточность базы данных

Любая база данных предназначена для хранения информации. И при **проектирование базы данных** следует учесть то, что какая-то информация может повторяться несколько раз. А каждая повторяющаяся запись – это занятое место на диске. То есть превышение количества информации необходимого для хранения данных. Но **информационная избыточность** ведет не только к увеличению требуемого объема памяти для хранения информации содержащейся в базе данных.

Избыточность данных в базе данных – это нежелательное явление еще и потому, что при работе с таблицами базы данных (которые еще называют отношениями), содержащими избыточные данные возникают проблемы связанные с обработкой информации

Пример информационной избыточности

Допустим, у нас есть таблица, в которой хранятся данные список преподавателей и список предметов, которые они ведут. Естественно, в этой таблице присутствует информационная избыточность.

Код преподавателя	Преподаватель	Предметы	Код предмета
1	Иванов	Мат. ан.	1
2	Петров	Химия	2
3	Сидоров	Физика	3
4	Иванов	Дискретная математика	4

Избыточность данных в этой таблице заключается в том, что любой преподаватель может вести несколько предметов, как преподаватель Иванов и для каждого нового предмета приходится добавлять новые записи в таблицу. Один преподаватель может вести разные предметы, а разные предметы могут вести разные преподаватели. Давайте посмотрим, какие аномалии могут произойти в данном конкретном случае и как можно избавиться от аномалий в конкретном случае.

Пример информационной избыточности - продолжение

Допустим, в нашей школе появился новый предмет и мы хотим его добавить в существующую таблицу базы данных, но мы еще не нашли преподавателя для этого предмета. А вписать в таблицу предмет нужно уже сейчас.

В этом случае мы должны присвоить значение NULL каждому атрибуту преподавателя, но делать это никак нельзя, так как атрибут «Код преподавателя» является первичным ключом отношения (первичным ключом таблицы). Результатом попытки создания такой записи будет нарушение целостности данных базы данных, а любая СУБД, в том числе и СУБД MySQL отклонит подобную попытку создания такой записи. Чтобы избавиться от возможных проблем нужно разбить таблицу на две: таблица преподавателей и таблица предметов.

Код преподавателя	Преподаватель	Предметы	Код предмета
1	Иванов	Мат. ан.	1
2	Петров	Химия	2
3	Сидоров	Физика	3
4	Иванов	Дискретная математика	4

Код преподавателя	Преподаватель
1	Иванов
2	Петров
3	Сидоров

Код предмета	Предмет
1	Мат.ан.
2	Химия
3	Физика
4	Дискретная математика

Вопрос на усвоение материала

Как избавиться от избыточности данных в следующей базе данных?

Код преподавателя	Преподаватель	Дата рождения	Предмет	Код предмета
1	Иванов	01.01.1960	Мат.ан.	1
2	Петров	02.02.1950	Химия	2
3	Сидоров	03.03.1940	Физика	3
4	Иванов	01.01.1960	Дискретная математика	4

Подключение к серверу базы данных из PHP

Прежде, чем начать работу с базой данных, необходимо создать соединение с сервером MySQL. Этим и занимается функция mysql_connect(). Эта функция устанавливает соединение с сервером MySQL и возвращает дескриптор соединения с базой данных, по которому все другие функции, принимающие этот дескриптор в качестве аргумента, будут однозначно определять выбранную базу данных. Вторым и третьим аргументами этой функции являются имя пользователя username и его пароль password соответственно

```
<?php
$dblocation = "localhost"; // Имя сервера
$dbuser = "root";          // Имя пользователя
$dbpasswd = "";            // Пароль
$dbcnx = @mysql_connect($dblocation,$dbuser,$dbpasswd);
if (!$dbcnx) // Если дескриптор равен 0 соединение не установлено
{
  echo("<P>В настоящий момент сервер базы данных не доступен, поэтому
        корректное отображение страницы невозможно.</P>");
  exit();
}
?>
```

Выбор базы данных для дальнейшего использования

Перед работой с базой данных необходимо ее выбрать с помощью функции

bool mysql_select_db (string database_name [, resource link_identifier])

Функция принимает в качестве аргументов название выбираемой базы данных database_name и дескриптор соединения resource. Функция возвращает true при успешном выполнении операции и false в противном случае

```
<?php
$dblocation = "localhost";
$dbname = "softtime";
$dbuser = "root";
$dbpasswd = "";
$dbcnx = @mysql_connect($dblocation,$dbuser,$dbpasswd);
if (!$dbcnx)
{
  echo( "<P>В настоящий момент сервер базы данных не доступен, поэтому
        корректное отображение страницы невозможно.</P>" );
  exit();
}
if (!@mysql_select_db($dbname, $dbcnx))
{
  echo( "<P>В настоящий момент база данных не доступна, поэтому
        корректное отображение страницы невозможно.</P>" );
  exit();
}
?>
```

Отправка серверу SQL запросов

Для отправки серверу SQL запросов необходимо использовать PHP функцию

mysql_query (string query)

```
include "config.php";
$ath = mysql_query("select * from authors;");
```

Получение значений из базы данных в виде массива

Данные из базы данных можно получить в виде ассоциативного массива, используя функцию **mysql_fetch_array**

```php
<?php
include "config.php";
$ath = mysql_query("select * from authors;");
if($ath)
{
  $author = mysql_fetch_array($ath);
  echo "<br>имя = ".$author['name']."<br>";
  echo "пароль = ".$author['passw']."<br>";
  echo "e-mail = ".$author['email']."<br>";
  echo "url = ".$author['url']."<br>";
  echo "ICQ = ".$author['icq']."<br>";
  echo "about = ".$author['about']."<br>";
  echo "photo = ".$author['photo']."<br>";
  echo "time = ".$author['time'];
}
else
{
  echo "<p><b>Error: ".mysql_error()."</b></p>";
  exit();
}
?>
```

Пример вывода данных из базы данных в цикле

```php
<?php
include "config.php";
$ath = mysql_query("select * from authors;");
if($ath)
{
  // Определяем таблицу и заголовок
  echo "<table border=1>";
  echo "<tr><td>имя</td><td>пароль</td><td>e-mail</td><td>url</td></tr>";
  // Так как запрос возвращает несколько строк, применяем цикл
  while($author = mysql_fetch_array($ath))
  {
    echo "<tr><td>".$author['name']." </td><td>".$author['passw']."
      </td><td>".$author['email']." </td><td>".
    $author['url']." </td></tr>";
  }
  echo "</table>";
}
else
{
  echo "<p><b>Error: ".mysql_error()."</b><p>";
  exit();
}
?>
```