

МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Методичні вказівки до лабораторних робіт
з курсу дисципліни

«Web-програмування»

Увага! Роботу над методичними вказівками не завершено, вже існуючі завдання змінюватись не будуть, але у теоретичному матеріалі та додатках зміни можливі.

Херсон 2012

Зміст

1	Взаємодія PHP з СКБД MySQL	8
1.1	Мова запитів SQL, та типи даних у СУБД MySQL	8
1.2	Функції для роботи з MySQL. Виконання SQL-запитів	20
1.3	Установка з'єднання. Вибір бази даних	20
1.4	Вибірка з таблиці, розбір результату вибірки	20
1.5	Додавання записів, оновлення записів	20
1.6	Очищення і видалення таблиць	20
1.7	Індивідуальне завдання	20
2	ООП. Робота з XML	22
2.1	Класи та об'єкти. Методи	22
2.1.1	Оголошення класу	23
2.1.2	Створення об'єкта	24
2.2	Наслідування	24
2.3	Конструктори та деструктори	24
2.4	Оператори «::» та «parent»	25
2.5	Об'єктна модель XML-документа	25
2.6	Розширення SAX та DOM для роботи з XML	25
2.7	Індивідуальне завдання	25
3	Робота з сокетом. Пересилання даних JSON	27
3.1	Структура сокета. Відкриття та закриття сокета	27
3.2	Запис в сокет та читання з сокета	27

3.3	Синтаксис JSON, його структура	27
3.4	Кодування та декодування JSON у PHP	27
3.5	Обмін даними з JavaScript-додатками	27
3.6	Індивідуальне завдання	27
4	Робота з зображеннями	29
4.1	Створення зображень	29
4.2	Малювання простих геометричних фігур	29
4.3	Малювання тексту на зображенні	29
4.4	Зміна розміру зображення	29
4.5	Функції бібліотеки GD	29
4.6	Індивідуальне завдання	29
A	Правила оформлення звіту	31
A.1	Титульна сторінка лабораторної роботи	31
A.2	Приклади блок-схем	32
A.3	Оформлення програмного коду	35
A.4	Оформлення скріншотів	35
Б	Матеріали до Л/Р №2	37
B.1	Елементи форм HTML	37
B.1.1	INPUT і його методи	37
B.1.2	Багаторядкове текстове поле	39
B.1.3	Списки з вибором	40
B.2	Змінні-функції	41
В	Матеріали до Л/Р №3	42
V.1	Змінні оточення web-сервера Apache	42
V.1.1	Формовані сервером змінні	42
V.1.2	Спеціальні змінні сервера Apache	43
V.1.3	Змінні HTTP-полів запиту	43
V.1.4	Суперглобальні масиви PHP	44
V.2	Пріоритети виконання операторів	45

Г	Матеріали до Л/Р №4	46
Г.1	Рядки та регулярні вирази	46
Г.1.1	Функції роботи з рядками	46
Г.1.2	Метасимволи та керуючі конструкцію регулярних виразів у MySQL	50
Д	Матеріали до Л/Р №5	53
Д.1	Функції для роботи з файловою системою	53
Д.2	Параметри функції «foren()»	57
Д.3	Функції для роботи з каталогами	58
Е	Матеріали до Л/Р №6	59
Е.1	Налаштування сесії у PHP	59
Е.2	Повний перелік змінних у «php.ini»	59
Е.3	Опис функцій для роботи з сесіями	61
Є	Матеріали до Л/Р №7	64
Є.1	Синтаксис команди CREATE TABLE	64
Є.2	Синтаксис команди ALTER TABLE	66
Ж	Матеріали до Л/Р №8	67
З	Матеріали до Л/Р №9	68
И	Матеріали до Л/Р №10	69

Перелік ілюстрацій

A.1	Приклад нескладної блок-схеми	34
A.2	Оригінал зображення	35
A.3	Оброблене зображення	36
A.4	Необроблене зображення	36

Перелік таблиць

А.1 Основны елементи блок-схем	32
В.1 Пріоритети виконання операторів	45
Г.1 Повний список функцій роботи з рядками	46
Г.2 Спецпослідовності регулярних виразів POSIX 1003.2 для MySQL	51
Г.3 Опис метасимволів регулярних виразів POSIX 1003.2 для MySQL	51
Г.4 Класи символів регулярних виразів POSIX 1003.2 для MySQL .	52
Д.1 Перелік функцій для роботи з ФС	53
Д.2 Другий параметр функції «foren()»	57
Д.3 Перелік функцій для роботи з каталогами	58
Е.1 Перелік параметрів у файлі php.ini	59
Е.2 Повний параметрів у файлі «php.ini» та значень за замовчува- нням	60

Listings

2.1 Синтаксис оголошення класів	23
2.2 Приклад створення методів	23

Лабораторна робота № 1

Взаємодія PHP з СКБД MySQL

Мета роботи

Навчитися працювати з СУБД MySQL, записувати та отримувати табличні дані. Опрацювати взаємодію мови PHP з СУБД MySQL.

1.1 Мова запитів SQL, та типи даних у СУБД MySQL

Основні команди

Команди SQL можна поділити на дві категорії:

- ◇ команди означення даних
- ◇ команди обробки даних

Перша категорія команд відповідає за створення або видалення баз даних, таблиць баз даних, редагування структури таблиць та типів даних полів таблиць. До цієї категорії належать наступні команди:

- ◇ CREATE
- ◇ DROP
- ◇ ALTER
- ◇ RENAME

Друга категорія команд відповідає за внесення даних до таблиць, оновлення та видалення даних. До цієї категорії належать наступні команди:

- ◇ SELECT

- ◇ INSERT
- ◇ DELETE
- ◇ UPDATE
- ◇ HANDLER
- ◇ TRUNCATE
- ◇ REPLACE
- ◇ LOAD DATA INFILE
- ◇ DO

Нижче більш детально розглянуто синтаксис команд.

Команди означення даних

CREATE

Для створення бази даних необхідно використати наступну послідовність команд:

```
CREATE DATABASE [IF NOT EXISTS] db_name
```

Оператор `CREATE DATABASE` створює базу даних з вказаним іменем. Якщо база даних вже існує і не зазначений ключовий параметр `IF NOT EXISTS`, то виникає помилка виконання команди.

Скорочений вид синтаксису команди створення таблиці виглядає наступним чином:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS]
tbl_name [(create_definition, ...)]
[Table_options] [select_statement]
```

Оператор створює таблицю. Якщо таблиця вже існує і не зазначений параметр `IF NOT EXISTS`, то виникає помилка виконання команди. Параметри таблиці та перелік полів вказують після означення імені таблиці. Приклад створення таблиці надано нижче:

```
CREATE TABLE table123 (f1 INT, f2 INT)
```

Створюється таблиця з двома полями цілочисельного типу.

Повний перелік можливих параметрів команди дано у додатку [Є.1](#).

Для створення індексів у СУБД MySQL існує команда `CREATE INDEX`. Синтаксис команди надано нижче:

```
CREATE [UNIQUE | FULLTEXT] INDEX index_name  
ON tbl_name (col_name [(length)], ...)
```

Команда `CREATE INDEX` у версіях MySQL до 3.22 не виконує ніяких дій. У версії 3.22 і більш пізніх `CREATE INDEX` відповідає команді `ALTER TABLE` в частині створення індексів. Зазвичай всі індекси створюються в таблиці під час створення самої таблиці командою `CREATE TABLE`. `CREATE INDEX` дає можливість додати індекси до існуючих таблиць.

DROP

Для видалення бази даних існує наступна команда:

```
DROP DATABASE [IF EXISTS] db_name
```

Оператор `DROP DATABASE` видаляє всі таблиці в зазначеній базі даних і саму базу. Якщо ви виконуєте `DROP DATABASE` на базі даних, символічно пов'язаної з іншою, то видаляється як посилання, так і оригінальна база даних. Будьте дуже уважні при роботі з цією командою!

Оператор `DROP DATABASE` повертає кількість файлів, які були видалені з директорії бази даних. Як правило, це число дорівнює кількості таблиць, помноженому на три, оскільки зазвичай кожна таблиця представлена трьома файлами: `.MYD`-файлом, `.MYI`-файлом і `.Frm`-файлом.

Команда DROP DATABASE видаляє з директорії зазначеної бази даних всі файли з наступними розширеннями: «.BAK», «.DAT», «.HSH», «.ISD», «.ISM», «.ISM», «.MRG», «.MYD», «.MYI», «.Db», «.Frm».

Всі піддиректорії, імена яких складаються з двох цифр (RAID-директорії), також видаляються.

У версії MySQL 3.22 і більш пізніх можна використовувати ключові слова IF EXISTS для попередження помилки, якщо зазначена база даних не існує.

Щоб видалити таблицю з поточної бази даних необхідно виконати наступну команду:

```
DROP TABLE [IF EXISTS] tbl_name  
[, tbl_name, ...] [RESTRICT | CASCADE]
```

Оператор DROP TABLE видаляє одну або декілька таблиць. Всі табличні дані та визначення видаляються, так що будьте уважні при роботі з цією командою!

У версії MySQL 3.22 і більш пізніх можна використовувати ключові слова IF EXISTS , щоб попередити помилку, якщо зазначені таблиці не існують.

Для видалення індексів з таблиць необхідно використати команду

```
DROP INDEX index_name ON tbl_name
```

Оператор DROP INDEX видаляє індекси, зазначені в index_name з таблиці tbl_name. DROP INDEX не виконує ніяких дій у версіях MySQL до 3.22. У версіях 3.22 і більш пізніх DROP INDEX відповідає команді ALTER TABLE в частині видалення індексів.

ALTER

Скорочений синтаксис команди для зміни структури таблиць виглядає наступним чином:

```
ALTER [IGNORE] TABLE tbl_name alter_spec  
[, alter_spec ...]
```

Оператор `ALTER TABLE` забезпечує можливість змінювати структуру існуючої таблиці. Наприклад, можна додавати або видаляти стовпці, створювати або знищувати індекси або перейменовувати стовпці або саму таблицю. Можна також змінювати коментар для таблиці та її тип.

Нижче надано приклади використання команди:

Для того щоб змінити тип стовпця з `INTEGER` на `TINYINT NOT NULL` (залишаючи ім'я колишнім) і змінити тип стовпця `b` з `CHAR(10)` на `CHAR(20)` з перейменуванням його з `b` на `c` :

```
ALTER TABLE t2 MODIFY a TINYINT NOT NULL,  
CHANGE bc CHAR (20);
```

Для того щоб додати індекс до стовпцю `d` і зробити стовпець `a` первинним ключем:

```
ALTER TABLE t2 ADD INDEX (d), ADD PRIMARY KEY (a);
```

Повні дані щодо синтаксису `ALTER TABLE` дано в додатку [Є.2](#).

RENAME

Для перейменування таблиць, починаючи з версії 3.22, введено команду `RENAME TABLE`.

```
RENAME TABLE tbl_name TO new_tbl_name  
[, tbl_name2 TO new_tbl_name2, ...]
```

Операція перейменування повинна здійснюватися як атомарна, тобто при виконанні перейменування ніякому іншому потоку не дозволяється доступ до зазначених таблиць. Завдяки цьому можливе заміщення таблиці порожньої таблицею:

```
CREATE TABLE new_table (...);  
RENAME TABLE old_table TO backup_table,  
new_table TO old_table;
```

Перейменування проводиться зліва направо. Таким чином, для обміну іменами між двома таблицями необхідно виконати наступні дії:

```
RENAME TABLE old_table TO backup_table,  
            new_table TO old_table,  
            backup_table TO new_table;
```

Для двох баз даних, що знаходяться на одному і тому ж диску, можна також здійснювати обмін іменами:

```
RENAME TABLE current_db.tbl_name  
TO other_db.tbl_name;
```

При виконанні команди RENAME не повинні мати місце заблоковані таблиці або активні транзакції. Необхідно також мати привілеї ALTER і DROP для початкової таблиці і привілеї CREATE і INSERT — для нової.

Якщо MySQL стикається з якою-небудь помилкою при перейменуванні декількох таблиць, то станеться зворотнє перейменування для всіх перейменованих таблиць, щоб повернути все в початковий стан.

Команди обробки даних

SELECT

Оператор SELECT має наступну структуру:

```
SELECT [STRAIGHT_JOIN]  
      [SQL_SMALL_RESULT] [SQL_BIG_RESULT]  
      [SQL_BUFFER_RESULT] [SQL_CACHE | SQL_NO_CACHE]  
      [SQL_CALC_FOUND_ROWS] [HIGH_PRIORITY]  
      [DISTINCT | DISTINCTROW | ALL]  
  
select_expression, ...  
  
[INTO {OUTFILE | DUMPFILE}  
'file_name' export_options]  
  
[FROM table_references  
  [WHERE where_definition]  
  [GROUP BY {unsigned_integer |  
    col_name | formula} [ASC | DESC], ...]
```

```
[HAVING where_definition]
[ORDER BY {unsigned_integer |
col_name | formula} [ASC | DESC], ...]
[LIMIT [offset,] rows]
[PROCEDURE procedure_name]
[FOR UPDATE | LOCK IN SHARE MODE]]
```

SELECT застосовується для вилучення рядків, вибраних з однієї або декількох таблиць. Вираз `select_expression` задає стовпці, в яких необхідно проводити вибірку. Крім того, оператор SELECT можна використовувати для витягання рядків, обчислених без посилання якусь таблицю. Наприклад запит `SELECT 1 + 1`; поверне значення «2».

INSERT

```
INSERT [LOW_PRIORITY | DELAYED]
      [IGNORE] [INTO] tbl_name [(col_name, ...)]
      VALUES (expression, ...), (...), ...
```

```
або INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
      [INTO] tbl_name [(col_name, ...)]
      SELECT ...
```

```
або INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
      [INTO] tbl_name
      SET col_name = expression,
      col_name = expression, ...
```

Оператор INSERT вставляє нові рядки в існуючу таблицю. Форма даної команди `INSERT ... VALUES` вставляє рядки відповідно до точно зазначеними в команді значеннями. Форма `INSERT ... SELECT` вставляє рядки, обрані з іншої таблиці або таблиць. Форма `INSERT ... VALUES` зі списком з декількох значень підтримується у версії MySQL 3.22.5 і більш пізніх. Синтаксис вираження `col_name=expression` підтримується у версії MySQL 3.22.10 і більш пізніх.

Команда INSERT ...SELECT забезпечує можливість швидкого внесення великої кількості рядків у таблицю з однієї або більше таблиць.

```
INSERT INTO tblTemp2 (fldID)
    SELECT tblTemp1.fldOrder_ID
    FROM tblTemp1
    WHERE tblTemp1.fldOrder_ID > 100;
```

DELETE

```
DELETE [LOW_PRIORITY | QUICK] FROM table_name
    [WHERE where_definition]
    [ORDER BY ...]
    [LIMIT rows]
```

или

```
DELETE [LOW_PRIORITY | QUICK] table_name[.*] [,table_name[.*] ...]
    FROM table-references
    [WHERE where_definition]
```

оили

```
DELETE [LOW_PRIORITY | QUICK]
    FROM table_name[.*], [table_name[.*] ...]
    USING table-references
    [WHERE where_definition]
```

Оператор DELETE видаляє з таблиці table_name рядки, що задовольняють заданим в where_definition умовам, і повертає число віддалених записів.

Якщо оператор DELETE запускається без визначення WHERE, то видаляються всі рядки. При роботі в режимі AUTOCOMMIT це буде аналогічно використанню оператора TRUNCATE. В MySQL 3.23 оператор DELETE без визначення WHERE поверне нуль як число віддалених записів.

UPDATE

```
UPDATE [LOW_PRIORITY] [IGNORE] tbl_name
      SET col_name1=expr1 [, col_name2=expr2, ...]
      [WHERE where_definition]
      [LIMIT #]
```

Оператор UPDATE оновлює стовпці відповідно до їх новими значеннями в рядках існуючої таблиці. У виразі SET вказується, які саме стовпці слід модифікувати і які величини повинні бути в них встановлені. У виразі WHERE, якщо воно присутнє, задається, які рядки підлягають оновленню. В інших випадках оновлюються всі рядки. Якщо задано вираз ORDER BY, то рядки будуть оновлюватися у зазначеному в ньому порядку.

Якщо вказується ключове слово LOW_PRIORITY, то виконання даної команди UPDATE затримується до тих пір, поки інші клієнти не завершать читання цієї таблиці.

Якщо вказується ключове слово IGNORE, то команда оновлення не буде перервана, навіть якщо при оновленні виникне помилка дублювання ключів. Рядки, через які виникають конфліктні ситуації, оновлені не будуть.

Якщо доступ до стовпцю із зазначеного виразу здійснюється по аргументу tbl_name, то команда UPDATE використовує для цього стовпця його поточне значення.

HANDLER

Оператор HANDLER забезпечує прямий доступ до інтерфейсу дескриптора таблиці MyISAM, мінаючи оптимізатор SQL. Отже, цей оператор працює швидше, ніж SELECT.

Перша форма оператора HANDLER відкриває таблицю, роблячи її доступною для послідовності команд HANDLER ...READ. Цей об'єкт недоступний іншим потокам і не буде закритий, поки даний потік не викличе HANDLER tbl_name CLOSE або сам потік не буде знищений.

Друга форма вибирає один рядок (або більше - у відповідності з установкою в вираженні LIMIT), для якої (их) зазначений індекс відповідає заданій умові і умова в виразі WHERE також виконується. Якщо індекс складається з декількох частин (охоплює декілька стовпців), то складові його величини вказуються у вигляді розділеного комами списку. Забезпечуються величини тільки для декількох перших шпальт.

Третя форма вибирає один рядок (або більше - у відповідності з установкою в вираженні LIMIT), з таблиці; гаразд вказівки індексів у відповідності з умовою WHERE.

Четверта форма (без зазначення індексів) вибирає один рядок (або більше - у відповідності з установкою в вираженні LIMIT), з таблиці, використовуючи природний порядок рядків (як вони зберігаються у файлі даних), у відповідності з умовою WHERE. Ця форма працює швидше, ніж HANDLER tbl_name READ index_name, в тих випадках, коли бажаний перегляд всієї таблиці.

Оператор HANDLER ... CLOSE закриває таблицю, відкриту оператором HANDLER ... OPEN.

TRUNCATE

TRUNCATE TABLE table_name

TRUNCATE TABLE має наступні відмінності від DELETE FROM ...:

- ◇ Ця операція видаляє і відтворює таблицю, що набагато швидше, ніж почергове видалення рядків.
- ◇ Операція є нетранзакційною; якщо одночасно виконується транзакція або активна блокування таблиці, то можна отримати помилку.
- ◇ Не повертає кількість вилучених рядків.
- ◇ Поки існує коректний файл «table_name.frm», таблицю можна відтворити з його допомогою, навіть якщо файли даних або індексів пошкоджені.

TRUNCATE є розширенням Oracle SQL.

REPLACE

```
REPLACE [LOW_PRIORITY | DELAYED]
      [INTO] tbl_name [(col_name,...)]
      VALUES (expression,...),(...),...
```

или REPLACE [LOW_PRIORITY | DELAYED]

```
      [INTO] tbl_name [(col_name,...)]
      SELECT ...
```

или REPLACE [LOW_PRIORITY | DELAYED]

```
      [INTO] tbl_name
      SET col_name=expression, col_name=expression,...
```

Оператор REPLACE працює точно так само, як INSERT, за винятком того, що якщо стара запис у даній таблиці має те ж значення індексу UNIQUE або PRIMARY KEY, що і нова, то стара запис перед занесенням нової буде видалена.

LOAD DATA INFILE

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name.txt'
      [REPLACE | IGNORE]
      INTO TABLE tbl_name
      [FIELDS
        [TERMINATED BY '\t']
        [[OPTIONALLY] ENCLOSED BY '']
        [ESCAPED BY '\\\']
      ]
      [LINES TERMINATED BY '\n']
      [IGNORE number LINES]
      [(col_name,...)]
```

Команда `LOAD DATA INFILE` читає рядки з текстового файлу і вставляє їх у таблицю з дуже високою швидкістю. Якщо задано ключове слово `LOCAL`, то файл читається з клієнтського хоста. Якщо ж `LOCAL` не вказується, то файл повинен знаходитися на сервері. (Опція `LOCAL` доступна у версії MySQL 3.22.6 і більш пізніх.)

Якщо текстові файли, які потрібно прочитати, знаходяться на сервері, то з міркувань безпеки ці файли повинні або розміщуватися в директорії бази даних, або бути доступними для читання всім користувачам. Крім того, для застосування команди `LOAD DATA INFILE` до серверних файлів необхідно володіти привілеями `FILE` для серверного хосту.

DO

`DO expression, [expression, ...]`

Виконує даний вираз, але не повертає небудь результат. Є скороченою формою оператора `SELECT expression, expression`, але перевага його полягає в тому, що він працює трохи швидше, якщо немає необхідності в поверненні результату.

Оператор головним чином корисний при використанні з функціями, які мають побічні ефекти, такими як `RELEASE_LOCK`.

1.2 Функції для роботи з MySQL. Виконання SQL-запитів

1.3 Установка з'єднання. Вибір бази даних

1.4 Вибірка з таблиці, розбір результату вибірки

1.5 Додавання записів, оновлення записів

1.6 Очищення і видалення таблиць

1.7 Індивідуальне завдання

Завдання до лабораторної роботи

1. Вивчити теоретичний матеріал
2. Відповісти на контрольні запитання
3. Скласти алгоритм (блок-схему) програми
4. Виконати практичне завдання
5. Скласти звіт
6. Захистити роботу

Контрольні запитання

1. Що таке рядки?
2. Яким чином можна задати рядковий літерал?
3. Яка особливість рядків у подвійних лапках?

4. Яка особливість синтаксису `Heredoc`?
5. Яким чином вивести HTML-код на веб-сторінці?
6. Що таке регулярні вирази?
7. Яка особливість метасимволу «\»?
8. Для чого використовуються регулярні вирази?
9. Які функції для роботи з регулярними виразами ви знаєте?
10. Яким чином можна використовувати регулярні вирази в SQL-запитах?

Практичні завдання

Написати HTML-сторінку з формою, що складається з:

1. однорядкового поля вводу, поля вводу пароля та кнопки відправлення форми.
- 2.

Лабораторна робота № 2

ООП. Робота з XML

Хоча PHP володіє загальними об'єктно-орієнтованими можливостями, він не є повноцінною ОО-мовою (наприклад, такою, як C++ або Java). Зокрема, в PHP не підтримуються наступні об'єктно-орієнтовані можливості:

- ◇ множинне спадкування;
- ◇ автоматичний виклик конструкторів (якщо ви хочете, щоб при конструюванні об'єкта похідного класу викликався конструктор базового класу, вам доведеться викликати його явно);
- ◇ абстрактні класи;
- ◇ перевантаження методів;
- ◇ перевантаження операторів (це пов'язано з тим, що PHP є мовою з вільною типізацією); закритий і відкритий доступ, віртуальні функції;
- ◇ деструктори.

Але і без усього перерахованого ви все одно зможете отримати користь з об'єктно-орієнтованих можливостей, підтримуваних PHP. Реалізація ООП в PHP надає колосальну допомогу в модульному оформленні функціональності вашої програми.

2.1 Класи та об'єкти. Методи

Класи утворюють синтаксичну базу об'єктно-орієнтованого програмування. Їх можна розглядати як свого роду «контейнери» для логічно пов'язаних даних і функцій (методів). Клас являє собою шаблон, по якому створюються конкретні екземпляри, використовуються в програмі. Екземпляри класів називаються об'єктами.

2.1.1 Оголошення класу

Клас можна розглядати як тип даних, а об'єкт — як змінну. Програма може одночасно працювати з декількома об'єктами одного класу як і з декількома змінними. Загальний формат класів РНР приведений в наступному лістингу.

Лістинг 2.1 – Синтаксис оголошення класів

```
class Class_name {  
var $attribute_1;  
//...  
var $attribute_N;  
function function1() {  
//...  
}  
//...  
function functionN() {  
//...  
}  
} // end Class_name
```

Оголошення класу має починатися з ключового слова **class** (подібно до того, як оголошення функції починається з ключового слова **function**). Кожному оголошенню атрибуту, що міститься в класі, має передувати ключове слово **var**. Атрибути можуть відноситися до будь-якого типу даних, підтримуваних в РНР. Після оголошень атрибутів слідують оголошення методів, дуже схожі на типові оголошення функцій.

Методи часто використовуються для роботи з атрибутами класів. При посиланнях на атрибути всередині методів використовується спеціальна змінна **\$this**. Синтаксис методів продемонстрований в наступному прикладі:

Лістинг 2.2 – Приклад створення методів

```
<?  
class Webpage {  
var $bgcolor;  
function setBgColor($color) {
```

```
$this->bgcolor = $color;
}
function getBgColor() {
return $this->bgcolor;
}
}
?>
```

Змінна `$this` посилається на екземпляр об'єкта, для якого викликається метод. Оскільки в будь-якому класі може існувати декілька екземплярів об'єктів, уточнення `$this` необхідно для посилань на атрибути, що належать поточному об'єкту. При використанні цього синтаксису зверніть увагу на дві обставини:

1. атрибут, на який ви посилаєтеся в методі, не потрібно передавати у вигляді параметра функції;
2. знак долара (\$) ставиться перед змінною `$this`, але не перед ім'ям атрибута (як у звичайної змінної).

2.1.2 Створення об'єкта

2.2 Наслідування

2.3 Конструктори та деструктори

Досить часто при створенні об'єкта потрібно задати значення деяких атрибутів. На щастя, розробники технології ООП врахували цю обставину і реалізували його в концепції конструкторів. Конструктор являє собою метод, який задає значення деяких атрибутів (а також може викликати інші методи). Конструктори викликаються автоматично при створенні нових об'єктів. Щоб це стало можливим, ім'я методу-конструктора повинне збігатися з ім'ям класу, в якому він міститься. Приклад конструктора приведений в лістингу 2.

2.4 Оператори «::» та «parent»

2.5 Об'єктна модель XML-документа

2.6 Розширення SAX та DOM для роботи з XML

2.7 Індивідуальне завдання

Завдання до лабораторної роботи

1. Вивчити теоретичний матеріал
2. Відповісти на контрольні запитання
3. Скласти алгоритм (блок-схему) програми
4. Виконати практичне завдання
5. Скласти звіт
6. Захистити роботу

Контрольні запитання

1. Що таке рядки?
2. Яким чином можна задати рядковий літерал?
3. Яка особливість рядків у подвійних лапках?
4. Яка особливість синтаксису Heredoc?
5. Яким чином вивести HTML-код на веб-сторінці?
6. Що таке регулярні вирази?
7. Яка особливість метасимволу «\»?

8. Для чого використовуються регулярні вирази?
9. Які функції для роботи з регулярними виразами ви знаєте?
10. Яким чином можна використовувати регулярні вирази в SQL-запитах?

Практичні завдання

Написати HTML-сторінку з формою, що складається з:

1. однорядкового поля вводу, поля вводу пароля та кнопки відправлення форми.
- 2.

Лабораторна робота № 3

Робота з сокетами. Пересилання даних JSON

3.1 Структура сокета. Відкриття та закриття сокета

3.2 Запис в сокет та читання з сокета

3.3 Синтаксис JSON, його структура

3.4 Кодування та декодування JSON у PHP

3.5 Обмін даними з JavaScript-додатками

3.6 Індивідуальне завдання

Завдання до лабораторної роботи

1. Вивчити теоретичний матеріал
2. Відповісти на контрольні запитання
3. Скласти алгоритм (блок-схему) програми
4. Виконати практичне завдання
5. Скласти звіт
6. Захистити роботу

Контрольні запитання

1. Що таке рядки?

2. Яким чином можна задати рядковий літерал?
3. Яка особливість рядків у подвійних лапках?
4. Яка особливість синтаксису `Heredoc`?
5. Яким чином вивести HTML-код на веб-сторінці?
6. Що таке регулярні вирази?
7. Яка особливість метасимволу «\»?
8. Для чого використовуються регулярні вирази?
9. Які функції для роботи з регулярними виразами ви знаєте?
10. Яким чином можна використовувати регулярні вирази в SQL-запитах?

Практичні завдання

Написати HTML-сторінку з формою, що складається з:

1. однорядкового поля вводу, поля вводу пароля та кнопки відправлення форми.
- 2.

Лабораторна робота № 4

Робота з зображеннями

4.1 Створення зображень

4.2 Малювання простих геометричних фігур

4.3 Малювання тексту на зображенні

4.4 Зміна розміру зображення

4.5 Функції бібліотеки GD

4.6 Індивідуальне завдання

Завдання до лабораторної роботи

1. Вивчити теоретичний матеріал
2. Відповісти на контрольні запитання
3. Скласти алгоритм (блок-схему) програми
4. Виконати практичне завдання
5. Скласти звіт
6. Захистити роботу

Контрольні запитання

1. Що таке рядки?
2. Яким чином можна задати рядковий літерал?

3. Яка особливість рядків у подвійних лапках?
4. Яка особливість синтаксису **Heredoc**?
5. Яким чином вивести HTML-код на веб-сторінці?
6. Що таке регулярні вирази?
7. Яка особливість метасимволу «\»?
8. Для чого використовуються регулярні вирази?
9. Які функції для роботи з регулярними виразами ви знаєте?
10. Яким чином можна використовувати регулярні вирази в SQL-запитах?

Практичні завдання

Написати HTML-сторінку з формою, що складається з:

1. однорядкового поля вводу, поля вводу пароля та кнопки відправлення форми.
- 2.

Додаток А

Правила оформлення звіту

А.1 Титульна сторінка лабораторної роботи

МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Звіт до лабораторної роботи №123
з дисципліни «Web-програмування»

Тема: «Основи мережі Internet»

Виконав
ст.групи хПР1

Пупкін А.А.

Перевірив
ст.викладач

Іванов Б.Б.


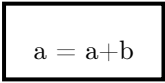
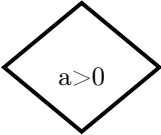
Херсон 2012

А.2 Приклади блок-схем

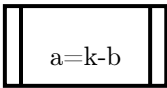
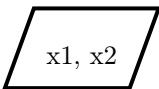
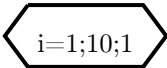

Правила виконання блок-схем задані наступними документами:

- ◇ ГОСТ 19.701-90. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения
- ◇ ГОСТ 19.002-80. Схемы алгоритмов и программ. Правила выполнения
- ◇ ГОСТ 19.003-80. Схемы алгоритмов и программ. Обозначения условные графические

Табл. А.1. Основны елементи блок-схем

Найменування	Позначення	Призначення
Блок початок-кінець		Елемент відображає вхід із зовнішнього середовища або вихід з неї (найбільш часто застосовування - початок і кінець програми).
Блок обчислень		Виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування).
Логічний блок		Відображає рішення або функцію перемикача типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов.

Продовження

Найменування	Позначення	Призначення
Зумовлений процес		Символ відображає виконання процесу, що складається з однієї або декількох операцій, який визначений в іншому місці програми (в підпрограмі, модулі).
Дані		Перетворення даних у форму, придатну для обробки (введення) або відображення результатів обробки (висновок).
Кордон циклу		Символ складається з двох частин — відповідно, початок і кінець циклу - операції, що виконуються всередині циклу, розміщуються між ними. Умови циклу і збільшення записуються всередині символу початку або кінця циклу - в залежності від типу організації циклу.
З'єднувач		Символ відображає вхід в частину схеми і вихід з іншої частини цієї схеми.

Приклад типової блок-схеми з вводом даних, розгалуженням, обчисленням та виводом даних дано на малюнку [A.1](#).

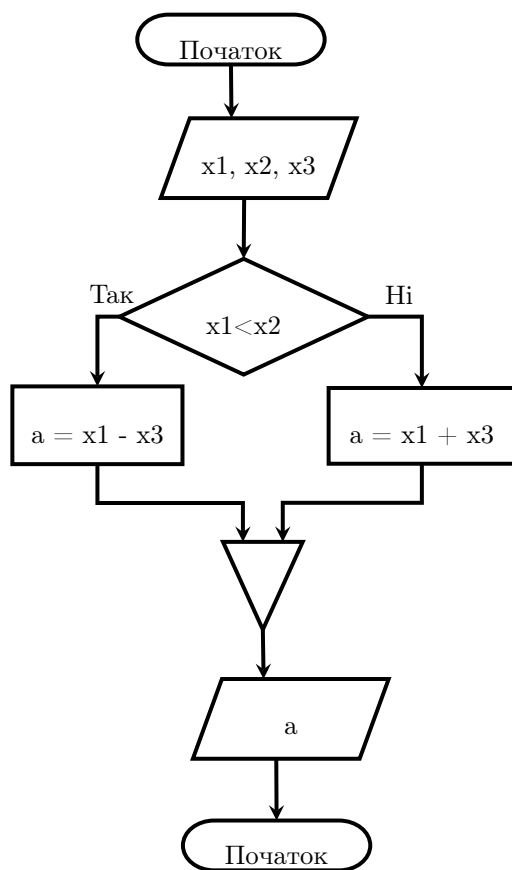


Рис. А.1 – Приклад нескладної блок-схеми

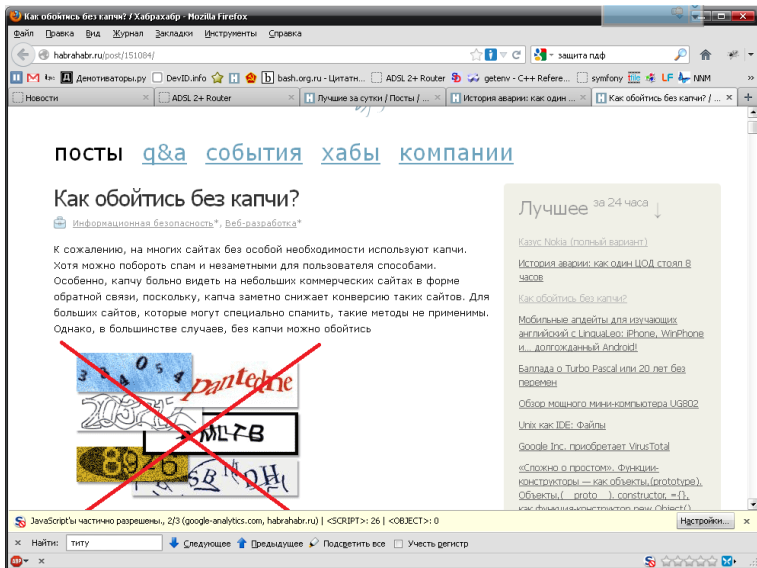


Рис. А.2 – Оригінал зображення

А.3 Оформлення програмного коду

Програмний код необхідно оформлювати у друкованому вигляді, бажано використовуючи моноширинний шрифт Courier або Courier New розміром 12 – 14pt. Приклад оформлення лістингів надано нижче.

А.4 Оформлення скріншотів

Скріншоти HTML-форм або РНР-сценаріїв бажано оформлювати без рамки веб-браузера по 1 – 2 зображення на сторінку. При використанні на фоні HTML-сторінки темних кольорів можливе редагування зображення з метою висвітлення кольорів. Приклад зображення, що отримано з комбінацій клавіш **Alt+PrtCs** зображена на малюнку А.2, та бажане зображення на малюнку малюнку А.3. Небажане оформлення скріншота дано на малюнку А.4.

посты [q&a](#) [события](#) [хабы](#) [компании](#)

Как обойтись без капчи?

[Информационная безопасность*, Веб-разработка*](#)

К сожалению, на многих сайтах без особой необходимости используют капчи. Хотя можно побороть спам и незаметными для пользователя способами. Особенно, капчу больно видеть на небольших коммерческих сайтах в форме обратной связи, поскольку, капча заметно снижает конверсию таких сайтов. Для больших сайтов, которые могут специально спамить, такие методы не применимы. Однако, в большинстве случаев, без капчи можно обойтись



Лучшее за 24 часа ↓

[Какую Nokia \(полный вариант\)](#)

[История аварии: как один ЦОД стоил 8 часов](#)

[Как обойтись без капчи?](#)

[Мобильные приложения для изучения английского с LinguaLeo: iPhone, WinPhone и... долгожданный Android!](#)

[Батпапа о Turbo Pascal или 20 лет без перемен](#)

[Обзор мощного мини-компьютера U9802](#)

[Утил как IDE: Файлы](#)

[Google Inc. приобретает VirusTotal](#)

[«Спокойно о простом»: функции, конструкции — как объекты \(prototype\), объекты \(.. proto ..\) constructor = 0, как функции \(constructor.name \(Object\)\)](#)

Рис. А.3 – Обработанное изображение

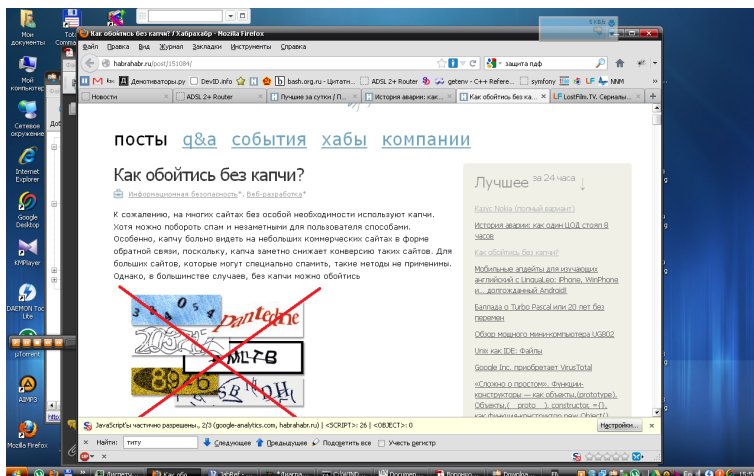


Рис. А.4 – Необработанное изображение

Додаток Б

Матеріали до Л/Р №2

Б.1 Елементи форм HTML

Б.1.1 INPUT і його методи

Однорядкові поля введення даних

```
<input type=text name=им'я_параметра [value=значення]  
[size=розмір_поля] [maxlength=довжина_поля]>
```

Даний тег створює поле вводу з максимально допустимою довжиною тексту «maxlength» і розміром в size «знакомест». Якщо вказаний атрибут «value», то в поле буде спочатку відображатися значення даного атрибуту. У квадратних дужках [] позначені необов'язкові атрибути.

Поля вводу паролів

```
<input type=password name=им'я_параметра [value=значення]  
[size=розмір_поля] [maxlength=довжина_поля]>
```

Структура даного тегу така сама як і у <input type=text>. Різниця лише у відображенні даних, що вводить користувач.

Приховане текстове поле

```
<input type=hidden name=им'я_параметра [value=значення]>
```

Такі поля передають дані серверу, але не відображаються на сторінці. Значення атрибуту «value» встановлюється при формуванні сторінки, або JavaScript-сценарієм.

Незалежні перемикачі

```
<input type=checkbox name=им'я_параметра [value=значення]
[checked]>
```

Перемикач виду «прапорець». У разі встановлення прапорця при відправленні форми серверу будуть передані параметри «им'я_параметра=значення». Якщо прапорець не встановлено серверу взагалі нічого не буде відправлено.

Перемикач за замовчуванням вимкнтий, щоб зробити його увімкнтим за замовчуванням треба встановити атрибут «checked».

Стан перемикача не залежить від стану інших перемикачів цього типу.

Залежні перемикачі

Залежний перемикач, так само як і незалежний перемикач, може заходитись у двох станах, в залежності від атрибуту «checked». При цьому на формі може бути увімкнений лише один перемикач серед групи перемикачів з однаковим атрибутом «name».

```
<form action="http://localhost/script.php" method="GET">
<input type=radio name=answer value=yes checked>Да
<input type=radio name=answer value=no>Нєт
<input type=submit value=Отправить>
</form>
```

Кнопки відправлення та очищення параметрів форми

Кнопка відправки служить для передачі серверу змісту форми на сторінці. Атрибут «value» визначає текст, що відображається на кнопці. Під час відправлення форми серверу будуть передані дані кнопки у вигляді «им'я_параметра=значення».

```
<input type=submit [name=им'я_параметра] value=значення>
```

У разі використання кнопки із зображенням серверу передадуться координати кліку відносно зображення.

```
<input type=submit [name=им'я_параметра] src=зображення>
```

Кнопка очищення форми знищує всі зміни внесені користувачем сайту у дану форму.

```
<input type=reset [name=им'я_параметра] value=значення>
```

Поле вибору файлу

Тег `<input>` також дозволяє активувати діалогове вікно вибору файлу та завантажувати його на сервер при відправленні форми.

```
<input type=file name=имя [value=имя_файла]>
```

Б.1.2 Багаторядкове текстове поле

Синтаксис багаторядкового поля виглядає наступним чином:

```
<textarea name=имя [cols=ширина_в_символах]
[rows=высота_в_символах] wrap=тип_переноса>
текст за замовчуванням
</textarea>
```

Хоча висота і ширина поля необов'язкові параметри їх бажано вказувати. Атрибут «wrap» відповідає за перенос і може приймати наступні значення:

1. Virtual — справа від тексту з'являється полоса прокрутки, а текст розбивається на рядки.
2. Physical — залежить від браузера і виглядає по-різному
3. None — текст залишається у тому вигляді, в якому користувач його ввів, з'являються горизонтальна і вертикальна полоси прокрутки.

Б.1.3 Списки з вибором

Списки з одиночним вибором

Списки з одиночним вибором реалізуються за допомогою наступної конструкції:

```
<select name=day size=1>
<option value=1>Понедельник</option>
<option value=2>Вторник</option>
<option value=3 selected>Среда</option>
<option value=4>Четверг</option>
<option value=5>Пятница</option>
<option value=6>Суббота</option>
<option value=7>Воскресенье</option>
</select>
```

При відправленні форми сервер отримає дані виду «им'я_параметра=значення». За замовчуванням може бути обраний пункт списку серед атрибутів якого є **selected**.

Списки з множинним вибором

Список з множинним вибором відрізняється лише атрибутом **multiple** в середині тега **<select>**.

```
<select name=day size=7 multiple>
<option value=1>Понедельник</option>
<option value=1>Вторник</option>
<option value=1>Среда</option>
<option value=1>Четверг</option>
<option value=1>Пятница</option>
```



```
<option value=1>Суббота</option>
<option value=1>Воскресенье</option>
</select>
```

Б.2 Змінні-функції

Приклад використання змінної-функції:

```
<?php
?php
function foo() {
    echo "In foo()<br />\n";
}
function bar($arg = '')
{
    echo "In bar(); argument was '$arg'.<br />\n";
}
// Функция-обертка для echo
function echoit($string)
{
    echo $string;
}

$func = 'foo';
$func();          // Вызывает функцию foo()
$func = 'bar';
$func('test');    // Вызывает функцию bar()
$func = 'echoit';
$func('test');    // Вызывает функцию echoit()
```

Додаток В

Матеріали до Л/Р №3

В.1 Змінні оточення web-сервера Apache

В.1.1 Формовані сервером змінні

AUTH_TYPE Використовується схема аутентифікації. Зазвичай BASIC

CONTENT_LENGTH Довжина вмісту

CONTENT_TYPE MIME-тип вмісту

GETAWAY_INTERFACE Версія CGI, наприклад CGI/1.1

PATH_info HTTP-шлях до сценарію

PATH_TRANSLATED Повний шлях до сценарію

REMOTE_ADDR IP-адреса запитуваного комп'ютера-клієнта

REMOTE_HOST Доменне ім'я запитувача комп'ютера (якщо доступно). Доменне ім'я визначається веб-сервером за допомогою служби DNS. Директива HostnameLookups сервера Apache дозволяє (або забороняє) перетворення IP-адреси в доменне ім'я.

REMOTE_PORT Порт, закріплений за браузером для отримання відповіді від сервера

REMOTE_USER Ім'я користувача, що пройшов аутентифікацію

QUERY_STRING Рядок переданих серверу параметрів

SERVER_ADDR IP-адресу сервера

SERVER_NAME Доменне ім'я сервера. Визначається директивою ServerName файлу конфігурації

SERVER_PORT TCP-порт Web-сервера. Зазвичай 80

SERVER_PROTOCOL Версія протоколу HTTP. Наприклад, HTTP/1.1

SERVER_SOFTWARE Програмне забезпечення сервера

SCRIPT_NAME HTTP-шлях до сценарію

SCRIPT_FILENAME Файл сценарію в файловій системі сервера (фізичний шлях). Наприклад, `/var/www/cgi-bin/script.cgi`

В.1.2 Спеціальні змінні сервера Apache

DOCUMENT_ROOT Фізичний шлях до кореневого WWW-каталогу сервера. Наприклад, `/var/www.html/`

SERVER_ADMIN Адреса електронної пошти адміністратора сервера

SERVER_SIGNATURE Підпис сервера. Наприклад, «Apache/1.3.3 сервера на `www.somefirm.com` порт 80»

В.1.3 Змінні HTTP-полів запиту

HTTP_HOST Ім'я віртуального хоста, якому адресовано запит

HTTP_USER_AGENT Програмне забезпечення віддаленого користувача. Зазвичай ця змінна оточення містить назву і версію браузера

HTTP_ACCEPT Список підтримуваних клієнтом типів інформації.

HTTP_ACCEPT_LANGUAGE Список підтримуваних мов в порядку переваги, наприклад, RU, EN

HTTP_ACCEPT_ENCODING Список підтримуваних методів стиснення

HTTP_ACCEPT_CHARSET Список підтримуваних кодувань

HTTP_CONNECTION Тип з'єднання. Можливі два варіанти: •

- **Keep-Alive** — якщо після відповіді на запит не потрібно розривати з'єднання;
- **Close** — якщо потрібно закрити з'єднання відразу після відповіді на запит.

HTTP_REFERER Значення поля `REFERER`. У цьому полі браузер передає URL ресурсу, який посилається на наш сервер. Наприклад, якщо

користувач перейшов на сайт зі сторінки `http://www.somehost.com/page.php`, то значення поля `REFERER` буде `http://www.somehost.com/page.php`.

HTTP_X_FORWARDED_FOR Якщо користувач працює через проксі-сервер, то в цьому полі буде IP-адреса комп'ютера, який звернувся до проксі-сервера. Якщо це поле вже містить значення, то нове значення буде додано через кому.

B.1.4 Суперглобальні масиви PHP

\$GLOBALS — масив всіх глобальних змінних (у тому числі і для користувача).

\$_SERVER — містить безліч інформації про поточний запит і сервер.

\$_ENV — поточні змінні середовища. Їх набір специфічний для кожної конкретної платформи, на якій виконується сценарій.

\$_GET — асоціативний масив з параметрами GET-запиту. У початковому вигляді ці параметри доступні в **\$_SERVER** [`'QUERY_STRING'`] і в **\$_SERVER** [`'REQUEST_URI'`] в складі URI.

\$_POST — асоціативний масив значень полів HTML-форми при відправки методом POST.

\$_FILES — асоціативний масив з відомостями про надіслані методом POST файлах. Кожен елемент має індекс ідентичний значенню атрибута `name` у формі і, в свою чергу, також є масивом з наступними елементами:

1. **\$_FILES['name']** — вихідне ім'я файлу на комп'ютері користувача.
2. **\$_FILES['type']** — зазначений агентом користувача MIME — тип файлу.
3. **\$_FILES['size']** — розмір файлу в байтах.
4. **\$_FILES['tmp_name']** — повний шлях до файлу в тимчасовій папці.
5. **\$_FILES['error']** — код помилки.

\$_COOKIE — асоціативний масив з переданими агентом користувача значеннями cookie.

`$_REQUEST` — загальний масив ввідних даних запиту користувача як в масивах **`$_GET`**, **`$_POST`**, **`$_COOKIE`**. Починаючи з версії PHP 4.1 включається і вміст **`$_FILES`**.

B.2 Пріоритети виконання операторів

Табл. B.1. Пріоритети виконання операторів

Асоціативність	Оператор
неасоціативна	<code>new</code>
права	<code>[</code>
неасоціативна	<code>++ --</code>
неасоціативна	<code>! ~ -(int) (float) (string) (array) (object) @</code>
ліва	<code>* / %</code>
ліва	<code>+ - .</code>
ліва	<code><< >></code>
неасоціативна	<code>< <= > >=</code>
неасоціативна	<code>== != === !==</code>
ліва	<code>&</code>
ліва	<code>^</code>
ліва	<code> </code>
ліва	<code>&&</code>
ліва	<code> </code>
ліва	<code>? :</code>
права	<code>= += -= *= /= .= %= &= = ^= <<= >>=</code>
ліва	<code>and</code>
ліва	<code>xor</code>
ліва	<code>or</code>
ліва	<code>,</code>

Додаток Г

Матеріали до Л/Р №4

Г.1 Рядки та регулярні вирази

Г.1.1 Функції роботи з рядками

Табл. Г.1. Повний список функцій роботи з рядками

Функція	Опис
addslashes	екрануючі спецсимволи в стилі мови С
addslashes	екрануючі спецсимволи в рядку
bin2hex	Перетворює бінарні дані у шістнадцятірільне подання
chr	Повертає символ за його кодом
chunk_split	Розбиває рядок на фрагменти
convert_cyr_string	Перетворює рядок з одного кириличної кодування в інше
count_chars	Повертає інформацію про символи, що входять в рядок
crc32	Обчислює CRC32 для рядка
crypt	Необоротне шифрування (хешування)
echo	Виводить одну чи більше рядків
explode	Розбиває рядок на підрядки
fprintf	Записує отформатованну рядок у потік
get_html_translation_table	Повертає таблицю перетворень
hebrew	Перетворює текст на івриті з логічного кодування у візуальне

Продовження

Функція	Опис
hebrevc	Перетворює текст на івриті з логічного кодування у візуальне з перетворенням в переклад
htmlentities	Перетворює символи у відповідні HTML теги
htmlspecialchars	Перетворює спеціальні символи в HTML теги
html_entity_decode	Перетворює HTML теги в відповідні символи
implode	Об'єднує елементи масиву в рядок
localeconv	Повертає інформацію про числові формати
ltrim	Видаляє пробіли з початку рядка
md5	Повертає MD5-хеш рядка
md5_file	Повертає MD5-хеш файлу
metaphone	Повертає ключ metaphone для рядка
nl2br	Вставляє HTML-код розриву рядка перед кожним переведенням рядка
number_format	Форматує число з поділом груп
ord	Повертає ASCII-код символу
parse_str	Розбирає рядок у змінні
print	Виводить рядок
printf	Виводить відформатований рядок
quoted_printable_decode	розкодує рядок, закодовану методом quoted printable
quotemeta	екрануючі спеціальні символи
rtrim	Видаляє пробіли з кінця рядка
sha1	Повертає SHA1-хеш рядка
sha1_file	Повертає SHA1-хеш файлу
similar_text	Обчислює ступінь схожості двох рядків
soundex	Повертає ключ soundex для рядка
sprintf	Повертає відформатований рядок

Продовження

Функція	Опис
sscanf	Розбирає рядок у відповідності із заданим форматом
strcasecmp	Порівняння рядків без урахування регістра, безпечне для даних у двійковій формі
strcmp	Порівняння рядків, безпечне для даних у двійковій формі
strcoll	Порівняння рядків з урахуванням поточної локалі
strcspn	Повертає довжину ділянки на початку рядка, не відповідного масці
stripslashes	Видаляє екранування символів, вироблене функцією addslashes ()
stripos	Повертає позицію першого входження підрядка без урахування регістра
stripslashes	Видаляє екранування символів, вироблене функцією addslashes ()
strip_tags	Видаляє HTML і PHP теги з рядка
stristr	Аналог функції strstr, але незалежний від регістру
strlen	Повертає довжину рядка
strnatcasecmp	Порівняння рядків без урахування регістра з використанням алгоритму
strnatcmp	Порівняння рядків з використанням алгоритму "природнього упорядкування"
strncasecmp	порівняння перших n символів рядків без урахування регістра, безпечне для даних у двійковій формі

Продовження

Функція	Опис
strncmp	порівняння перших n символів рядків без урахування регістра, безпечне для даних у двійковій формі
strpos	Знаходить перше входження підрядка в рядок
strrchr	Знаходить останнє входження символу в рядок
strrev	Перевертає рядок
strrpos	Повертає позицію останнього входження підрядка без урахування регістра
strrpos	Знаходить останнє входження символу в рядок
strspn	Повертає довжину ділянки на початку рядка, відповідного масці
strstr	Знаходить перше входження підрядка
strtok	Розбиває рядок
strtolower	Перетворює рядок у нижній регістр
strtoupper	Перетворює рядок у верхній регістр
strtr	Перетворює задані символи
str_ireplace	Регістро-незалежний варіант функції str_replace ().
str_pad	Доповнює рядок іншим рядком до заданої довжини
str_repeat	Повертає повторювану рядок
str_replace	Замінює рядок пошуку на рядок заміни
str_rot13	Виконує над рядком перетворення ROT13
str_shuffle	переміщує символи в рядку
str_split	Розбиває рядок в масив

Продовження

Функція	Опис
str_word_count	Повертає інформацію про слова, що входять в рядок
substr	Функція повертає частину рядка
substr_count	Підраховує кількість входжень підрядка в рядок
substr_replace	Замінює частину рядка
trim	Видаляє пробіли з початку та кінця рядка
ucfirst	Перетворює перший символ рядка в верхній регістр
ucwords	Перетворює у верхній регістр перший символ кожного слова в рядку
vprintf	Виводить відформатований рядок
vsprintf	Повертає відформатований рядок
wordwrap	Виконує перенесення рядка на дану кількість символів з використанням символу розриву рядка

Г.1.2 Метасимволи та керуючі конструкцію регулярних виразів у MySQL

У додатку надано перелік метасимволів та керуючих конструкцій для регулярних виразів, що підтримуються у СУБД MySQL.

Табл. Г.2. Спецпослідовності регулярних виразів POSIX 1003.2 для MySQL

Позначення	Опис
\t	символ табуляції.
\f	конець файла.
\n	символ переведення строки.
\r	символ возврата каретки.
\\	символ обратного слэша \.

Табл. Г.3. Опис метасимволів регулярних виразів POSIX 1003.2 для MySQL

Позначення	Опис
^	Відповідає початку рядка.
\$	Відповідає кінцю рядка.
.	Відповідає будь-якому символу.
a*	Відповідає будь-якій послідовності з 0 або більше символів «a».
a+	Відповідає будь-якій послідовності з 1 або більше символів «a».
a?	Відповідає 0 або 1 символу «a».
de abc	Відповідає послідовності «de» або «abc».
(abc)*	Відповідає 0 або більше послідовностям «abc».
[a-dX], [^a-dX]	Відповідає будь-якому символу, який є (або не є) являєтся, якщо використовується ^) будь-яким із символів a, b, c, d або X. Символ '-' між двома іншими символами утворює інтервал.

Табл. Г.4. Класи символів регулярних виразів POSIX 1003.2 для MySQL

Позначення	Опис
<code>[:alnum:]</code>	алфавітно цифрові символи.
<code>[:alpha:]</code>	символи алфавіту.
<code>[:blank:]</code>	символи пробілу і табуляції.
<code>[:cntrl:]</code>	керуючі символи.
<code>[:digit:]</code>	десяткові цифри (0-9).
<code>[:graph:]</code>	графічні (видимі) символи.
<code>[:lower:]</code>	символи алфавіту в нижньому регістрі.
<code>[:print:]</code>	графічні або невидимі символи.
<code>[:punct:]</code>	знаки пунктуації.
<code>[:space:]</code>	символи пробілу, табуляції, нового рядка або повернення каретки.
<code>[:upper:]</code>	символи алфавіту в верхньому регістрі.
<code>[:xdigit:]</code>	шістнадцяткові цифри.

Додаток Д

Матеріали до Л/Р №5

Д.1 Функції для роботи з файловою системою

Табл. Д.1. Перелік функцій для роботи з ФС

функція	Опис
<code>basename</code>	Повертає останній компонент імені з вказаного шляху
<code>chgrp</code>	Змінює групу власників файлу
<code>chmod</code>	Змінює режим доступу до файлу
<code>chown</code>	Змінює власника файлу
<code>clearstatcache</code>	Очищає кеш стану файлів
<code>copy</code>	Копіює файл
<code>delete</code>	См.опис функції <code>unlink</code> або <code>unset</code>
<code>dirname</code>	Повертає ім'я батьківського каталогу з зазначеного шляху
<code>disk_free_space</code>	Повертає розмір доступного простору в каталозі або в файловій системі
<code>disk_total_space</code>	Повертає загальний розмір каталогу або розділу файлової системи
<code>diskfreespace</code>	Псевдонім <code>disk_free_space</code>
<code>fclose</code>	Закриває відкритий дескриптор файлу
<code>feof</code>	Перевіряє, чи досягнутий кінець файлу
<code>fflush</code>	Скидає буфер виводу у файл
<code>fgetc</code>	Зчитує символ з файлу
<code>fgetcsv</code>	Читає рядок з файлу і виробляє розбір даних CSV

Продовження

функція	Опис
<code>fgets</code>	Читає рядок з файлу
<code>fgetss</code>	Читає рядок з файлу і відкидає HTML-теги
<code>file_exists</code>	Перевіряє наявність вказаного файлу або каталогу
<code>file_get_contents</code>	Читає вміст файлу в рядок
<code>file_put_contents</code>	Пише рядок в файл
<code>file</code>	Читає вміст файлу і поміщає його в масив
<code>fileatime</code>	Повертає час останнього доступу до файлу
<code>filectime</code>	Повертає час зміни індексного дескриптора файлу
<code>filegroup</code>	Отримує ідентифікатор групи файлу
<code>fileinode</code>	Повертає індексний дескриптор файлу
<code>filemtime</code>	Повертає час останньої зміни файлу
<code>fileowner</code>	Повертає ідентифікатор власника файлу
<code>fileperms</code>	Повертає інформацію про права на файл
<code>filesize</code>	Повертає розмір файлу
<code>filetype</code>	Повертає тип файлу
<code>flock</code>	Переносима консультативна блокування файлів
<code>fnmatch</code>	Перевіряє збіг імені файлу з шаблоном
<code>fopen</code>	Відкриває файл або URL
<code>fpasssthru</code>	Виводить всі залишилися дані з файлового покажчика
<code>fputcsv</code>	Форматує рядок у вигляді CSV і записує його в файловий покажчик
<code>fputs</code>	Псевдонім <code>fwrite</code>
<code>fread</code>	бінарних-безпечне читання файлу

Продовження

функція	Опис
<code>fscanf</code>	Обробляє дані з файлу відповідно до формату
<code>fseek</code>	Встановлює зсув у файловому покажчику
<code>fstat</code>	Отримує інформацію про файл використовуючи відкритий файловий покажчик
<code>ftell</code>	Повідомляє поточну позицію читання/запису файлу
<code>ftruncate</code>	Врізає файл до вказаної довжини
<code>fwrite</code>	бінарно-безпечний запис в файл
<code>glob</code>	Знаходить файлові шляхи, що збігаються з шаблоном
<code>is_dir</code>	Визначає, чи є ім'я файлу директорією
<code>is_executable</code>	Визначає, чи є файл виконуваним
<code>is_file</code>	Визначає, чи є файл звичайним файлом
<code>is_link</code>	Визначає, чи є файл символічним посиланням
<code>is_readable</code>	Визначає наявність файлу і доступний він для читання
<code>is_uploaded_file</code>	Визначає, чи був файл завантажений за допомогою HTTP POST
<code>is_writable</code>	Визначає, чи доступний файл для запису
<code>is_writeable</code>	Псевдонім <code>is_writable</code>
<code>lchgrp</code>	Змінює групу, якій належить символічне посилання
<code>lchown</code>	Змінює власника символічного посилання
<code>link</code>	Створює жорстке посилання
<code>linkinfo</code>	Повертає інформацію про посилання

Продовження

функція	Опис
<code>lstat</code>	Повертає інформацію про фото або символічної посиланню
<code>mkdir</code>	Створює директорію
<code>move_uploaded_file</code>	Переміщає завантажений файл у нове місце
<code>parse_ini_file</code>	Обробляє конфігураційний файл
<code>parse_ini_string</code>	Розбирає рядок конфігурації
<code>pathinfo</code>	Повертає інформацію про шлях до файлу
<code>pclose</code>	Закриває файловий показчик процесу
<code>popen</code>	Відкриває файловий показчик процесу
<code>readfile</code>	Виводить файл
<code>readlink</code>	Повертає файл, на який вказує символічне посилання
<code>realpath_cache_get</code>	Отримує записи з кешу реального шляху
<code>realpath_cache_size</code>	Отримує розмір кеша реального шляху
<code>realpath</code>	Повертає канонізований абсолютний шлях до файлу
<code>rename</code>	Перейменовує файл або директорію
<code>rewind</code>	Скидає курсор у файлового показчика
<code>rmdir</code>	Видаляє директорію
<code>set_file_buffer</code>	Псевдонім <code>stream_set_write_buffer</code>
<code>stat</code>	Повертає інформацію про файл
<code>symlink</code>	Створює символічну посилання
<code>tempnam</code>	Створює файл з унікальним ім'ям
<code>tmpfile</code>	Створює тимчасовий файл
<code>touch</code>	Встановлює час доступу і модифікації файлу
<code>umask</code>	Змінює поточну <code>umask</code>
<code>unlink</code>	Видаляє файл

Д.2 Параметри функції «fopen()»

Табл. Д.2. Другий параметр функції «fopen()»

Параметр	Опис
R	Відкриває файл тільки для читання; поміщає покажчик в початок файлу.
R+	Відкриває файл для читання і запису; поміщає покажчик в початок файлу.
W	Відкриває файл тільки для запису; поміщає покажчик в початок файлу і обрізає файл до нульової довжини. Якщо файл не існує — намагається його створити.
W+	Відкриває файл для читання і запису; поміщає покажчик в початок файлу і обрізає файл до нульової довжини. Якщо файл не існує — намагається його створити.
A	Відкриває файл тільки для запису; поміщає покажчик в кінець файлу. Якщо файл не існує — намагається його створити.
A+	Відкриває файл для читання і запису; поміщає покажчик в кінець файлу. Якщо файл не існує — намагається його створити.
X	Створює і відкриває тільки для запису; поміщає покажчик в початок файлу. Якщо файл вже існує, виклик <code>fopen()</code> закінчиться невдачею, поверне <code>FALSE</code> і видасть помилку <code>E_WARNING</code> . Якщо файл не існує, спробує його створити.
X+	Створює і відкриває для читання і запису інакше поверне <code>FALSE</code> .

Продовження

Параметр	Опис
<code>c</code>	Відкриває файл тільки для запису. Якщо файл не існує, то він створюється. Якщо ж файл існує, то він не обрізається, і виклик цієї функції не викликає помилку. Показчик на файл буде встановлений на початок файлу.
<code>c+</code>	Відкриває файл для читання і запису інакше функція має ту ж поведінку, що і з використанням « <code>c</code> ».

Д.3 Функції для роботи з каталогами

Табл. Д.3. Перелік функцій для роботи з каталогами

Функція	Опис
<code>chdir</code>	Змінює каталог
<code>mkdir</code>	Створює каталог
<code>rmdir</code>	Видаляє каталог
<code>is_dir</code>	Перевіряє чи є об'єкт каталогом
<code>chroot</code>	Змінює кореневий каталог
<code>closedir</code>	Звільняє дескриптор каталогу
<code>dir</code>	Повертає екземпляр класу <code>Directory</code>
<code>getcwd</code>	Отримує ім'я поточного робочого каталога
<code>opendir</code>	Відкриває дескриптор каталогу
<code>readdir</code>	Отримує елемент каталогу за його дескриптору
<code>rewinddir</code>	Скинути дескриптор каталогу
<code>scandir</code>	Отримує список файлів і каталогів, розташованих по зазначеному шляху

Додаток Е

Матеріали до Л/Р №6

Е.1 Налаштування сесії у PHP

Табл. Е.1. Перелік параметрів у файлі `php.ini`

Опція	Опис
<code>session.save_path</code>	визначає, де на сервері зберігатимуться дані сесії.
<code>session.use_cookies</code>	визначає, чи використовувати cookies при роботі з сесіями.
<code>session.cookie_lifetime</code>	задає тривалість життя cookies в секундах.
<code>session.name</code>	визначає ім'я сесії
<code>session.auto_start</code>	дозволяє автоматично запускати сесії
<code>session.serialize_handler</code>	задає спосіб кодування даних сесії
<code>session.cache_expire</code>	визначає, через скільки хвилин застаріває документ в кеші

Ім'я сесії `session.name` за умовчанням встановлюється як `PHPSESSID` і використовується в cookies як ім'я змінної, в якій зберігається ідентифікатор сесії. Автоматичний запуск сесій за замовчуванням відключений, але його можна задати, зробивши значення `session.auto_start` рівним «1». Для кодування даних сесії по замовчуванню використовується `php`. Старіння даних, збережених в кеші, відбувається через 180 хвилин.

Е.2 Повний перелік змінних у «`php.ini`»

Табл. Е.2. Повний параметрів у файлі «php.ini» та значень за замовчуванням

Опція	Значення
<code>session.save_path</code>	<code>""</code>
<code>session.name</code>	<code>"PHPSESSID"</code>
<code>session.save_handler</code>	<code>"files"</code>
<code>session.auto_start</code>	<code>"0"</code>
<code>session.gc_probability</code>	<code>"1"</code>
<code>session.gc_divisor</code>	<code>"100"</code>
<code>session.gc_maxlifetime</code>	<code>"1440"</code>
<code>session.serialize_handler</code>	<code>"php"</code>
<code>session.cookie_lifetime</code>	<code>"0"</code>
<code>session.cookie_path</code>	<code>"/"</code>
<code>session.cookie_domain</code>	<code>""</code>
<code>session.cookie_secure</code>	<code>""</code>
<code>session.cookie_httponly</code>	<code>""</code>
<code>session.use_cookies</code>	<code>"1"</code>
<code>session.use_only_cookies</code>	<code>"1"</code>
<code>session.referer_check</code>	<code>""</code>
<code>session.entropy_file</code>	<code>""</code>
<code>session.entropy_length</code>	<code>"0"</code>
<code>session.cache_limiter</code>	<code>"nocache"</code>
<code>session.cache_expire</code>	<code>"180"</code>
<code>session.use_trans_sid</code>	<code>"0"</code>
<code>session.bug_compat_42</code>	<code>"1"</code>
<code>session.bug_compat_warn</code>	<code>"1"</code>
<code>session.hash_function</code>	<code>"0"</code>
<code>session.hash_bits_per_character</code>	<code>"4"</code>
<code>url_rewriter.tags</code>	<code>"a=href, area=href, frame=src, form=, fieldset="</code>
<code>session.upload_progress.enabled</code>	<code>"1"</code>

Опція	Значення
<code>session.upload_progress.cleanup</code>	"1"
<code>session.upload_progress.prefix</code>	"upload_progress_"
<code>session.upload_progress.name</code>	"PHP_SESSION_UPLOAD_PROGRESS"
<code>session.upload_progress.freq</code>	"1%"
<code>session.upload_progress.min_freq</code>	"1"

Е.3 Опис функцій для роботи з сесіями

Нижче дано перелік функцій, що необхідні для роботи із сесіями. Деякі функції налаштування дублюють функціонал опцій у файлі «php.ini», на такі функції буде звернено увагу.

`session_destroy()`

Функція скасовує дію `session_start()`. Викликати потрібно після виклику `session_start()`. Можна застосовувати, щоб знищувати сесію користувача, а потім відразу викликати в програмі вдруге `session_start()`, вийти зовсім новий відвідувач з новим ідентифікатором і чистої сесією.

`session_name()` або `session.name`

PHP для зберігання ідентифікатора використовує якусь змінну. В cookies записують змінну: значення змінної — ідентифікатор, назва змінної — PHPSESSID. PHPSESSID — це назва, яку використовують за замовчуванням. Її можна перейменувати у щось більш коротке, наприклад SID. А для цього треба замінити параметр `session.name` на значення SID:

можна замінити php.ini: `session.name = SID`

можна створити .htaccess файл у каталозі з скриптами вашого сайту і помістити рядок `php_value session.name SID`

Щоб отримати назву змінної, яку використовують для зберігання ідентифікатора, треба скористатися функцією без параметра `$name = session_name()`.

Щоб встановити таку змінну в довільну назву, скористайтеся функцією з параметром `session_name("МояНазва")`. Зрозуміло, якщо Вам чомусь знадобилося змінити ім'я змінної для cookies за допомогою цієї функції, то її треба викликати перед `session_start()` або `session_register()`.

`session_module_name()` або **`session_save_handler`**

Отримати або встановити поточний модуль сесії. PHP може зберігати сесії різному вигляді. За замовчуванням — в файлах.

`session_save_path()` або **`session.save_path`**

Отримати або встановити каталог, в якому будуть зберігатися файли сесії. `$path = session_save_path()` — отримати. `session_save_path("/mydir/temp")` — встановити. Параметр `session.save_path` можна задавати в `php.ini` або «`.htaccess`».

`session_id()`

Отримати або встановити ідентифікатор відвідувача (128-бітове число, представлене у вигляді рядка в 32 байти). У нормальних умовах Вам не потрібно встановлювати номер сесії. Але якщо Ви хочете для всіх своїх відвідувачів використовувати одну сесію, то перед `session_start()` подумайте будь-яке ім'я (наприклад, `session_id("my_name")`) або отримаєте справжній ідентифікатор іншим чином. Виклик функції без параметрів поверне Вам поточний номер сесії (в такому випадку — викликати після `session_start()`).

`session_register()`

Зареєструвати одну або кілька змінних. Передавати треба імена змінних, а не їх значення: `session_register("змінна1", "змінна2", ...)`.

`session_unregister()`

Видалити з сесії необхідну змінну. Можна передати тільки одне ім'я змінної за один виклик функції.

`session_unset()`

Очистити всі змінні сесії. У відмінності від `session_destroy()` сесія і ідентифікатор залишається.

`session_is_registered()`

Перевірити, зареєстрована якась змінна в поточній сесії чи ні:

```
if (session_is_registered("змінна")) ...
```

`session_get_cookies_params()` і

`session_set_cookies_params()`

Отримати інформацію про cookies, що зберігає змінну з ідентифікатором сесії, можна наступним чином:

```
echo "<pre> session INFO: \ n";  
print_r (session_get_cookies_params());
```

Так можна дізнатися про час життя змінної, домен та шляхи cookies.

Функцією `session_set_cookies_params()` можна перевстановити параметри (хоча всі ці параметри задані в `php.ini`).

`session_decode()` і **`session_encode()`**

Декодування закодованих рядків у сесії і кодування змінних у рядок сесії.

`session_set_save_handler()`

Можливо, Вас не влаштовують варіанти зберігання сесій, що пропонуються в PHP. Може бути, ви хочете зберігати сесії в простих текстових файлах, щоб їх можна було легко редагувати. Тоді Вам потрібно створити декілька функцій, що відповідають за функціонування сесій і передати назви цих функцій в `session_set_save_handler()`.

Додаток Є

Матеріали до Л/Р №7

Є.1 Синтаксис команди CREATE TABLE

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS]
```

```
tbl_name [(create_definition, ...)]
```

```
[Table_options] [select_statement]
```

create_definition:

```
col_name type [NOT NULL | NULL] [DEFAULT default_value]
```

```
[AUTO_INCREMENT] [PRIMARY KEY] [reference_definition]
```

```
a6o PRIMARY KEY (index_col_name, ...)
```

```
a6o KEY [index_name] (index_col_name, ...)
```

```
a6o INDEX [index_name] (index_col_name, ...)
```

```
a6o UNIQUE [INDEX] [index_name] (index_col_name, ...)
```

```
a6o FULLTEXT [INDEX] [index_name] (index_col_name, ...)
```

```
a6o [CONSTRAINT symbol] FOREIGN KEY [index_name]
```

```
(index_col_name, ...) [Reference_definition]
```

```
a6o CHECK (expr)
```

type:

```
TINYINT [(length)] [UNSIGNED] [ZEROFILL]
```

```
a6o SMALLINT [(length)] [UNSIGNED] [ZEROFILL]
```

```
a6o MEDIUMINT [(length)] [UNSIGNED] [ZEROFILL]
```

```
a6o INT [(length)] [UNSIGNED] [ZEROFILL]
```

```
a6o INTEGER [(length)] [UNSIGNED] [ZEROFILL]
```

```
a6o BIGINT [(length)] [UNSIGNED] [ZEROFILL]
```

```
a6o REAL [(length, decimals)] [UNSIGNED] [ZEROFILL]
```

```
a6o DOUBLE [(length, decimals)] [UNSIGNED] [ZEROFILL]
```



```
a6o FLOAT [(length, decimals)] [UNSIGNED] [ZEROFILL]
a6o DECIMAL (length, decimals) [UNSIGNED] [ZEROFILL]
a6o NUMERIC (length, decimals) [UNSIGNED] [ZEROFILL]
a6o CHAR (length) [BINARY]
a6o VARCHAR (length) [BINARY]
a6o DATE
a6o TIME
a6o TIMESTAMP
a6o DATETIME
a6o TINYBLOB
a6o BLOB
a6o MEDIUMBLOB
a6o LONGBLOB
a6o TINYTEXT
a6o TEXT
a6o MEDIUMTEXT
a6o LONGTEXT
a6o ENUM (value1, value2, value3, ...)
a6o SET (value1, value2, value3, ...)
```

index_col_name:

```
col_name [(length)]
```

reference_definition:

```
REFERENCES tbl_name [(index_col_name, ...)]
    [MATCH FULL | MATCH PARTIAL]
    [ON DELETE reference_option]
    [ON UPDATE reference_option]
```

reference_option:

```
RESTRICT | CASCADE | SET NULL |
```

NO ACTION | SET DEFAULT

table_options:

TYPE = {BDB | HEAP | ISAM | InnoDB |
MERGE | MRG_MYISAM | MYISAM}

a6o AUTO_INCREMENT = #

a6o AVG_ROW_LENGTH = #

a6o CHECKSUM = {0 | 1}

a6o COMMENT = "string"

a6o MAX_ROWS = #

a6o MIN_ROWS = #

a6o PACK_KEYS = {0 | 1 | DEFAULT}

a6o PASSWORD = "string"

a6o DELAY_KEY_WRITE = {0 | 1}

a6o ROW_FORMAT = {default | dynamic |
fixed | compressed}

a6o RAID_TYPE = {1 | STRIPED | RAID0}

RAID_CHUNKS = # RAID_CHUNKSIZE = #

a6o UNION = (table_name, [table_name ...])

a6o INSERT_METHOD = {NO | FIRST | LAST}

a6o DATA DIRECTORY = "абсолютный шлях до каталогу"

a6o INDEX DIRECTORY = "абсолютный шлях до каталогу"

select_statement:

[IGNORE | REPLACE] SELECT ...

(Будь-який коректний вираз SELECT)

Є.2 Синтаксис команды ALTER TABLE

Додаток Ж
Матеріали до Л/Р №8

Додаток 3

Матеріали до Л/Р №9

Додаток И

Матеріали до Л/Р №10

dgfwasg [1] jsfhgsfthsdhgdgfwasg

Бібліографія

- [1] Котеров Д., Костарев А. РНР 5 В Подлиннике. — 3-е edition. — В подлиннике : БХВ-Петербург, 2006. — Р. 1104.