МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНІЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ

до самостійної роботи студентів та виконання контрольної роботи з дисципліни

ТЕХНОЛОГІЇ ТА СТАНДАРТИ ВЗАЄМОДІЇ У ГЛОБАЛЬНИХ МЕРЕЖАХ

для студентів 4 курсу заочної форми навчання

Освітнього рівню – «Бакалавр»

Напрям підготовки – «Комп'ютерні науки»

Узгоджено
Завідувач навчально-
консультаційного центру
_____ Волошина О.В.

Затверджено
на засіданні кафедри інформатики
Протокол №__ від _____
Зав. кафедри _____ Мещеряков В.І.

ОДЕСА 2016

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНІЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ

до самостійної роботи студентів та виконання контрольної роботи з дисципліни

ТЕХНОЛОГІЇ ТА СТАНДАРТИ ВЗАЄМОДІЇ У ГЛОБАЛЬНИХ МЕРЕЖАХ

для студентів 4 курсу заочної форми навчання

Освітнього рівню – «Бакалавр»

Напрям підготовки – «Комп'ютерні науки»

ОДЕСА 2016

МЕТОДИЧНІ ВКАЗІВКИ до самостійної роботи студентів та виконання контрольної роботи дисципліни “Технології та стандарти взаємодії у глобальних мережах” для студентів заочної форми навчання, напряму підготовки – комп’ютерні науки.

Укладачі:

ГНАТОВСЬКА Г.А., к.т.н., доцент кафедри інформатики

ВОЛОЩУК Л.А., к.т.н., доцент кафедри інформатики

ВОХМЕНЦЕВА Т.Б., старший викл. кафедри інформатики

ЗМІСТ

ПЕРЕДМОВА

Методичні вказівки призначені для студентів заочного факультету ОДЕКУ до самостійної роботи студента та виконанню контрольної роботи з дисципліни «Технології та стандарти взаємодії у глобальних мережах» бакалаврів з галузі знань “Інформатика та обчислювальна техніка”, напряму підготовки 6.050101 “Комп’ютерні науки”.

Мета методичних вказівок – забезпечити отримання студентами теоретичних знань і практичних навичок щодо сучасних технологій, архітектур та стандартів організації доступу до даних у глобальних мережах. Отримані студентами знання та вміння можуть використовуватися при здійсненні дипломного проектування інформаційних Інтернет-систем.

Поєднання Інтернет–технологій та технологій СУБД як способу організації доступу до даних має ряд безумовних недоліків, та вимагає не лише знання цих технологій, але й вміння аналізувати та обирати оптимальні архітектури подібних інформаційних систем.

Ці методичні вказівки містять рекомендації по вивченню розділів дисципліни для виконання контрольної роботи, контрольні запитання для самоперевірки, приклад виконання контрольної роботи та завдання. В методичних вказівках розглядаються питання, які відповідають навчальній програмі дисципліни.

1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО ДИСЦИПЛІНУ

1.1 Мета дисципліни та її місце у навчальному процесі

Дисципліна «Технології та стандарти взаємодії у глобальних мережах» входить до складу варіативної частини навчального плану підготовки бакалаврів з галузі знань “Інформатика та обчислювальна техніка”, напряму підготовки 6.050101 “Комп’ютерні науки”. Викладається відповідно до освітньо-професійної програми, освітньо-кваліфікаційної характеристики та навчального плану підготовки бакалаврів.

Курс присвячений вивченню сучасного рівня WEB технологій та принципів побудови Internet-додатків з використанням сучасних технологій та стандартів взаємодії у глобальних мережах. Це можуть бути не лише відкриті Інтернет-системи, але й “закриті” корпоративні автоматизовані системи управління, розподілені на великі території та відстані. Істотну відзнаку таких систем складає здійснення віддаленого доступу до сховищ інформації – баз даних. Поєднання Інтернет–технологій та технологій СУБД як способу організації доступу до даних має ряд безумовних недоліків, та вимагає не лише знання цих технологій, але й вміння аналізувати та обирати оптимальні архітектури подібних інформаційних систем.

Мета дисципліни – полягає у ознайомленні з існуючими архітектурами, технологіями та стандартами побудови WEB-систем, отриманні практичних навичок їх створення для функціонування у глобальних мережах.

Для підготовки курсу використовано велика низка новітніх наукових та навчально-методичних робіт, як зарубіжних так і вітчизняних фахівців у галузі програмування для Інтернет, характеристики архітектур інформаційних систем, технологій та стандартів програмного забезпечення для розробки інформаційних WEB-систем.

1.2 Зміст дисципліни

1. *Архітектура та сучасні технології інформаційних Інтернет-систем у глобальних мережах.*

   Література [1 с.3–34, 2 с.76–98, 3 с.18–58, 4 с. 215–268]

Принципи функціонування інформаційних Інтернет-систем. Обґрунтування вибору архітектур інформаційних Інтернет-систем. Internet-технологій в якості платформи для WEB-додатки з базами даних. Огляд різних архітектур інформаційних систем з базами даних. Налаштування Web-сервера і MySQL-сервера для розробки Web-систем. Інформаційні Інтернет-системи з розподіленою та централізованою веб-орієнтованою архітектурою.

2. *Комплекс стандартів XML.*

   Література [1 с.35–50, 2 с.115–201, 3 с.346-382]

XML-дані та XML-документи у середовищі глобальної мережі Інтернет. Передумови та джерела технології XML. Призначення та функціональні можливості мови XML. Організацію та функціональні можливості платформи XML. Стандарти Веб-сервісів, спадкоємність з технологіями HTML. Технології семантичного WEB. Основні принципи розробки WEB-додатків на PHP і MySQL. Проектування та створення інформаційної Web-системи з базою даних.

3. *Особливості XML-даних та їх моделювання*

   Література [1 с. 51–57, 2 с.346–376, 3 с.116-128]

Створення переносимих XML-документів у середовищі глобальної мережі Інтернет. Структуровані та слабоструктуровані XML-дані. Багаторівневе представлення XML-даних. Засоби розробки Web-додатків з модулями розширення Web-сервера та з модулями розширення Web-оглядача.

4. *XML-орієнтовані бази даних*

   Література [1 с.58–67, 3 с.446–476, 5 с.216-248]

Публікація баз даних з використанням XML. Функціональні можливості XML-орієнтованих СУБД. Формування XML-документа на основі бази даних. Класифікація XML-орієнтованих СУБД. Напрямки та перспективи розвитку XML-орієнтованих баз даних.

При вивченні даної дисципліни використовується наступна навчальна література:

Основная

1. Конспект лекцій «Технології та стандарти взаємодії у глобальних мережах» Г.А. Гнатовська, Л.А. Волощук – ОДЕКУ, 2016. – 67с.
2. Грейвс Марк. Проектирование баз данных на основе XML. Пер. с англ. – М. : Издательский дом "Вильямс", 2002. – 640 с.

3. Филимонов А. Ю. Протоколы Интернета. – СПб.: БХВ-Петербург, 2003. – 528 с.
4. Дюбуа Поль. MySQL Пер.с анг.: Уч. пос.- М.: Издательский дом «Вильямс», 2004г., – 1056с.
5. Колисниченко Д.Н. Самоучитель PHP 5. –СПб.: Наука и Техника, 2006г., – 576 с.
6. www.library-odeku.16mb.com.

Додаткова

1. Томсон Лаура, Веллинг Люк. Разработка WEB-приложений на PHP и MySQL:- Пер.с анг.- К.:Издательство «ДиаСофт», 2003г., – 672с.
2. Благодатских, В. А. Стандартизация разработки программных средств : учеб. пособие / В. А. Благодатских, В. А. Волнин, К. Ф. Поскакалов ; под ред. О. С. Разумова. – М.: Финансы и статистика, 2003. – 288 с.
3. Аткинсон Леон, Сураски Зеев. PHP5. Библиотека профессионала, 3-е издание. М.: Издательский дом «Вильямс», 2006. – 944 с.
4. 7. Питер Уэйнрайт. Apache для профессионалов. – М.: Издательство «Лори», 2001г. – 472с.

1.3 Перелік знань та вмінь

Вивчення дисципліни базується на знаннях, отриманих в курсах програмування, «WEB-технології та WEB-дізайн», «Технологія створення програмних продуктів», «Розробка прикладних СБД»

***Знати**:*

1. *Архітектура та сучасні технології інформаційних Інтернет-систем у глобальних мережах*
   - Принципи функціонування та обґрунтування вибору архітектур інформаційних Інтернет-систем.
   - Архітектури територіально-розподілених WEB–додатків, що публікують бази даних.
   - Інформаційні Інтернет-системи з розподіленою та централізованою веб-орієнтованою архітектурою.
2. *Комплекс стандартів XML*
   - XML-дані та XML-документи у середовищі глобальної мережі Інтернет;
   - Передумови та джерела технології XML.
   - Призначення та функціональні можливості мови XML
   - Організацію та функціональні можливості платформи XML
   - Стандарти Веб-сервісів, спадкоємність з технологіями HTML
   - Технології семантичного WEB;

3. Особливості XML-даних та їх моделювання

- XML-дані у середовищі глобальної мережі Інтернет
- створення переносимих XML-документів у середовищі глобальної мережі Інтернет
- Структуровані та слабоструктуровані XML-дані
- Багаторівневе представлення XML-даних
- Метадані XML.

4. XML-орієнтовані бази даних

- Класифікація XML-орієнтованих СУБД
- Функціональні можливості XML-орієнтованих СУБД.
- Напрямки та перспективи розвитку XML-орієнтованих баз даних

Вміти:

- обрати архітектуру технології реалізації Web-системи, використовуючи поняття архітектури та знання основних

принципів побудови інформаційних систем у глобальній мережі INTERNET;

- спроектувати базу даних в даної предметної області та створити її в СУБД MySQL;
- розробити алгоритм та написати програми на мові PHP для забезпечення формування динамічних Web-сторінок згідно з запитом клієнта та вмістом БД.
- створювати переносимі документи для глобальних мереж, використовуючи мову XML-як універсального способу обміну даними.
- здійснювати формування XML-документу на основі бази даних
- виконувати обробка XML-документів та налаштовувати зв'язок XML-документа з HTML-сторінкою

1.4 Контрольні заходи

Вивчення дисципліни «Технології та стандарти взаємодії у глобальних мережах» для студентів четвертого курсу заочної форми навчання складається з лекційних та лабораторних занять, самостійної роботи студентів (СРС) по засвоєнню теоретичного курсу та виконання контрольної роботи.

Контроль самостійної роботи студентів заочної форми навчання здійснюється шляхом перевірки контрольних робіт, які передаються на кафедру студентами у встановлені деканатом терміни, а також шляхом опитування під час лабораторних занять.

1.5 Організація самостійної роботи студентів

Основною формою навчання студента є самостійна робота над навчальним матеріалом. СРС повинна сприяти активізації творчого мислення студентів, підвищенню самостійності студентів та індивідуалізації процесу навчання. СРС складається з наступних елементів: вивчення матеріалу

конспекту лекцій та рекомендованих підручників, виконання лабораторних та контрольної роботи.

В процесі вивчення курсу студенти виконують контрольну роботу, що містить як теоретичні запитання, так і практичні завдання. Студент повинен виконати практичні завдання. По результатам виконання контрольної роботи студенти складають звіт, який містить формулювання завдань та відповіді. Студент може звертатися до викладача з питаннями для одержання письмової чи усної консультації. Однак студент повинен пам’ятати, що допомога викладача виявиться досить ефективною лише за умови систематичної і завзятої роботи студента.

Варіанти завдань вибираються як номер останньої цифри залікової книжки, якщо ця цифра дорівнює нулю, то обирають 10 варіант.

Загальні поради до СРС:

- зміст кожної теми дисципліни вивчається за допомогою наведеного переліку навчальної літератури;
- після засвоєння кожної теми необхідно відповісти на запитання для самоперевірки, що надаються в цих методичних вказівках наприкінці кожної теми;
- завдання з контрольної роботи необхідно виконувати згідно наведених вимог;
- якщо виникли питання при опрацюванні теоретичного матеріалу або при виконанні контрольної роботи, студент може звернутися до викладача, який читав установчі лекції, письмово на адресу ОДЕКУ звичайною поштою або на адресу кафедри інформатики ОДЕКУ електронною поштою (kaf.inf.odeku@gmail.com). У темі електронного листа необхідно вказати назву дисципліни.

2 ТЕОРЕТИЧНІ МАТЕРІАЛИ ДО САМОСТІЙНОЇ РОБОТИ СТУДЕНТА ТА ВИКОНАННЯ КОНТРОЛЬНОЇ РОБОТИ

Загальні поради по вивченню дисципліни «Технології та стандарти взаємодії у глобальних мережах» та виконанню контрольної роботи.

У процесі самостійного вивчення курсу студент повинен керуватися його програмою та вивчити за конспектом лекцій та літературою, що рекомендована викладачем, відповідний теоретичний матеріал:

1. Архітектура та сучасні технології інформаційних Інтернет-систем у глобальних мережах;
2. Вибір і обґрунтування архітектури інформаційної Інтернет-системи
3. Сучасні технології розробки Web-додатків
4. Використання PHP і СУБД My SQL для створення WEB-систем
5. Комплекс стандартів XML;
6. Особливості XML-даних та їх моделювання;
7. XML-орієнтовані бази даних.

Ці методичні вказівки містять перелік тем, які повинні бути розглянуті, завдання, які є необхідними для рішення поставленої задачі, контрольні питання та завдання в межах виконання контрольної роботи, які охоплюють наступні теми: архітектури інформаційних Internet систем, використання MySQL та PHP для проектування та реалізації інформаційної Web-системи з базою даних, технології сучасних Web – додатків та публікування баз даних в Internet, публікація баз даних з використанням мови XML, що дають можливість студентам найкраще закріпити теоретичні знання, отримані на лекціях з дисципліни.

2.1 Архітектура та сучасні технології інформаційних Інтернет-систем у глобальних мережах

Публікація баз даних в Internet здійснюється за допомогою технологій, в яких реалізується можливість відображення на web-сторінках інформації з баз даних, що зберігаються на web-сервері. Об'єднання Internet технології та технології СУБД як спосіб доступу до даних має свої плюси:

– уніфікований підхід для доступу в Internet – єдність програмних браузерів, що дозволяє стандартизувати користувальницький інтерфейс;

– обмін інформацією між браузером і web-сервером здійснюється за допомогою переносних незалежного протоколу http, що дозволяє спростити і стандартизувати представлення даних;

– багаторівнева архітектура мережі Internet має стандартні способи підвищення можливостей браузера і web-сервера (доступ до сервісів Internet з корпоративних мереж, обмін між СУБД, що працюють на різних платформах);

– застосування СУБД у середовищі Internet дозволяє ввести стандарти, організувати якісне зберігання, захист інформації, управління транзакціями за допомогою SQL.

Інформаційна система з базою даних в залежності від відносного розташування програми клієнта і бази даних (БД) може бути локальною – база даних і додаток користувача знаходяться на одному комп'ютері і розподіленою – база даних віддалених від клієнтських додатків в рамках комп'ютерної мережі.

Залежно від ширини доступу визначають однокористувацький і багатокористувацький режими роботи.

2.2 Вибір і обґрунтування архітектури інформаційної Інтернет-системи

Архітектури інформаційних систем (ІС), що використовують багатокористувацькі СУБД, в залежності від реалізованих технологій можна розбити на наступні типи:

– архітектура «Телеобробка»;

– архітектура «Файловий сервер»;

– архітектура «Клієнт-сервер»;

Архітектура «Телеобробка» – це архітектура 70-80-х років, в якій один комп'ютер з одним центральним процесором був з'єднаний з багатьма не інтелектуальними терміналами (рис.2.1).

Основне навантаження припадало на центральну ЕОМ – мейнфрейм, який виконував дії прикладних програм СУБД, а також віддалених терміналів.

В даний час дорогі мейнфрейми замінені мережами персональних ЕОМ, що призвело до появи двох архітектур ІС – архітектури з файловим сервером і клієнт-серверній.

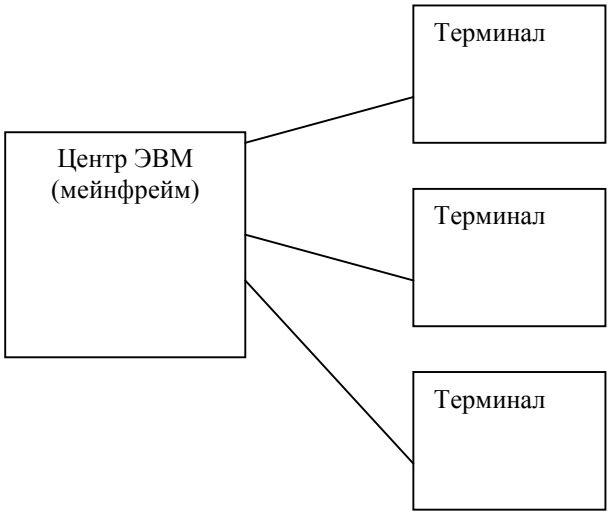

Рисунок 2.1. – Архітектура «Телеобробка»

Архітектура «Файловий сервер» – зароджувалась в найпростішої локальної комп'ютерної мережі. Файли бази даних розташовувалися на сервері

локальної мережі, а користувальницькі додатки і сама СУБД знаходилися на окремих робочих станціях і зверталися до файлів даних тільки в міру необхідності (рис.2.2.).

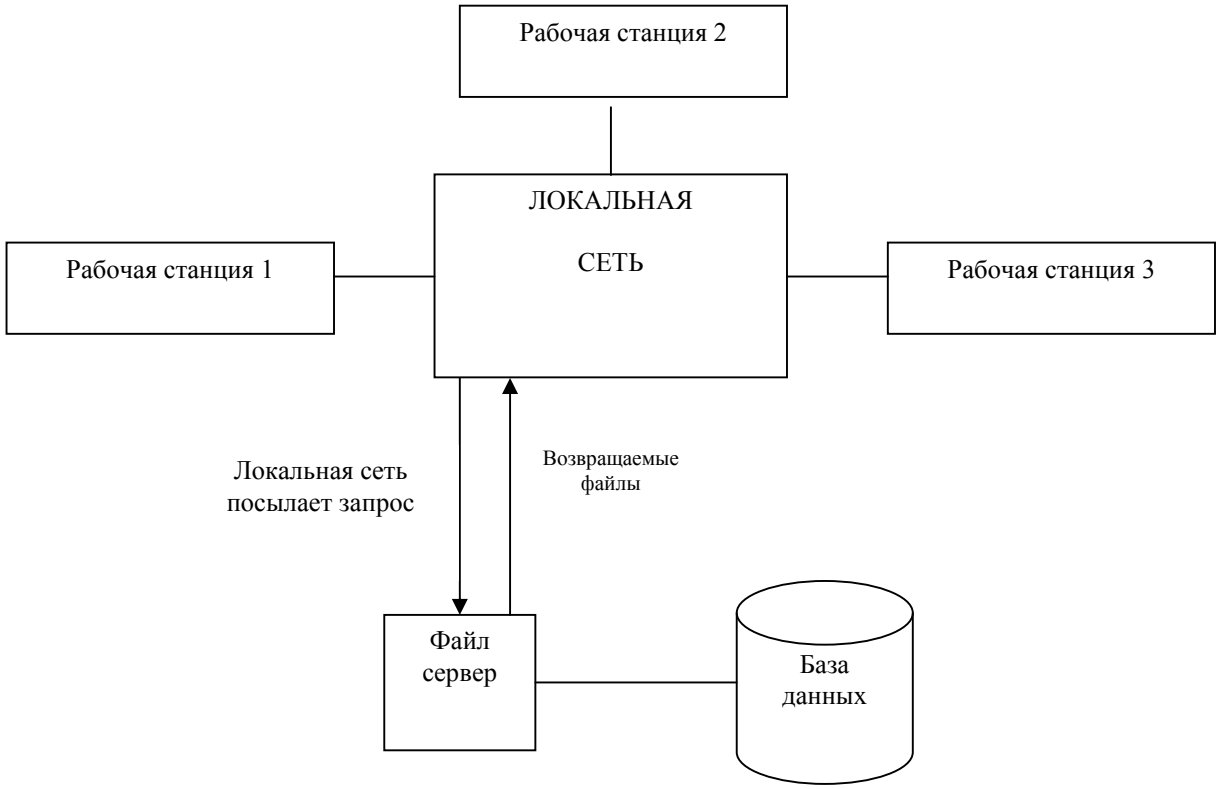

Рисунок 2.2. – Архітектура «Файловий сервер»

Переваги даної архітектури – простота реалізації.

Недоліки – великий обсяг мережевого трафіку, так як по мережі пересилаються файли даних для реалізації їх обробки на робочій станції, при цьому на кожній робочій станції повинна знаходиться повна копія СУБД.

Архітектура «Дволанковий клієнт-сервер» – передбачає, що база даних (БД) розміщена на сервері локальної комп'ютерної мережі.

На робочих станціях розміщують клієнтське додаток, що формує і відсилає запити віддаленого сервера з БД, який забезпечує виконання запиту і видачу клієнту результату.

Вся обробка інформації відбувається на віддаленому сервері (рис.2.3).

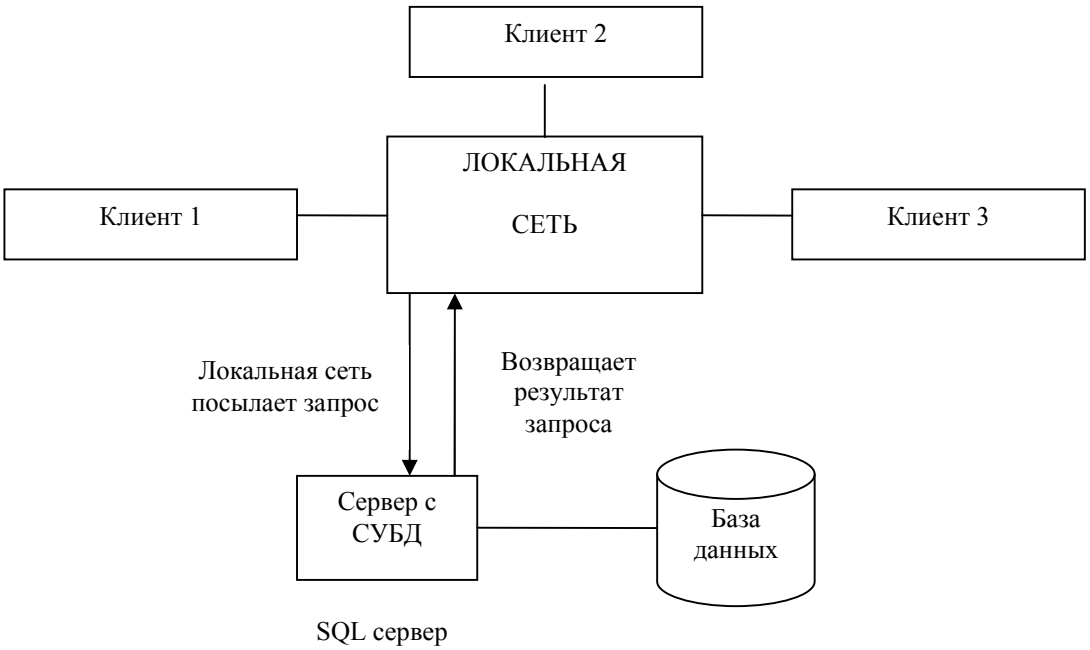

Рисунок 2.3. – Архітектура «Дволанковий клієнт-сервер»

Функції, що виконуються клієнтом і сервером:

клієнт:

- управляє користувача інтерфейсом;
- програмує і перевіряє синтаксис виданого користувачем запиту;
- генерує запит до бази даних і передає його серверу;
- відображає дані, отримані користувачем;

сервер:

- приймає і відображає запити до бази даних з боку клієнта;
- перевіряє повноваження користувача;
- гарантує обмеження цілісності;
- виконує запити поновлення і повертає результати клієнту;
- підтримує системний каталог;
- забезпечує паралельність доступу до бази даних;
- забезпечує управління відновленням.

У дволанковій архітектурі «Клієнт-сервер» клієнтське додатки називаються «сильним» або «товстим» клієнтом. У ролі SQL-сервера сьогодні виступають такі СУБД як Oracle, Informix, Microsoft SQL-server, DB2, InterBase.

Переваги:

– зменшення навантаження на мережу, тому– передається не вся інформація, а результати запиту;

– підвищення рівня захисту інформації;

– підвищення загальної продуктивності системи.

Архітектура «Триланкової клієнт-сервер» була запропонована в 1995 році (рис.2.4.).

Рисунок 2.4. – Архітектура «Триланковий клієнт-сервер»

Це подальше розширення триланкової архітектури, при якій функціональна частина колишнього «товстого» клієнта ділиться на дві частини:

перша – «тонкий», не інтелектуальний клієнт на робочій станції;

друга – середній рівень, керуючий всієї предметної логікою додатка.

На верхньому рівні залишається сервер бази даних.

Переваги:

- «тонкий» клієнт знижує вартість апаратного забезпечення робочої станції;
- централізація бізнес-логіки дозволяє централізувати супровід додатка (не потрібно інсталювати програмне забезпечення на всі робочі станції);
- модульність спрощує модифікацію і заміну;
- як наслідок рівномірного розподілу навантажень відбувається розвантаження сервера.

Триланкова архітектура природно відображається на середу WEB, де web-браузер виконує роль «тонкого» клієнта, а web-сервер – роль сервера додатків, при цьому триланкова архітектура може бути розширена до N-рівнів.

Архітектура WEB-додатків, які публікують бази даних включає додаткові рівні, такі як сервер бази даних, сервер додатків, джерела даних. В залежності від того, як розподіляються ланки ІС на вузлах комп'ютерної мережі, можуть бути визначені дворівневі, трирівневі або N-рівневі WEB-додатки.

Структура ***дворівневого WEB-додатка*** наведена на рис. 2.5.

База даних зберігається на тому ж комп'ютері, де знаходиться WEB-сервер, для доступу до бази даних використовуються додаткові програмні модулі розширення, розроблені на основі сучасних WEB-технологій.

Схема функціонування:

- WEB-оглядач передає URL адресу головної сторінки додатка WEB-сервера;
- WEB-сервер передає головну HTML-сторінку WEB-додатку оглядачеві;
- якщо необхідна інформація з бази даних оглядач по посиланню, що знаходиться в HTML сторінці, про формує URL запит до модуля розширення сервера (при цьому використовуються різні технології PHP, ASP, CGI і інші);

– WEB-сервер викликає необхідний модуль розширення і передає йому параметри URL;
– модуль розширення сервера формує SQL запит до бази даних.

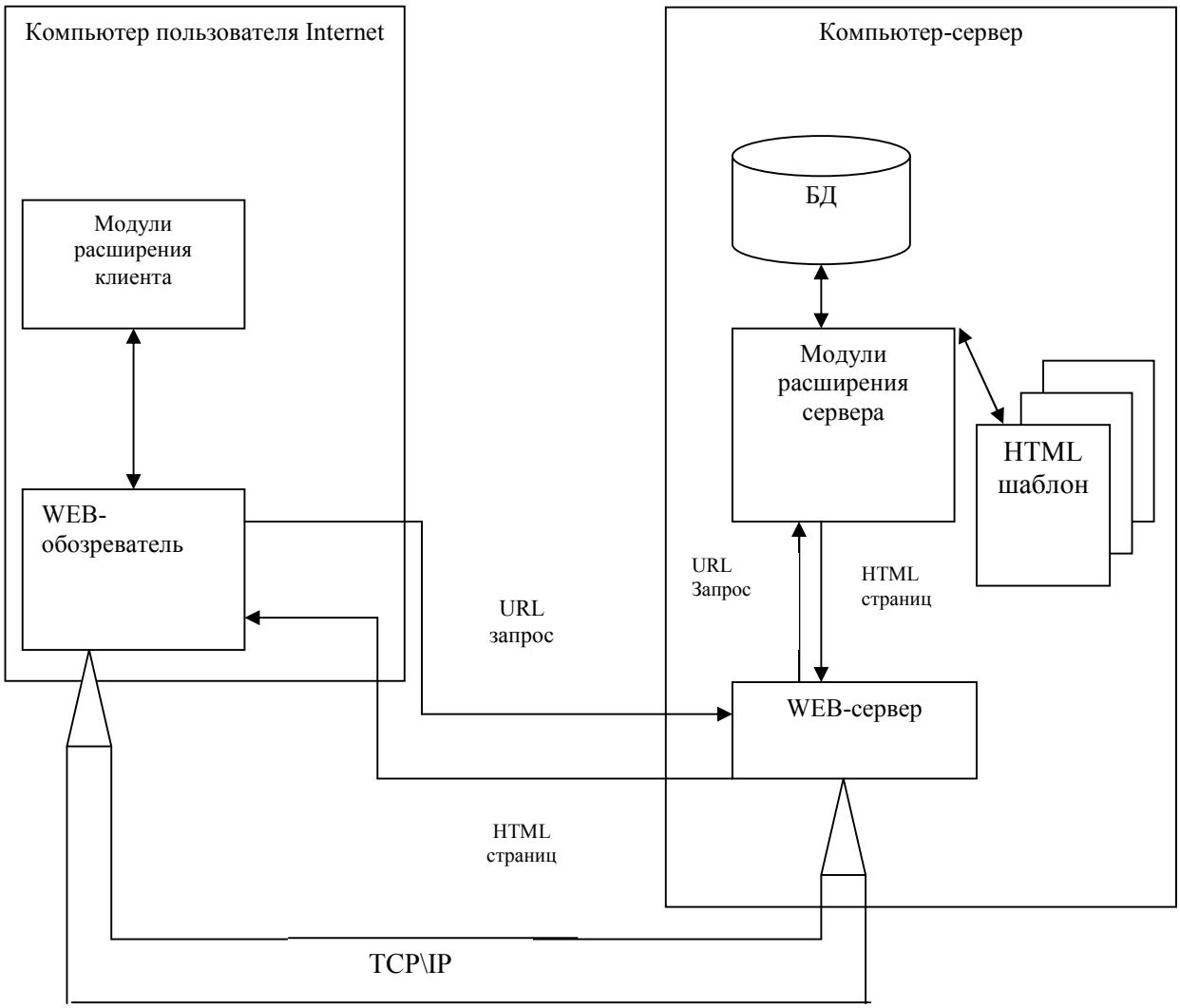

Рисунок 2.5. – Структура дворівневого WEB-додатка

Недоліки:

– підвищене навантаження на WEB-сервер, що полягає в тому, що обробка URL-запитів, вилучення інформації з БД і формування HTML сторінок виконується WEB-сервером і модулями розширення;
– низька безпека, пов'язана з неможливістю, наприклад, забезпечити необхідних рівень захисту інформації в БД від збоїв під час звернення

до БД з модуля розширення сервера або конфіденційність інформації БД від адміністратора WEB-вузла.

Структура ***трирівневого WEB-додатка*** наведена на рисунку 2.6.

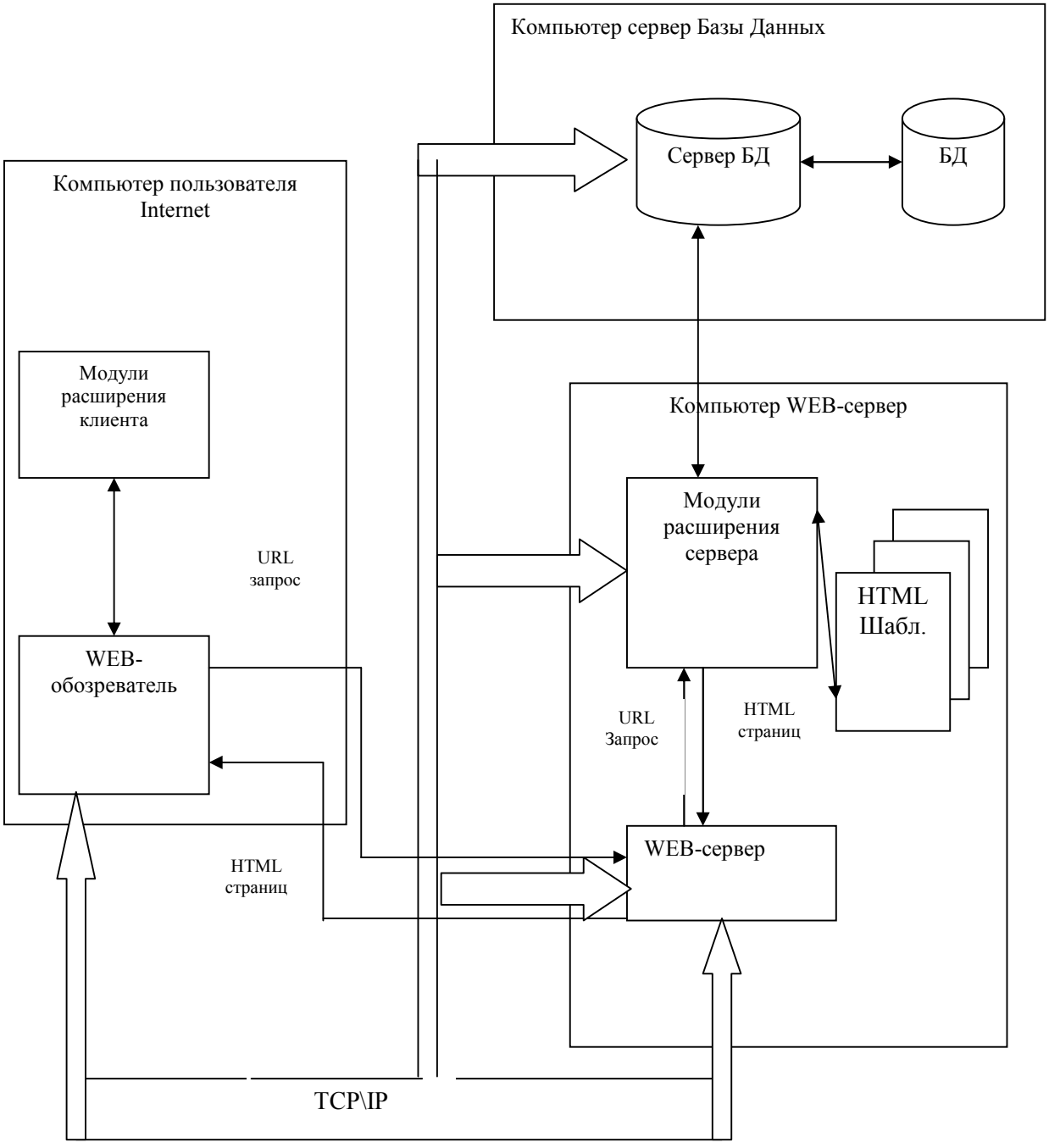

Рисунок 2.6. – Структура трирівневого WEB-додатка

WEB-сервер і модулі розширення утворюють проміжний рівень між клієнтом-оглядачем і сервером бази даних. Для отримання даних модуль

розширення WEB-сервера формує і відсилає SQL запит віддаленого сервера бази даних. SQL-сервер забезпечує виконання запиту і видачу модулю розширення WEB-сервера результат запиту т.е. обробка запиту виконується на віддаленому сервері.

Переваги в порівнянні з дворівневою архітектурою:

– – підвищення рівня безпеки інформації (сервер БД управляє доступом до БД, забороняє одночасне зміна одного запису різними користувачами, реалізуючи механізм транзакцій);
– підвищення стійкості WEB додатків до збоїв;
– зниження складнощів модулів розширення WEB-серверів, так як відсутня програмних код, пов'язаний з контролем БД, розмежуванням доступу;
– забезпечення взаємозамінності компонентів архітектури та модульності.

Недоліки – підвищення часу обробки запитів, що пов'язано з додатковим зверненням по мережі до сервера БД.

Підводячи підсумки розгляду існуючих архітектур ІС можна сформулювати наступні переваги та недоліки інтеграції СУБД в середу WEB.

Переваги інтеграції полягають у наступному:

– простота реалізації;
– незалежність платформи;
– графічний інтерфейс користувача;
– міжплатформна підтримка;
– перевага використання функцій СУБД;
– прозорий мережевий доступ;
– масштабування розгортання.

Недоліки інтеграції:

– слабка захищеність і недостатня надійність;

– обмежена функціональність мови HTML;

– недостатня продуктивність;

– недосконаліть інструментів розробки;

– високі вимоги до пропускної здатності мережі.

На рисунку 2.7 приведена функциональна модель типової ІС, в тому числі створюваної для середовища WEB.

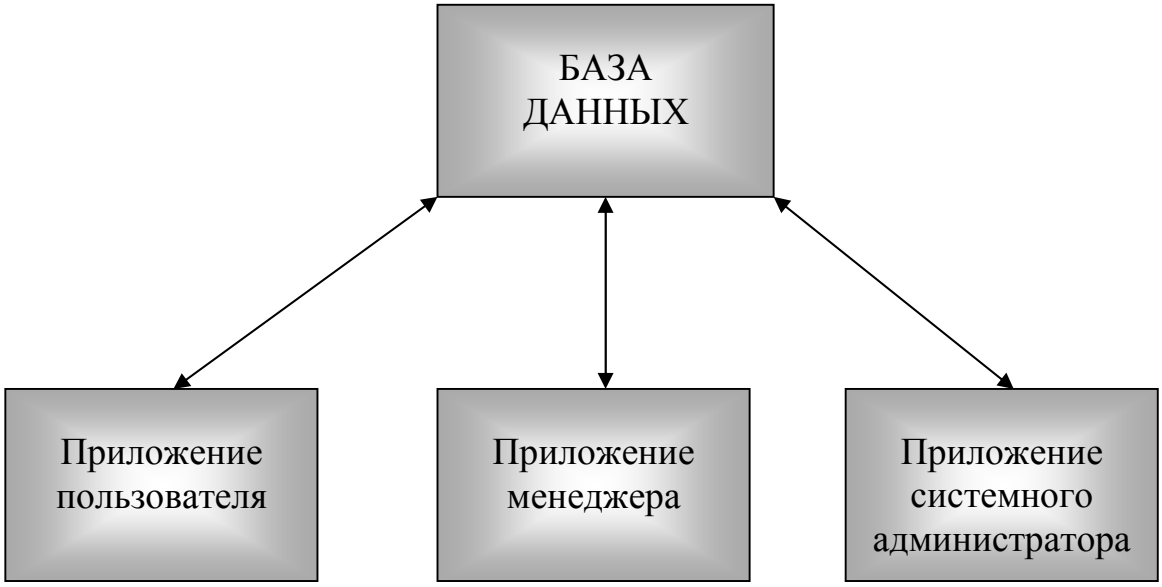

Рисунок 2.7 – Функциональна модель Інтернет-системи

2.3 Технології сучасних Web-додатків

Сучасні Web-додатки будуються як багатоланкові додатки, що публікують БД. Для розширення функціональних можливостей Web-системи, яка реалізує бізнес логіку конкретних предметних областей необхідно створення функціональних модулів, званих модулями розширення, які можуть розміщуватися як на Web-сервері, так і на комп'ютері клієнта.

Для реалізації таких модулів розширення використовується велика різноманітність технологій, причому в цьому напрямку здійснюється дуже динамічний розвиток. На рисунку 2.8, що ілюструє, представлені найбільш широко використовувані технології.

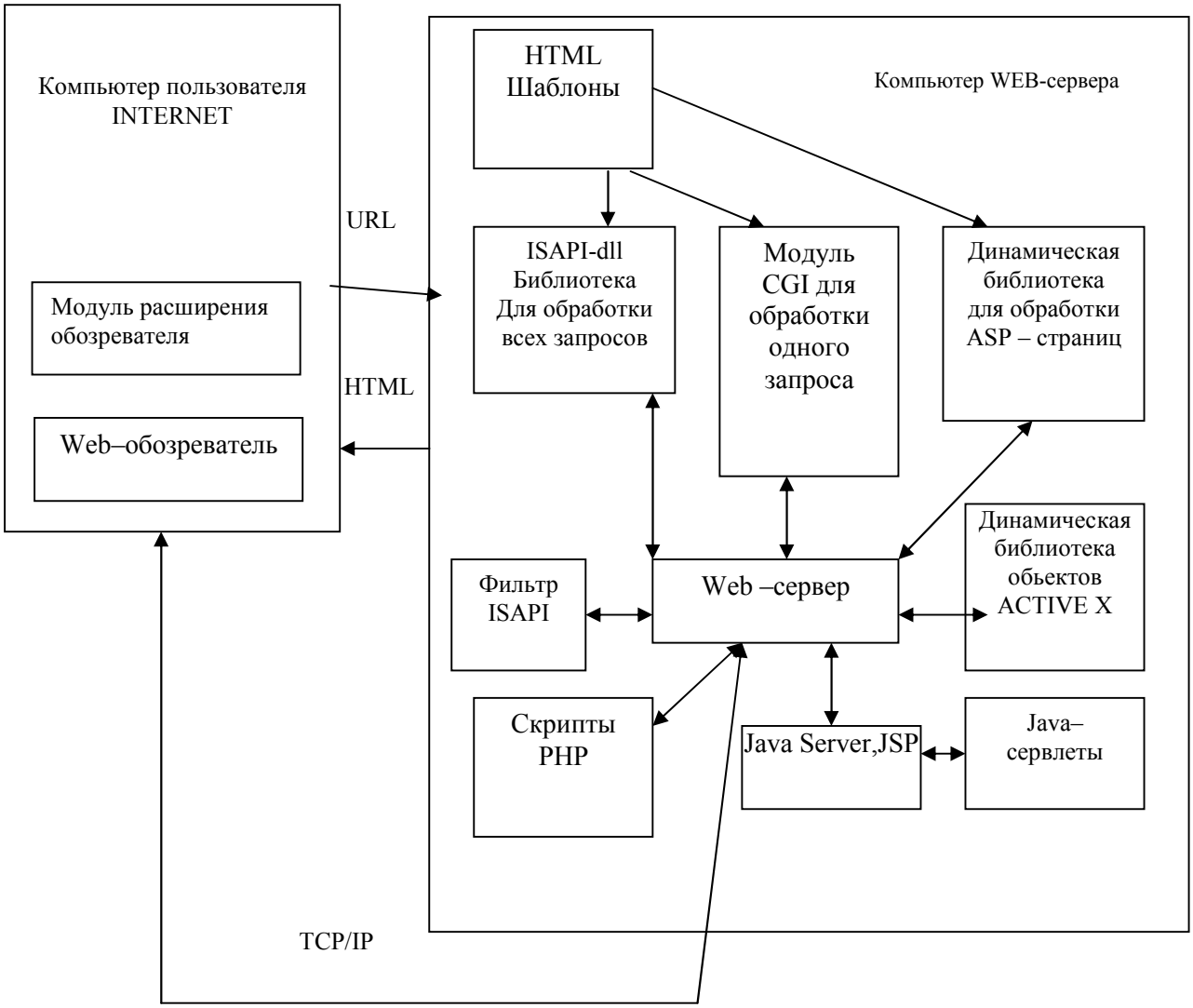

Рисунок 2.8. – Технології реалізації модулів розширення на Web-сервері

Web-додатки з модулями розширення на Web-сервері можуть включати наступні стандартні модулі реалізовані за технологіями:

- модулі, розроблені на основі інтерфейсів CGI, ISAPI, NSAPI;
- об'єкти Active X;
- Java-сервлети;
- динамічні бібліотеки (DLL), що реалізують технології ASP, JSP;
- скрипти PHP і інші.

При цьому основна функція Web-сервера це обробка запитів Web-оглядачів користувачів мережі, виклик відповідного модуля розширення і

передача йому параметрів запиту та, в результаті обробки запитів модулями розширення на основі різних HTML шаблонів, формування Web-документа. Готовий HTML документ відсилається Web-оглядачеві у форматі протоколу HTTP. При цьому можливо пасивний стан Web-сервера, сформований документ містить тільки статичну інформацію і відсутні засоби введення і обробки запитів і активний стан сервера – динамічне створення Web-документа у відповідь на запит користувача.

Інтерфейс CGI (***Common Gateway Interface*** – загальний шлюзовий інтерфейс) – є стандартним протоколом взаємодії між Web-сервером і модулями розширення, які можуть застосовуватися для виконання додаткових функцій, не підтримуваних сервером. Для написання CGI-програм підходить будь-яка мова програмування.

Недоліки:

– невисока швидкість обробки запитів;

– підвищена завантаження Web-сервера;

ISAPI/NSAPI (***INTERNET Server API / Netscape server API***) – більш перспективні, ніж CGI і призначені для розробки модулів розширення Web-серверів, які реалізовані вигляді бібліотек динамічного компонування (dll). У багатокористувательському режимі роботи сервера завантаження ISAPI модуля розширення l відбувається один раз при першому зверненні. При обробці сервером подальших запитів до модуля розширення сервер використовує вже завантажений екземпляр динамічної бібліотеки.

Переваги:

– економія ресурсів сервера;

– збільшення швидкості обробки;

ISAPI-фільтри – на відміну від модулів ISAPI можуть використовуватися для контролю усього потоку даних між сервером і оглядачем на рівні протоколу http. Застосовуються для динамічного перекодування, шифрування, збору статистичної інформації про роботу сервера, фільтрації.

Елементи керування Active X – модулі розширення використовувані на стороні клієнта і сервера, створюються на основі технології COM. Контейнером для елементів управління Active X може бути тільки той додаток, що підтримує технологію COM і надає можливість маніпулювати вбудовуваними компонентами. До таких додатків зазвичай ставляться докладання фірми MICROSOFT написані на мовах Visual Basic, VBScript, Access і т.д.

Переваги:

– не треба витрачай зусилля на програмування функціональних можливостей додатків, так як можна скористатися реалізованими компонентами;

– Active X – стандартні будівельні блоки, найбільш часто використовуються для задач контролю, шифрування, мультимедіа, анімація.

На рисунку 2.9 наведено технології, які переважно використовуються на стороні клієнта.

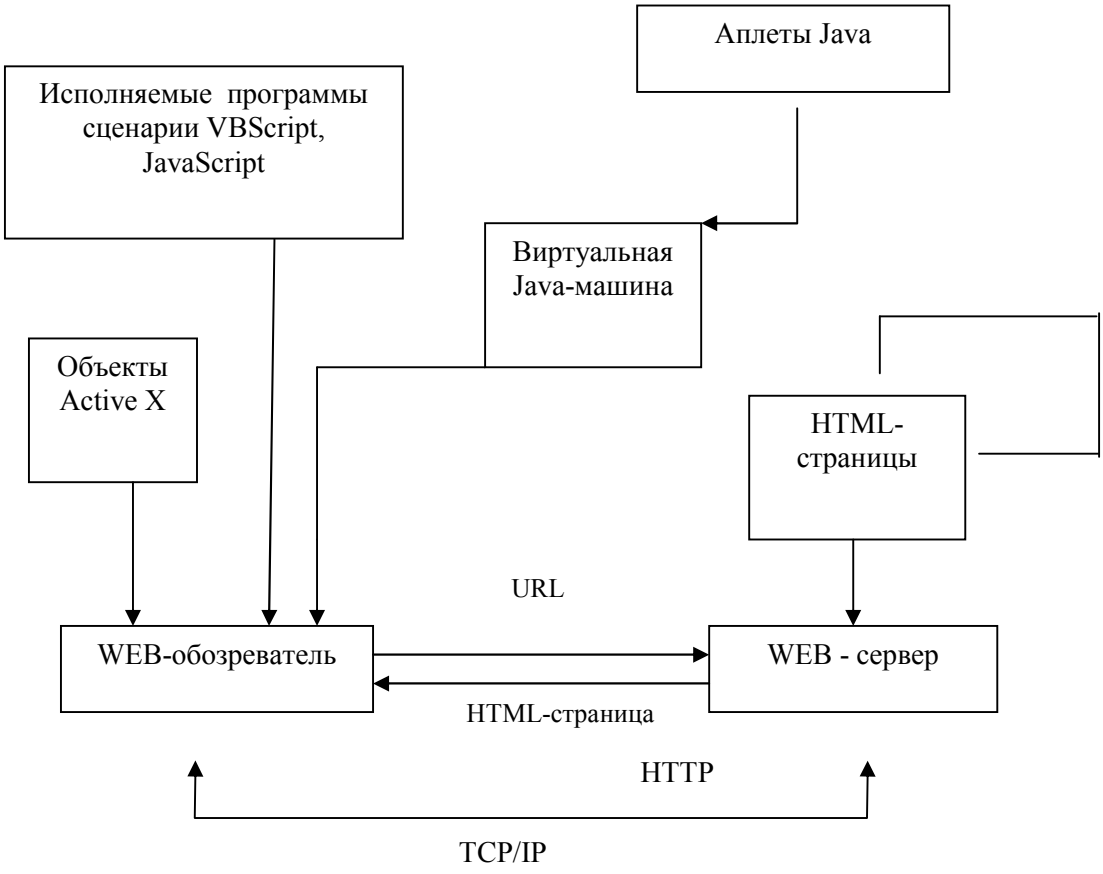

Рисунок 2.9 – Технології реалізації модулів розширення на клієнтської частині

У разі модулів розширення клієнтської частини використовують:

– програми, які підключаються;
– сценарії VBScript, JavaScript;
– об'єкти Active X;
– аплети Java.

Об'єкти Active X можуть використовуватися як на сервері так і на клієнтській машині. З точки зору забезпечення безпеки локальних даних користувачів використання Active X не завжди виправдано, оскільки механізм їх роботи дозволяє одержати з програмного коду необмежений доступ до локальних ресурсів.

Виконувані програми – перше покоління клієнтських розширень при цьому попередньо необхідно сконфігурувати оглядач. Досить важко забезпечити сумісність таких програм з різними оглядачами.

Java – мова програмування, розроблений Sun Microsystems в 1995 році. Java технологія включає в себе, власне, мова програмування і платформу.

Головне достоїнство мови Java в порівнянні з іншими мовами полягає в тому, що код аплету не залежить від апаратної платформи і програмних засобів.

Програма на Java транслюється компілятором в спеціальний байтовий код – J-код, для виконання якого потрібен спеціальний інтерпретатор, розміщений на WEB-браузері.

Інтерпретатор Java – це додаток, призначений для конкретної апаратно-програмної платформи (IBM PC, UNIX, APPLE MACINTOSH).

При цьому J-код не залежить від платформи, що забезпечує архітектурну незалежність і переносимість програм, тобто J-код набір машинних команд деякої віртуальної машини реалізованої інтерпретатором (рис. 2.10).

а) выполнение Java программа

б) выполнение Си программы

Рисунок 2.10 – Виконання програм Си та Java

Платформа Java складається з віртуальної машини і інтерфейсу прикладного програмування Java API. Віртуальна машина призначена для виконання J-коду на різних платформах. Інтерфейс прикладного програмування Java API представляє собою велику колекцію класів.

У середовищі Java існують два основних типи програм:

– *додатки* – самостійні програми для компіляції потрібно тільки віртуальна машина Java;

– *аплети* – програма, яка виконується тільки в складі WEB-браузера або програми перегляду аплетів.

Схема застосування аплетів наведена на рисунку 2.11.

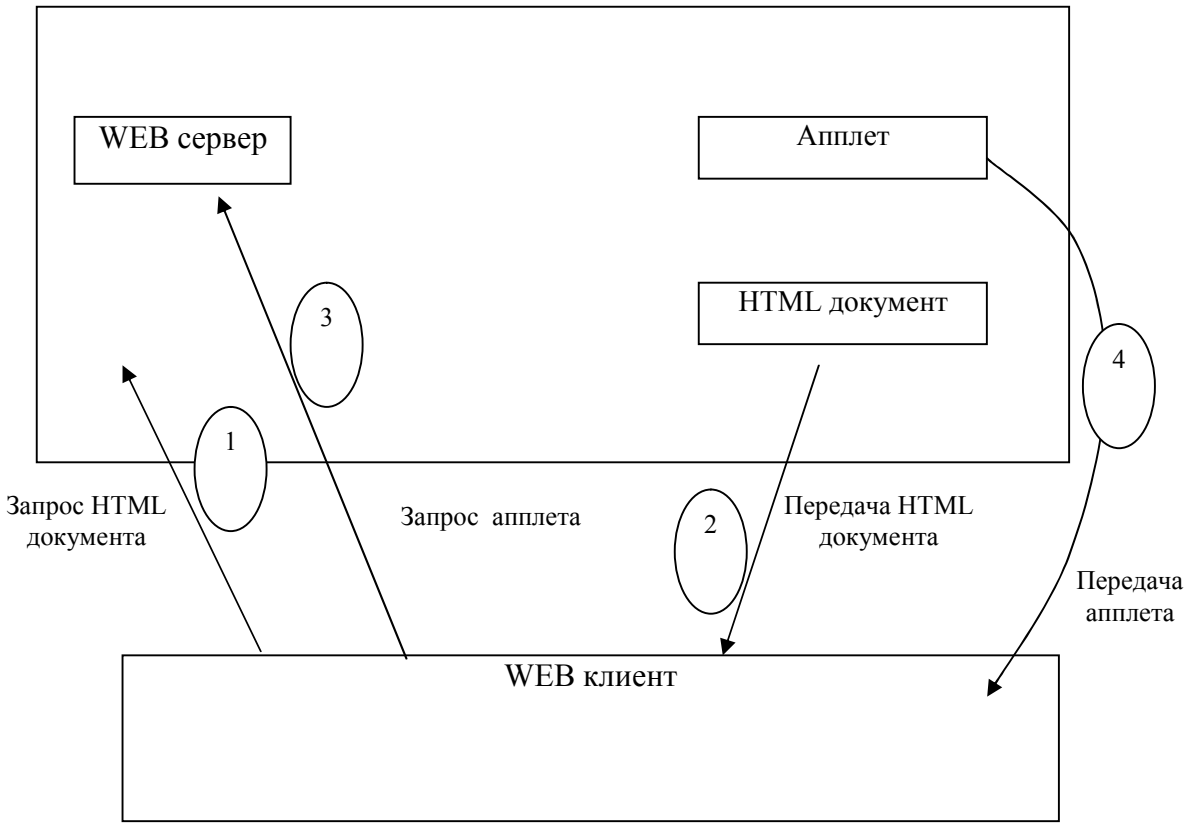

Рисунок 2.11. – Схема застосування Java-аплетів

Аплет створюється, компілюється, зберігається на WEB сервері.

У HTML документі при використанні спеціального тега міститься посилання на місце розташування аплету.

При отриманні документа з сервера браузер завантажує аплет і виконує його на віртуальній машині. Тобто браузер в своєму складі містить віртуальну машину і набір класів. Java-об'єктно-орієнтована мова.

Використання аплету на Java – досить безпечно, тому існує обмеження набору операцій, які він може виконувати.

Java сервлети – маленькі програми, які виконуються виключно на стороні сервера для розширення функціональних можливостей, використовуються при цьому бібліотеки класів і спеціальні набори інструментальних засобів (JSDK).

2.4 Використання PHP і СУБД MySQL для створення WEB-систем

PHP – надбудова в HTML є мовою серверних скриптів, що інтерпретуються і виконуються на сервері.

PHP є препроцесором HTML. До того, як сервер "віддасть" файл браузеру, його переглядає препроцесор-інтерпретатор. Для того, щоб це відбулося, файли, що підлягають обробці препроцесором, повинні мати визначене розширення (як правило це .php) і містити (хоча це не обов'язкова вимога) код для препроцесору. Перед відправкою сторінки PHP-код програється на сервері й броузеру видається результат у вигляді HTML-сторінки. Звичайні ж сторінки, що мають розширення .html чи .htm Web-сервер буде відправляти броузеру без будь-якої обробки.

Основна відзнака від CGI-скриптів, написаних на інших мовах типу Perl чи C – це те, що в CGI-програмах ви самі пишете виведений HTML-код. При застосуванні PHP ви вбудовуєте свою программу-скрипт у готову HTML-сторінку, використовуючи відкриваючий і закриваючий теги (<?php і ?> чи <? і ?>), які можна вставлять у будь-яке місце HTML-сторінки.

PHP називається мовою серверних скриптів на відзнаку від JavaScript/Jscript/VBScript, що є мовами клієнтських скриптів. Це означає, що PHP-скрипт виконується на сервері, а клієнту передається результат його роботи, тоді як у JavaScript- код цілком передається на клієнтську машину і тільки там виконується броузером.

На PHP можна зробити усе, що можна зробити за допомогою CGI-програм, а саме: обробляти дані з форм, генерувати динамічні сторінки. Крім цього в PHP включено підтримку багатьох баз даних (БД), що спрощує написання Web-додатків з використанням БД.

Приклад простої PHP-програми:

```
<html>
<head>
<title>Приклад</title>
</head>
<body>
<?php
```

```
echo "Привіт, я PHP-програма!";
?>
</body>
</html>
```

Найчастіше серверні скрипти використовуються для обробки результатів заповнення форм. Наприклад, у гостьовій книзі відвідувач уводить дані у форму, що потім обробляється на сервері. Відповідаючи на яке-небудь опитування, користувач, аналогічно, встановлює значення визначених полів форми.

Які теги й атрибути повинна містити форма?

```
<FORM NAME="ім'я_форми" ACTION="шлях_до_оброблювача"
METHOD="метод_передачі_перемінних">
…поля введення...
</FORM>
```

Оброблювач – це скрипт на сервері, у якому будуть передані значення полів вводу.

Кожне поле вводу має атрибут NAME, що буде переданий до оброблювача разом зі своїм значенням.

Існує два методи передачі даних: GET і POST. Їх відмінність полягає в тому, що при використанні методу GET значення полів приєднуються до URL, зазначеного в атрибуті ACTION. Відбувається це в такий спосіб:

```
http://server.net/action.php?ім'я=значення&...ім'я=значення
```

Пари "ім'я=значення" створюються для кожного елементу вводу, для якого зазначено ім'я атрибутом NAME.

У випадку використання методу POST значення полів передаються в тілі запиту до сервера "непомітно" для звичайного користувача.

При виконанні скрипта мовою PHP створюються перемінні з іменами, що відповідають іменам полів й утримують відповідні значення.

Припустимо, що ми створили форму наступного виду:

```
<FORM ACTION="mult.php" METHOD="GET">
<INPUT TYPE="text" NAME="first" SIZE="4" MAXLENGTH="4">
<INPUT TYPE="text" NAME="second" SIZE="4" MAXLENGTH="4">
<INPUT TYPE="Submit" VALUE="Помножити">
</FORM>
```

Скрипт, що міститься у файлі mult.php, може мати такий вигляд:

```
<?php
echo "$first помножити на $second вийде ", $first*$second;
?>
```

Робота з БД у PHP здійснюється в основному шляхом роботи з різними SQL-серверами, причому SQL-сервер у будь-якому випадку розглядається як віддалений, тобто створюється з'єднання мережі. Завдяки цьому можливо відкривати з одного скрипта або кілька сесій користувача, або працювати з різними SQL-серверами. Після встановлення з'єднання з сервером, обирається робоча база даних, після чого можна відправляти й обробляти запити (тому що SQL є клієнт-серверною архітектурою, будь-яка робота з даними здійснюється за допомогою запитів до SQL-сервера на одержання чи зміну даних). При виконанні запиту створюється деякий об'єкт, у якому зберігається результат виконання запиту, після чого можна одержувати окремі терміни шляхом виконання спеціальних функцій.

Загальна послідовність дій при взаємодії із сервером MySQL виглядає так:

- встановити з'єднання з сервером MySQL. Якщо спроба завершується невдачею, вивести відповідне повідомлення й завершити процес;
- обрати базу даних сервера MySQL. Якщо спроба вибору завершується невдачею, вивести відповідне повідомлення і завершити процес. Припустимим є одночасне відкриття декількох баз даних для обробки запитів;
- обробити запити до обраної бази (чи баз);
- після завершення обробки запитів закрити з'єднання із сервером баз даних.

Далі описуються функції PHP для роботи із СУБД MySQL.

Функція mysql_connect() встановлює зв'язок із сервером MySQL Після успішного підключення до MySQL можна переходити до вибору баз даних, що обслуговуються цим сервером. Синтаксис функції mysql_connect():

```
int mysql_connect ([string хост [:порт] [:/шлях//до/сокету]
[, string ім'я користувача] [, string пароль])
```

У параметрі хост передається ім'я хостового комп'ютера, зазначене в таблицях привілеїв сервера MySQL. Воно ж використовується для переспрямування запитів на web-сервер, на якому працює MySQL, оскільки до серверу MySQL можна підключатися у віддаленому режимі. Поряд з ім'ям хоста можуть бути вказаними необов'язкові параметри — номер порту, а також шлях до сокету (для локального хоста). Параметри ім'я_користувача і пароль повинні відповідати імені користувача і паролю, заданим у таблицях привілеїв MySQL. Усі параметри є необов'язковими, оскільки таблиці привілеїв можна настроїти таким чином, щоб вони допускали з'єднання без перевірки. Якщо параметр хост не заданий, mysql_connect() намагається встановить зв'язок з локальним хостом.

Функція mysql_pconnect() забезпечує підтримку відновлюваних з'єднань. У багакористувальницких середовищах рекомендується використовувати mysql_pconnect() замість mysql_connect() з метою економії системних ресурсів. За типами параметрів і значенню, що повертається, функція mysql_pconnect() у точності збігається з mysql_connect().

Після успішного з'єднання з MySQL необхідно вибрати базу даних, що знаходиться на сервері. Для цього використовується функція mysql_select_db(). Синтаксис функції mysql_select_db():

```
int mysql_select_db (string ім'я_бази_даних
[, int ідентифікатор_з'єднання])
```

Параметр ім'я_бази_даних визначає обирану базу даних, ідентифікатор якої повертається функцією mysql_select_db(). Зверніть увагу: параметр

ідентифікатор_з'єднання необов'язковий лише при одному відкритому з'єднанні із сервером MySQL. При наявності декількох відкритих з'єднань цей параметр має бути вказаним.

Якщо в програмі вибирається лише одна база даних, зберігати її ідентифікатор не обов'язково. Однак при виборі декількох баз даних ідентифікатори, що повертаються, зберігаються, щоб ви могли послатися на потрібну базу при обробці запиту. Якщо ідентифікатор не зазначений, використовується остання обрана база даних.

Після завершення роботи із сервером MySQL з'єднання необхідно закрити. Функція mysql_close() закриває з'єднання, обумовлене необов'язковим параметром. Якщо параметр не заданий, функція mysql_close() закриває останнє відкрите з'єднання. Синтаксис функції mysql_close():

```
int mysql_close ([int ідентифікатор_з'єднання])
```

Функція mysql_query() забезпечує інтерфейс для звертання з запитом до баз даних. Синтаксис функції mysql_query():

```
int mysql_query (string запросо [, int ідентифікатор_з'єднання])
```

Параметр запросу містить текст запиту мовою SQL. Запит передається або з'єднанню, обумовленому необов'язковим параметром ідентифікатор_з'єднання, або, при відсутності параметра, останньому відкритому з'єднанню.

При успішному виконанні команди SQL SELECT повертається ідентифікатор результату, що згодом передається функції mysql_result() для наступного форматування і відображення результатів запиту. Якщо обробка запиту завершилася невдачею, функція повертає FALSE. Кількість записів, що беруть участь у запиті, визначається за допомогою функції mysql_num_rows().

Якщо викликає занепокоєння те, що при обробці запитів витрачається занадто багато пам'яті, можна викликати стандартну функцію PHP

mysql_free_result(). При виклиці їй передається ідентифікатор результату, що повертається mysql_query(). Функція mysql_free_result() звільняє всю пам'ять, зв'язану з даним запитом.

У багатьох ситуаціях потрібно знати кількість записів, що беруть участь у запиті SQL з командами INSERT, UPDATE, REPLACE чи DELETE. Задача зважується функцією mysql_affected_rows(). Синтаксис функції:

```
int mysql_affected_rows ([int ідентифікатор_з'єднання])
```

Параметр ідентифікатор_з'єднання не є обов'язковим. Якщо не вказується, mysql_affected_rows() намагається використовувати останнє відкрите з'єднання.

Функція mysql_affected_rows() не працює з запитами, заснованими на команді SELECT. Для визначення кількості записів, повернутих при виклику SELECT, використовується функція mysql_num_rows().

Функція mysql_num_rows() визначає кількість записів, що повертаються командою SELECT. Синтаксис функції mysql_num_rows():

```
int mysql_num_rows(int результат)
```

Функція mysql_result() використовується в поєднанні з mysql_query() (при виконанні запиту з командою SELECT) для одержання набору даних. Синтаксис функції mysql_resu1t():

```
int mysql_result (int ідентифікатор_результату, int запис
[. mixed поле"]')
```

У параметрі ідентифікатор_результату передається значення, повернуте функцією mysql_query(). Параметр запис посилається на визначений запис набору даних, зумовленого параметром ідентифікатор_результату. Нарешті, у необов'язковому параметрі поле можуть передаватися:

```
поля в таблиці;
ім'я поля;
```

```
ім'я поля у форматі ім'я_поля_ім'я_таблиці.
```

Функція mysql_result() зручна для роботи з відносно невеликими наборами даних, однак існують й інші функції, що працюють набагато ефективніше, - а саме, функції mysql_fetch_row() і mysql_fetch_array( ).

Звичайно набагато зручніше відразу привласнити значення всіх полів запису елементам індексованого масиву (починаючи з індексу 0), ніж багаторазово викликати mysql_result() для одержання окремих полів. Задача зважується функцією mysql_fetch_row(), що має наступний синтаксис:

```
array mysql_fetch_row (int результат)
```

Функція mysql_fetch_array() аналогічна mysql_fetch_row(), однак за замовчуванням значення полів записи зберігаються в асоціативному масиві. Втім, ви можете вибрати тип індексації (асоціативна, числова чи комбінована). Синтаксис функції mysql_fetch_array():

```
array mysql_fetch_array (int ідентифікатор результату
[, тип_індексації])
```

У параметрі ідентифікатор_результату передається значення, повернуте функцією mysql_query(). Необов'язковий параметр тип_індексації приймає одне з наступних значень:

`MYSQL_ASSOC` – функція `mysql_fetch_array()` повертає асоціативний масив. Якщо параметр не зазначений, це значення використовується за замовчуванням;

`MYSQL_NUM` – функція `mysql_fetch_array()` повертає масив з числовою індексацією;

`MYSQL_BOTH` – до полів запису, що повертається, можна звертатися як за числовими, так й за асоціативними індексами.

2.5 Питання для самоперевірки

1) Охарактеризувати архітектуру Інтернет системи.
2) Пояснити принципи проектування бази даних у вибраній наочній області.
3) Пояснити принципи реалізації функціональності Інтернет системи в технології PHP скриптів.
4) Як виконується PHP-код?
5) Які теги й атрибути повинна містити форма?
6) Методи передачі даних: GET і POST.
7) Загальна послідовність дій при взаємодії із сервером MySQL.
8) Яка функція PHP встановлює зв'язок із сервером MySQL ?
9) Яка функція PHP після успішного з'єднання з MySQL вибирає базу даних?
10) Яка функція PHP закриває з'єднання з MySQL?
11) Яка функція PHP забезпечує інтерфейс для звертання з запитом до баз даних?
12) Як використовується функція mysql_result() ?

3 ПРИКЛАД ВИКОНАННЯ ЗАВДАНЬ КОНТРОЛЬНОЇ РОБОТИ

3.1. Приклад розробки WEB–додатку з використанням мови сценаріїв PHP і СУБД MySQL

Завдання 1

Здійснити проектування бази даних, яка містить в собі дані о студентах, дисциплінах, які вивчають студенти, оцінках, викладачах, які мають ставити оцінки. Розробити модель бази даних «Сутність – Зв’язок», згідно з переліченими даними. Виконати реалізацію бази даних у СУБД MySQL.

Припустимо, нам необхідно спроектувати базу даних, що містить дані щодо студентів, дисципліни, що вони вивчають, оцінки, та викладачів, що виставляють оцінки.

Проектування почнемо з виділення основних сутностей та їх атрибутів. Основними сутностями (атрибути перераховані в дужках) у даній предметній області є:

- ***Студенти*** (номер залікової книжки, ПІБ, адреса, телефон, дата народження, група);
- ***Викладачі*** (код, прізвище, ім’я, по-батькові);
- ***Дисципліни*** (код, назва);
- ***Оцінки*** (код, назва).

Виділення сутності ***Групи*** (код групи, назва) дозволить скоротити обсяг даних та знизити ймовірність виникнення аномалій при роботі з даними про студентів (при видаленні даних про студентів якої-небудь групи може зникнути інформація про дану групу).

З метою виключення зв'язків типу «численні-до-численних» між виділеними поняттями (наприклад, між поняттями ***Студенти й Оцінки*** – один студент може отримувати багато різних оцінок («5», «4», «3», «2»), а з іншого боку, ту саму оцінку можуть отримати багато студентів, вводимо додаткову

сутність (***Звіт***), що буде зберігати необхідну інформацію про те, хто, коли, з якого предмету яку оцінку отримав і ким цю оцінку поставлено. Таким чином, сутність ***Звіт*** буде характеризуватися атрибутами: код, код студента, дата, код предмету, код оцінки, код викладача.

Далі описані зв'язки між сутностями .

- Між сутностями ***Групи і Студенти*** існує зв’язок типу «один-до-багатьох». Наявність такого зв'язку означає, що в одній групі може навчаться багато студентів.
- Між сутностями ***Предмети і Звіт*** існує зв'язок типу «один-до-багатьох». Наявність такого зв'язку означає, що з одного предмету можуть отримувати оцінки різні студенти.
- Між сутностями ***Оцінки і Звіт*** існує зв'язок типу «один-до-багатьох». Наявність такого зв'язку означає, що ту саму оцінку можуть одержати багато студентів.
- Між сутностями ***Викладачі і Звіт*** існує зв’язок типу «один-до-багатьох». Наявність такого зв'язку означає, що один викладач може поставити багато оцінок.
- Між сутностями ***Студенти і Звіт*** існує зв’язок типу «один-до-багатьох». Наявність такого зв'язку означає, що один студент може одержати багато оцінок.

На рисунку 3.1 зображено модель «сутність-зв'язок» для розглянутої предметної області. Жирним шрифтом виділені первинні ключі.

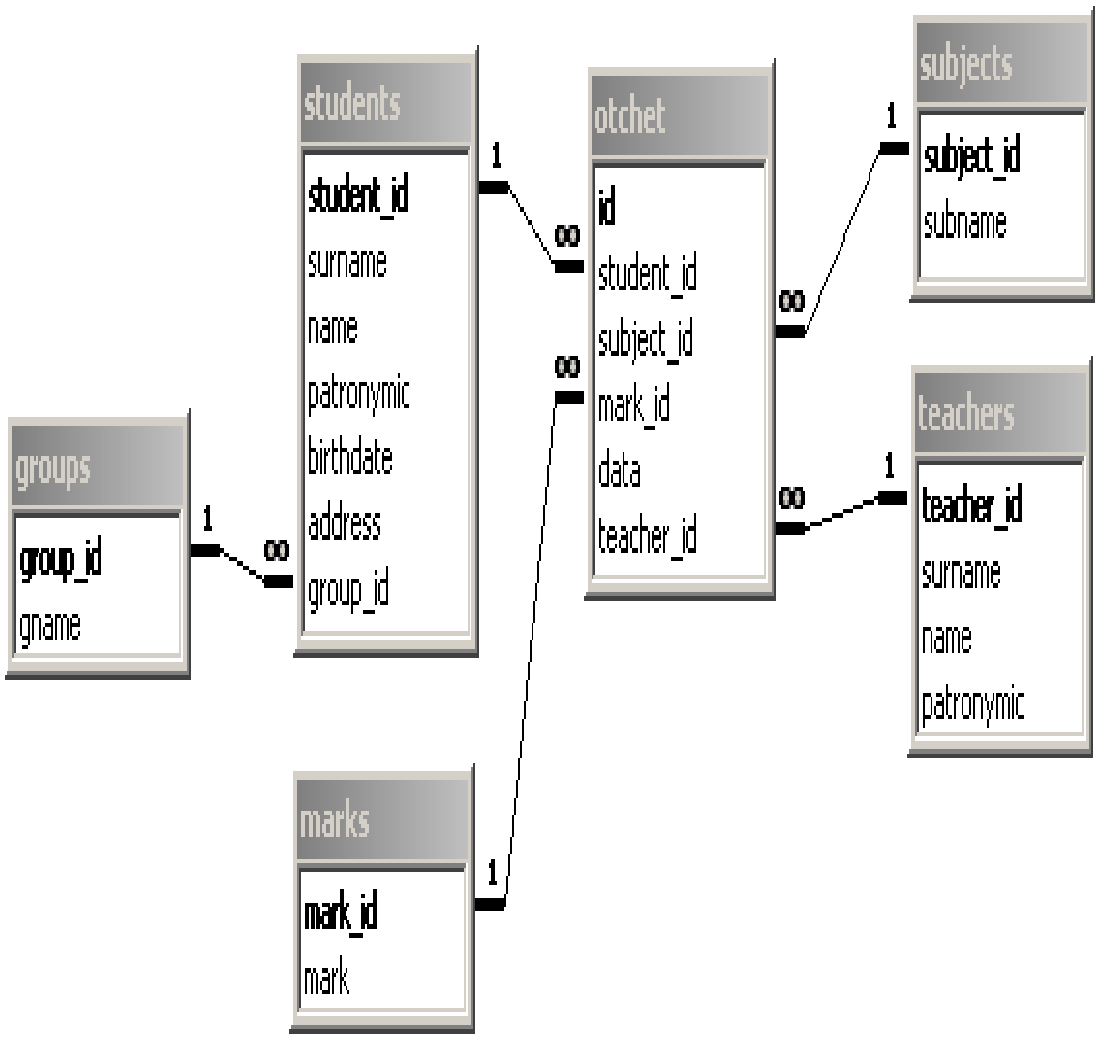

Рисунок 3.1 – Модель «Сутність-зв'язок»

На подальших етапах проектування кожна сутність стає таблицею, атрибути – стовпчиками таблиць, а значення атрибутів формують рядки таблиць. Для кожного стовпчика необхідно вказати, значення якого типу будуть у ньому зберігатися, чи може він містити порожні значення. Наприклад, для таблиці

Групи:

Ім'я стовпчика	Тип даних	Порожні значення	Додатково
group_id	integer	not null	primary key, auto_increment
gname	char(4)	not null	

Для таблиці **Студенти**:

Ім'я стовпчика	Тип даних	Порожні значення	Додатково
student_id	char(6)	not null	primary key
surname	text	not null	
name	text	not null	
patronymic	text	not null	
birthdate	date	not null	
address	text	not null	
group_id	integer	not null	

Опція `auto_increment` означає, що при додаванні нового запису в таблицю значення поля будуть автоматично збільшуватися на 1 (для забезпечення унікальності значень первинних ключів).

Далі наведено опис спроектованої БД мовою SQL.

```
create table groups (
    group_id integer not null auto_increment,
    gname char(4) not null,
    primary key (group_id)
   );

create table subjects (
    subject_id integer not null auto_increment,
    subname text not null,
    primary key (subject_id)
    );

create table students (
    student_id char(6) not null,
    surname text not null,
    name text not null,
    patronymic text,
    birthdate date not null,
    address text not null,
    group_id integer,
    primary key (student_id)
    );
```

```
create table marks (
    mark_id integer not null auto_increment,
    mark text not null,
    primary key (mark_id)
    );

create table teachers (
    teacher_id integer not null auto_increment,
    surname text not null,
    name text not null,
    patronymic text not null,
    primary key (teacher_id)
    );

create table otchet (
    id integer not null auto_increment,
    student_id integer not null,
    subject_id integer not null,
    mark_id integer not null,
    data date not null,
    teacher_id integer not null,
    primary key (id)

    );
```

Завдання 2

Розробити Web-інтерфейс для роботи з базою даних з використанням мови розмітки HTML і мови скриптів PHP: реалізувати перегляд інформації; реалізувати додавання інформації в БД; реалізувати видалення інформації з БД; реалізувати функціональність додатка.

Головна сторінка клієнтського застосування для роботи з БД (`index.html`) є заснаннями на сторінки, призначені для виконання операцій над відповідними таблицями БД. Ця сторінка створена за допомогою мови HTML, тобто є статичною, і її вихідний код приведений нижче.

```
<html>
<head>
```

```
<title>Client Application</title>
</head>
<body>
<h1 align=center>Main Menu</h1>
<center>
<table width=30% border=//присвоение 10 cellspacing=10>
<tr><td align=left>
<ul>
<li><a href=groups.php>Groups</a><p>
<li><a href=students.php>Students</a><p>
<li><a href=subjects.php>Subjects</a><p>
<li><a href=teachers.php>Teachers</a><p>
<li><a href=marks.php>Marks</a><p>
<li><a href=otchet.php>Otchet</a><p>
<li><a href=search.php>Search</a><p>
</ul>
</td></tr>
</table>
</body>
</html>
```

Далі приведений вихідний код динамічної сторінки `groups.php`, створеною за допомогою PHP-сценарію.

```
<html>
<head>
<title>Groups</title>
</head>
<body>
<h1 align=center>Groups</h1>
<center>
<table border=1>
<tr><th>Group ID</th><th>Name</th><th> </th></tr>
<?
mysql_connect("host","user","password"); //соединение с БД
mysql_select_db("dbname"); // выбор БД
$result=mysql_query("select * from groups"); //запрос к БД
// присвоение значений полей, возвращаемых в результате
// выполнения запроса, элементам ассоциативного массива
while($r=mysql_fetch_array($result)){
// печать значений полей таблицы
    print "<tr><td>$r[0]</td><td>$r[1]</td>";
    print                                          <td><a
href=group_del.php?id=$r[0]>Delete</a></td>";
```

```
    }

mysql_close(); // закрытие соединения с БД
?>
</table>
<p><a href=group_add.html>Add new group</a>
<p><a href=index.html>Go back</a>
</body>
</html>
```

Вихідний код сторінки `group_add.html`, призначеної для формування форми для занесення інформації про нову групу:

```
<html>
<head>
<title>Groups</title>
</head>
<body>
<h1 align=center>Add new group</h1>
<center>
<form action=group_add.php>
Type new group name here
<input type=text name=n><p>
<input type=submit value="  Add  ">
</form>
<a href=groups.php>Back</a>
</body>
</html>
```

Исходный код сценария `group_add.php`, выполняющего занесение информации о новой группе в БД.

```
<html>
<head>
<title>Groups</title>
</head>
<body>
<h1 align=center>Groups</h1>
<center>
<table border=1>
<tr><th>Group ID</th><th>Name</th><th> </th></tr>
<?
mysql_connect("host","user","password");
```

```
mysql_select_db("dbname");
$result=mysql_query("insert into groups values(0,'$n')");
$result=mysql_query("select * from groups");
while($r=mysql_fetch_array($result)){
    print "<tr><td>$r[0]</td><td>$r[1]</td>";
    print"<td><a
href=group_del.php?id=$r[0]>Delete</a></td>";
    }

mysql_close();
?>
</table>
<p><a href=group_add.html>Add new group</a>
<p><a href=index.html>Go back</a>
</body>
</html>
```

Вихідний код сценарію `group_del.php`, що виконує видалення інформації про вибрану групу з БД.

```
<html>
<head>
<title>Groups</title>
</head>
<body>
<h1 align=center>Groups</h1>
<center>
<table border=1>
<tr><th>Group ID</th><th>Name</th><th> </th></tr>
<?
mysql_connect("host","user","password");
mysql_select_db("dbname");
$result=mysql_query("delete from groups where group_id=$id");
$result=mysql_query("select * from groups");
while($r=mysql_fetch_array($result)){
    print "<tr><td>$r[0]</td><td>$r[1]</td>";
    print                                          <td><a
href=group_del.php?id=$r[0]>Delete</a></td>";
    }

mysql_close();
?>
</table>
```

```
<p><a href=group_add.html>Add new group</a>
<p><a href=index.html>Go back</a>
</body>
</html>
```

Робота з останніми батьківськими таблицями *(subjects, marks і teachers*) виробляється аналогічним чином.

Розглянемо приклад роботи з дочірніми таблицями на прикладі таблиці students. Нижче приведений вихідний код сценарію `students.php`.

```
<html>
<head>
<title>Students</title>
</head>
<body>
<h1 align=center>Students</h1>
<center>
<table border=1>
<tr><th>Student
ID</th><th>Surname</th><th>Name</th><th>Patronymic</th>
<th>Birthdate</th><th>Address</th><th>Group</th>
<th> </th></tr>
<?
mysql_connect("host","user","password");
mysql_select_db("dbname");
$result=mysql_query("select
student_id,surname,name,patronymic,birthdate,address,gname     from
students inner join groups on students.group_id=groups.group_id");
while($r=mysql_fetch_array($result)){
    print
"<tr><td>$r[0]</td><td>$r[1]</td><td>$r[2]</td><td>$r[3]</td>";
  print "<td>$r[4]</td><td>$r[5]</td><td>$r[6]</td>";
  print<td><a href=student_del.php?id=$r[0]>Delete</a></td>";
    }
?>
</table>
<p><a href=student_add.php>Add new student</a>
<p><a href=index.html>Go back</a>
</body>
</html>
```

Далі приводиться сценарій для формування форми, призначеної для занесення інформації про нового студента в БД (*student_add.php*). На відміну від сторінки *group_add.html*, яка є статичною, дана сторінка є динамічною,

оскільки в таблиці *students* присутнє заслання на таблицю *groups* (атрибут *group_id*) і значення для стовпця `group_id` в таблиці `students` повинні відповідати значенням із *стовпця* `group_id` таблиці `groups`.

```
<html>
<head>
<title>Students</title>
</head>
<body>
<h1 align=center>Students</h1>
<center>
<form action=student_add_.php>
Enter new student ID
<input type=text name=id><p>
Enter new student surname
<input type=text name=surname><p>
Enter new student name
<input type=text name=name><p>
Enter new student patronymic
<input type=text name=patronymic><p>
Enter new student birthdate
<input type=text name=birthdate><p>
Enter new student address
<input type=text name=address><p>
Enter new student group
<select name=group>
<?
mysql_connect("host","user","password");
mysql_select_db("dbname");
$result=mysql_query("select * from groups");
while($r=mysql_fetch_array($result)){
    print "<option value=$r[0]>$r[1]";
    }
mysql_close();
?>
</select><p>
<input type=submit value="  Add  ">
</form>
<p><a href=students.php>Back</a>
</body>
</html>
```

Вихідний код сценарію *student_add_.php,* що виконує занесення інформації про нового студента в БД.

```
<html>
<head>
<title>Students</title>
</head>
<body>
<h1 align=center>Students</h1>
<center>
<table border=1>
<tr><th>Student
ID</th><th>Surname</th><th>Name</th><th>Patronymic</th>
<th>Birthdate</th><th>Address</th><th>Group</th>
<th> </th></tr>
<?
mysql_connect("host","user","password");
mysql_select_db("dbname");
$result=mysql_query("insert into students
values('$id','$surname','$name','$patronymic','$birthdate','$addre
ss',$group)");
$result=mysql_query("select
student_id,surname,name,patronymic,birthdate,address,gname from
students inner join groups on students.group_id=groups.group_id");
while($r=mysql_fetch_array($result)){
    print "<tr><td>$r[0]</td><td>$r[1]</td><td>$r[2]</td>
<td>$r[3]</td>";
    print "<td>$r[4]</td><td>$r[5]</td><td>$r[6]</td>";
    print "<td><a href=student_del.php?id=$r[0]>Delete</a>
</td>";
    }
mysql_close();
?>
</table>
<p><a href=student_add.php>Add new student</a>
<p><a href=index.html>Go back</a>
</body>
</html>
```

Вихідний код сценарію *student_del.php,* що виконує видалення інформації про вибраного студента з БД.

```
<html>
<head>
```

```
<title>Students</title>
</head>
<body>
<h1 align=center>Students</h1>
<center>
<table border=1>
<tr><th>Student
ID</th><th>Surname</th><th>Name</th><th>Patronymic</th>
<th>Birthdate</th><th>Address</th><th>Group</th>
<th> </th></tr>
<?
mysql_connect("host","user","password");
mysql_select_db("dbname");
$result=mysql_query("delete from students where
student_id=$id");
$result=mysql_query("select
student_id,surname,name,patronymic,birthdate,address,gname from
students inner join groups on students.group_id=groups.group_id");
while($r=mysql_fetch_array($result)){
    print "<tr><td>$r[0]</td><td>$r[1]</td><td>$r[2]</td>
<td>$r[3]</td>";
    print "<td>$r[4]</td><td>$r[5]</td><td>$r[6]</td>";
    print "<td><a href=student_del.php?id=$r[0]>Delete</a>
</td>";    }
mysql_close();
?>
</table>
<p><a href=student_add.php>Add new student</a>
<p><a href=index.html>Go back</a>
</body>
</html>
```

Робота з дочірньою таблицею *otchet* виробляється аналогічним чином.

Розглянемо приклад пошуку в БД. Нижче приведений вихідний код сценарію *search.php*, що виконує пошук оцінок вибраного студента.

```
<html>
<head>
<title>Students' marks</title>
</head>
<body>
<center>
<?
mysql_connect("host","user","password");
mysql_select_db("dbname");
```

```
?>
<h1 align=center>View students' marks</h1>
// форма для выбора студента
<form action=search.php>
Select student
<select name=id>
<?
$res=mysql_query("select
student_id,surname,name,patronymic,gname from students inner join
groups on students.group_id=groups.group_id order by
gname,surname,name,patronymic");
while ($r=mysql_fetch_array($res)){
    print "<option value=$r[0]>$r[1] $r[2] $r[3], $r[4]";
    }
?>
</select>
<p>
<input type=submit value="Show marks">
</form>
// виведення знайдених оцінок вибраного студента
<?
if($QUERY_STRING){
    print "<hr>";
    $result=mysql_query("select
subname,surname,teachers.name,patronymic,data,mark from otchet
inner join teachers on teachers.teacher_id=otchet.teacher_id inner
join subjects on subjects.subject_id=otchet.subject_id inner join
marks on marks.mark_id=otchet.mark_id where student_id=$id");
    $f=mysql_result($res,0,1);
    $i=mysql_result($res,0,2);
    $o=mysql_result($res,0,3);
    print "<h2 align=center>$f $i $o</h2>";
    print "<table border=1>";
    print "<tr><th>Subject</th><th>Mark</th><th>Teacher</th>
<th>Date</th></tr>";
    while ($r=mysql_fetch_array($result)){
print "<tr><td>$r[0]</td><td align=center>$r[5]</td>
<td>$r[1] $r[2] $r[3]</td><td>$r[4]</td></tr>";
}
    print "</table>";
    mysql_close();
    }
print "<p><a href=index.html>Back</a>";
?>
</body>
</html>
```

4 ВАРІАНТИ ЗАВДАНЬ

В процесі вивчення дисципліни студенти виконують контрольну роботу в міжсесійний період. Контрольна робота містить 2 завдання. При виконанні завдань студент повинен вибрати варіант.

Варіант завдання визначається, як остання цифра залікової книжки, якщо ця цифра 0, то обирають варіант №10.

Звіт про виконання контрольної роботи представити у вигляді пояснювальної записки з описом процесу виконання роботі. В додатку навести програмні коді, що містять текст SQL-сценарію для створення базі даних і PHP-сценарію, що реалізують клієнтський додаток. До записки пояснення необхідно прикласти CD-диск або DVD-диск, що містить текст sql-сценарію для створення бази даних і PHP-сценарію, що реалізовують клієнтський додаток.

Для виконання контрольної роботи використовуються персональний комп'ютер під управлінням ОС Windows. Для швидкої організації роботи з web-системами, створеними за допомогою технологій PHP, MYSQL на комп'ютерах з ОС Windows, необхідно скористатися пакетом програм **Denwer.** Для установки призначеної для користувача програми на персональний комп'ютер необхідно скористатися наступним електронним ресурсом: *http://www.denwer.ru/dis/Base/.* Як інструмент для збереження результатів виконання завдань і оформлення контрольної роботи може використовуватися будь-який текстовий редактор (наприклад, MS Word).

4.1 Завдання 1

Спроектувати базу даних:

– побудувати модель «Сутність-Зв'язок» бази даних обраної згідно з варіантом предметної області;

– реалізувати базу даних предметної області з використанням СУБД MySQL.

4.2. Завдання 2

Розробити Web-інтерфейс для роботи з базою даних обраної предметної області з використанням мови розмітки HTML і мови скриптів PHP:

– реалізувати перегляд інформації;

– реалізувати додавання інформації в БД;

– реалізувати видалення інформації з БД;

– реалізувати функціональність додатку, зазначену у варіанті завдання.

Варіанти завдань

Варіант 1

"Продуктовий магазин"

В базі даних повинна зберігатися наступна інформація про товари: виробник, тип (овочі, напої, кондитерські вироби, та ін.), назва, дата виготовлення, термін зберігання, кількість товару в магазині та ціна за одиницю.

Реалізувати можливість отримання списка товарів, що виробляються обраним виробником, а також пошук інформації про товари, ціна яких не перевищує ту, що задав користувач.

Варіант 2

"Відділ кадрів"

В базі даних повинна зберігатися наступна інформація про співробітників: ПІБ, дата народження, адреса, посада, кваліфікація, відділ.

Реалізувати можливість отримання списку співробітників, що працюють на посаді начальника в обраному відділі, а також пошук інформації про співробітників по прізвищах.

Варіант 3

"Поліклініка"

В базі даних повинна зберігатися наступна інформація про пацієнтів: ПІБ, адреса, телефон, рік народження, захворювання, лікар, що лікує, а також інформація про лікарів (ПІБ, спеціальність).

Реалізувати можливість отримання інформації про лікарів, що мають певну спеціальність, а також пошук інформації про пацієнтів по даним адреси.

Варіант 4

"Готель"

В базі даних повинна зберігатися наступна інформація про номери готелю: клас, поверх, вартість за добу, кількість місць, зайнятий чи вільний, а також інформація про людей, що проживають в готелі (ПІБ, номер паспорта, номер, в якому проживає, кількість днів проживання).

Реалізувати можливість отримання інформації про вільні номери з вартістю проживання, яка не перевищує ту, що задав користувач. При оформленні замовлення видати клієнту загальну вартість за всі дні проживання.

Варіант 5

"Ювелірне ательє"

В базі даних повинна зберігатися наступна інформація про ювелірні вироби з урахуванням типа виробу (каблучка, ланцюжок, сережки та ін.), кількості матеріалів (камінь, метал), що необхідні для створення виробів, назви виробу та вартості роботи.

Реалізувати можливість отримання інформації про вироби обраного типу, а також функцію оформлення замовлення з автоматичним підрахуванням його загальної вартості.

Варіант 6

"Телефонний довідник"

В базі даних повинна зберігатися інформація як про підприємства, так і про приватних осіб.

Про підприємства: тип підприємства, адреса, номер телефону, факс.

Про приватних осіб: ПІБ, адреса, номер телефону.

Реалізувати можливість пошуку необхідних даних по деякої сукупності букв, що входять в назву підприємства або прізвище приватної особи, а також можливість пошуку інформації про підприємство або приватну особу по даним адреси.

Варіант 7

"Деканат"

В базі даних повинна зберігатися наступна інформація про студентів: ПІБ, дата народження, адреса, номер телефону, група, підсумкові оцінки по предметах за поточний семестр.

Реалізувати можливість пошуку інформації про студентів по прізвищу, а також відображення списку студентів обраної групи з зазначенням їх підсумкові оцінок за семестр.

Варіант 8

"Бібліотека"

В базі даних повинна зберігатися наступна інформація про книги: назва, автор, жанр, рік видання, видавництво, кількість вільних екземплярів, а також інформація про читачів-студентів (ПІБ, група, які книги має на руках).

Реалізувати можливість пошуку інформації про наявність певної книги. Реалізувати процес видачі книги читачеві та автоматизувати процес переходу книг з "вільних" в "ті, що на руках".

Варіант 9

"Щоденник"

Зберігає інформацію про певні події на певні дату та час з зазначенням їх пріоритету (дуже важливо, важливо, ...).

Реалізувати можливість отримання інформації про події на певну дату, при цьому виводити список подій в порядку їх настання та шрифтом певного кольору в залежності від пріоритету. Додати функцію нагадування про подію за добу до його настання

Варіант 10

"Дитячий садок"

В базі даних повинна зберігатися наступна інформація про дітей: ПІБ, рік народження, адреса, телефон, інформація про батьків, група, вихователь.

Реалізувати можливість перегляду списку дітей групи, що закріплена за обраним вихователем, а також пошук інформації про дитину по деякої сукупності букв, що входять в її прізвище або ім'я.

Варіант 11

"Музичний магазин"

В базі даних повинна зберігатися наступна інформація про музичні альбоми: назва, виконавець, рік виходу, носій (аудіокасета, компакт-диск, відеокасета та ін.), кількість екземплярів.

Реалізувати можливість пошуку інформації про товар по першим буквам назви (імені) виконавця та назви альбому, а також перегляд списку музичної продукції, тимчасово відсутньої в магазині.

Варіант 12

"Кулінарна книга"

В базі даних повинна зберігатися наступна інформація про страви: назва страви, тип (напої, м'ясні страви, овочеві страви, перші страви та ін.), склад, описання процесу приготування, калорійність.

Реалізувати можливість пошуку інформації про спосіб приготування страви по першим буквам її назви, а також пошук страв з калорійністю, яка не перевищує ту, що задав користувач.

Варіант 13

"Футбольна ліга"

В базі даних повинна зберігатися наступна інформація про футболістів: ПІБ, дата народження, зріст, вага, позиція на полі, а також команда, за яку він грає. Передбачити також зберігання даних про тренерів команд.

Реалізувати можливість перегляду інформації про команду, яку тренує обраний тренер з зазначенням списку її гравців, а також пошук інформації про футболістів по їх ігровому амплуа.

Варіант 14
"Періодичні видання"

В базі даних повинна зберігатися наступна інформація про періодичні видання: індекс, назва, періодичність виходу, тип (газета, журнал), видавництво, вартість одного номеру.

Реалізувати можливість перегляду інформації про видання обраного типу, а також оформлення передплати з автоматичним підрахуванням її загальної вартості.

Варіант 15
"Розклад уроків"

В базі даних повинна зберігатися наступна інформація про шкільні заняття: день тижня, номер уроку, назва предмету, клас, ПІБ вчителя, номер шкільної аудиторії.

Реалізувати можливість перегляду розкладу на тиждень для певної шкільної аудиторії, а також пошук інформації про всі заняття, що проводяться в певний день тижня.

5 ОРГАНІЗАЦІЯ КОНТРОЛЮ ЗНАНЬ ТА ВМІНЬ СТУДЕНТІВ

5.1 Система контролю знань та вмінь студентів

Поточний контроль з дисципліни «Технології та стандарти взаємодії у глобальних мережах» здійснюється на протязі навчального курсу (семестру) за наступними формами:

– перевірка контрольної роботи, яка виконується у міжсесійний період;

– перевірка знань та вмінь студента під час аудиторних занять протягом заліково-екзаменацйної сесії.

Підсумковий контроль з дисципліни «Технології та стандарти взаємодії у глобальних мережах» проводиться на основі накопиченої (інтегральної) суми балів, яку отримав студент по підсумках поточного контролю та підсумкового семестрового контролю у виглядізаліку.

5.2 Форми контролю знань та вмінь студентів

При вивченні дисципліни «Технології та стандарти взаємодії у глобальних мережах» використовується накопичувальна система оцінювання. Максимальна сума балів, що може одержати студент, дорівнює 100 балів. Накопичувальна підсумкова оцінка засвоєння студентом навчальної дисципліни «Технології та стандарти взаємодії у глобальних мережах» складається з:

№п/п	Види завдань, за які нараховують бали за змістовними модулями	Максимальна кількість балів, яка нараховується за виконання певного виду завдання
1.	Виконання Контрольної роботи в міжсесійний період.	60 балів
2.	Усне опитування студента перед початком виконання лабораторних робіт у сесійний період.	15 балів
3	Підготовка та виконання лабораторних робіт під час заліково-екзаменаційної сесії.	15 балів
4.	Залікова (підсумкова) контрольна робота	10 балів
	Разом	100 балів

Поточна та підсумкова оцінка рівня знань студентів здійснюється за модульною системою організації навчання та контролю знань студентів. Суми балів які отримав студент за всіма модулями КСРС навчальної дисципліни, формують інтегральну оцінку поточного контролю даного студента з навчальної дисципліни.

Оцінювання самостійної роботи студента в міжсесійний період здійснюється у формі оцінки виконання контрольної роботи. Таким чином загальна максимальна оцінка за виконання Контрольної роботи складає 60 балів:

– Завдання №1 – максимальна оцінка – 30 балів

– Завдання №2 – максимальна оцінка – 30 балів

Контрольна робота зараховується, якщо студент отримав сумарну оцінку не менш 36 балів (тобто не менш 60% від максимальної суми в 60 балів). Студент, який отримав за виконання контрольної роботи сумарну оцінку меншу за 36 балів (тобто – «незадовільно») не допускається до підсумкового контролю.

Оцінка роботи студента при проведенні лабораторних занять по дисципліні під час екзаменаційно-залікової сесії складається з:

– оцінки знань студента під час усного опитування перед початком виконання лабораторних робіт (максимально 15 балів);

– оцінки знань та вмінь студента при виконанні лабораторної роботи в комп'ютерному класі (максимально 15 балів).

– оцінки залікової (підсумкової) контрольної роботи (максимально 10 балів).

Загальна максимально оцінка за цей вид поточного контролю складає 40 балів.

Студенти, які виконали контрольну роботу у міжсесійний період (отримали не менше ніж 36 балів (60%)), а також виконали лабораторні роботи у сесійний період та підсумкову контрольну роботу (отримали не менше ніж 24 бали (60%)), мають залік з дисципліни.

Максимальна сума, що може одержати студент – 100 балів:

60 балів – міжсесійна контрольна робота,

40 балів – робота студента у сесійний період.

Кількісні та якісні критерії оцінки :

Діапазон оцінки відповіді, %	Якісні критерії оцінки відповіді
90 – 100	відмінне виконання лише з незначною кількістю помилок
82 – 89,9	вище середнього рівня з кількома помилками
74 – 81,9	в загальному правильна робота з певною кількістю помилок
64 – 73,9	непогано, але зі значною кількістю помилок
60 – 63,9	відповідь в цілому достатня, що свідчить про певні знання студента з поставленого питання, але у відповіді є суттєві помилки або виявляються прогалини у знаннях з поставленого питання
35 – 59,9	є окремі вірні думки, але в цілому відповідь недостатня або багато помилок, які формують в цілому невірну відповідь
1 – 34,9	студент не відповів зовсім на питання або відповідь у більшій частині невірна

Одержана накопичена підсумкова оцінка виставляється викладачем у заліково-екзаменаційну відомість встановленого зразка, відповідно до шкали ЄКТАС:

Шкала оцінювання за системою ЄКТАС та за системою університету

За шкалою ЄКТАС	За національною системою	Визначення	За системою університету (в %)
A	5 (відмінно)	відмінне виконання лише з незначною кількістю помилок	**90 – 100**
B	4 (добре)	вище середнього рівня з кількома помилками	**85 – 89**
C	4 (добре)	в загальному правильна робота з певною кількістю помилок	**75 – 84**
D	3 (задовільно)	непогано, але зі значною кількістю помилок	**68 – 74**
E	3 (задовільно)	виконання задовольняє мінімальні критерії	**60 – 67**
FX	2 (незадовільно)	з можливістю перекласти	**35 – 59**
F	2 (незадовільно)	з обов’язковим повторним курсом навчання	**1 – 34**

Студенти, які не отримали за контрольну роботу мінімальної кількості балів (36 балів), повинні виконати інший варіант контрольної роботи, який видається викладачем, або виправити помилки попереднього варіанту та отримати відповідну кількість балів.

5.3 Перелік базових знань та вмінь

Узагальнюючи інформацію, що викладена вище, можна навести повний перелік базових знань та вмінь з дисципліни «Технології та стандарти взаємодії у глобальних мережах».

1. Тема: «Архітектура та сучасні технології інформаційних Інтернет-систем у глобальних мережах

- Принципи функціонування та обґрунтування вибору архітектур інформаційних Інтернет-систем.
- Архітектури територіально-розподілених WEB–додатків, що публікують бази даних.
- Інформаційні Інтернет-системи з розподіленою та централізованою веб-орієнтованою архітектурою.

2. Тема: «Комплекс стандартів XML»

- XML-дані та XML-документи у середовищі глобальної мережі Інтернет.
- Використання MySQL та PHP для проектування та реалізації інформаційної Web-системи з базою даних.
- Передумови та джерела технології XML.
- Призначення та функціональні можливості мови XML.
- Організацію та функціональні можливості платформи XML.
- Стандарти Веб-сервісів, спадкоємність з технологіями HTML.
- Технології семантичного WEB.

3. Тема: «Особливості XML-даних та їх моделювання»

- XML-дані у середовищі глобальної мережі Інтернет.
- створення переносимих XML-документів у середовищі глобальної мережі Інтернет.
- Технології сучасних Web-додатків та публікування баз даних в Internet.
- Призначення, загальна схема роботи та середовище роботи Web-сервера. Технології сучасних Web-додатків.
- Засоби розробки Web-додатків з модулями розширення Web- сервера та Web-оглядача.

4. Тема: «XML-орієнтовані бази даних»

- Публікація баз даних з використанням XML
- XML, як універсальний спосіб обміну даними і створення переносимих документів.
- Класифікація XML-орієнтованих СУБД.
- Функціональні можливості XML-орієнтованих СУБД.